Computer Science Logic 2013

22nd Annual Conference of the EACSL
CSL'13, September 2–5, 2013, Torino, Italy

Edited by

Simona Ronchi Della Rocca

Editor

Simona Ronchi Della Rocca
Dipartimento di Informatica
Università di Torino
ronchi@di.unito.it

ACM Classification 1998
A.0 Conference Proceedings, F Theory of Computation, C.2.4 Distributed Systems,
D.2.4 Software/ Programs Verifications, D.3.1 Formal Definitions and Theory,
D.3.3 Languages Constructs and Features, I.2.4 Knowledge Representations Formalisms and Methods

ISBN 978-3-939897-60-6

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-939897-60-6.

Publication date
September, 2013

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Digital Object Identifier: 10.4230/LIPIcs.CSL.2013.i

ISBN 978-3-939897-60-6 **ISSN 1868-8969** **http://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8969

www.dagstuhl.de/lipics

Contents

Editor's Preface ix

Conference Organization xi

External Reviewers xiii

Report on the Ackermann Award 2013

The Ackermann Award 2013
Anuj Dawar, Thomas A. Henzinger, and Damian Niwiński 1

Abstracts of Invited Talks

Res Publica: The Universal Model of Computation
Nachum Dershowitz 5

Three lightings of logic
Jean-Yves Girard 11

From determinism, non-determinism and alternation to recursion schemes for P, NP and Pspace
Isabel Oitavem 24

Means and Limits of Decision
Lidia Tendera 28

Contributed Papers

On closure ordinals for the modal mu-calculus
Bahareh Afshari and Graham E. Leigh 30

Realizability and Strong Normalization for a Curry-Howard Interpretation of HA + EM1
Federico Aschieri, Stefano Berardi, and Giovanni Birolo 45

Bounds for the quantifier depth in finite-variable logics: Alternation hierarchy
Christoph Berkholz, Andreas Krebs, and Oleg Verbitsky 61

Unambiguity and uniformization problems on infinite trees
Marcin Bilkowski and Michał Skrzypczak 81

A characterization of the Taylor expansion of λ-terms
Pierre Boudes, Fanny He, and Michele Pagani 101

Team building in dependence
Julian Bradfield 116

Saturation-Based Model Checking of Higher-Order Recursion Schemes
Christopher Broadbent and Naoki Kobayashi 129

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Descriptive complexity of approximate counting CSPs
Andrei Bulatov, Victor Dalmau, and Marc Thurley 149

What is Decidable about Partially Observable Markov Decision Processes with omega-Regular Objectives
Krishnendu Chatterjee, Martin Chmelik, and Mathieu Tracol 165

Infinite-state games with finitary conditions
Krishnendu Chatterjee and Nathanaël Fijalkow 181

Annotation-Free Sequent Calculi for Full Intuitionistic Linear Logic
Ranald Clouston, Jeremy Dawson, Rajeev Goré, and Alwen Tiu 197

Deciding the weak definability of Büchi definable tree languages
Thomas Colcombet, Denis Kuperberg, Christof Löding, and Michael Vanden Boom 215

Innocent Game Semantics via Intersection Type Assignment Systems
Pietro Di Gianantonio and Marina Lenisa 231

Cuts for circular proofs: semantics and cut-elimination
Jérôme Fortier and Luigi Santocanale 248

Hierarchies in independence logic
Pietro Galliani, Miika Hannula, and Juha Kontinen 263

Inclusion Logic and Fixed Point Logic
Pietro Galliani and Lauri Hella 281

Theories for Subexponential-size Bounded-depth Frege Proofs
Kaveh Ghasemloo and Stephen A. Cook 296

The Structure of Interaction
Stéphane Gimenez and Georg Moser 316

The Fixed-Parameter Tractability of Model Checking Concurrent Systems
Stefan Göller 332

One-variable first-order linear temporal logics with counting
Christopher Hampson and Agi Kurucz 348

On the locality of arb-invariant first-order logic with modulo counting quantifiers
Frederik Harwath and Nicole Schweikardt 363

When is Metric Temporal Logic Expressively Complete?
Paul Hunter 380

Proving Strong Normalisation via Non-deterministic Translations into Klop's Extended λ-Calculus
Kentaro Kikuchi 395

Kleene Algebra with Products and Iteration Theories
Dexter Kozen and Konstantinos Mamouras 415

Internalizing Relational Parametricity in the Extensional Calculus of Constructions
Neelakantan R. Krishnaswami and Derek Dreyer 432

Modal Logic and Distributed Message Passing Automata
Antti Kuusisto 452

Global semantic typing for inductive and coinductive computing
Daniel Leivant 469

Two-Variable Logic on 2-Dimensional Structures
Amaldev Manuel and Thomas Zeume 484

Categorical Duality Theory: With Applications to Domains, Convexity, and the Distribution Monad
Yoshihiro Maruyama 500

Axiomatizing Subtyped Delimited Continuations
Marek Materzok 521

On dialogue games and coherent strategies
Paul-André Melliès 540

Elementary Modal Logics over Transitive Structures
Jakub Michaliszyn and Jan Otop 563

A Fully Abstract Game Semantics for Parallelism with Non-Blocking Synchronization on Shared Variables
Susumu Nishimura 578

Extracting Herbrand trees in classical realizability using forcing
Lionel Rieg 597

The Complexity of Abduction for Equality Constraint Languages
Johannes Schmidt and Michał Wrona 615

A New Type Assignment for Strongly Normalizable Terms
Rick Statman 634

Semantics of Intensional Type Theory extended with Decidable Equational Theories
Qian Wang and Bruno Barras 653

Editor's Preface

The annual conference of the European Association for Computer Science Logic (EACSL), CSL'13, was held in Torino, Italy, from September 2 to September 5, 2013. CSL started as a series of international workshops on Computer Science Logic, and became at its sixth meeting the Annual Conference of the EACSL. This conference was the 27th workshop and 22th EACSL conference; it was organized by the Dipartimento di Informatica of the Università di Torino (UNITO).

The Ackermann Award is the EACSL Outstanding Dissertation Award for Logic in Computer Science. This year, the jury decided to give the Ackermann Award for 2013 to Matteo Mio. The awards were officially presented at the conference (September 3). The citation of the awards, an abstract of the thesis, and a biographical sketch of the recipients written by Anuj Dawar, Tom Henzinger and Damian Niwinski may be found in the proceedings.

A total of 130 abstracts were registered and 108 of these were followed by full papers submitted to CSL'13. After a two weeks electronic meeting, the Program Committee selected 37 papers for presentation at the conference and publication in these proceedings. Each paper was assigned to at least three PC members. The overall quality of the submissions was really high. The program committee did not fix a strict a priori limit on the number of accepted papers and wished to accept as many good papers as possible. However, at the end some of them had to be rejected due to lack of space.

In addition to the contributed talks, CSL'13 had four invited speakers: Nachum Dershovitz (Tel Aviv University), Jean Yves Girard (CNRS, Marseille), Isabel Oitavem (Universidade Lisboa), Lidia Tendera (University of Opolskiego). Abstracts of the invited talks are included in the proceedings. A welcome talk by Piergiorgio Odifreddi has been held on September 1, but it is not included in these proceedings.

I wish to warmly thank the PC and all external reviewers for their precious help in reviewing the papers. Our thanks also go to the members of the Organizing Committee, for their considerable efforts in organizing the conference, to Luca Padovani for his great work in preparing the proceedings, and to Marc Herbstritt (Dagstuhl Publishing) for his technical help.

The conference received support from the Dipartimento di Informatica of the Università di Torino, from the EACSL, from the GNSAGA group of INDAM (Istituto Nazionale di Alta Matematica "F.Severi"), from the Regione Piemonte, which offered the conference location, and from the Università di Torino, which offered the location for the welcome talk and the associated aperitif. I thank these organizations for their generous supports.

September 2013 Simona Ronchi Della Rocca

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Conference Organization

Program Committee

Zena Ariola	University of Oregon
Arnon Avron	Tel-Aviv University
Roberto Bagnara	University of Parma and BUGSENG srl
Christel Baier	Technical University of Dresden
Marc Bezem	University of Bergen
Paola Bruscoli	University of Bath
Agata Ciabattoni	TU Wien
Thierry Coquand	Chalmers University
Ugo Dal Lago	Università di Bologna
Valeria De Paiva	Nuance Communications
Reinhard Kahle	Universidade Nova de Lisboa
Stephan Kreutzer	Technical University Berlin
Olivier Laurent	CNRS - ENS Lyon
Carsten Lutz	Universität Bremen
Jean-Yves Marion	Université de Lorraine
Damian Niwinski	Warsaw University
Frank Pfenning	Carnegie Mellon University
Elaine Pimentel	Universidade Federal de Minas Gerais
Ruzica Piskac	MPI-SWS Saarbrücken
Simona Ronchi Della Rocca (chair)	Università di Torino
Jan Rutten	CWI Amsterdam
Helmut Schwichtenberg	LMU Munich
Phil Scott	University of Ottawa
Peter Selinger	Dalhousie University
Makoto Tatsuta	NII Tokyo
Tachio Terauchi	Nagoya University

Organizing Committee

Erika De Benedetti	Università di Torino - ENS Lyon
Luca Paolini	Università di Torino
Paola Giannini	Università del Piemonte Orientale
Simona Ronchi Della Rocca	Università di Torino
Mauro Piccolo	Università di Bologna
Luca Roversi	Università di Torino
Luca Padovani	Università di Torino
Angelo Troina	Università di Torino

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

External Reviewers

Abel, Andreas
Abramsky, Samson
Accattoli, Beniamino
Aczel, Peter
Adamek, Jiri
Alama, Jesse
Arbiser, Ariel
Asperti, Andrea
Atserias, Albert
Autexier, Serge
Baelde, David
Baillot, Patrick
Balabonski, Thibaut
Bartha, Miklos
Ben-Amram, Amir
Berardi, Stefano
Berger, Ulrich
Bernardy, Jean-Philippe
Berwanger, Dietmar
Bodirsky, Manuel
Bollig, Benedikt
Bonfante, Guillaume
Boudes, Pierre
Bradfield, Julian
Brotherston, James
Buchholz, Wilfried
Buss, Sam
Cardone, Felice
Chaudhuri, Kaustuv
Clairambault, Pierre
Cleaveland, Rance
Cockett, Robin
Colcombet, Thomas
D'Agostino, Giovanna
Dal Palù, Alessandro
Dawar, Anuj
De Freitas, Renata
De Nivelle, Hans
de Vries, Fer-Jan
de'Liguoro, Ugo
Decker, Normann
Demri, Stéphane
Dimitrova, Rayna
Dittmann, Christoph
Dubslaff, Clemens
Durand, Arnaud
Dyer, Martin
Ehlers, Ruediger
Faber, Wolfgang
Facchini, Alessandro
Fearnley, John
Fiore, Marcelo
Gabbrielli, Maurizio
Gaboardi, Marco
Galesi, Nicola
Geuvers, Herman
Ghica, Dan
Gimbert, Hugo
Goncharov, Sergey
Goyet, Alexis
Graham-Lengrand, Stéphane
Grigorieff, Serge
Grädel, Erich
Guerrini, Stefano
Göller, Stefan
Hardin, Thérèse
Harwath, Frederik
Haveraaen, Magne
Heijltjes, Willem
Hirschkoff, Daniel
Hodkinson, Ian
Hofstra, Pieter
Hovland, Dag
Hyvernat, Pierre
Ilik, Danko
Immerman, Neil
Jacobs, Bart
Jahren, Eivind
Jeřábek, Emil
Johannsen, Jan
Jonsson, Peter
Kameyama, Yukiyoshi
Kartzow, Alexander
Kashev, Alexander
Kieronski, Emanuel
Kimura, Daisuke
Klop, Jan Willem
Klüeppelholz, Sascha
Konev, Boris
Kontchakov, Roman
Krivine, Jean-Louis
Krupski, Vladimir
Kullmann, Oliver
Kupferman, Orna
Kuske, Dietrich
Kuznets, Roman
La Torre, Salvatore
Laird, James
Lee, Gyesik
Lenzi, Giacomo
Lescanne, Pierre
Levy, Jordi

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Levy, Paul Blain
Löding, Christof
Makowsky, Johann
Mann, Allen
Mastroeni, Isabella
Matthes, Ralph
Mazza, Damiano
McCusker, Guy
McKinley, Richard
Merz, Stephan
Michalewski, Henryk
Mio, Matteo
Mueller, Moritz
Nakazawa, Koji
Nigam, Vivek
Niksic, Filip
Norman, Gethin
Normann, Dag
Olarte, Carlos
Oliva, Paulo
Ong, Luke
Otto, Martin
Paolini, Luca
Parys, Pawel
Pattinson, Dirk
Pedicini, Marco
Petrić, Zoran
Piccolo, Mauro
Piecha, Thomas
Pilipczuk, Michal
Pinna, Giovanni Michele
Polonsky, Andrew
Pottier, François
Pous, Damien
Pédrot, Pierre-Marie
Quaas, Karin
Rabinovich, Alex
Ramanayake, Revantha
Ramirez, Carlos
Ranise, Silvio
Razborov, Alexander
Reus, Bernhard
Reynolds, Mark
Riba, Colin
Rose, Kristoffer
Sacerdoti Coen, Claudio
Salibra, Nino
Salzer, Gernot
Sandu, Gabriel
Saurin, Alexis
Schnoor, Henning
Seiller, Thomas
Setzer, Anton
Shan, Chung-chieh
Shavrukov, Volodya
Siebertz, Sebastian
Simmons, Robert
Skalka, Christian
Spendier, Lara
Spiwack, Arnaud
Straßburger, Lutz
Suter, Philippe
Sznajder, Nathalie
Sørensen, Morten Heine
Tasson, Christine
Tendera, Lidia
Terui, Kazushige
Thapen, Neil
Thomas, Wolfgang
Tiu, Alwen
Tolmach, Andrew
Tsukada, Takeshi
Tzevelekos, Nikos
Urzyczyn, Pawel
Uustalu, Tarmo
Valeriote, Matt
Van Breugel, Franck
Varacca, Daniele
Vaux, Lionel
Veith, Helmut
Vianu, Victor
Vickers, Steve
Viswanathan, Mahesh
Vollmer, Heribert
Walukiewicz, Igor
Wang, Bow-Yaw
Weirich, Stephanie
Weis, Philipp
Weller, Daniel
Westerbaan, Bram
Wies, Thomas
Witkowski, Piotr
Wooldridge, Michael
Zanuttini, Bruno
Zeilberger, Noam
Zhou, Chunlai
Zuliani, Paolo

The Ackermann Award 2013

Anuj Dawar, Thomas A. Henzinger, and Damian Niwiński

Members of EACSL Jury of the Ackermann Award

The ninth Ackermann Award is presented at CSL'13, held in Turin, Italy. This year, as in the previous three years, the EACSL Ackermann Award is generously sponsored by the Kurt Gödel Society. Besides providing financial support for the Ackermann Award, the Kurt Gödel Society has also committed to inviting the recipient of the Award for a special lecture to be given to the Society in Vienna.

The 2013 Ackermann Award was open to PhD dissertations in topics specified by the CSL and LICS conferences, which were formally accepted as theses for the award of a PhD degree at a university or equivalent institution between 1 January 2011 and 31 December 2012. The Jury received fifteen nominations for the Ackermann Award 2013. The candidates came from a number of different countries across the world. The institutions at which the nominees obtained their doctorates represent nine countries in Europe, North America, and the Middle East.

All submissions were of a very high standard and contained remarkable contributions to their particular fields. The Jury wishes to extend its congratulations to all nominated candidates for their outstanding work. The Jury encourages them to continue their scientific careers and hopes to see more of their work in the future.

With such an outstanding field of nominees, the task of the jury was difficult. In the end, after much discussion, one thesis stood out. The **2013 Ackermann Award** winner is:

- Matteo Mio from Italy, for his thesis
  Game Semantics for Probabilistic Modal μ-Calculi
  approved by the University of Edinburgh, UK, in 2012,
  supervised by Alex Simpson.

Matteo Mio

Citation. Matteo Mio receives the *2013 Ackermann Award* of the European Association of Computer Science Logic (EACSL) for his thesis

Game Semantics for Probabilistic Modal μ-Calculi.

His thesis builds an extension of the modal μ-calculus suitable for reasoning about non-deterministic probabilistic systems. It advances previous approaches, and adds a quantitative dimension to the game semantics of fixed-point logics, via a novel concept of a tree game, integrating randomness and concurrency.

Background of the Thesis. The modal μ-calculus lies at the very heart of logics and algorithms for computer-aided verification: it provides a powerful framework for comparing specification formalisms and devising model-checking algorithms for discrete dynamical systems, such as hardware and software systems. In order to model uncertainty in the behavior of such systems, it is natural to extend both state transition models and property specification languages with probabilistic aspects; the first such probabilistic temporal logic was introduced by Hansson and Jonsson in the early 1990s, and probabilistic extensions of the modal μ-calculus followed quickly. The resulting field of "probabilistic verification" has received much attention in the past two decades, which saw the solution of many probabilistic model-checking problems, the development of corresponding verification tools, and their application to case studies ranging from networking to systems biology. Yet the field still lacks

Computer Science Logic 2013 (CSL'13). doi: 10.4230/LIPIcs.CSL.2013.i.
Editor: Simona Ronchi Della Rocca; pp. 1–4

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a convincing canonical and foundational framework for specifying and comparing probabilistic properties. The thesis by Mio presents a promising step in this direction.

Mio's Thesis. In his thesis, Matteo Mio adopts a quantitative approach to temporal logics over probabilistic transition systems, introduced by Huth and Kwiatkowska, and independently by Morgan and McIver, where a formula holds in a state with some probability. An interpretation of a formula is therefore a mapping from the set of states to $[0, 1]$. In the probabilistic μ-calculus pLμ introduced by Morgan and McIver, conjunction and disjunction are interpreted as min and max over reals, respectively. The first contribution of the thesis extends to all models the equivalence between denotational and game semantics of the logic pLμ, established previously by Morgan and McIver for finite models. The logic pLμ, however, is not a completely satisfactory generalization of its classical counterpart to the probabilistic setting. Indeed, in contrast to the propositional μ-calculus Lμ, which subsumes most of the temporal logics known in the literature, pLμ fails to contain the probabilistic version of the most basic temporal logic CTL as its sublogic. A remedy proposed by Mio consists in using different real extensions of the Boolean operators *and* and *or* and combining them in a single logic. These interpretations have been already considered by Huth and Kwiatkowska as alternatives: in addition to min and max used in pLμ, also product $x\dot{y}$ (for *and*), and its dual co-product

$$x \odot y \quad = \quad x + y - xy$$

(for *or*), as well as the strong conjunction and disjunction of the Łukasiewicz logic

$$\begin{aligned} x \ominus y &= \max(0, x + y - 1) \\ x \oplus y &= \min(1, x + y). \end{aligned}$$

Mio shows that with all these operators one can express the probabilistic version of CTL, whereas the first two suffice for the qualitative fragment of this logic. Thus a new powerful fixed-point logic has emerged, whose expressive power and algorithmic properties are not yet completely understood. This will likely be the subject of active research in future years. What Matteo Mio contributes in his thesis is the game semantics for the new logic.

A known feature of μ-calculi is that, in contrast to, e.g., first-order logic or temporal logic, they did not arise as a formalization of natural language constructs, but rather as equational systems. As a result, fixed-point formulas are relatively hard to understand by humans. This difficulty only increases for a probabilistic version of the logic involving three variants of conjunction and disjunction. In the classical case, a helpful way of understanding the μ-calculus formulas is via games. More specifically, a formula φ of Lμ and a model M induce a perfect-information two-person game of possibly infinite duration, a so-called parity game, such that the satisfaction $M \models \varphi$ is equivalent to the existence of a winning strategy for the proponent in this game. This characterization is also at the basis of many model-checking algorithms, which thus boil down to solving games. As we have already mentioned, the thesis settles a similar characterization for the probabilistic μ-calculus pLμ. However, the main contribution of the thesis consists in establishing the game semantics for the full probabilistic μ-calculus pL$\mu^{\odot}_{\oplus}$ described above. *A priori* it is not obvious that this is possible, as the real functions used in this μ-calculus do not have any apparent game interpretation. Now Matteo Mio makes an unexpected twist in the very paradigm of game playing. He admits that a play need not be a linear process, but can instead split in several threads, which form of a tree. This tree can serve as an arena of a new (inner) game, and the payoff of the original (outer) game is defined in terms of winning the inner game. Here, the

outer game is usually stochastic, whereas the inner game is a perfect-information game. This construction, referred to by the author as a *tree game*, leads to the concept of meta games, parametrized by the class of inner games. The game semantics of the extended probabilistic μ-calculus is provided by meta parity games.

Mio also discovers a number of remarkable properties of tree games, which make this concept interesting in its own right. In particular, the tree games turn out to comprise (under suitable encoding) the Blackwell games, which is a class of infinite stochastic games with imperfect information that is well-studied in game theory. The determinacy of Blackwell games established by Donald Martin in 1998 is considered to be one of the strongest determinacy results provable in ZFC. Another feature of tree games is that they can be derandomized; i.e., the stochastic player *Nature* can be eliminated, its role taken by the concurrent branching mechanism.

The game semantics of the μ-calculus $\mathrm{pL}\mu^{\odot}_{\oplus}$ relies on the determinacy result for the meta parity games. This is the most technically difficult part of the thesis. Indeed, the argument requires some properties of sets in $\mathbf{\Delta^2_1}$, which do not, in general, hold in ZFC. Therefore, the author proves his results in ZFC extended by the Martin axiom for the first uncountable cardinal, $\mathrm{MA}_{\aleph_1}$.

Biographical Sketch. Matteo Mio was born on 5 July 1983. He was a student at the University of Udine in Italy during the period 2002-2007, studying for the *Laurea Triennale* and *Laurea Specialistica* in Computer Science. In 2007 he joined the University of Edinburgh in Scotland to pursue a PhD degree, which he completed in February 2012. Since then, he has spent a year as a postdoctoral researcher at the École Polytechnique in Paris and is currently a postdoctoral researcher at the Centrum Wiskunde & Informatica (CWI) in Amsterdam, funded by an ERCIM Alain Bensoussan fellowship.

Jury

The Jury for the **Ackermann Award 2013** consisted of eight members, two of them *ex officio*, namely, the president and the vice-president of EACSL. A member of the LICS organising committee is also normally a member of the jury. On this occasion, this member withdrew owing to a conflict of interest and a replacement was named.

The members of the jury were:

- Thierry Coquand (Chalmers University of Gothenburg),
- Anuj Dawar (University of Cambridge), the president of EACSL,
- Thomas A. Henzinger (IST Austria),
- Daniel Leivant (Indiana University, Bloomington),
- Damian Niwiński (University of Warsaw),
- Catuscia Palamidessi (École Polytechnique, Paris),
- Simona Ronchi della Rocca (University of Torino), the vice-president of EACSL,
- Wolfgang Thomas (RWTH, Aachen).

Previous winners

Previous winners of the Ackermann Award were

2005, Oxford:

Mikołaj Bojańczyk from Poland,

Konstantin Korovin from Russia, and
Nathan Segerlind from the USA.

2006, Szeged:
Balder ten Cate from The Netherlands, and
Stefan Milius from Germany.

2007, Lausanne:
Dietmar Berwanger from Germany and Romania,
Stéphane Lengrand from France, and
Ting Zhang from the People's Republic of China.

2008, Bertinoro:
Krishnendu Chatterjee from India.

2009, Coimbra:
Jakob Nordström from Sweden.

2010, Brno:
No award given.

2011, Bergen:
Benjamin Rossman from USA.

2012, Fontainebleau:
Andrew Polonsky from Ukraine, and
Szymon Toruńczyk from Poland.

Detailed reports on their work appeared in the CSL proceedings and are also available on the EACSL homepage.

Res Publica: The Universal Model of Computation

Nachum Dershowitz

School of Computer Science, Tel Aviv University, Ramat Aviv, Israel
nachum.dershowitz@cs.tau.ac.il

Abstract

We proffer a model of computation that encompasses a broad variety of contemporary generic models, such as cellular automata—including dynamic ones, and abstract state machines—incorporating, as they do, interaction and parallelism. We ponder what it means for such an intertwined system to be effective and note that the suggested framework is ideal for representing continuous-time and asynchronous systems.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Models of computation, cellular automata, abstract state machines, causal dynamics, interaction

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.5

Category Invited Talk

> The nature of the process is truly characterized by Glaucon, when he describes himself as a companion who is not good for much in an investigation, but can see what he is shown, and may, perhaps, give the answer to a question more fluently than another.
> —Plato, *The Republic*

1 Purpose

The goal of this study is to design a model of computation that encompasses various and sundry generic models, such as dynamic cellular automata [1], as well as interactive and parallel abstract state machines [2, 3]. Furthermore, the model should be capable of dealing with continuous-time and asynchronous systems.

We employ a political metaphor.

2 The State Model

Blocs. A *bloc* is an interconnected collection of *states* that evolve over time. The number of states in a bloc may be finite or infinite. States communicate with each other via (communication) *channels*. Not only do the internals of states evolve, but their connections may be reorganized. Furthermore, it may be possible for new states to be created and connected to existing ones.

Maps. We draw channels as pipes (looking like hoses) emanating from the client state (on the requesting end) and connected to the serving state (which owns the data that is being made public). A serving state may allow its clients to update sections of the shared data.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca ; pp. 5–10

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Arrows along the channel can be used to indicate that data flows along a channel in one direction only.

States. Each state is a logical structure (consisting of a domain, first-order vocabulary, and interpretations for the operations in its vocabulary) whose evolution is governed by its native *policy*—which may be *natural* (fixed by laws), *algorithmic* (dictated by a program), or *arbitrary* (controlled by some external agency)—and may react to its environment. As such, a state contains interpretations for the *functions* in its vocabulary (constants may be viewed as scalar functions and relations as truth-valued functions). Only the interpretations given by a state to its functions may change during evolution; the domain and vocabulary are fixed throughout.

Domains. All states in a bloc share the same domain, but can have different vocabularies. Domains may be finite (*automata*), countably infinite (*machines*), or uncountable (*processes*).

Names. States have (unique) identifying *names*, taken from a *namespace* that is included in the domains of states. Pipes in a graphical representation of this model of computation depict the use of names.

Resources. A subset of each state's vocabulary are designated *public*. Their values are made visible to other states; *private* functions are not. A *resource* is a (named) state along with one of its public functions. One can consider a framework in which public resources can be accessed but not modified by others; think of them as (read-only) *communiqués*. Alternatively, some resources can be designated *shared* and allow for modification by foreign states. No bound is placed on the number of channels connected to a state or the number of shared resources.

Assets and Agents. From the point of view of the client of a resource, a shared resource to which it is connected is its *agent*, while a public resource that is not modifiable is an *asset* of its.

Vassals. A state can only modify the values of its own functions or of shared resources to which it has access. To provide differential access to its public data, a state can set up *vassals* (or "satellites"), each of which connects to it by a private one-way channel, keeping the name of the controlling state secret (not publicly available). The vassal state can continuously retrieve the relevant part of the data from its master state and pass it on to whichever states are connected to it, the vassal.

Realignments. The topology of a bloc can change due to modifications of (the values of) its channels. In particular, if the value of a resource is itself a name, then a state can change an outgoing channel to refer to the state named by the resource.

Locations. *Locations* in a state are determined by function symbols (from the vocabulary) and domain values for its arguments (as per the arity of the symbol); it is the contents of locations that change when an interpretation is updated.

Puppets. A state may also create a *puppet*, which is a state with the same domain and vocabulary, running the same policy. Before releasing the puppet to run on its own, the controlling state may set various values in the puppet; all other locations in the puppet will retain their default values.

3 State Evolution

Time. States evolve over time, where *time* $\mathbb{T}$, in general, can be any linearly-ordered domain, with ordering $\leq$ and minimal element, denoted 0. Let $\mathbb{S}$ be the initial intervals $[0, t)$ for all $t \in \mathbb{T}$.

Discrete Time. For discrete systems, time is the natural numbers $\mathbb{N}$, with initial segments $\mathbb{S} = [0..n)$, for $n \in \mathbb{N}$.

Continuous Time. For continuous behavior, time $\mathbb{T}$ would be the non-negative reals.

Signals. Each resource to which a state is connected provides it with a *signal*, which is a function from an interval in $\mathbb{S}$ to the domain of the bloc. A signal defined for an interval $[0, t)$ has *length* t. Concatenation of a signal of length s with one of length t gives a signal of length $s + t$ in the obvious way.

Interaction. Channels provide a means for communication between states, but there is no special mechanism for explicitly responding to requests. Clearly, the signal emitted by one resource may depend on signals emitted by others. That is the nature of interaction.

Environments. The ensemble of signals reaching a state constitutes its *environment*. Let the possible environments, Σ, be all tuples of signals of the same length. The *width* of an environment the number of components in the tuple. The concatenation $\alpha\beta$ of environments $\alpha, \beta \in \Sigma$ of the same width is the tuple of concatenated signals. Write $\alpha \leq \gamma$ if there exists a β such that $\alpha\beta = \gamma$.

Evolutions. Policies are described by *transition* functions τ (perhaps multivalued) that map states and environments to states. That is, $\tau : X \times \Sigma \rightrightarrows X$. The *evolution* of a state x for a given environment γ is the sequence of states obtained in this way: $\{\tau_\alpha(x)\}_{\alpha \leq \gamma}$.

Causality. Let τ be a transition function. Transitions must be *causal* ("retrospective"), depending only on the past, so that $\tau_{\alpha\beta}(x) = \tau_\beta(\tau_\alpha(x))$ for all states x, where $\alpha\beta$ is a concatenated environment. If τ is multivalued, then τ_β should be understood as extended to sets. Put differently, $\tau_{\alpha\beta} = \tau_\alpha \circ \tau_\beta$, as relations.

Federations. One can view a subset of the states as one *federated* state. The transitions of the federation depend on its external environment, mediated by channels from the outside.

Globe. The *global* federation consists of the totality of states, or at least those states that are governed by programs or processes.

4 State Programs

Programs. Algorithmic policies may be described by programs. Programs operating in discrete time must define the one-step transition relation. This may be done in the basic language of abstract state machines [6], which includes the following at a minimum:

- general assignments: $f(s_1, \ldots, s_k) := t$ (terms s_i, t in the vocabulary of the state)
- conditionals: **if** c **then** P (Boolean term c and program P), and
- parallel composition: $P \parallel Q$ (programs P, Q).

In addition, we want

- higher-order assignments: $f := g$, where f and g are functions (of the state vocabulary) of the same arity, and
- serial composition: $P; Q$ (programs P, Q).

Channels. A channel is a name-valued location. A *foreign* location is indicated by an expression of the form $p.\ell$, where p is a channel and ℓ is a location. Only local locations and shared resources may appear on the left of assignments. A foreign resource on the left of an assignment is an agent; if it only appears on the right side or in conditions, it is an asset (that is read-only).

Dependence. A new state may be conceived with a

- creation assignment: $p :=$ **new** $f, g, \ldots$ **allow** $h, k, \ldots$.

The new puppet state, pointed to by p, with have public functions $f, g, \ldots, h, k, \ldots$, with the second half of the list shared freely. When launched, the puppet will run the same programmed policy as its parent. Assignments may be made to locations in unlaunched puppets (high-level assignments are of help here); flags can be used to specialize the behavior of puppets.

Independence. The

- launch command: **free** p

activates the program in the puppet pointed to by p, at which point the parent can no longer modify it on its own. The puppet is now independent.

Federations. The program of the federation as a whole is just the union of the programs of its constituent states, with functions disambiguated by the name of the state they reside in. (Of course, some states might not be governed by programs, but rather provide measurements of natural phenomena like temperature and barometric pressure.) Whereas an individual programmed state has a bounded number of channels it owns, a federation can create more and more new states, each of which is connected to non-federated states.

Flows. For continuous-time systems, the discrete programming language is extended with

- continuous (explicit) assignments: $f(s_1, \ldots, s_k) :\approx t$,

which stay in force until a new assignment is made to the same *term* by some program.

Jumps. Jumps are effected by conditionals. Additional constraints on algorithmic evolution make sense in the continuous context. These include that tests should test for conditions that have non-zero duration and that the dynamics of a system change only finitely often in a finite period of time.

Flows and Jumps. A *jump* in the evolution of a continuous-time state is a change in its dynamics, in contrast with *flows*, during which the dynamics are fixed. See [5].

Conflicts. Programs as described above can cause conflicts ("clashes") when different (discrete or continuous) assignments (in one or more state programs) attempt to assign different values (at one and the same moment) to a single location. The outcome of such a conflict is any one of the possibilities. (These nondeterministic semantics are preferable to a system crash.)

Continuity. Continuous assignments may involve infinitesimal time, dt, provided the outcome is independent of the choice for dt. This is a continuity requirement of sorts. One can conceive of implicit specifications of continuous behavior, as well.

5 State Policies

Clocks. To achieve synchronous behavior in a continuous-time environment, there would need to be a global clock to which other states are connected, directly or indirectly.

Archives. When foreign locations provide only read-only resources, write abilities to a public (but not shared) memory need to be achieved via requests—as in modern hardware. A state p can allocate resources for requests r, addresses a, and values v, which it makes available to a memory module. The latter runs a program of the sort **if** $p.r$ **then** $m(p.a) := p.v$, for some "storage" function m. A similar setup may be used to serve stored values.

Queues. When unboundedly many states use the same controlled archive, some queueing mechanism needs to be set up, by means of which individual states can place requests while the archive deals with them one at a time.

Data. If (automata) states share a finite domain (as in cellular models [1]), then unbounded memory is achievable by means an unbounded number of connected states, in which case an unbounded number of steps may be needed to access a particular datum.

Interfaces. To model a physical or biological system in which units are each governed by rules, but adjacent units exchange values or signals, one could represent their interface as a channel. For example, the temperature of a wall would be a public function over $\mathbb{R}^2$ of one side or the other.

Effectiveness. In general, for a system to be deemed *effective*, not only should its transitions and evolutions be describable by a finite text, but also the initial states with the operations they are endowed with. For a bloc to be effective, it should have finitely many states, each governed by an effective algorithm [4]. The number of states and their inter-connections may grow unboundedly during its evolution.

Positions. This model does not directly model positions in space (of physical or biological systems). Each state might keep track of its own position; neighboring states would need to be in contact to avoid overlap.

Delays. There could be a time delay between a request for a value from a serving state and its receipt by the client. This would hold up execution of that part of the client process that awaits the requested value.

6 Conclusion

We believe that most of the usual and unusual models of computation are instances of this paradigm.

References

1 Pablo Arrighi and Gilles Dowek, July 2012, "Causal graph dynamics", *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP 2012)*, Warwick, UK, Lecture Notes in Computer Science, vol. 7392, Part II, pp. 54–66. Available at `http://arxiv.org/pdf/1202.1098v3` (viewed July 10, 2013).

2 Andreas Blass and Yuri Gurevich, 2006, "Ordinary interactive small-step algorithms, Part I". *ACM Transactions on Computational Logic*, 7(2), pp. 363–419. Available at `http://tocl.acm.org/accepted/blass04.ps` (viewed July 10, 2013).

3 Andreas Blass and Yuri Gurevich, June 2008, "Abstract state machines capture parallel algorithms: Correction and extension", *ACM Transactions on Computation Logic*, 9(3), Article 19. Available at `http://research.microsoft.com/en-us/um/people/gurevich/Opera/157-2.pdf` (viewed July 10, 2013).

4 Udi Boker and Nachum Dershowitz, August 2010, "Three paths to effectiveness", in Andreas Blass, Nachum Dershowitz, and Wolfgang Reisig, editors, *Fields of Logic and Computation: Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*, volume 6300 of *Lecture Notes in Computer Science*, pp. 36–47, Springer, Berlin. Available at `http://nachum.org/papers/ThreePathsToEffectiveness.pdf` (viewed July 10, 2013).

5 Olivier Bournez, Nachum Dershowitz, and Evgenia Falkovich, May 2012, "Towards an axiomatization of simple analog algorithms", in Manindra Agrawal, S. Barry Cooper, and Angsheng Li, editors, *Proceedings of the 9th Annual Conference on Theory and Applications of Models of Computation (TAMC 2012, Beijing, China)*, volume 7287 of *Lecture Notes in Computer Science*, pp. 525–536. Springer, Berlin. Available at `http://nachum.org/papers/SimpleAnalog.pdf` (viewed July 11, 2012).

6 Yuri Gurevich, 1995, "Evolving algebras 1993: Lipari guide", in Egon Börger, editor, *Specification and Validation Methods*, pp. 9–36. Oxford University Press. Available at `http://research.microsoft.com/~gurevich/opera/103.pdf` (viewed July 10, 2012).

Three lightings of logic

Jean-Yves Girard

CNRS, Institut de Mathématiques de Luminy
UMR 6206, 163 Avenue de Luminy, Case 907, 13288 Marseille Cedex 09, France
girard@iml.univ-mrs.fr

Abstract

Whether we deal with foundations or computation, logic relates questions and answers, typically formulas and proofs: a very entangled relation due to the abuse of *presuppositions*.

In order to analyse syntax, we should step out from language, which is quite impossible. However, it is enough to step out from *meaning*: this is why our first lighting of logic is that of *answers*: it is possible to deal with them as meaningless artifacts assuming two basic states, *implicit* and *explicit*. The process of *explicitation* (a.k.a. normalisation, execution), which aims at making explicit what is only implicit, is fundamentally hazardous.

The second light is that of *questions* whose choice involves a formatting ensuring the convergence of explicitation, i.e., the existence of "normal forms". This formatting can be seen as the emergence of *meaning*. It is indeed a necessary nuisance; either too laxist or too coercitive, there is no just format. Logic should avoid the pitfall of Prussian, axiomatic, formats by trying to understand which *deontic* dialogue is hidden behind logical restrictions.

The third lighting, *certainty* deals with the adequation between answers and questions: how do we know that an answer actually matches a question? *Apodictic* certainty — beyond a reasonable doubt — is out of reach: we can only hope for *epidictic*, i.e., limited, reasonable, certainty. Under the second light (questions), we see that the format is made of two opposite parts, namely *rights* and *duties*, and that logical deduction relies on a strict balance between these two opposite terms, expressed by the *identity group* "A is A and conversely". The issue of certainty thus becomes the interrogation: "Can we afford the rights of our duties?"

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Proof theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.11

Category Invited Talk

1 First light: what is an answer?

1.1 Implicit vs. explicit

A simple-minded approach to answers would reduce them to something completely explicit, e.g., `yes` or ||| (the number 3 in Cro-Magnon numeration). However, *implicit* answers, those given by programs or proofs, are more interesting, since *portable*. Indeed, the two sorts of answers, implicit and explicit are linked by *explicitation*: the execution of a program (cut-elimination, normalisation) reduces the implicit to the explicit. To sum up, an implicit answer is a program before execution.

Explicit answers form the solid ground for logic, the ultimate reality, which is made possible by the fact that they convey *strictly no meaning*. But how do we reckon that something is explicit? Is explicit what belongs in the realm of *constatation*, i.e., what is *analytic*. On a traditional typing machine, all keys are constative: they can but add

new text, typically the "↲ " key which opens a new line. On a computer, keys can also assume a *performative* function: "↲ " launches programs. The two aspects, constative and performative, are mingled to the point that one easily launches a program by accident. In logic, the constative and performative aspects of implication were mingled in XIX$^{\text{th}}$ century syntax (Axiom + Rules): *Modus Ponens*. The XX$^{\text{th}}$ century reading (sequent calculus) distinguishes carefully between *implication* $\Rightarrow$ which is handled by the constative left introduction and *entailment* $\vdash$ which is handled by the performative cut rule.

The distinction between implicit and explicit is purely subjective: we *decide* that an object is *finished*, i.e., explicit enough for our taste. A cheque is the typical implicit answer: we must cash it, then spend the money, both operations being hazardous. But we can decide — say, it is a cheque of Paul Erdös — to pin it above the desk. In the same way, a program need not be executed: it can be frozen, or opened with a developer. In logic, a cut on A can be replaced with an left introduction of $A \Rightarrow A \vdash$ (or $\vdash A \otimes {\sim}A$)[1]. This shows that there is no real distinction between, say, programs and data: they all belong to the same analytic space in which explicitation takes place; indeed, the program of explicitation itself must be part of the space.

Although negated by totalitarian ideologies, starting with Bentham's *panoptic* prototype of *Big Brother*, the distinction between implicit and explicit is basic and incompressible. The first evidence is to be found in *incompleteness*: there are questions without answers, typically the Gödel sentence. This evidence is however bridled by the iron discipline of formal systems; we should concentrate on all means of producing explicit answers, including those proscribed by logic. In this lax context, Turing's *undecidability* yields a partial recursive function that cannot be extended into a total one, thus forbidding us to foretell the convergence of execution, i.e., explicitation. By the way, computational complexity deals with a less brutal approach to the distinction between implicit and explicit: some answers are more implicit (harder to compute) than others.

Almost anything can serve as analytic space, for instance the binary integers used in machine code. However, in view of the necessary relation to be made with questions, some choices are more interesting than others. In particular, explicitation should be as natural as possible, so that implicit answers look as much as possible as as *their own execution* — and not as data to which an external program is applied.

A good candidate for an analytic space remains pure λ-calculus; among its good properties, Church-Rosser which states that the implicit contents, if any, is unique. The rewriting style (basically one equation), although external, remains very natural. The limitations are those of the functional paradigm with no direct access to other types of data, e.g., pairs. Also, the treatment of bound variables (α-conversion, substitution) is particularily *ad hoc*. λ-calculus is indeed already too formated: the only abnormality is that of a never-ending normalisation. The absence of deadlocks in pure λ-calculus is both a measure of its intrinsic qualities and of its limitations as an analytic space: deadlocks do exist!

Experience, that of linear logic and parallel computation, compels us to find a more primitive notion, free from functionality, but still deterministic. The various versions of *Geometry of Interaction* eventually stabilised into an analytic space based upon Herbrand's technique of *unification*, which is more primitive, less *ad hoc*, than rewriting: execution can be seen as a sort of physical plugging. This was, by the way, the strongest point in the late *Logic Programming*.

---

[1] This remark can, surprinsingly, be traced back to Lewis Carroll who made a mess of it.

1.2 Stars and galaxies

1.2.1 Unification

Consider a term language with infinitely many functional symbols of each arity. An equation $t = u$ between terms can be solved by means of *substitutions*: t, u are *unifiable* when $t\theta = u\theta$ for some *unifier* θ. The point is that substitutions do compose, hence:

▶ Theorem 1 (Herbrand, 1930). If t, u are unifiable, there is a mother θ_0 of all unifiers for t, u: any unifier θ for t, u can be uniquely written $\theta_0\theta'$.

1.2.2 Flows

A *flow* is an expression $t \leftharpoonup t'$ where t, t' are terms with quite the same variables. These common variables are internal to the flow, in other terms *bound*. In particular, when combining two flows, one must always rename the variables so as to make them distinct. Composition between $t \leftharpoonup t'$ and $u \leftharpoonup u'$ is obtained by *matching* t' and u: matching is the particular case of unification where the terms have no variable in common, what is the case when the variables of t', u have been made distinct. If θ is the principal unifier, we define composition by $(t \leftharpoonup t')(u \leftharpoonup u') := t\theta \leftharpoonup u'\theta$. Composition is thus a partial operation; if we formally add an empty flow 0 to take care of a possible failure of the matching: $(t \leftharpoonup t')(u \leftharpoonup u') := 0$, composition becomes associative, with neutral $I := x \leftharpoonup x$.

If $\mathcal{T}$ is the set of closed terms, then any functional term t induces a subset $[t] \subset \mathcal{T}$, namely the set of all closed t_0 which unify with t; t, t' are *disjoint* when $[t] \cap [t'] = \emptyset$. Any flow $t \leftharpoonup t'$ induces a partial bijection $[t \leftharpoonup t']$ between the subsets $[t']$ and $[t]$ of $\mathcal{T}$. Let us fix a copnstant c; if t_0 is closed, then $[t \leftharpoonup t']t_0$ is defined when $(t \leftharpoonup t')(t_0 \leftharpoonup c) \neq 0$, in case it writes $[t \leftharpoonup t']t_0 \leftharpoonup c$. The condition "quite the same variables" ensures that $[t \leftharpoonup t']t_0$ is closed and that $[t \leftharpoonup t']$ is injective. Any flow $u \leftharpoonup u$ is idempotent; its associated function is the identity of the subset $[u] \subset \mathcal{T}$.

1.2.3 The convolution algebra

One can introduce the *convolution algebra* of the monoid, i.e., the set of finite formal sums $\sum \lambda_i \phi_i$ where the ϕ_i are flows and the λ_i are complex coefficients, the improper flow 0 being identified with the empty sum. This algebra acts on the Hilbert space $\ell^2(\mathcal{T})$ by means of $(t \leftharpoonup t')(\sum_i \lambda_i t_i) := \sum_i \lambda_i [t \leftharpoonup t']t_i$. The involution $(\sum \lambda_i (t_i \leftharpoonup t_i'))^* := \sum \bar{\lambda}_i (t_i' \leftharpoonup t_i)$ is implemented by the usual adjunction. The idempotents $t \leftharpoonup t$ correspond to the projections on the subspaces $\ell^2([t])$ and $t \leftharpoonup t'$ induces a partial isometry of *source* $\ell^2([t'])$ and *target* $\ell^2([t])$. The early versions of GoI did associate to proofs finite sums of flows. These sums were *partial isometries*; $u = \sum t_i \leftharpoonup t_i'$ is a partial isometry (i.e., $uu^*u = u$) if the targets t_i are pairwise *disjoint*, not unifiable, *idem* for the t_i'. The operators of GoI are indeed *partial symmetries* ($u = u^3 = u^*$): typically the identity axioms $(t \leftharpoonup t') + (t' \leftharpoonup t)$ (t, t' disjoint).

The unification algebra internalises the major algebraic constructions.

Matrixes

If I is a finite set of closed terms, the $I \times I$ matrix (λ_{ij}) can be naturally represented by $\sum_{ij} \lambda_{ij} (i \leftharpoonup j)$.

Direct sums

The flows $P := p(x) \leftharpoonup x, Q := q(x) \leftharpoonup x$ induce an isometric embedding of $\ell^2(\mathcal{T}) \oplus \ell^2(\mathcal{T})$ in $\ell^2(\mathcal{T})$: $x \oplus y \mapsto [P]x + [Q]y$. The isometricity comes from $P^*P = Q^*Q = I$, $P^*Q = Q^*P = 0$. The embedding is not surjective: this would require $PP^* + QQ^* = I$, in other terms that every term matches either $p(x)$ or $q(x)$.

P and Q have been heavily used in the early GoI, in particular for multiplicatives — and, modulo tensorisation with I, for contraction. They enable one to change the size of matrices in a flexible way. Usually, the only possibility is to *divide* the size, typically $\mathcal{M}_{mn}(\mathbb{C}) \simeq \mathcal{M}_m(\mathcal{M}_n(\mathbb{C}))$ replaces a $mn \times mn$ matrix with a $m \times m$ matrix whose entries are $n \times n$ matrices, i.e., blocks of size $n \times n$. Thanks to P, Q, one can replace a 3×3 matrix with a 2×2 one (with four "blocks" of sizes $2 \times 2, 2 \times 1, 1 \times 2, 1 \times 1$).

Tensor products

The tensor product of two flows makes use of a binary function "$\cdot$" and is defined by $(t \leftharpoonup t') \otimes (u \leftharpoonup u') := t \cdot u \leftharpoonup t' \cdot u'$; the variables of the two flows must first be made distinct. This corresponds to an internalisation of the tensor product, which plays an essential role in the handling of exponentials, i.e., of repetition. The flow $T := (x \cdot y) \cdot z \leftharpoonup x \cdot (y \cdot z)$ compensates the want of associativity of the internal tensor: $T^*(((t \leftharpoonup t') \otimes (u \leftharpoonup u')) \otimes (v \leftharpoonup v'))T = (t \leftharpoonup t') \otimes ((u \leftharpoonup u') \otimes (v \leftharpoonup v'))$.

Crown products

In the same style as T, the flow
$\sigma := x_1 \cdot (x_2 \cdot (\dots (x_{n-1} \cdot x_n) \dots)) \leftharpoonup x_{\sigma(1)} \cdot (x_{\sigma(2)} \cdot (\dots (x_{\sigma(n-1)} \cdot x_{\sigma(n)}) \dots))$ induces a permutation of the constituents af a n-ary tensor.

1.3 Stars and galaxies

1.3.1 Stars

A *star* $[\![t_1, \dots, t_{n+1}]\!]$ consists in $n+1$ terms; these terms, the *rays* of the star, must be pairwise *disjoint*, i.e., not matchable, which is strictly stronger than *not unifiable*.

Stars generalise the unification algebra; thus, the axiom link $(t \leftharpoonup t') + (t' \leftharpoonup t)$ becomes $[\![t, t']\!]$. However, since our objects are no longer operators, there are some difficulties in defining the analogue of composition. For this we shall use *coloured stars*. We select pairs of complementary colours, e.g., (`green`, `magenta`) together with the *neutral* colour `black`; a *coloured star* is a star in which each ray has been given a colour: typically, $[\![t, u, v, w]\!]$. Disjointness is required only for rays of the same colour, which comes from the fact that coloured stars are not yet another notion, just a shorthand: indeed, consider three unary functions g, m, b and replace $[\![t, u, v, w]\!]$ with $[\![g(t), g(u), b(v), m(w)]\!]$. t is thus *a priori* disjoint from u.

1.3.2 Galaxies

A *galaxy* is a finite set of coloured stars. Cut-free proofs will be represented by black galaxies, whereas the cut-rule will make use of complementary colours. The implicit thus lies in the use of colours, this explains why it is relative and contextual: by making everything black, a galaxy becomes explicit at no cost. Colours thus indicate that we consider the data as unfinished, thus initiating a *normalisation* process.

In order to normalise a galaxy, we first form its *diagrams*. By this I mean any tree (in the topological acception) obtained by attaching $N+1$ stars of the galaxy by means of N *vertices*. By a *vertex*, I mean a pair $t = u$ of rays of complementary colours. Since the same star may be used several times in a diagram, a galaxy is likely to generate infinitely many diagrams.

The *unification* of a diagram consists in unifying its vertices, so that $t\theta = u\theta$ becomes an actual equality. Most unifications will fail; we are basically concerned with *correct* diagrams, those for which unification succeeds.

1.3.3 Normalisation

In usual GoI, the cut-rule is handled by a partial symmetry σ; the normal form of the proof (u, σ) is given by:

$$(I - \sigma^2)u(I - \sigma u)^{-1}(I - \sigma^2)$$

Here σ corresponds to the swapping of complementary colours: σ exchanges `green` and `magenta` and "kills" `black`. Under reasonable hypotheses (nilpotency), $u(I - \sigma u)^{-1}$ can be written as a finite sum $u + u\sigma u + u\sigma u\sigma u + \ldots$, which corresponds to the plugging of u with itself through complementary colours. The two $I - \sigma^2$ correspond to the restriction to the "black stars".

Strong normalisation) generalise the nilpotency of σu:

1. There are only finitely many correct diagrams. In other terms, for an appropriate N, all diagrams of size $N+1$ fail; this finite N accounts for *strong* normalisation.
2. No correct diagram is *closed*, i.e., without a free ray. The condition thus excludes the closed diagram $\{[\![\, t \,]\!], [\![\, t \,]\!]\}$ (vertex $t = t$).
3. In a correct diagram, identify complementary colours, e.g., replace `magenta` with `green`; then the free rays are disjoint. The simplest diagram thus excluded consists of a single binary star: $\{[\![\, t, u \,]\!]\}$, with t, u not disjoint.

The normal form is obtained by collecting the correct diagrams whose free rays are black. And to replace them with their *residual* star, i.e., the star whose rays are their free rays.

A galaxy $\mathcal{G}$ is *isometric* when rays of the same colour occurring in $\mathcal{G}$ are pairwise disjoint. The normal form of an isometric galaxy is easily shown to be isometric.

1.3.4 Church-Rosser

In the presence of two pairs of complementary colours, there are three possible ways of normalising:

1. Identify `green` = `blue`, `magenta` = `yellow` and normalise.
2. Normalise the cuts `blue`/`yellow`, then the residual cuts `green`/`magenta`.
3. Normalise the cuts `green`/`magenta`, then the residual cuts `blue`/`yellow`.

The Church-Rosser property equates (in any possible sense) these three possibilities. This property will later be used to show the *compositionality* of cut, hence to develop various functional, i.e., category-theoretic, interpretations. Hence one pair of colours is enough, at least for theoretical considerations.

2 Second light: what is a question?

2.1 Formatted vs. informal

An implicit answer, a program, may have no explicit contents: normalisation may diverge. Fixing that point amounts at *formatting*; the emergence of meaning wholly lies in this formatting. A synonym for meaning is *question*: the meaning of the answer is the question it is supposed to solve. Now, there is a great divide between the formatted, typed, logical approach and the unformal, untyped, "free" approach.

The lesson of incompleteness is that the format is a *necessary nuisance*, think of Family, Justice, Police, etc. Indeed, the informal approach to logic is inconsistant — if we prefer, the untyped approach to computation does not normalise: this account for the "necessary". On the other hand, a typing discipline always misses something. This remark is already present in Richard's Paradox (1905): "The smallest integer not definable in less than twenty words". The informal acception of "definable" makes it inconsistent, while a formated version — say DEFINABLE — avoids the pitfall while producing a definition out of the scope of "DEFINABLE".

The same totalitarian ideologies that claim that everything is explicit, transparent, would consistently vouch for informality: witness the various *qualunquists* (libertarians, populists, etc.) which pretend to approach politics without politicians, taxes, laws. When in charge, these people turn out to be worse than the politicians they were opposing to. This is due the fact that one cannot escape formatting: and then, better an explicit than a hidden one!

The real question is thus not that of the necessity of a format, but that of its nature, its emergence. XIX$^{\text{th}}$ logic solved the problem by means of *axiomatics*, i.e., principles that one cannot discuss. There must be something of the like, but we should at least understand what we accept: axiomatics is too Prussian to be honest[2]. In logic, the format is usually invisible; besides the choice of a language to avoid inconsistencies, it also occurs in the *form* preserved by category-theoretic *morphisms* or in the *rule* of game-theoretic semantics. Can we discuss these choices, or better: is this discussion part of logic?

The situation of an opaque *deontic*, normative[3], kernel did not change till the invention of *linear logic* in the mid eighties. Indeed, the existing formats, especially *natural deduction*, were satisfactory enough to make us forget their axiomatic, Prussian, character. Linear logic, with the introduction of classical features — basically an involutive negation — within the constructive universe, posed a novel question, namely the handling of several simultaneous conclusions, a problem hitherto avoided by the tree-like format which pinpoints both the conclusion and the last rule applied. In *proof-nets*, the last rule is implicit to the point that it is not even uniquely defined. What makes a proof-net correct, i.e., what compels it to have a last rule and, this recursively, is a purely deontic question.

The question was not quite novel, since Herbrand's theorem solved it in the limited context of quantification. In a prenex form, the existentials should be given as functions $y_i = t[x_1, \ldots, x_n]$ of the universals. Assuming we forgot the step-by-step construction of t, Herbrand replaces x with $f(y)$ in the case of a formula $\exists y \forall x$; if x actually occurs in t, then we get a cycle (failed unification) $y = t[f(y)]$.

The sort of dialogue at work in Herbrand's theorem — more generally in proof-nets — is not basically designed to tell truth from falsity, but what is permitted from what is illegal. This dialogue is *deontic* (instead of *alethic*): it deals with permissions, obligations, and not with

[2] In modern Greek, *axiomatikos* means "officer"!
[3] This adjective may convey a derogatory approach to the format; "deontic" is more neutral.

truth. A typical deontic dialogue is "Objection your Honor! Objection sustained/overruled". The dialogue has nothing to do with the truth/falsity of the statement under discussion: it concerns its relevance to the case. One perfectly understands that not every question should be taken into consideration; but also that this necessary deontic dialogue may be a way to sweep things under the carpet.

Popper's notion of *falsifiablity* is a limited form of deontic dialogue accounting for purely universal, Π^0_1, formulas of arithmetic, e.g., $\forall x\ (x+1)^2 = x^2 + 2x + 1$. Falsifiability does not hold beyond Π^0_1 complexity, for the simple reason that falsifiability is itself Π^0_1: "for all tests…". Beyond the Π^0_1 case, the deontic dialogues becomes completely symmetric: if an objection is overruled, something goes wrong, but we cannot foretell which side "is right": when the judge says "sustained", he may be dismissed!

2.2 Vehicles and gabarits

We restrict our presentation to the familiar multiplicative case of linear logic.

2.2.1 Proof-nets

We should get rid of syntactical decorations so as to describe multiplicative proof-nets in a purely *locative* way: in order to represent a proof of $\vdash A, B, C$ unary functions p_A, p_B, p_C will be used to distinguish between the various *locations* available in the sequent; I could as well use p_1, p_2, p_3, but this would compel me into a systematic reindexing.

2.2.2 Vehicles: cut-free case

Let us choose, once for all, distinct constants $\mathtt{l}, \mathtt{r}$ and a binary function letter "$\cdot$". To each proof π we associate its *vehicle*, i.e., a galaxy $\pi^\bullet$; this galaxy is black in the cut-free case.

Identity axiom: if π is the axiom $\vdash A, {\sim}A$, then $\pi^\bullet := \{[\![\, p_A(x), p_{\sim A}(x) \,]\!]\}$.

$\parr$-rule: if the proof π of $\vdash \Gamma, A \parr B$ has been obtained from a proof ν of $\vdash \Gamma, A, B$, then $\pi^\bullet := \nu^\bullet$ in which p_A and p_B are now *defined* by
$p_A(x) := p_{A\otimes B}(\mathtt{l}\cdot x)$, $p_B(x) := p_{A\otimes B}(\mathtt{r}\cdot x)$.

$\otimes$-rule: if the proof π of $\vdash \Gamma, A \otimes B$ has been obtained from proofs ν of $\vdash \Gamma, A$ and μ of $\vdash B, \Delta$, then $\pi^\bullet := \nu^\bullet \cup \mu^\bullet$, with p_A, p_B *defined* by
$p_A(x) := p_{A\parr B}(\mathtt{l}\cdot x)$, $p_B(x) := p_{A\parr B}(\mathtt{r}\cdot x)$.

The vehicle is thus a galaxy of axiom-links, seen as stars. The rules $\parr, \otimes$ have been used to relocate these links. For instance, the axiom $[\![\, p_A(x), p_{\sim A}(x) \,]\!]$ may relocate as $[\![\, p_{A\parr(\sim A\otimes B)}(\mathtt{l}\cdot x), p_{A\parr(\sim A\otimes B)}(\mathtt{r}\cdot(\mathtt{l}\cdot x)) \,]\!]$.

2.2.3 Vehicles: general case

In presence of cuts, coloured functions will be needed. We shall use a pair of complementary colours, typically $p_B, p_{\sim B}$ and $p_B, p_{\sim B}$. The interpretation $\pi^\bullet$ now looks as a union $\mathcal{V} \cup \mathcal{C}$: $\mathcal{V}$ (in black and `green`) is the vehicle proper, $\mathcal{C}$ — its *feedback* — is easily identified as the `magenta` part of the vehicle.

Cut rule: if the proof π of $\vdash \Gamma$ has been obtained from proofs ν of $\vdash \Gamma, A$ and μ of $\vdash {\sim}A, \Delta$, then $\pi^\bullet := \nu^\bullet \cup \mu^\bullet \cup \{[\![\, p_A(x), p_{\sim A}(x) \,]\!]\}$; furthermore, in $\nu^\bullet \cup \mu^\bullet$, $p_A, p_{\sim A}$ have been painted `green`: $p_A(t) \mapsto p_A(t)$, $p_{\sim A}(t) \mapsto p_{\sim A}(t)$.

2.2.4 Gabarits (I)

We must now make sense of the lower part of the proof-net, the one dealing with the $⅋, \otimes$ and Cut links. The main problem is to give a precise definition of the switching discipline leading to the correctness condition. Indeed, to each switch, we shall associate an *ordeal*, i.e., a coloured galaxy. This finite set of ordeals is called the *gabarit*.

We already defined the unary functions $p_A(x)$ for each formula and subformula of the proof-net. We now introduce $q_A(x) := p_A(\mathtt{g} \cdot x)$, where $\mathtt{g}$ is yet another constant. The replacement of p_A with q_A in the context of gabarits is due to the fact that $p_{A\otimes B}(x)$ is not disjoint from $p_A(x) := p_{A\otimes B}(\mathtt{l} \cdot x)$, whereas $q_{A\otimes B}(x)$ and $q_A(x)$ are disjoint: the q_A provide disjoint locations for the formulas occurring in the lower part of the proof-net.

Given a proof-net of conclusions Γ, a switch L/R of its $⅋$-links induces an *ordeal*, namely the coloured galaxy made of the following stars:
$\boldsymbol{X, \sim X}$: $[\![\, p_A(x), q_A(x) \,]\!]$ when A is a *literal* $X, Y, \sim X, \sim Y, \ldots$.
$\otimes$: $[\![\, q_{A\otimes B}(x), q_A(x), q_B(x) \,]\!]$.
$⅋_L$: $[\![\, q_{A⅋B}(x), q_A(x) \,]\!]$ and $[\![\, q_B(x) \,]\!]$. In terms of graphs, $[\![\, q_B(x) \,]\!]$ "terminates" all $[\![\, q_B(t) \,]\!]$.
$⅋_R$: $[\![\, q_{A⅋B}(x), q_B(x) \,]\!]$ and $[\![\, q_A(x) \,]\!]$ which "terminates" all $[\![\, q_A(t) \,]\!]$.
Cut: $[\![\, q_A(x), q_{\sim A}(x) \,]\!]$.
Conclusion: $[\![\, q_A(x), p_A(x) \,]\!]$ when $A \in \Gamma$, i.e., is a conclusion.
An ordeal thus normalises into a galaxy in `black` (conclusions) and `blue` (literals).

2.2.5 Correctness, a.k.a. completeness

Let $\mathcal{V}$ be $\mathcal{V}$ painted `yellow`. The *correctness criterion* thus writes as:

> For any ordeal $\mathcal{S}$, the galaxy $\mathcal{V} \cup \mathcal{S}$ strongly normalises into $\{[\![\, p_A(x) \;;\; A \in \Gamma \,]\!]\}$.

This condition is obviously necessary; its sufficiency is the most elaborate form of *completeness* that one can imagine, since it relates the symbolic testing by means of the ordeals with the proofs in a logical system.

The main technical problem with completeness is that usual proof-nets are, so to speak, "preconstrained": the identity links relate complementary formulas $A, \sim A$, whereas nothing of the kind has been so far required. In other terms, our treatment of literals is completely indistinct: $X, \sim X, Y$ are the same, up to their locations. How can we force an axiom link to relate X with a $\sim X$ (and not a Y, nay another X)?

Here, we must remember that predicate or propositional calculi are convenient structures, but that part of them belongs in the worst kind of *a priori*. Typically, the so-called propositional "constants" X, Y and their negations: we are embarrassed since they mean nothing by themselves. The real logic is a second order system — a sort of system $\mathbf{F}$ — in which there is no propositional constants, but in which formulas are *closed*. What we call first order logic indeed corresponds to those formulas $\forall X_1 \ldots \forall X_n A$, with A quantifier-free: the behaviour of such formulas is extremely simple, especially in view of completeness issues, e.g., the subformula property. The restriction to those formulas renders the universal prefix compulsory —hence the possibility to omit it. To make the long story short, when dealing with a proof-net, we must take into account the *implicit* second order quantification $\forall X$ on all propositional "constants". What follows is a glimpse of the future treatment of second order logic; indeed the easy case of the quantifier $\forall X$.

Every propositional "constant" must be switched; each switch has three positions, so that n propositional constants induce 3^n possibilities. The switching corresponds to the choice $\boldsymbol{\Theta}$ of a substitution $X_i \rightsquigarrow \mathbf{c}_i$ for each of the "constants" X_i, the $\mathbf{c}_i$ ranging over the

three possibilities $\mathbf{a}, \mathbf{a} \otimes \mathbf{b}, \mathbf{a} \parr \mathbf{b}$, where $\mathbf{a}, \mathbf{b}$ are propositional letters. Now, to switch our net consists in:

1. First switch the constants, thus yielding a substitution $\mathbf{\Theta}$.
2. Then switch $\mathbf{\Theta}(\Gamma)$ as explained above.

This should be enough to ensure that literals are linked according to the book. As to general axiom links (not between literals) an argument based upon ηxpansion should exclude "illegal" links.

2.2.6 Gabarits (II): virtual switches

Let us turn our attention towards an exotic multiplicative, namely the "linear affine" implication $A \multimap B$. "$\multimap$" yields a purely multiplicative second-order reduction of additives: $A \oplus B := \forall X((A \multimap X) \multimap ((B \multimap X) \multimap X))$[4].

Indeed, $A \multimap B$ is an intuitionistic implication without reuse of premises; this is why it interests us. The associated disjunction $A \ltimes B := {\sim}A \multimap B$ is problematic in terms of gabarits. Indeed, the $\ltimes$-link:

$$\frac{[A] \quad B}{A \ltimes B}$$

is problematic: the premise A (written $[A]$ for this reason[5]) might be absent, hence the switch "L" is hazardous: it may destroy everything in case of absence. On the other hand, we cannot content ourselves with the sole "R", hence the idea of a *virtual switch*, i.e., a sort of compensation for the missing switch.

Virtual switches are inspired from the proof by Mogbil and de Naurois of the NL complexity of multiplicative proof-nets; improving the idea of *contractibility* introduced by Danos, the authors show that it is enough to switch $\parr$ on one side, e.g., always "R"; an additional order condition (3 below) compensates for the missing switches. The point is that this alternative approach can be used in case we cannot switch the $\parr$ on "L", typically if the actual presence of the premise A is dubious. This is the case with the marginal connective $\ltimes$, a multiplicative which *actually* needs virtual switches.

A *virtual switch* is a star $[\![t; u_1, \dots, u_n]\!]$, with a distinguished ray, its *root* t. $u_1, \dots, u_n$ must be pairwise disjoint; each variable occurring in t must still occur in the u_i.

The notion of ordeal is modified as follows, so as to include an auxiliary galaxy of virtual switches. Typically, in the case of $A \ltimes B$, besides $[\![q_{A \ltimes B}(x), q_B(x)]\!]$ and $[\![q_A(x)]\!]$, we add the auxiliary stars $[\![q_{A \ltimes B}(x); q_A(x)]\!]$.

Consider the unique correct diagram in $\mathcal{V} \cup \mathcal{S}$, and let us unify it, so as to get a galaxy $\mathcal{G}$. For each virtual switch $[\![t; u_1, \dots, u_n]\!]$, consider all rays obtained by unification from some u_i; since $u_i\theta = u_i\theta'$ implies $t\theta = t\theta'$, each such ray "comes from" a specific instanciation of t, its "root". We require that:

1. If $u_i\theta \in \mathcal{G}$, then its root $t\theta$ occurs in $\mathcal{G}$.
2. $u_i\theta$ is "upwards connected" to $t\theta$, i.e., the connection does not transit through the vertex $t_1\theta_1 = t\theta$.

[4] Instead of $\forall X((A \multimap X) \Rightarrow ((B \multimap X) \Rightarrow X))$.
[5] The graphism is reminiscent of the discharged hypotheses of natural deduction.

For each $t\theta$ in $\mathcal{G}$, we can consider the set $\mathcal{G}_{t\theta}$ of all rays standing in between $t\theta$ and some $u_i\theta'$ with root $t\theta$ (i.e., s.t. $t\theta = t\theta'$) including extremities; $\mathcal{G}_{t\theta}$ is this a sort of tree, rooted in $t\theta$. We define $t_1\theta_1 \preceq_1 t_2\theta_2$ by $t_1\theta_1 \in \mathcal{G}_{t_2\theta_2}$. If $\preceq$ is the reflexive and transitive closure of $\preceq_1$, we require that:

3. $\preceq$ is an order relation.

These conditions (especially 3) are clearly CO-NL, hence their complexity-theoretic import. To understand how virtual switches work, let us assume that the ordeal $\mathcal{S} \in \mathcal{G}$ switches the $⅋$ link with conclusion $A ⅋ B$ on "R" and that its virtual part contains $[\![q_{A⅋B}(x), q_A(x)]\!]$; we can get rid of this virtual switch by adding to $\mathcal{G}$ the ordeal $\mathcal{S}'$, namely the twin of $\mathcal{S}$ with the same $⅋$ switched on "L": conditions 1 – 3 precisely allow for this replacement. We can thus eliminate the virtual switch $[\![q_{A⅋B}(x), q_A(x)]\!]$ from $\mathcal{G}$ at the price of a duplication of the number of its ordeals.

Virtual switches are well-adapted to weakening, since they cope with the possible uncertainty as to the presence of a specific premise. Moreover, since u_i may contain variables not in t, there is no limitation as to the number of $u_i\theta'$ rooted in a given $t\theta$: the extra variables thus account for contraction. The treatment of exponentials and additives makes a heavy use of virtual switches.

3 Third light: what conveys certainty?

3.1 Epidictic vs. apodictic

The main difference between XIX$^{\text{th}}$ century, pre-Gödelian, and XX$^{\text{th}}$ century logics is perhaps the issue of *certainty*. Before incompleteness, a proof was supposed to be valid beyond any doubt; hence the adjective *apodictic*, which corresponds to this absence of doubt, but whose etymology is simply "proven". Incompleteness opens the possibility of a reasonable doubt, hence to a change of status for proofs: they are no longer apodictic, they can only be *epidictic*, i.e., they only guarantee a reasonable form of certainty. Common sense can explain this failure: deduction is a rational form of prediction, but prediction cannot be 100% rational. Just like rating agencies were unable to prevent the subprime crisis, there is no absolute certainty as to cheques, before cashing. The only absolutely reliable bank is completely explicit: it directly delivers the goods you are looking for, the cow and the butter: but then, forget money! In the same way, the only absolutely reliable formal system would be purely analytic, limited to down to earth constatations of the form $2 + 2 = 4$.

How come that our certainty is no longer that certain? We must remember that it never occurred to XIX$^{\text{th}}$ century logicians, e.g., Russell, Hilbert, that the logical format could "miss" some "truth", unless the definition was intentionally ambiguous. For instance, Euclide's Postulate left open the question of parallels, but this was made explicit by alternative models, the sphere or the one-sheet hyperboloid; this question being fixed, nothing else was "missing". In the case of incompleteness, nothing specific is actually missing in the sense that it would suffice to add it. But there is a definite shortage of counter models: nobody has ever seen the tail of a refutation of the Gödel sentence — the book says that such a refutation must exist — but this "evidence" follows from incompleteness, while it should establish it. This is why this "model" is styled *non standard*, i.e., good for nothing.

Back in the 1920s, the only possibility was that of proving too much, like in Burali-Forti's or Russell's antinomies. Hence the reduction of certainty to *consistency*: if a deductive system cannot prove A and $\neg A$, then it should be perfectly sound, i.e., conveys certainty. However, $\mathbf{PA} + \neg G$, Peano Arithmetic extended with the negation of the Gödel sentence is

equi-consistent with **PA**, although plainly wrong! An analogy: many criminals are found "not guilty" on the grounds of some legal trick, say a statute of limitations; but an acquittal based on a deontic use of Law can by no means restore confidence. In other terms, although the negation $\neg G$ avoids inconsistency, it is still far from plausible: consistency does not entail certainty.

We must however reckon that consistency is (a minor) *part of* certainty. Here the second incompleteness destroys the ultimate illusion of XIX$^{\text{th}}$ century logic: consistency itself cannot be established beyond a reasonable doubt.

Gödel's incompleteness is the final firework of XIX$^{\text{th}}$ logic. XX$^{\text{th}}$ logic begins with Gentzen's cut-elimination (the distinction implicit/explicit), Herbrand's theorem (the emergence of format) and the "functional" interpretation of proofs, a.k.a. BHK[6]. Typically, a proof of $\forall x\ A[x]$ is a function associating to each integer n a proof $f(n)$ of $A[n]$. The definition is interesting and problematic under the three lights:

Answers: f cannot be quite a function, since a function is an infinite object. It must thus be a finite artifact, a program yielding the output $f(n)$ when feeding it with n.

Questions: f must be of the right kind, i.e., associate to each n a proof of $A[n]$, whatever that means. Deontically speaking, this means that f must pass infinitely many tests: first choose n, then test whether $f(n)$ is a proof of $A[n]$. Something of the like occurs with Popper's *falsifiablity*.

Certainty: how do we know that the proof is actually a proof, in other terms, that it passes the deontic tests which are infinetely many? In the Π^0_1 case, this *proof that the proof is a proof* is indeed the proof irself: the function, something like $f(n) := \texttt{true}$ is known in advance, so the only thing at stake is to determine whether $A[n] = \texttt{true}$ for all n, i.e., $\forall x\ A[x]$.

BHK can thus be seen as an archaic prefiguration the most recent developments in terms of answers and questions: in that respect, it fully belongs in XX$^{\text{th}}$ century logic. It also poses the problem of certainty: and, to start with, how come, in XX$^{\text{th}}$ century terms, that we lost absolute certainty?

3.2 Derealism

3.2.1 Proof-nets and certainty

The correctness criterion for proof-nets yields a form of apodictic certainty: yes, we can be sure that a would-be proof is actually a proof. This is due to the combination of several facts:

Finiteness: correction relates a vehicle with a gabarit. This involves finitely many finite verifications, leaving no room for reasonable doubt.

Compositionality: the gabarit for A and the gabarit for $\sim A$ do match so as to ensure the identity group, especially cut-elimination.

The great divide of logic is between first and second order. Indeed, if we take a second order approach to logic (with quantifiers on predicates or propositions), the first order part is the one in which second order quantifiers occur as universal prefixes $\forall X_1 \dots \forall X_n$: the formula $X \Rightarrow X$ is thus a shorthand for $\forall X\,(X \Rightarrow X)$. Using the Dedekind translation of natural numbers, arithmetic becomes part of second order logic: indeed, Π^0_1 formulas involve second order existentials. First order is complete and apodictic, while second order proper — i.e., using $\exists X$ — is incomplete and can only be epidictic.

[6] Indeed Brouwer-Heyting-Kolmogorov.

Something puzzling is that the proof-net technology basically applies to full logic. The fact that we lose certainty must be ascribed to second order quantification, more precisely, to the the existential quantifier. The study of system **F** shows that this quantifier concentrates most of the logical complexity: its interpretation through *candidats de réductibilité* involves comprehension axioms, which cannot convey absolute certainty.

Indeed, in a second-order proof-net, we must indicate the existential witnesses T corresponding to the rules deducing $\exists X\, A[X]$ from $A[T]$. And, relative to these witnesses T (which carry their own gabarits), we can get absolute certainty, at least on the grounds of *finiteness*. The issue of compositionality is, however, a cat of a different colour: indeed, X occurs several times in $A[X]$, in practice both positively and negatively. This means that we must provide gabarits for both T and $\sim T$. But how do we know that they actually match?

We already mentioned, concerning Popper, that his approach was too simplistic: like in the Gospel, the judges must be judged. This means that the matching between the normativity for T and the normativity for $\sim T$ is the most intricate thing one can imagine, surely something not of this world. The reasonable doubts and the reasonable certainties as to reasoning concentrate in this hazardous matching.

3.2.2 Épures

The deontic pair $T/\sim T$ corresponds to the *rights* and *duties* attached to T. The identity axiom $T \vdash T$, or, better, $\vdash \sim T, T$ is still valid when we relinquish our rights — and/or exaggerate our duties. But the cut rule enables to pass from $\vdash \Gamma, T$ and $\vdash \Delta, \sim T$, to $\vdash \Gamma, \Delta$ on the basis that we have the rights (T) of our duties ($\sim T$). By the way, replacing T with $\sim T$ will not alter the pattern, since the rights of $\sim T$ are the duties of T.

This schizophrenic approach to deduction first occurred in Schütte's *partial valuations*, in other terms, three-valued models. The fact that one can relinquish our rights is expressed by a third value, $\boldsymbol{i}$. Hence, in terms of rights, A is not false, while in terms of duties, A is true. The fact that "true" implies "not false" accounts for the identity axiom. But the cut rule requires the reverse implication, which is the case only in the usual, two-valued case. This semantics is, as far as I know, the unique legitimate occurence of an exotic truth value. Its technical interest is almost void: since $\boldsymbol{i} \Rightarrow \boldsymbol{i} = \boldsymbol{i}$, the third value has a propension to swallow the real ones... and what is the use of a model where almost everything takes the value $\boldsymbol{i}$? The only interest of the third value is that of a sort of "side wheels" helping us from mixing rights and duties.

A much better incarnation of the same idea is the category-theoretic notion of *dinaturality*: the entailment $A \vdash A$ between rights and duties becomes a morphism. And the failure of compositionality can be ascribed the want of commutativity of certain "hexagons". But this is still semantics, not yet the real thing.

The French "épure" means the representation of an object through three planar projections. I propose to use this term for the combination $\mathcal{V} + \mathcal{G}$ of a vehicle and a gabarit: indeed, both an object and several ways (the ordeals of $\mathcal{G}$) of structuring it. The inclusion of a gabarit as part of the épure renders quantification over gabarits possible: this should answer for the problematic aspects of second order logic. By the way, if a proof is an épure, the missing "auxiliary proof" of BHK is its gabarit.

3.2.3 The derealistic program

If we except first order quantification and equality, our understanding of first order logic is quite satisfactory. This is not the case with second order logic, especially under the light

of *certainty*. The new approach — épures — should improve the the existing systems and their interpretations: in particular, fix the limitations of the the usual realistic, semantic, approach which collapsed in front of natural numbers. By introducing deontic components — the gabarits at work in the épures —, we should able to find *derealist* integers explaining — say — why the Gödel sentence is not provable.

As to certainty, the final pattern should look like:

- A solid, i.e., non-deductive, analytic, rock in which reasoning takes place as a combination of épures.
- Depending upon the choice of the right gabarits, the access to a reasonable form of deduction.

Certainty can only be *epidictic*, i.e., rely upon a covenant between rights and duties: such a pact belongs in the realm of beliefs. But the day of true believers, of axiomatic certainty is over: the idea of an épure is to make everything, including the covenant, part of the logical artifact. Making suppositions part of the object is a way to get, once for all, rid of these presuppositions which so badly hinder logic.

References

1 J.-Y. Girard. **The Blind Spot: lectures on logic**. European Mathematical Society, Zürich, 2011. 550 pp.

Institut de Mathématiques de Luminy,
UMR 6206 – **CNRS**,
163, Avenue de Luminy, Case 907,
F-13288 Marseille Cedex 09
girard@iml.univ-mrs.fr

NON SI NON LA

From determinism, non-determinism and alternation to recursion schemes for P, NP and Pspace

Isabel Oitavem

CMAF, Universidade de Lisboa and
DM, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
2829-516 Caparica, Portugal
oitavem@fct.unl.pt

Abstract

Our goal is to approach the classes of computational complexity *P*, *NP*, and *Pspace* in a recursion-theoretic manner. Here we emphasize the connection between the structure of the recursion schemes and the underlying models of computation.

1998 ACM Subject Classification F.4.1 Mathematical Logic, F.1.1 Models of Computation, F.1.2 Modes of Computation, F.1.3 Complexity Measures and Classes

Keywords and phrases Computational complexity, Recursion schemes, P, NP, Pspace

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.24

Category Invited Talk

1 Introduction

P, *NP* and *Pspace* are well-known classes of computational complexity that can be described following different approaches. Here we describe them in a machine independent manner, using recursion schemes, which turn the known inclusions $P \subseteq NP \subseteq Pspace$ obvious. This work contributes to a better understanding of the involved classes, but no separation result is foreseen.

Recursion-theoretic approaches lead to classes of functions instead of predicates (or boolean functions). Therefore, instead of *P* and *Pspace* we reach the classes *FPtime* and *FPspace*. As a class of functions corresponding to *NP* we choose $FPtime \cup NP$, and we adopt the notation *FNP*.

Our strategy is, as always in recursion-theoretic contexts, to start with a set of initial functions — which should be basic from the complexity point of view — and to close it under composition and recursion schemes. The recursion schemes can be bounded or unbounded depending on the chosen approach. In the first case we consider the Cobham characterization of *FPtime* [3], in the second case we consider the Bellantoni-Cook characterization of *FPtime* [2]. In both cases we work over $\mathbb{W}$, instead of $\mathbb{N}$, where $\mathbb{W}$ is interpreted over the set of 0-1 words. ϵ stands for the empty word, and S_0 and S_1 stand for concatenation, respectively, with 0 and 1. Therefore, as initial functions one considers ϵ, S_0, S_1, P (binary predecessor) and C (case distinction).

We look to these three classes of complexity — *FPtime*, *FNP* and *FPspace* — as resulting from three different models of computation — deterministic, non-deterministic and alternating Turing machines (as described in [1]) — and imposing the same resource constraint, polynomial time. Thus the adopted recursion schemes should somehow reflect

the "increasing" computational power of the computation model. For *FNP*, besides the calibration of the recursion schemes, we have an additional problem since one is dealing with a class which, in principle, is not closed under composition (because *NP* is, in principle, not closed under negation).

2 Bounded recursion schemes

Bounded recursion schemes are recursion schemes where the length of the outputs are, at every step, bounded. In the cases we treat here, the bound of the lengths is polynomial. The bounds are functions explicitly definable from ϵ, S_0, S_1, string concatenation and string product (corresponding to the smash function of Buss). The bound is, at each step of the recursion, imposed via truncation. We use $x_{|y}$ to denote x truncated to the length of y.

2.1 FPtime

The bounded recursion scheme for *FPtime* described below is based on Cohbam's work [C64] and reproduces the sequential structure of deterministic computations. We denote it by *bounded recursion over* $\mathbb{W}$ *(BR)*:

$$\begin{aligned} f(\epsilon, \bar{x}) &= g(\epsilon, \bar{x}) \\ f(y0, \bar{x}) &= h(y0, \bar{x}, f(y, \bar{x}))_{|t(y0,\bar{x})} \\ f(y1, \bar{x}) &= h(y1, \bar{x}, f(y, \bar{x}))_{|t(y1,\bar{x})} \end{aligned}$$

Notice that, for instance, the definition of $f(11)$ by *BR* (based on g and h and t) leads to $h(11, h(1, g(\epsilon)))$ (t is omitted), which corresponds to the sequence

$$\begin{array}{c} h \\ | \\ h \\ | \\ g \end{array}$$

2.2 FPspace

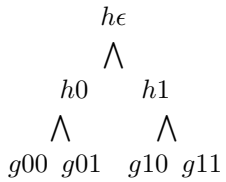
It is well-known that: a function f (over $\mathbb{W}$) is computable in polynomial space if, and only if, f is bitwise computable by an alternating Turing machine in polynomial time, and the length of the outputs of f is polynomial in the length of the inputs.

Alternating Turing machines lead to trees of computation. Therefore, the corresponding recursion scheme, instead of a sequential structure, has a tree structure. It is defined analogously to BR, but we double the recursive call and we distinguish them from each other via a pointer (denoted by p).

Bounded tree recursion over $\mathbb{W}$*(BTR)*, also called *bounded recursion with pointers*:

$$\begin{aligned} f(p, \epsilon, \bar{x}) &= g(p, \epsilon, \bar{x}) \\ f(p, y0, \bar{x}) &= h(p, y0, \bar{x}, f(p0, y, \bar{x}), f(p1, y, \bar{x}))_{|t(p,y0,\bar{x})} \\ f(p, y1, \bar{x}) &= h(p, y1, \bar{x}, f(p0, y, \bar{x}), f(p1, y, \bar{x}))_{|t(p,y1,\bar{x})} \end{aligned}$$

If $f(\epsilon, 11)$ is defined by *BTR* on its second input based on g, h and t, then (omitting, once more, the bound t) one obtains $h(\epsilon, h(0, g(00), g(01)), h(1, g(10), g(11)))$. The corresponding tree is



The mentioned input is the pointer, and it gives the address from the root of the tree to the current node. The tree structure of BTR is clear. It is also clear that if h and g do not depend on their first input (the pointer), then the tree structure collapses to a sequential one. Therefore, BTR trivially extends BR.

More about this characterization of *FPspace* can be found in [4].

2.3 FNP

Non-deterministic Turing machines can be seen, simultaneously, as an extension of the concept of deterministic Turing machines and a restriction of alternating Turing machines. Thus our goal is, also simultaneously, to extend BR and restrict BTR in an appropriated way.

We would like to do it via a single recursion scheme, however so far that was not achieved. This issue is also related with the restricted form of composition one may have in *FNP*.

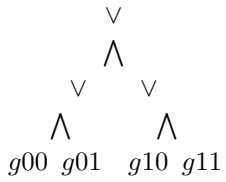
What we describe here is a recursion scheme which should be taken in addition to BR. We call it $TR[\vee]$ because it results from BTR by fixing the step function h — h is the disjunction of its last two inputs (the recursive calls). More precisely, *disjunctive tree recursion over* $\mathbb{W}$ ($TR[\vee]$) is the scheme:

$$\begin{aligned} f(p,\epsilon,\bar{x}) &= g(p,\epsilon,\bar{x}) \\ f(p,y0,\bar{x}) &= \vee(f(p0,y,\bar{x}),f(p1,y,\bar{x})) \\ f(p,y1,\bar{x}) &= \vee(f(p0,y,\bar{x}),f(p1,y,\bar{x})), \end{aligned}$$

where $\vee(u,v)$ returns 1 if at least one of its inputs ends with 1, and 0 otherwise.

Notice that there is no need of imposing bounds — a single bit is returned at every step of the recursion (with possible exception of the base level, where g is computed).

Let us look at our example once more. If $f(\epsilon,11)$ is defined by $TR[\vee]$ based on g, then one has $\vee(\epsilon,\vee(0,g(00),g(01)),\vee(1,g(10),g(11)))$, which corresponds to the tree



Therefore one gets a tree structure as before, but only the addresses of the leaves are available. All internal nodes have the same (disjunctive) label. The parallel with non-deterministic Turing machines is obvious.

Notice that if, in $TR[\vee]$, g does not depend on the pointer, then the scheme loses his tree structure. However, since the step function is fixed (it is $\vee$) this scheme does not extend BR. As mentioned above, $TR[\vee]$ is taken in addition to BR.

See [5] for more about this characterization of *FNP*.

3 Final considerations

With a simple example one is able to illustrate the structure of the recursion schemes used to describe the classes of computational complexity *FPtime*, *FNP* and *FPspace*. The connection between the structure of the recursion and the underlying model of computation is of interest and it might deserve some further thoughts. Some work is being developed concerning the levels of the polynomial hierarchy of time.

FPtime, *FNP* and *FPspace* are reached in a recursion-theoretic manner by successively "extending" the characterization of *FPtime* given in 1964 by Cobham. That is achieved by introducing pointers in the recursion schemes. Recursion with pointers can be understood as a restrict form of recursion with substitution. Leivant and Marion have work in this direction.

What is here stated using recursion schemes with bounds can be done in other frameworks. The polynomial bounds explicitly address the resource constraint of the studied complexity classes. There exist several ways of enriching the syntax, in order to build in the classes some internal control on the growth of the functions terms. This can be done, for instance, via ranks (which measure the syntactical complexity of the functions terms), distinguishing sorts of variables (Leivant style), or sorts of input-positions (Bellantoni-Cook style).

Acknowledgements. I want to thank the funding of the projects PTDC/MAT/104716/2008 and PEst-OE/MAT/UI0209/2011, from Fundação para a Ciência e a Tecnologia.

References

1 J. L. Balcázar, J. Díaz, J. Gabarró, *Structural Complexity I and II*, Springer-Verlag, (1990)
2 S. Bellantoni and S. Cook, *A new recursion-theoretic characterization of Polytime functions*, Computational Complexity, vol. 2 (1992), pp. 97–110.
3 A. Cobham, *The intrinsic computational difficulty of functions*, Proc. of the 1964 International Congress for Logic, Methodology, and the Philosophy of Science, ed. Y. Bar-Hillel, North Holland, Amsterdam (1965), pp. 24–30.
4 I. Oitavem, *Characterizing Pspace with pointers*, Mathematical Logic Quarterly, vol. 54 (2008), no. 3, pp. 317–323.
5 I. Oitavem, *A recursion-theoretic approach to NP*, Annals of Pure and Applied Logic, vol. 162 (2011), no. 8, pp. 661–666.

Means and Limits of Decision

Lidia Tendera

Institute of Mathematics and Informatics
Opole University, Poland
tendera@math.uni.opole.pl

Abstract

In this talk we survey recent work in the quest for expressive logics with good algorithmic properties, starting from the two-variable fragment of first-order logic and the guarded fragment. While tracing the boundary between decidable and undecidable fragments we describe their power, limitations, similarities and differences in order to stress out key properties responsible for their good or bad behaviour. We also highlight tools and techniques that have proven most effective for designing optimal algorithms, special attention giving to the more universal ones.

1998 ACM Subject Classification F.4.1 Mathematical Logic, F.1.1 Models of Computation, F.4.3 Formal Languages

Keywords and phrases classical decision problem, decidability, computational complexity, two-variable first-order logic, guarded logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.28

Category Invited Talk

1 Overview

In Computer Science, the use of logical formalisms to describe, query or manipulate structured data is now firmly embedded in both theory and practice. Having data described/specified within a logical formalism we often want such a specification to undergo static analysis – an automated procedure that optimizes the specification with respect to some correctness and efficiency criteria. Static analysis of specifications described in logical formalism often boils down to verifying one of the two basic logical properties, namely satisfiability and finite satisfiability.

Undecidability of the classical decision problem (=the satisfiability problem for first-order logic) results in two possible responses. The first one is to develop programs to test satisfiability of arbitrary collection of first-order formulas, accepting that, however well they generally work in practise, there will always be problem instances that defeat them. The second is to restrict attention to a *fragment* of first-order logic for which the satisfiability problem is decidable, exploiting the fact that in many real-life situations, the formulas we encounter fit comfortably into such fragments.

In this talk we overview recent work in the quest for expressive logics with good algorithmic properties. We concentrate mainly on fragments of first-order logic defined by restricting the number of variables (to gain decidability – to *two* [9, 7]) and usage of quantifiers to *guarded* quantification [1], and their variants or extensions motivated by real-life applications. We are equally interested in satisfiability and finite satisfiability, as in many application areas we want to model systems and computation to be essentially finite.

While tracing the boundary between decidable and undecidable fragments we study their similarities and differences to understand their power and limitations and to stress out key

properties responsible for (un)decidability or (in)tractability. We give examples of fragments enjoying the *finite model property*: any satisfiable formula is true in some finite structure, and the *tree model property*: any satisfiable formula is true is some tree-like structure. We present fragments for which these two key properties led to optimal decision procedures (e.g. [4], [3], [11]) and contrast these fragments with their extensions where more sophisticated reasoning is required (e.g. [6, 5, 10]). We give special attention to linear and integer programming techniques that have recently proved useful to design optimal algorithms to decide uniformly both, the finite and the unrestricted satisfiability problems for certain expressive fragments.

The talk involves recent and ongoing work with Emanuel Kieroński, Jakub Michaliszyn, Ian Pratt-Hartmann, Wiesław Szwast, Georg Gottlob and Andreas Pieris.

The title of the talk has been inspired by Quine [8].

References

1. H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *ILLC Research Report ML-1996-03*, University of Amsterdam, 1996. Journal version in: *J. Philos. Logic*, 27 (1998), no. 3, 217–274.
2. G. Gottlob, A. Pieris, and L. Tendera. Qyerying the guarded fragment. In *ICALP*, pages 293–304, 2013.
3. E. Grädel. Decision procedures for guarded logics. In *16th International Conference in Artificial Intelligence*, volume LNCS 1932, pages 31–51. Springer, 1999.
4. E. Grädel, P. Kolaitis, and M. Vardi. On the decision problem for two-variable first-order logic. *Bull. of Symb. Logic*, 3(1):53–69, 1997.
5. E. Kieroński, J. Michaliszyn, I. Pratt-Hartmann, and L. Tendera. Two-variable first-order logic with equivalence closure. In *Proc. of LICS2012*, pages 431–440. IEEE, 2012.
6. E. Kieroński and L. Tendera. On finite satisfiability of two-variable first-order logic with equivalence relations. In *Proc. of LICS2009*, pages 123–132, 2009.
7. M. Mortimer. On languages with two variables. *Zeitschr. f. Logik und Grundlagen d. Math.*, 21:135–140, 1975.
8. W. V. Quine. *Theories and Things*, chapter On the Limits of Decision, pages 156–163. Harvard University Press, Cambridge, MA, 1981. A shorter version of this paper appeared in the *Akten des XIV. internationalen Kongresses für Philosophie*, vol. 3, 1969.
9. D. Scott. A decision method for validity of sentences in two variables. *J. Symb. Logic*, 27:477, 1962.
10. W. Szwast and L. Tendera. FO^2 with one transitive relation is decidable. In Natacha Portier and Thomas Wilke, editors, *STACS*, volume 20 of *LIPIcs*, pages 317–328. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2013.
11. M. Y. Vardi. Why is modal logic so robustly decidable? *DIMACS Series in Discrete Mathematics and Theoretical Computer Science,*, 31:149–184, 1997.

On closure ordinals for the modal μ-calculus

Bahareh Afshari[1] and Graham E. Leigh[2]

1 **Department of Computer Science, University of Oxford, Parks Road, Oxford OX1 3QD, UK**
bahareh.afshari@cs.ox.ac.uk

2 **Faculty of Philosophy, University of Oxford, Woodstock Road, Oxford OX2 6GG, UK**
graham.leigh@philosophy.ox.ac.uk

Abstract

The closure ordinal of a formula of modal μ-calculus $\mu X\varphi$ is the least ordinal κ, if it exists, such that the denotation of the formula and the κ-th iteration of the monotone operator induced by φ coincide across all transition systems (finite and infinite). It is known that for every $\alpha < \omega^2$ there is a formula φ of modal logic such that $\mu X\varphi$ has closure ordinal α [3]. We prove that the closure ordinals arising from the alternation-free fragment of modal μ-calculus (the syntactic class capturing $\Sigma_2 \cap \Pi_2$) are bounded by ω^2. In this logic satisfaction can be characterised in terms of the existence of tableaux, trees generated by systematically breaking down formulæ into their constituents according to the semantics of the calculus. To obtain optimal upper bounds we utilise the connection between closure ordinals of formulæ and embedded order-types of the corresponding tableaux.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Closure ordinals, Modal mu-calculus, Tableaux

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.30

1 Introduction

Modal μ-calculus is often referred to as the "mother of all temporal logics". Indeed the majority of temporal logics, including LTL (Linear Time Logic), CTL (Computational Tree Logic) and their various extensions, can be easily interpreted and analysed in μ-calculus making the study of this logic of high interest in the research community. The defining feature of the modal μ-calculus is the expression of fixpoints. In this calculus the syntax of modal logic is extended by least and greatest fixpoint quantifiers (μ and ν) that bind propositional variables. The formulæ $\mu X\varphi$ and $\nu X\varphi$ are interpreted respectively as the least and greatest fixpoints of the monotone operator induced by φ. In analogy to the hierarchies defined in second order logic, one can alternate the fixpoint quantifiers to define a hierarchy of formulæ. Although we have a relatively good understanding of least and greatest fixpoints, when nested their meaning and behaviour is easily lost. As a result many fundamental properties of this calculus have remained unanswered even after decades of attention from logicians and computer scientists.

An interesting open problem for μ-calculus is that of closure ordinals, the number of iterations required for a fixpoint to close across all structures. Given an arbitrary formula, its closure ordinal may not exist, such as in the case of $\mu X\Box X$. On the other hand mere syntactic analysis suggests that the fixpoint iterations in this context cannot exhaust the power of ordinals beyond certain levels. Hence one may ask the following question.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca ; pp. 30–44

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

For which ordinals α is there a formula of modal μ-calculus with closure ordinal α?

In the case of finite ordinals the formulæ $\mu X.(\Diamond X \wedge \Box^n \bot) \vee \Box\bot$, which express that all paths in a model of the formula have length at most n, are guaranteed to close across all structures after n iterations. By expressing the existence of arbitrarily long finite paths, through the formula $\mu X.\Diamond X \vee \Box\bot$ for example, transfinite closure ordinals are obtained. In fact it is known that for every $\alpha < \omega^2$ there is a formula φ of modal logic such that $\mu X\varphi$ has closure ordinal α [3].

In this paper we establish optimal upper bounds on closure ordinals, showing that no formula of the alternation-free fragment can have a closure ordinal equal or greater than ω^2, even if iterations of all quantifiers occurring in the formula are taken into account. We begin with a syntactic analysis on a fragment of the Σ_1-formulæ in section 2. This study, despite applying only to operators induced by particular formulæ of modal logic, provides the motivation for the general solution. The main result of the paper is given in section 3 and consists of a semantic analysis of the problem by means of tableaux constructions. We present a strong characterisation of closure ordinals in terms of order-types of tableaux for formulæ without genuine dependencies between their alternating fixpoint quantifiers. This correspondence will prove sufficient to bound closure ordinals of these formulæ by their logical complexity.

1.1 Syntax and semantics of modal μ-formulæ

Let VAR be an infinite set of propositional variables and PROP an infinite set of propositional constants. The set of μ-formulæ is defined inductively as follows.

$$\varphi := p \mid \bar{p} \mid X \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \mu X\,\varphi \mid \nu X\,\varphi$$

where $p \in$ PROP and $X \in$ VAR. Also define $\bot := p \wedge \bar{p}$ and $\top := p \vee \bar{p}$ for some propositional constant p. A variable X in φ is called a μ-variable (respectively, ν-variable) if the quantifier μX (resp. νX) occurs in φ. We assume that all quantifiers occur uniquely. This can be achieved through implicit α-conversion.

A *transition system* is a tuple $T = (S, \to, \lambda)$ where $(S, \to)$ is a directed graph and $\lambda\colon S \to \mathcal{P}(\text{PROP})$ is an assignment of propositional constants to states. Given a transition system $T = (S, \to, \lambda)$ and a valuation $\mathcal{V}\colon \text{VAR} \to \mathcal{P}(S)$ of free variables, the set of states satisfying a formula φ, denoted by $\|\varphi\|^T_{\mathcal{V}}$, is defined inductively as follows.

$$\begin{aligned}
&\|p\|^T_{\mathcal{V}} = \{x \in S : p \in \lambda(x)\}\\
&\|\bar{p}\|^T_{\mathcal{V}} = \{x \in S : p \notin \lambda(x)\}\\
&\|X\|^T_{\mathcal{V}} = \mathcal{V}(X)\\
&\|\varphi \wedge \psi\|^T_{\mathcal{V}} = \|\varphi\|^T_{\mathcal{V}} \cap \|\psi\|^T_{\mathcal{V}}\\
&\|\varphi \vee \psi\|^T_{\mathcal{V}} = \|\varphi\|^T_{\mathcal{V}} \cup \|\psi\|^T_{\mathcal{V}}\\
&\|\Box\varphi\|^T_{\mathcal{V}} = \{x \in S : \forall y(x \to y \Rightarrow y \in \|\varphi\|^T_{\mathcal{V}})\}\\
&\|\Diamond\varphi\|^T_{\mathcal{V}} = \{x \in S : \exists y(x \to y \wedge y \in \|\varphi\|^T_{\mathcal{V}})\}\\
&\|\mu X\varphi(X)\|^T_{\mathcal{V}} = \bigcap\{U \subseteq S : \|\varphi\|^T_{\mathcal{V}[X\mapsto U]} \subseteq U\}\\
&\|\nu X\varphi(X)\|^T_{\mathcal{V}} = \bigcup\{U \subseteq S : U \subseteq \|\varphi\|^T_{\mathcal{V}[X\mapsto U]}\}
\end{aligned}$$

In the above $\mathcal{V}[X \mapsto U]$ is the valuation that maps X into U and agrees with $\mathcal{V}$ on all other variables. Note that a formula φ gives rise to a function $f_\varphi\colon \mathcal{P}(S) \to \mathcal{P}(S)$ given by $U \mapsto \{x \in S : x \in \|\varphi(X)\|^T_{\mathcal{V}[X\mapsto U]}\}$. As f_φ is a monotone function on the powerset lattice

$\langle \mathcal{P}(S), \subseteq \rangle$, by the Knaster-Tarski Theorem its least (and greatest) fixpoint exists, and is equal to the least prefixed point (resp. greatest postfixed point) of f_φ, the set $\|\mu X \varphi\|^T_{\mathcal{V}}$ (resp. $\|\nu X \varphi\|^T_{\mathcal{V}}$).

1.2 Alternation-free fragment

The alternation of fixpoint quantifiers is the major source of potency, and a fundamental measure of logical strength in the study of fragments of μ-calculus. The number of genuine alternations between least and greatest fixpoint quantifiers is called the *depth* of the formula. Bradfield [1] showed that there are modal fixpoint properties which require arbitrarily large depth, and hence the modal μ-calculus alternation hierarchy is strict. Formally, the Niwiński hierarchy is defined as follows. A formula φ is in the classes Π_0 and Σ_0 if it contains no fixpoint quantifiers, i.e. it is a formula of modal logic. The class Σ_{n+1} (Π_{n+1}) is the closure of $\Sigma_n \cup \Pi_n$ under the following rules.

- If $\varphi, \psi \in \Sigma_{n+1}$ (Π_{n+1}), then $\varphi \wedge \psi, \varphi \vee \psi, \Box\varphi, \Diamond\varphi \in \Sigma_{n+1}$ (Π_{n+1}).
- If $\varphi \in \Sigma_{n+1}$ (Π_{n+1}), then $\mu X \varphi \in \Sigma_{n+1}$ ($\nu X \varphi \in \Pi_{n+1}$).
- If $\varphi, \psi \in \Sigma_{n+1}$ (Π_{n+1}), then $\varphi(\psi) \in \Sigma_{n+1}$ (Π_{n+1}), provided the free variables of ψ do not become bound by quantifiers in φ.

In comparison the *alternation-free fragment* of the modal μ-calculus is the class of formulæ with no real dependencies between alternating fixpoint quantifiers. This fragment is the closure of $\Sigma_1 \cup \Pi_1$ under Boolean and modal operators and substitutions that preserve the alternation depth. Despite the restrictions imposed, this class of properties still forms a remarkably expressive fragment encompassing the majority of logics used in the verification of systems. It is known that this class coincides with the collection of all formulæ semantically equivalent to both a Σ_2-formula and a Π_2-formula [5]. Moreover, this fragment is the limit of the weak index hierarchy as introduced in [6]; thus, the languages defined by alternation-free formulæ are also referred to as *weakly definable languages*.

1.3 Trees

A *tree* is a pair $t = (V, \to)$ with a distinguished node ρ_t such that $(V, \to)$ is a connected directed graph, there are no transitions into ρ_t and for every $v \in V \setminus \{\rho_t\}$ there is exactly one $v_0 \in V$ such that $v_0 \to v$. The node ρ_t is referred to as the *root* of the tree and any node without outgoing transitions is called a *leaf*. For a tree t and a node v in t, we write $t{\restriction_v}$ to denote the sub-tree rooted at v. If there is no cause for confusion we identify a tree with its domain. Tree $t_0 = (V_0, \to_0)$ is a *pruning* of $t = (V, \to)$ if $V_0 \subseteq V$, $\to_0 = \to \cap V_0^2$ and if $u \to v \notin V_0$ then $\{w \in V_0 : u \to_0 w\} = \emptyset$.

A *path* through a tree $t = (V, \to)$ is an enumerable set $\mathbb{P} \subseteq V$ such that $\rho_t \in \mathbb{P}$, if $v_0 \to v \in \mathbb{P}$ then $v_0 \in \mathbb{P}$, and for every $v \in \mathbb{P}$ either v is a leaf or there exists exactly one $u \in V$ such that $v \to u$ and $u \in \mathbb{P}$. For a path $\mathbb{P}$ given by a sequence $\rho_t = v_0 \to v_1 \to v_2 \to \ldots \to v_n \to \ldots$, we write $\mathbb{P}(n)$ to denote v_n. For nodes $u, v \in t$ we write $u <_t v$ (resp. $u \leq_t v$) if for some path $\mathbb{P}$ through t and $i < j$ (resp. $i \leq j$), $\mathbb{P}(i) = u$ and $\mathbb{P}(j) = v$.

A *tree transition system* (TTS) is a transition system $T = (S, \to, \lambda)$ for which $(S, \to)$ is a tree. We say a TTS T *satisfies* φ, written $T \models \varphi$, if $\rho_T \in \|\varphi\|^T_{\mathcal{V}}$. In this case T is a *model* of φ and φ is *satisfiable*. Note that modal μ-calculus has the *tree model property*, namely every satisfiable formula has a model which is a TTS (see e.g. [2]).

1.4 Closure ordinals

The definition of semantics for μ-formulæ can be generalised to also take into account approximations to fixpoint variables. For each formula φ, set of bound variables $\mathcal{X}$ occurring in φ and ordinal α, we define a set $\|\varphi^\alpha\|^T_\mathcal{V}$ by induction on α. Let $T = (S, \to, \lambda)$ be a transition system and $\mathcal{V}$ a valuation on T. For every α, define

$$\|p^\alpha\|^T_\mathcal{V} = \|p\|^T_\mathcal{V}$$
$$\|\bar{p}^\alpha\|^T_\mathcal{V} = \|\bar{p}\|^T_\mathcal{V}$$
$$\|Z^\alpha\|^T_\mathcal{V} = \mathcal{V}(Z)$$
$$\|(\varphi \wedge \psi)^\alpha\|^T_\mathcal{V} = \|\varphi^\alpha\|^T_\mathcal{V} \cap \|\psi^\alpha\|^T_\mathcal{V}$$
$$\|(\varphi \vee \psi)^\alpha\|^T_\mathcal{V} = \|\varphi^\alpha\|^T_\mathcal{V} \cup \|\psi^\alpha\|^T_\mathcal{V}$$
$$\|(\Box\varphi)^\alpha\|^T_\mathcal{V} = \{x \in S : \forall y(x \to y \Rightarrow y \in \|\varphi^\alpha\|^T_\mathcal{V})\}$$
$$\|(\Diamond\varphi)^\alpha\|^T_\mathcal{V} = \{x \in S : \exists y(x \to y \wedge y \in \|\varphi^\alpha\|^T_\mathcal{V})\}$$
$$\|(\mu X\varphi)^\alpha\|^T_\mathcal{V} = \begin{cases} \bigcup_{\gamma<\alpha} \|\varphi[\mu X\varphi/X]^\gamma\|^T_\mathcal{V}, & \text{if } X \in \mathcal{X}, \\ \|\mu X\varphi^\alpha\|^T_\mathcal{V}, & \text{otherwise.} \end{cases}$$
$$\|(\nu X\varphi)^\alpha\|^T_\mathcal{V} = \begin{cases} \bigcap_{\gamma<\alpha} \|\varphi[\nu X\varphi/X]^\gamma\|^T_\mathcal{V}, & \text{if } X \in \mathcal{X}, \\ \|\nu X\varphi^\alpha\|^T_\mathcal{V}, & \text{otherwise.} \end{cases}$$

For every formula φ there exists an ordinal κ such that $\|\varphi\|^T_\mathcal{V} = \|\varphi^\kappa\|^T_\mathcal{V} = \|\varphi^{\kappa+1}\|^T_\mathcal{V}$. The least such κ is called the *closure ordinal of φ with respect to T and $\mathcal{X}$* and is denoted $CO_{T,\mathcal{X}}(\varphi)$. Note that a formula may have different closure ordinals depending on the transition system on which it is evaluated as well as the particular collection of variables analysed. For example the formula $\mu X\Box X$ is satisfied by all well-founded trees; its closure ordinal with respect to $\{X\}$ in each case is the order-type of the tree.

▶ Definition 1.1 (Closure Ordinal). The *closure ordinal* of a closed formula φ with respect to a non-empty set $\mathcal{X}$ of variables, denoted by $CO_\mathcal{X}(\varphi)$, is the ordinal $\sup_T CO_{T,\mathcal{X}}(\varphi)$, if this ordinal exists.

2 Syntactic analysis

Let $\overline{\text{Prop}} := \{\bar{p} : p \in \text{Prop}\}$ and $P_1, P_1', P_2, P_2', \ldots, P_n, P_n'$ be finite subsets of $\text{Prop} \cup \overline{\text{Prop}}$. Each such set, when referred to as a formula, denotes the conjunction of its elements. We say a formula of modal logic is *primary* if it is of the form

$$(P_1 \wedge \Box P_1' \wedge \nabla_1 X) \vee (P_2 \wedge \Box P_2' \wedge \nabla_2 X) \vee \ldots \vee (P_n \wedge \Box P_n' \wedge \nabla_n X) \vee \Box\bot \tag{1}$$

where $\nabla_i \in \{\Diamond, \Box\}$ for each i. Czarnecki's analysis in [3] establishes that every ordinal below ω^2 is the closure ordinal of the least fixpoint of some primary formula. In this section we establish a strong converse: if the primary formula given in (1) has closure ordinal α, then $\alpha < \omega.(n+1)$. For the following let ψ denote the formula in (1) and $\varphi = \mu X\psi$.

▶ Lemma 2.1. Fix a transition system T and a valuation $\mathcal{V}$. Suppose κ is a limit ordinal. If $x \in \|\varphi^{\kappa+1}\|^T_\mathcal{V} \setminus \|\varphi^\kappa\|^T_\mathcal{V}$, then there is no $j \leq n$ such that $x \in \|P_j \wedge \Box P_j' \wedge \nabla_j\varphi^\kappa\|^T_\mathcal{V}$ and $\nabla_j = \Diamond$.

Proof. Suppose $T = (S, \to, \lambda)$ and let $\|\varphi^\alpha\|$ abbreviate $\|\varphi^\alpha\|^T_\mathcal{V}$. Suppose $x \in \|\varphi^{\kappa+1}\| \setminus \|\varphi^\kappa\|$. By way of contradiction suppose also $x \in \|P_j \wedge \Box P_j' \wedge \nabla_j\varphi^\kappa\|$ and $\nabla_j = \Diamond$ for some $j \leq n$. If $\{y \in S : x \to y\} = \emptyset$ then $x \in \|\varphi^1\| \subseteq \|\varphi^\kappa\|$ which cannot be, so let $x \to y$ be such that $y \in \|\varphi^\kappa\|$. Thus there exists $\gamma < \kappa$ such that $y \in \|\varphi^\gamma\|$, and hence $x \in \|\varphi^{\gamma+1}\| \subseteq \|\varphi^\kappa\|$ yielding a contradiction. ◀

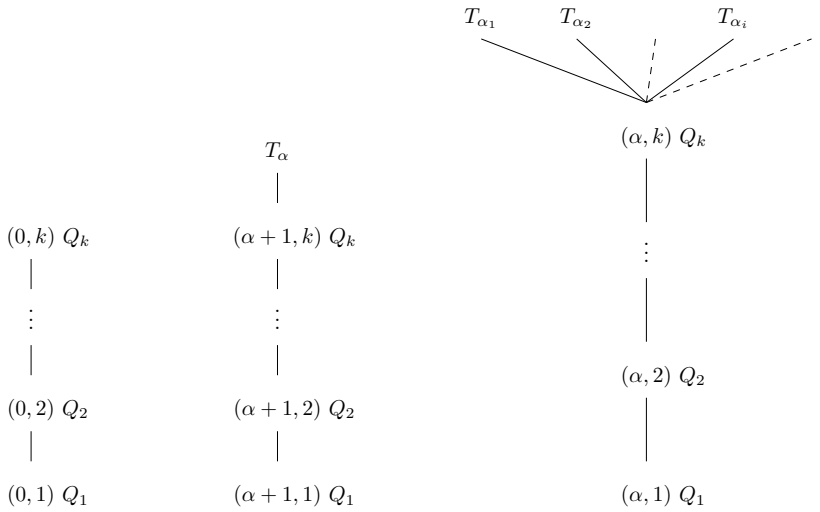


Figure 1 T_0, $T_{\alpha+1}$ and T_α (in the case $\alpha = \sup_i \alpha_i$) in the proof of lemma 2.3.

▶ Corollary 2.2. If $\nabla_i = \Diamond$ for every $i \leq n$ then the closure ordinal of $\mu X\psi$ exists and is no greater than ω.

▶ Lemma 2.3. Suppose there exist consistent sets of propositions Q_1, Q_2,...,Q_{k+1} and numbers $i_1, i_2, \ldots, i_k < n$ such that $P_{i_j} \wedge \Box P'_{i_j} \wedge \nabla_{i_j} X$ is a subformula of ψ with $P_{i_j} \subseteq Q_j$ and $P'_{i_j} \subseteq Q_{j+1}$ for each $j \leq k$. Furthermore, suppose $\nabla_{i_k} = \Box$ and there is no $j \leq n$ such that $P_j \subseteq Q_k$, $P'_j \subseteq Q_{k+1}$ and $\nabla_j = \Diamond$. If $Q_{k+1} = Q_1$, then $\mu X\psi$ does not have a closure ordinal.

Proof. Let λ: On $\times \{i : i \leq k+1\} \to \mathcal{P}(\text{Prop})$, where On is the class of all ordinals, be defined by $p \in \lambda((\alpha, j))$ if and only if $p \in Q_j$. Furthermore, let $T_0^\alpha = (S_0^\alpha, \to_0^\alpha, \lambda)$ be the TTS where

$$S_0^\alpha = \{(\alpha, j) : 0 < j \leq k\},$$
$$\to_0^\alpha = \{((\alpha, j), (\alpha, j+1)) : 0 < j < k\}.$$

For each countable ordinal α we define a tree T_α as follows. Let $T_0 = T_0^0$ and $T_{\alpha+1} = (S_{\alpha+1}, \to_{\alpha+1}, \lambda)$ where $S_{\alpha+1} = S_0^{\alpha+1} \cup S_\alpha$ and $\to_{\alpha+1} = \to_0^{\alpha+1} \cup \to_\alpha \cup \{((\alpha+1, k), (\alpha, 1))\}$. If α is a limit ordinal, then $S_\alpha = S_0^\alpha \cup \bigcup_{\beta<\alpha} S_\beta$ and $\to_\alpha = \to_0^\alpha \cup \bigcup_{\beta<\alpha} \to_\beta \cup \{(\alpha, k), (\beta, 1)) : \beta < \alpha\}$.

Let f be the function $\kappa \mapsto k.\kappa$. We will show that for each $\kappa \leq \alpha$ and $0 \leq j < k$,

$$(\kappa, k-j) \in \|\varphi^{f(\kappa)+j+1}\|_{\mathcal{V}}^{T_\alpha} \setminus \|\varphi^{f(\kappa)+j}\|_{\mathcal{V}}^{T_\alpha} \tag{2}$$

whereby it will be clear that the formula φ does not possess a closure ordinal. The argument proceeds by transfinite induction on $\kappa \leq \alpha$ with an auxiliary induction on $j < k$. If $j \neq 0$ then (2) follows from the fact that $(\kappa, k-(j-1))$ is the unique successor of $(\kappa, k-j)$ and the definition of λ. Thus suppose $j = 0$, whence three sub-cases manifest:

- $\kappa = 0$. Then $f(\kappa) = 0$ and (κ, k) is a leaf of T_α, so (2) trivially holds.
- $\kappa = \kappa' + 1$. By the definition of T_α, (κ, k) has a unique successor, namely $(\kappa', 1)$, whence (2) follows from the induction hypothesis
- κ limit. The successors of $(\kappa, k-j)$ in this case are the nodes $(\gamma, 1)$ for $\gamma < \kappa$. By the induction hypothesis we know $(\gamma, 1) \in \|\varphi^{f(\gamma+1)}\|_{\mathcal{V}}^{T_\alpha} \setminus \|\varphi^{f(\gamma)+k-1}\|_{\mathcal{V}}^{T_\alpha}$ for each $\gamma < \kappa$. Notice

that $\|\varphi^{f(\kappa)}\|^{T_\alpha}_{\mathcal{V}} = \bigcup_{\gamma<\kappa}\|\varphi^{f(\gamma)}\|^{T_\alpha}_{\mathcal{V}}$. Since $P_{i_k} \subseteq Q_k$, $P'_{i_k} \subseteq Q_1$ and $\nabla_{i_k} = \Box$, it follows that $(\kappa, k) \in \|\varphi^{f(\kappa)+1}\|^{T_\alpha}_{\mathcal{V}}$. If, however, $(\kappa, k) \in \|\varphi^{f(\kappa)}\|^{T_\alpha}_{\mathcal{V}}$, then $(\kappa, k) \in \|\varphi^{f(\gamma)+k}\|^{T_\alpha}_{\mathcal{V}}$ for some $\gamma < \kappa$. But then $(\gamma, 1) \in \|\varphi^{f(\gamma)+k-1}\|^{T_\alpha}_{\mathcal{V}}$ by the assumption on φ. ◀

▶ Proposition 2.4. The closure ordinal of φ is strictly less than ω^2.

Proof. Let α be the closure ordinal of φ and suppose $\alpha \geq \omega^2$. Fix $N \geq 2^{|\varphi|+1}$ where $|\varphi|$ denotes the number of symbols occurring in φ. Let T be a TTS such that for every $i \leq N$, $\|\varphi^{\omega.i}\|^T_{\mathcal{V}}$ is a proper subset of $\|\varphi^{\omega.i+1}\|^T_{\mathcal{V}}$. Then there exists a path $\mathbb{P}$ through T, $m_N < m_{N-1} < \cdots < m_0 < \omega$ and a function $f\colon \omega \times \omega \to \omega$ such that for every $i \leq N$ and $j < m_i - m_{i+1}$, $f(i,j) \leq n$ and

$$\mathbb{P}(m_i - j) \in \|P_{f(i,j)} \wedge \Box P'_{f(i,j)} \wedge \nabla_{f(i,j)}\varphi^{\omega.i+j}\|^T_{\mathcal{V}} \setminus \|\varphi^{\omega.i+j}\|^T_{\mathcal{V}}.$$

Define for each $j < \omega$, $Q_j = \lambda_T(\mathbb{P}(j)) \cup \{\bar{p} : p \notin \lambda_T(\mathbb{P}(j))\}$. For some $i_0 < i_1 \leq N$ it must be the case that

$$Q_{m_{i_0}} \cap \textsc{Prop}_\varphi = Q_{m_{i_1}} \cap \textsc{Prop}_\varphi$$

where $\textsc{Prop}_\varphi = \bigcup_{i\leq n}(P_i \cup P'_i)$. The sequence $Q_{m_{i_1}}, \ldots, Q_{m_{i_0}}$ therefore fulfils the hypothesis of lemma 2.3 whence, contrary to our assumption, φ does not have a closure ordinal. ◀

The above analysis can also be applied to formulæ of the form

$$(\psi_1 \wedge \nabla_1 X) \vee (\psi_2 \wedge \nabla_2 X) \vee \ldots \vee (\psi_n \wedge \nabla_n X) \vee \Box\bot \tag{3}$$

where $\psi_1, \ldots, \psi_n$ are closed formulæ of modal logic. Replacing literals with arbitrary modal formulæ in each disjunct alters the "proposition paths" that can occur. Therefore, in order to find a repetition as in the proof of proposition 2.4, one will need to look at larger segments of a suitable model. As such a proof would be technically cumbersome, in the next section we will employ a semantic analysis which will include (3) and extend the bounds to formulæ of the alternation-free fragment of μ-calculus.

3 Semantic analysis

For the remainder of the paper, formulæ are assumed to be closed and guarded unless otherwise stated. A formula φ is *guarded* if in every subformula $\sigma Z.\psi$ of φ, every occurrence of the bound variable Z in ψ appears within the scope of a modal operator. The restriction to the guarded fragment is not significant as every formula is equivalent to one in guarded form (see e.g. [7]). Moreover, by following the approach of [4] it is possible to carry out the analysis below for unguarded formulæ.[1]

Upper-case Greek letters such as Γ and Δ denote *sequents*, finite sets of formulæ. $\Box\Gamma$ abbreviates the set $\{\Box\varphi : \varphi \in \Gamma\}$ and $\Diamond\Gamma$ is defined analogously. We write Γ, φ for $\Gamma \cup \{\varphi\}$, and Γ, Δ to denote $\Gamma \cup \Delta$. The *Fischer-Ladner closure* of a formula φ, denoted by $\mathrm{FL}(\varphi)$, is the smallest set such that

- $\varphi \in \mathrm{FL}(\varphi)$,
- if $\psi_0 \circ \psi_1 \in \mathrm{FL}(\varphi)$ where $\circ \in \{\vee, \wedge\}$ then $\psi_0, \psi_1 \in \mathrm{FL}(\varphi)$,
- if $\nabla\psi \in \mathrm{FL}(\varphi)$ where $\nabla \in \{\Diamond, \Box\}$ then $\psi \in \mathrm{FL}(\varphi)$,
- if $\sigma X\psi \in \mathrm{FL}(\varphi)$ where $\sigma \in \{\mu, \nu\}$ then $\psi[\sigma X\psi/X] \in \mathrm{FL}(\varphi)$.

Note that $|\mathrm{FL}(\varphi)| \leq |\varphi|$ where $|\varphi|$ denotes the number of symbols occurring in φ. For a sequent Γ we set $\mathrm{FL}(\Gamma) = \bigcup_{\gamma\in\Gamma}\mathrm{FL}(\gamma)$.

[1] We would like to thank the anonymous referee for drawing our attention to [4].

3.1 Tableaux

▶ Definition 3.1. Given a TTS T and a sequent Γ, a *pre-tableau* for (T, Γ) is a tree $t = (V, \to)$ together with functions $\tau_t \colon t \to T$ and $\lambda_t \colon t \to \mathcal{P}(\mathrm{FL}(\Gamma))$ such that the following conditions are satisfied.

- $\tau_t(\rho_t) = \rho_T$ and $\lambda_t(\rho_t) = \Gamma$.
- If $v \in t$ is a leaf then $\lambda_t(v) = \Box\Xi, \Theta$ where $\Theta \subseteq \textsc{Prop} \cup \overline{\textsc{Prop}}$, and either $\Xi = \emptyset$ or $\tau_t(v)$ is a leaf of T.
- If $\tau_t(u) = \tau_t(v)$ then either $u \leq_t v$ or $v \leq_t u$.
- For every $v \in t$, $\lambda_t(v) \cap \textsc{Prop} \subseteq \lambda_T(\tau_t(v)) \subseteq \{p \in \textsc{Prop} : \bar{p} \notin \lambda_t(v)\}$.
- For every $v_0 \to v_1 \in t$ with $\tau_t(v_i) = x_i$ and $\lambda_t(v_i) = \Gamma_i$, one of the following conditions hold.

($\wedge$) $x_0 = x_1$ and there are formulæ φ_0, φ_1 such that $\varphi_0 \wedge \varphi_1 \in \Gamma_0$ and $\Gamma_1 = (\Gamma_0 \setminus \{\varphi_0 \wedge \varphi_1\}) \cup \{\varphi_0, \varphi_1\}$. The formula $\varphi_0 \wedge \varphi_1$ is called *active at* v_0 and both φ_0 and φ_1 *residual at* v_1.

($\vee$) $x_0 = x_1$ and there are formulæ φ_0, φ_1 such that $\varphi_0 \vee \varphi_1 \in \Gamma_0$ and $\Gamma_1 = (\Gamma_0 \setminus \{\varphi_0 \vee \varphi_1\}) \cup \{\varphi_i\}$. The formula $\varphi_0 \vee \varphi_1$ is called *active at* v_0 and φ_i *residual at* v_1.

(σX) $x_0 = x_1$ and there is a formula φ and $\sigma \in \{\mu, \nu\}$ such that $\sigma X\varphi \in \Gamma_0$ and $\Gamma_1 = (\Gamma_0 \setminus \{\sigma X\varphi\}) \cup \{\varphi[\sigma X\varphi/X]\}$. The formula $\sigma X\varphi$ is called *active at* v_0 and $\varphi(\sigma X\varphi)$ *residual at* v_1.

(mod) $x_0 \to_T x_1$ and $\Gamma_0 = \Box\Xi, \Diamond\Delta, \Theta$ with $\Theta \subseteq \textsc{Prop} \cup \overline{\textsc{Prop}}$ and $\Xi \subseteq \Gamma_1 \subseteq \Xi \cup \Delta$. All formulæ in Γ_0 are considered *active at* v_0 and all formulæ in Γ_1 *residual at* v_1.

In the cases ($\wedge$), ($\vee$) and (σX) above, $|\{u : v_0 \to u\}| = 1$, while in the case of (mod), $\bigcup_{v_0 \to u} \lambda_t(u) = \Xi \cup \Delta$ and $\{\tau_t(u) : v_0 \to u\} = \{y : x_0 \to_T y\}$.

▶ Remark 3.2. Exactly one of the four conditions ($\wedge$), ($\vee$), (σX) and (mod) can apply to a non-leaf node of a pre-tableau; henceforth we will refer to them as *tableaux rules*. Note that in a pre-tableau branching only occurs at a (mod)-rule and may be infinite.

Suppose t is a pre-tableau for (T, Γ) and $\Psi = \{(\psi_i, v_i) : i \in I\} \subseteq \mathrm{FL}(\Gamma) \times t$ where I is an initial segment of natural numbers. Ψ is called a *trace* from (ψ, v) if $(\psi_0, v_0) = (\psi, v)$ and there exists a path $\mathbb{P}$ in t and natural number n such that for every $i \in I$,

- $v_i = \mathbb{P}(n + i)$,
- $\psi_i \in \lambda_t(v_i)$,
- if v_i is a leaf or $\psi_i \in \textsc{Prop} \cup \overline{\textsc{Prop}}$ is active at v_i then $i + 1 \notin I$,
- if $i + 1 \in I$ and ψ_i is active at v_i then ψ_{i+1} is an immediate subformula of ψ_i that is residual at v_{i+1},
- if $i + 1 \in I$ and ψ_i is not active at v_i then $\psi_{i+1} = \psi_i$.

In each infinite trace (i.e. if I is infinite) there exists a variable that appears infinitely often and subsumes all other infinitely occurring variables. If this unique variable is a μ-variable then the trace is called a *μ-trace*; otherwise it is a *ν-trace*.

▶ Definition 3.3. A pre-tableau for (T, Γ) is a *tableau* if every infinite trace is a ν-trace.

The following theorem which provides a characterisation of satisfaction in terms of the existence of tableaux is folklore; see for example [7].

▶ Theorem 3.4. $T \models \bigwedge \Gamma$ if and only if there is a tableau for (T, Γ).

3.2 Order-types of tableaux

Fix a TTS T and a sequent Γ. To each tableau for (T, Γ) and set of μ-variables $\mathcal{X}$ one can assign an order-type with respect to $\mathcal{X}$ in a natural way. The *order-type of ψ at a node v*, denoted by $\alpha_{\psi,v,\mathcal{X}}$, is defined recursively as follows. If there exists a trace $\Psi = \{(\psi_i, v_i) : i \in I\}$ from (ψ, v) such that for infinitely many $i \in I$, ψ_i has the form $\mu X\psi'$ for some $X \in \mathcal{X}$, or there are no traces $\Psi = \{(\psi_i, v_i) : i \in I\}$ from (ψ, v) for which ψ_i has the form $\mu X\psi'$ for some $i \in I$ and $X \in \mathcal{X}$, then $\alpha_{\psi,v,\mathcal{X}} = 0$. Otherwise,

- if $\psi = \mu X\psi'$ is active at v and $X \in \mathcal{X}$ then $\alpha_{\psi,v,\mathcal{X}} = \alpha_{\psi',u,\mathcal{X}} + 1$ where u is the unique successor of v in the tableau,
- if ψ is not of the form $\mu X\psi'$ for some $X \in \mathcal{X}$ or not active at v then $\alpha_{\psi,v,\mathcal{X}}$ is the supremum of $\alpha_{\psi_1,v_1,\mathcal{X}}$ for which there exists a trace $\Psi = \{(\psi_i, v_i) : i \in I\}$ from (ψ, v).

▶ Definition 3.5. The *order-type with respect to $\mathcal{X}$* of a tableau t for (T, Γ) is the ordinal $\sup\{\alpha_{\varphi,\rho_t,\mathcal{X}} : \varphi \in \Gamma\}$. A tableau is an *$\alpha$-tableau with respect to $\mathcal{X}$* if its order-type with respect to $\mathcal{X}$ is no greater than α.

To establish the connection between the closure ordinal of a formula and order-types of the corresponding tableaux we show that if φ is alternation-free and $\mathcal{X}$ a set of μ-variables,

$$x \in \|\varphi^\alpha\|^T_{\mathcal{V}} \text{ iff there exists an } \alpha\text{-tableau for } (T\restriction_x, \varphi) \text{ with respect to } \mathcal{X}.$$

We will prove the result for $\mathcal{X} = \{X\}$; the above statement is a direct generalisation of the next lemma.

▶ Lemma 3.6. Suppose $\psi(Y)$ is a formula with at most Y free and X a variable not occurring in ψ. Let $\mathcal{X} = \{X\}$ and T be a TTS. Then $x \in \|\psi(\mu X\varphi)^\alpha\|^T_{\mathcal{V}}$ if and only if there exists an α-tableau for $\big(T\restriction_x, \psi(\mu X\varphi)\big)$ with respect to $\mathcal{X}$.

Proof. By transfinite induction on α. For the base case suppose $\alpha = 0$. We want to show

$$x \in \|\psi(Z)\|^T_{\mathcal{V}[Z\mapsto\emptyset]} \text{ iff there exists a 0-tableau } \big(T\restriction_x, \psi(\mu X\varphi)\big).$$

Notice $x \in \|\psi(Z)\|^T_{\mathcal{V}[Z\mapsto\emptyset]}$ if and only if there is a tableau for $\big(T\restriction_x, \psi(\bot)\big)$. Consider a tableau for $\big(T\restriction_x, \psi(\bot)\big)$. Since $\bot$ cannot appear in the label of any node, this tableau can be used to create a tableau for $\big(T\restriction_x, \psi(\mu X\varphi)\big)$ in a trivial way: replace $\bot$ by $\mu X\varphi$ at relevant positions. The order-type of the emerging tableau is 0 as $\mu X\varphi$ can never appear in any trace. Conversely, since a tableau of order-type 0 means the (μX)-rule is never applied, replacing occurrences of $\mu X\varphi$ by $\bot$ in a tableau for $\big(T\restriction_x, \psi(\mu X\varphi)\big)$ yields a tableau for $\psi(\bot)$.

For the successor case we want to show

$$x \in \|\psi(\mu X\varphi)^{\alpha+1}\|^T_{\mathcal{V}} \text{ iff there exists an } (\alpha+1)\text{-tableau } \big(T\restriction_x, \psi(\mu X\varphi)\big).$$

Note that $x \in \|\psi(\mu X\varphi)^{\alpha+1}\|^T_{\mathcal{V}}$ if and only if $x \in \|(\psi \circ \varphi)(\mu X\varphi)^\alpha\|^T_{\mathcal{V}}$, if and only if there exists an α-tableau for $\big(T\restriction_x, (\psi \circ \varphi)(\mu X\varphi)\big)$ by the induction hypothesis. Hence it suffices to show how to construct an $(\alpha+1)$-tableau for $\big(T\restriction_x, \psi(\mu X\varphi)\big)$ from an α-tableau for $\big(T\restriction_x, \psi \circ \varphi(\mu X\varphi)\big)$ and vice versa. Given an α-tableau t for $\big(T\restriction_x, \psi \circ \varphi(\mu X\varphi)\big)$, along every path look for the first node v with $\lambda_t(v) = \Gamma, \varphi(\mu X\varphi)$ for some Γ, and replace all occurrences of $\varphi(\mu X\varphi)$ by $\mu X\varphi$ in nodes $u \leq_t v$. The sequent at v has therefore become $\Gamma, \mu X\varphi$. Between v and its successors, insert a new node labelled by $\Gamma, \varphi(\mu X\varphi)$. The added transition is a valid (μX)-rule so the resulting tableau is readily seen to be a tableau for $\big(T\restriction_x, \psi(\mu X\varphi)\big)$. Moreover, all traces from $(\mu X\varphi, v)$ have order-type at most $\alpha + 1$ and indeed, the tableau

for $\big(T\restriction_x, \psi(\mu X\varphi)\big)$ has order-type $\alpha+1$. Similarly, by replacing occurrences of $\mu X\varphi$ by $\varphi(\mu X\varphi)$ at the relevant nodes in an $(\alpha+1)$-tableau for $\big(T\restriction_x, \psi(\mu X\varphi)\big)$ and removing the first application of a (μX)-rule on every trace one obtains an α-tableau for $\big(T\restriction_x, \psi\circ\varphi(\mu X\varphi)\big)$.

For the limit case suppose $x \in \|\psi(\mu X\varphi)^\alpha\|^T_{\mathcal{V}}$. Let q be a fresh proposition and T^q a new TTS obtained by adjusting the labelling so that q holds at all nodes belonging to $\|(\mu X\varphi)^\alpha\|^T_{\mathcal{V}}$ i.e.

$$\lambda_{T^q}(x) = \begin{cases} \lambda_T(x)\cup\{q\}, & \text{if } x \in \|(\mu X\varphi)^\alpha\|^T_{\mathcal{V}}, \\ \lambda_T(x), & \text{otherwise.} \end{cases}$$

Since $\|\psi(\mu X\varphi)^\alpha\|^T_{\mathcal{V}} = \|\psi(q)\|^{T^q}_{\mathcal{V}}$ and $\psi(q)$ is closed, there is a tableau t for $(T^q\restriction_x, \psi(q))$ of order-type 0. It is possible that there are nodes of this tableau at which q is active. The key to obtaining a tableau for $\big(T\restriction_x, \psi(\mu X\varphi)\big)$ lies in replacing the occurrences of q at these nodes by tableaux for $\mu X\varphi$ of the relevant order-type. Suppose $\lambda_t(v) = \Box\Gamma, \Diamond\Delta, \Theta, q$, $\tau_t(v) = y$, q is active at v and for no $u <_t v$ is q active at u. Let $\beta < \alpha$ be such that $y \in \|(\mu X\varphi)^\beta\|^T_{\mathcal{V}}$. By the main induction hypothesis there is a β-tableau for $(T\restriction_y, \mu X\varphi)$. We can combine this tableau with the sub-tableau $t\restriction_v$ to obtain a β-tableau t_v for $(T^q\restriction_y, \Box\Gamma, \Diamond\Delta, \Theta, \mu X\varphi)$. Now we replace $t\restriction_v$ by t_v in t, substitute each occurrence of q by $\mu X\varphi$ in the trace from the root to (q, v) and repeat the procedure. In the limit a tableau for $(T\restriction_x, \psi(\mu X\varphi))$ is obtained. Moreover, the order-type of this tableau can be no greater than α.

The converse direction is equally straight forward. ◀

▶ Corollary 3.7. Suppose φ is a closed formula and $\mathcal{X}$ a set of μ-variables occurring in φ. For an arbitrary TTS T, set α_T to be 0 if $T \not\models \varphi$, and otherwise the infimum of the order-types of all possible tableaux for (T, φ) with respect to $\mathcal{X}$. Then $CO_{\mathcal{X}}(\varphi) = \sup\{\alpha_T : T \text{ a TTS}\}$.

With corollary 3.7 in mind, in order to rule out certain ordinals being closure ordinals we require a notion of minimality of order-types for tableaux.

▶ Definition 3.8. A tableau t for (T, Γ) is *minimal* if there are no tableau for (T, Γ) with smaller order-type, and *absolutely minimal* if for every node $v \in t$, $t\restriction_v$ is a minimal tableau for $(T\restriction_{\tau_t(v)}, \lambda_t(v))$.

▶ Remark 3.9. If $T \models \varphi$ then a minimal tableau t for (T, φ) exists. Moreover, as $T\restriction_{\tau_t(v)} \models \bigwedge \lambda_t(v)$ for each $v \in t$, the existence of an absolutely minimal tableau for (T, φ) is also guaranteed.

As a refinement of lemma 3.6 for limit ordinals we have the following.

▶ Proposition 3.10. Suppose φ is a formula with closure ordinal $\omega.\alpha > 0$ with respect to a set $\mathcal{X}$ of μ-variables. Then there exists a TTS T and a minimal tableau for $(T, \Box\varphi)$ with order-type $\omega.\alpha$ with respect to $\mathcal{X}$.

Proof. By corollary 3.7, for every $\beta < \omega.\alpha$ there exists a TTS T_β such that every tableau for (T_β, φ) has order-type greater than β. Let T be the TTS obtained by extending the disjoint union of $\{T_\beta : \beta < \omega.\alpha\}$ by a fresh node ρ_T whose immediate successors are $\{\rho_{T_\beta} : \beta < \omega.\alpha\}$. As $T \models \Box\varphi$, there exists a tableau for $(T, \Box\varphi)$. Moreover, every minimal tableau for $(T, \Box\varphi)$ has order-type $\omega.\alpha$ with respect to $\mathcal{X}$. ◀

3.3 Closure ordinals for the alternation-free fragment

In this section we determine upper bounds on the closure ordinals of alternation-free formulæ. The analysis breaks into two parts. First we prove that if an alternation-free formula φ has closure ordinal strictly less than ω^2 with respect to its external μ-variables, then this ordinal

is bounded by $\omega.2^{2^{|\varphi|+2}}$. Although primary formulæ can yield ordinals arbitrary close to ω^2 (from below), in the second part we show that the closure ordinal of any alternation-free formula is strictly less than ω^2.

We need only consider order-types for tableaux with respect to particular classes of μ-variables. Given a formula φ, a set of variables $\mathcal{X}$ of φ is called *principal* if whenever $X \in \mathcal{X}$ appears within the scope of a quantifier σY in φ, also $Y \in \mathcal{X}$. Let $\mathcal{X}_\varphi$ denote the largest principal set containing only μ-variables of φ.

An *ordinal assignment* on a tree t is a function $o\colon t \to \textsc{On}$ such that if x, y are nodes in t and $x \leq_t y$ then $o(y) \leq o(x)$. A tableau t for (T, Γ) induces a natural ordinal assignment on itself, denoted o_t, setting $o_t(u) = \sup\{\alpha_{\psi,u,\mathcal{X}_\Gamma} : \psi \in \lambda_t(u)\}$ for every $u \in t$, where $\mathcal{X}_\Gamma = \bigcup_{\varphi\in\Gamma} \mathcal{X}_\varphi$. Furthermore, the same tableau induces an ordinal assignment on T, also denoted o_t, by defining $o_t(x) = \sup\{o_t(u) : u \in t \wedge \tau_t(u) = x\}$ for each $x \in T$. The *order-type of a tableau* t, denoted $o(t)$, is the ordinal $o_t(\rho_t)$. A tableau is an *α-tableau* if its order-type is no greater than α.

▶ Lemma 3.11. If $T \models \varphi$ is a TTS with an infinite path $x_1 <_T x_2 <_T \cdots$ then there exists k such that for every $\Gamma \subseteq \mathrm{FL}(\varphi)$, every absolutely minimal tableau t for (T, Γ) and every $l > k$, $o_t(x_l) = 0$.

Proof. Suppose the contrary, namely for every i there exists $\Gamma_i \subseteq \mathrm{FL}(\varphi)$ and absolutely minimal tableau t_i for (T, Γ_i) such that $o_{t_i}(x_i) > 0$. For each m and i, let $\Delta_i^m \subseteq \mathcal{P}(\mathrm{FL}(\varphi))$ be the collection of sequents that are associated with x_m by t_i,

$$\Delta_i^m = \{\Delta : \exists u \in t_i(\tau_{t_i}(u) = x_m \wedge \lambda_{t_i}(u) = \Delta)\}.$$

For each m, there exists an infinite set $I \subseteq \omega$ with $\Delta_i^m = \Delta_j^m$ for every $i, j \in I$. Thus it is possible to define a sequence $(S_m)_{n\in\omega}$ such that for each m,

1. S_m is an infinite set,
2. $S_{m+1} \subseteq S_m$,
3. for every $i, j \in S_m$, $\Delta_i^m = \Delta_j^m$.

As for each i the tableau t_i is absolutely minimal, we have in fact

$$\forall i, j \in S_m\, o_{t_i}(x_m) = o_{t_j}(x_m)$$

for every m. Let $f\colon \omega \to S_0$ be a strictly increasing function such that $f(m) \in S_m$ for every m and set $\alpha_m = o_{t_{f(m)}}(x_m)$. Then the sequence $(\alpha_m)_{m\in\omega}$ is a weakly decreasing sequence of ordinals as

$$\begin{aligned}
\alpha_{m+1} &= o_{t_{f(m+1)}}(x_{m+1}) \\
&\leq o_{t_{f(m+1)}}(x_m), && \text{since } x_m <_T x_{m+1}, \\
&= o_{t_{f(m)}}(x_m), && \text{since } S_{m+1} \subseteq S_m, \\
&= \alpha_m.
\end{aligned}$$

As $f(m) \geq m$, we also have that $\alpha_m = o_{t_{f(m)}}(x_m) \geq o_{t_{f(m)}}(x_{f(m)}) > 0$. Thus, the sequence $(\alpha_{m.|\varphi|})_{m\in\omega}$ forms an infinite, strictly decreasing sequence of ordinals. ◀

Given T, Γ and a non-empty collection S of tableaux for (T, Γ), we define the *S-pruning of T* to be the TTS T' that alters T by setting, for each propositional constant q not appearing in Γ, $q \notin \lambda_{T'}(x)$ iff for some $s \in S$ and all $y <_T x$, $o_s(y) > 0$. If S is the collection of all absolutely minimal tableaux for (T, Γ), we write $\Gamma \star T$ for the S-pruning of T.

▶ Lemma 3.12 (Well-foundedness lemma). If T is a TTS, Γ is a finite set of formulæ all satisfied by T and q is a propositional constant not occurring in Γ then $\{x \in \Gamma \star T : q \notin \lambda_{\Gamma\star T}(x)\}$ forms a well-founded initial sub-tree of T.

Proof. Immediate consequence of lemma 3.11. ◀

The next three lemmata relate tableaux on $\Gamma \star T$ and T. Let T be a TTS, Γ a sequent and q a propositional constant not occurring in Γ.

▶ Lemma 3.13. If $y \in T$ and $o_s(y) \leq \alpha$ for every absolutely minimal tableau s for (T, Γ) then the set $\{x \in \Gamma \star T : q \notin \lambda_{\Gamma\star T}(x) \wedge y \leq_T x\}$ forms a well-founded tree of order-type no greater than $|\Gamma|.(1 + \alpha)$.

Proof. By transfinite induction on α. Notice that if $\tau_s(u) = y$ and $o_s(u) > 0$ then every trace in $s\restriction_u$ must pass through a (μX)-rule for which $\mu X\varphi$ is active, within the first $|\Gamma|$ occurrences of a (mod)-rule. ◀

▶ Lemma 3.14. If $\{x \in \Gamma \star T : q \notin \lambda_{\Gamma\star T}(x) \wedge y \leq_T x\}$ forms a non-empty (well-founded) tree of order-type $\omega.\alpha$ then

1. for every $\Delta \subseteq \Gamma$ and every absolutely minimal tableau t for (T, Δ), $o_t(y) \leq \omega.\alpha$,
2. there exists an absolutely minimal tableau s for (T, Γ) such that $o_s(y) = \omega.\alpha$.

Proof. 1 can be proved via transfinite induction, noting that since Γ is a set of guarded formulæ, between any two applications of the (σY)-rule on the same trace, the (mod)-rule must have been applied.

We prove 2. Suppose, in search of a contradiction, that for every absolutely minimal tableau s for (T, Γ), $o_s(y) < \omega.\alpha$. Consider the ordinal

$$\delta = \sup\{o_s(y) : s \text{ is an absolutely minimal tableau for } (T, \Gamma)\}.$$

By lemma 3.13 it must be the case that $\delta = \omega.\alpha$. But then for every $\beta < \alpha$ there exists an absolutely minimal tableau s for (T, Γ) such that $\beta < o_s(y) < \delta$; contradiction. ◀

For a formula $\varphi \in \Gamma$, let φ_q denote the formula resulting from replacing in φ each $X \in \mathcal{X}_\varphi$ by $\bar{q} \wedge X$, and set $\Gamma_q = \{\varphi_q : \varphi \in \Gamma\}$.

▶ Lemma 3.15. There exists an α-tableau for (T, Γ) iff there is an α-tableau for $(\Gamma \star T, \Gamma_q)$.

Proof. Suppose t is an α-tableau for (T, Γ). Then there exists an absolutely minimal tableau t' for (T, Γ) with $o(t') \leq \alpha$. An $o(t')$-tableau for $(\Gamma \star T, \Gamma_q)$ can be readily constructed from t'. For the converse, let t be a tableau for $(\Gamma \star T, \Gamma_q)$. By the definition of Γ_q, it follows that if $o_t(y) > 0$ then $y \in \{x \in \Gamma \star T : q \notin \lambda_{\Gamma\star T}(x)\}$ whence t can be modified to yield a tableau for (T, Γ) with the same order-type. ◀

Lemma 3.15 together with lemma 3.14 provide immediate upper bounds on the order-types of sub-tableaux for $(\Gamma \star T, \Gamma_q)$. We can now expand on these properties to obtain a more fine-grained version of lemma 3.14.

If B is a collection of nodes in a tableau s, ee write $v \leq_s B$ if for some $u \in B$, $v \leq_s u$. Let s be an arbitrary tableau, s_0 a pruning of s and suppose $A \subseteq s_0$ is the collection of leaves of s_0 that are inner nodes of s. A *filter over* (s, s_0) is a set $B \subseteq A$ such that for every $v \leq_s B$ if $\{u : v \to_s u \text{ and } u \leq_s A\}$ is infinite, so is $\{u : v \to_s u \text{ and } u \leq_s B\}$. An *ordinal for the filter* B is any α such that for every $v \leq_s B$, if $\{u : v \leq_s u \in A\}$ is infinite then for every $\beta < \alpha$ there is $w \in A$ such that $v \leq_s w$ and $\beta \leq o_s(w)$. It follows that for any tableau s and pruning s_0:

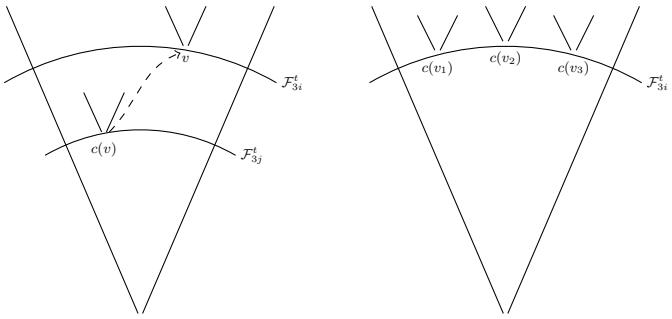


Figure 2 Tableaux t and $\hat{t}$ in the proof of lemma 3.18.

▶ Lemma 3.16. If $o(s) < \alpha + o(s_0)$ then there is no filter over s with ordinal $\alpha + \omega$.

▶ Lemma 3.17. If every ordinal for every filter over s is bounded by α, then $o(s) \leq \alpha + o(s_0)$.

Proof. Both lemmata are proved by transfinite induction on $o(s)$. For the second lemma, notice that for $u \leq_s B$, if $o_{s_0}(u) = \omega.\beta$ and for every $v > u$, $o_{s_0}(v) < \omega.\beta$, then for $o_s(u) > \alpha + o_{s_0}(u)$ to be the case we must have $o_s(v) > \alpha + o_{s_0}(u)$ for some $v \geq_{s_0} u$. ◀

We are now ready to prove the core lemma.

▶ Lemma 3.18. Let $N = 2^{2^{|\varphi|+2}}$. If there is a minimal tableau for (T, φ) of order-type $\alpha \in [\omega.N, \omega^2)$ then there exists a TTS $\hat{T}$ and a minimal tableau for $(\hat{T}, \varphi)$ with order-type strictly greater than α.

Proof. Suppose $\alpha = \omega.m_1 + m_2$ and q is a constant not appearing in φ. Let $T' = \varphi \star T$. For each $i \leq m_1$ define

$$\mathcal{F}_i = \{y \in T' : \{x \in T : q \notin \lambda_{T'}(x) \wedge y \leq_T x\} \text{ is a tree of order-type } \omega.i\}.$$

Since there is a minimal tableau for (T, φ) of order-type $\alpha \geq \omega.N$, the set $\mathcal{F}_i$ is non-empty for every $i \leq N$. Moreover, by lemma 3.15 there exists a tableau for (T', φ_q) with order-type precisely α. Denote this tableau by t and set $\mathcal{F}^t_i = \{v \in t : \tau_t(v) \in \mathcal{F}_i\}$. Let

$$\mathbf{\Delta}_i = \{\Delta : \text{there exists } v \in \mathcal{F}^t_i \text{ and an } \omega.i\text{-tableau for } (T'\restriction_{\tau_t(v)}, \Delta_q)\}.$$

Notice $\mathbf{\Delta}_i$ is non-empty for each $0 < i \leq N$. Moreover, as $\mathbf{\Delta}_i \subseteq \mathcal{P}(\mathrm{FL}(\varphi))$ and $m_1 \geq N$, there exists $0 < i < j \leq m_1$ such that $\mathbf{\Delta}_{3i} = \mathbf{\Delta}_{3j}$ and $\mathbf{\Delta}_{3i-1} = \mathbf{\Delta}_{3j-1}$. To each $v \in \mathcal{F}^t_{3i}$ is therefore associated a node $c(v) \in \mathcal{F}^t_{3j}$ such that for every $\Delta \subseteq \mathrm{FL}(\varphi)$,

1. there is a tableau for $(T'\restriction_{\tau_t(v)}, \Delta_q)$ if and only if there is a tableau for $(T'\restriction_{\tau_t(c(v))}, \Delta_q)$,
2. there exists an $\omega.(3i-1)$-tableau for $(T'\restriction_{\tau_t(v)}, \Delta_q)$ if and only if there exists an $\omega.(3j-1)$-tableau for $(T'\restriction_{\tau_t(c(v))}, \Delta_q)$.

Let $\hat{t}$ be the tableau obtained from t by replacing each node $v \in \mathcal{F}^t_{3i}$ by $t\restriction_{c(v)}$. $\hat{t}$ is a tableau for $(\hat{T}, \varphi_q)$ where $\hat{T}$ is obtained from T' by replacing the sub-tree at each $\tau_t(v) \in \mathcal{F}_{3i}$ by $T'\restriction_{\tau_t(c(v))}$. Denote by A the set of nodes of $\hat{t}$ corresponding to this change.

Let $\hat{s}$ be an absolutely minimal tableau for $(\hat{T}, \varphi)$ and $\hat{A} = \{u \in \hat{s} : \exists v \in A\ \tau_{\hat{s}}(u) = \tau_{\hat{t}}(v)\}$. It suffices to prove that $o(\hat{s}) > \alpha = \omega.m_1 + m_2$. Since $(\mathbf{\Delta}_{3i}, \mathbf{\Delta}_{3i-1}) = (\mathbf{\Delta}_{3j}, \mathbf{\Delta}_{3j-1})$, lemma 3.14 implies that for every $u \in \hat{A}$ there is a tableau, say t_u, for $(T'\restriction_{\tau_{\hat{s}}(u)}, \lambda_{\hat{s}}(u))$ with $o(t_u) \leq \omega.3i$, and $o(t_u) \leq \omega.(3i-1)$ if $o_{\hat{s}}(u) \leq \omega.(3j-1)$. From $\hat{s}$ we define a new tableau

s for (T', φ_q) replacing the sub-tableau $\hat{s}\restriction_u$ by t_u for each $u \in \hat{A}$. We remark that s and $\hat{s}$ have a common initial part, namely the pruning $s_0 = s \cap \{v : v \leq_s \mathcal{F}_{3i}\}$.

Assume $o(\hat{s}) \leq \alpha$. Every ordinal for a filter over (s, s_0) is no greater than $\omega.3i$ by lemma 3.14, so by lemma 3.17, $o(s_0) \geq \omega.(m_1 - 3i) + m_2$. Notice also that $o(s_0) < \omega.(m_1 - 3i) + \omega$. But then $o(\hat{s}) \leq \omega.m_1 + m_2 < \omega.(3i+1) + o(s_0)$ and lemma 3.16 implies that every ordinal for a filter over $\hat{s}$ is strictly below $\omega.(3i+2)$. Since $3i + 2 \leq 3j - 1$, in forming s a sub-tableau of order-type $< \omega.(3i+2)$ at A is replaced by a tableau of order-type $\omega.(3i-1)$. Therefore every filter over (s, s_0) has ordinal $\leq \omega.(3i-1)$, whence

$$\begin{aligned} o(s) &\leq \omega.(3i-1) + o(s_0) \\ &< \omega.(3i-1) + \omega.(m_1 - 3i) + \omega \leq \alpha \end{aligned}$$

Thus by lemma 3.15 there exists a tableau for (T, φ) with order-type $\beta < \alpha$, yielding a contradiction. ◀

▶ Corollary 3.19. Let φ be a closed formula of alternation-free μ-calculus. If φ has closure ordinal $\alpha < \omega^2$ with respect to $\mathcal{X}_\varphi$, in fact $\alpha < \omega.N$ where $N = 2^{2^{|\varphi|+2}}$.

Proof. Suppose $CO_{\mathcal{X}_\varphi}(\varphi) = \alpha \in [\omega.N, \omega^2)$. Proposition 3.10 implies the existence of a TTS T and an absolutely minimal tableau t for $(T, \Box\varphi)$ with order-type α. By lemma 3.18 there exists a TTS $\hat{T} \models \Box\varphi$ and a minimal tableau $\hat{s}$ for $(\hat{T}, \Box\varphi)$ with order-type greater than α, whence lemma 3.6 implies $CO_{\mathcal{X}_\varphi}(\varphi) \geq CO_{\hat{T}, \mathcal{X}_\varphi}(\varphi) > \alpha$. ◀

It remains to rule out closure ordinals of ω^2 or greater. To achieve this a more general version of the argument in the preceding proof is required.

▶ Lemma 3.20. If t is a minimal tableau for (T, φ) and $o(t) \geq \omega^2$, then there exists a TTS $\hat{T}$ and a minimal tableau for $(\hat{T}, \varphi)$ with order-type strictly greater than $o(t)$.

Proof. Suppose t is a minimal tableau for (T, φ) and $\omega^2 \leq \omega.\alpha_t \leq o(t) < \omega.(\alpha_t + 1)$. Set $T_0 = \varphi \star T$. Let q not appear in φ and for each $k < \omega$ let the set $\mathcal{F}_k$ be defined analogously to the previous lemma as the collection of nodes in $\varphi \star T$ such that the sub-tree $\{x \in T_0 : q \notin \lambda_{T_0}(y) \wedge y \leq_T x\}$ has order-type $\omega.k$. Now $\mathcal{F}_k$ is non-empty for every $k < \omega$, so there exist infinitely many indices, $0 < i < j(1) < j(2) < \ldots$ such that $j(n+1) \geq j(n) + 2$ and $(\boldsymbol{\Delta}_i, \boldsymbol{\Delta}_{i-1}) = (\boldsymbol{\Delta}_{j(n)}, \boldsymbol{\Delta}_{j(n)-1})$ for every n. Let $c_m \colon \mathcal{F}_i \to \mathcal{F}_{j(m)}$ be the function such that for each $x \in \mathcal{F}_i$, $\Delta \subseteq \mathrm{FL}(\varphi)$ and every $m < \omega$,

1. there is a tableau for $(T_0\restriction_x, \Delta_q)$ if and only if there is a tableau for $(T_0\restriction_{c_m(x)}, \Delta_q)$,
2. there is a tableau for $(T_0\restriction_x, \Delta_q)$ with order-type $\omega.(i-1)$ if and only if there is a tableau for $(T_0\restriction_{c_m(x)}, \Delta_q)$ with order-type $\omega.(j(m)-1)$.

Beginning with c_m, one can define iterated versions, c_m^α for each α: for $i \in \mathcal{F}_k$ with $k \geq i$, $c_m^0(x) = T_0\restriction_x$ and $c_m^1(x)$ is defined to be the result of replacing in $T_0\restriction_x$ each node $y \in \mathcal{F}_i$ by the tree $c_m(y)$; $c_m^{\alpha+1}(x)$ is the tree $c_m^1(x)$ in which each node $y \in \mathcal{F}_i$ is replaced by $c_m^\alpha(y)$; for a limit ordinal α, $c_m^\alpha(x)$ is the tree $c_m^1(x)$ in which, given a bijection $g_0 \colon \mathcal{F}_i \to \omega$,

- if $\alpha = \omega.\gamma + \omega$ then each node $y \in \mathcal{F}_i$ is replaced by the tree $c_{g_0(y)}^{\omega.\gamma + g_0(y)}(y)$,
- if $\alpha = \omega.\alpha_0$, α_0 is a limit ordinal and $g_1 \colon \mathcal{F}_i \to \alpha_0$ is a bijection, then each node $y \in \mathcal{F}_i$ is replaced by the tree $c_{g_0(y)}^{\omega.g_1(y)}(y)$.

The following two lemmata are obtained by generalising the argument in lemma 3.18 making essential use of lemmata 3.16 and 3.17.

▶ Sub-lemma 1. There exists a tableau for $(c_m^\alpha(x), \Delta_q)$ if and only if there exists a tableau for $(T_0\restriction_{c_m(x)}, \Delta_q)$.

▶ Sub-lemma 2. If $x \in \mathcal{F}_i$ and there exists a tableau for $(c^{\alpha}_m(x), \Delta_q)$ with order-type $< \omega.\alpha$ then there exists an $\omega.(i-1)$-tableau for $(T_0\restriction_x, \Delta_q)$.

The construction of the trees $c^{\alpha}_m(x)$ and the two previous sub-lemmata suffice to prove the main lemma. By lemma 3.15, t naturally induces an absolutely minimal tableau for (T_0, φ_q) of the same order-type. Let $T_0^{\alpha_t}$ be the tree obtained by replacing each sub-tree $T_0\restriction_y$ for $y \in \mathcal{F}_i$ by $c_i^{\alpha_t+\omega}(y)$. It is easy to see that $T_0^{\alpha_t} \models \varphi \wedge \varphi_q$.

Let $\hat{s}$ be an arbitrary absolutely minimal tableau for $(T_0^{\alpha_t}, \varphi_q)$ and s the collapse of $\hat{s}$ to a tableau for (T_0, φ_q): on each path replace the first $v \in \hat{s}$ such that $T_0^{\alpha_t}\restriction_{\tau_{\hat{s}}(v)} = c_i^{\alpha_t+\omega}(y)$ for some $y \in \mathcal{F}_i$ by the tableau for $(T_0\restriction_y, \lambda_{\hat{s}}(v))$ given by sub-lemma 2, if $o_{\hat{s}}(v) < \omega.\alpha_t + \omega$, and by sub-lemma 1 otherwise. Let S_0 denote the collection of absolutely minimal tableaux for T_0, and set S_0' to be the collection of tableaux for (T_0, φ_q) that arise as the collapse, in the manner described above, of an absolutely minimal tableau for $(T_0^{\alpha_t}, \varphi_q)$. If there is a minimal tableau for $(T_0^{\alpha_t}, \varphi_q)$ with order-type strictly greater than $o(t)$ then we are done. Otherwise, for every $r' \in S_0'$ there exists $r \in S_0$ such that for all x, if $o_{r'}(x) = \omega.i$ then $o_r(x) > \omega.i$. Now set T_1 to be the S_0'-pruning of T_0. T_1 has the same domain as T_0 and hence T. Moreover, if $\{z \in T_1 : q \notin \lambda_{T_1}(z) \wedge x \leq_T z\}$ has order-type $\omega.i$ then there exists $x <_T y$ such that the tree $\{z \in T_0 : q \notin \lambda_{T_0}(z) \wedge y \leq_T z\}$ has order-type $\omega.i$. Let the set S_1 comprise all absolutely minimal tableaux for (T_1, φ_q). Any $r \in S_1$ is also a tableau for (T_0, φ_q) and hence also for (T, φ). Thus consider tableaux for $(T_1^{\alpha_t}, \varphi_q)$ and set S_1' to be the collection of tableaux that are obtained from the collapse of absolutely minimal tableaux for $(T_1^{\alpha_t}, \varphi_q)$. Define S_2 to be the set of absolutely minimal tableaux for the S_1'-pruning of T_1. Similarly define S_3, S_4, etc. Every tableau in S_{n+1} "moves" the $\omega.i$-frontier of T closer to the root. Thus, either for some n there exists a minimal tableau for $(T_n^{\alpha_t}, \varphi_q)$ with order-type strictly greater than $o(t)$, or for every n there exists $x \in T$ and tableau $s_j \in S_j$ for every $j \leq n$ such that $o_{s_j}(x) < o_{s_{j+1}}(x)$. As the latter will yield a contradiction, we are done. ◀

As a consequence of lemmata 3.18 and 3.20 the closure ordinals of μ-formulæ will be sufficiently bounded.

▶ Theorem 3.21. Let $\mathcal{X}$ be a principal set of μ-variables for a closed and alternation-free formula φ. Then the closure ordinal of φ with respect to $\mathcal{X}$, if it exists, is strictly less than $\omega.2^{2^{|\varphi|+2}}$.

▶ Corollary 3.22. Suppose φ is a closed formula in the alternation-free fragment of the μ-calculus and $\mathcal{X}$ is a principal set of ν-variables only. Then $CO_{\mathcal{X}}(\varphi) < \omega.2^{2^{|\varphi|+2}}$ if the former ordinal exists.

Proof. Let $\bar{\varphi}$ denote the dual of φ and let $\mathcal{X}$ be a set of ν-variables principal in φ. That $CO_{\mathcal{X}}(\varphi) = CO_{\mathcal{X}}(\bar{\varphi})$ follows from the dual semantics of the μ-calculus, whence theorem 3.21 implies $CO_{\mathcal{X}}(\varphi) < \omega.2^{2^{|\varphi|+2}}$. ◀

▶ Theorem 3.23. Let φ be a closed alternation-free formula in guarded form and let $\mathcal{X}$ be the set of variables occurring in φ. If $CO_{\mathcal{X}}(\varphi)$ exists then $CO_{\mathcal{X}}(\varphi) < \omega^2$.

Proof sketch. Suppose $\varphi \in \Sigma_{n+1}$ in the weak hierarchy has closure ordinal κ with respect to the set of all variables in φ. By theorem 3.21 all μ-variables that do not appear under the scope of a ν-variable close off at some ordinal $\alpha < \omega^2$. Moreover, the structure of φ will induce, for each closed weak Π_n sub-formula ψ, a particular class of transition systems, say $\mathcal{T}$, such that ψ has closure ordinal κ with respect to trees in $\mathcal{T}$. In the case $n = 1$, by relativising the previous arguments to the class $\mathcal{T}$, one may deduce ψ has closure ordinal,

say α_ψ, strictly less than ω^2 with respect to $\mathcal{T}$. As the closure ordinal of φ is no greater than the sum of α and ordinals α_ψ, $CO_{\mathcal{X}}(\varphi) < \omega^2$. ◀

A profound consequence of lemma 3.20 and corollary 3.22 and one that also applies to theorem 3.23, is that there is no essential dependency between closure ordinals and alternation depth for the alternation-free fragment: the choice of N in these results depends only on the logical complexity of φ and the dependency on the alternation depth of φ is essentially trivial, necessitating a smaller increase in bounds than for the connectives and quantifiers. Whether this remains the case for formulæ outside the alternation-free fragment is unclear.

Acknowledgements. The authors' research was supported by the Engineering and Physical Sciences Research Council UK (EP/G012962/1 & EP/F036361/1) and the Arts and Humanities Research Council UK (AH/H039791/1) respectively. The authors also wish to thank the anonymous referees for their insightful comments.

References

1 J.C. Bradfield. The modal mu-calculus alternation hierarchy is strict. *Theoretical Computer Science* 195(2)(1998), 133–153.

2 J.C. Bradfield and C. Stirling. Modal mu-calculi. In: P. Blackburn, J. Van Benthem and F. Wolter eds. *Handbook of Modal Logic*, Studies in Logic and Practical Reasoning 3, Elsevier, 721–756, 2006.

3 M. Czarnecki. How fast can the fixpoints in modal μ-calculus be reached? In: L. Santocanale, ed. *Fixed Points in Computer Science 2010, Brno, August 2010*, 35–39, 2010.

4 O. Friedmann and M. Lange. The modal μ-calculus caught off guard. In: K. Brünnler and G. Metcalfe eds. *Proceedings of the 20th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, Bern, July 2011*, LNCS 6793, 149–163, 2011.

5 O. Kupferman and M.Y. Vardi. $\Pi_2 \cap \Sigma_2 \equiv$ AFMC. In: J.C.M. Baeten, J.K. Lenstra, J. Parrow and G.J. Woeginger, eds. *Proceedings of the 30th International Colloquium on Automata, Languages and Programming, Eindhoven, July 2003*, LNCS 2719, 697–713, 2003.

6 D.E. Muller, A. Saoudi and P.E. Schupp. Alternating automata, the weak monadic theory of the tree, and its complexity. In: L. Kott, ed. *Proceedings of the 13th International Colloquium on Automata, Languages and Programming, Rennes, July 1986*, LNCS 226, 275–283, 1986.

7 D. Niwiński and I. Walukiewicz. Games for the μ-calculus. *Theoretical Computer Science* 163(1–2)(1996), 99–116.

Realizability and Strong Normalization for a Curry-Howard Interpretation of HA + EM1

Federico Aschieri[*1], **Stefano Berardi**[2], **and Giovanni Birolo**[3]

1 **Laboratoire de l'Informatique du Parallélisme (UMR 5668, CNRS, UCBL) École Normale Supérieure de Lyon – Université de Lyon, France**
2 **Dipartimento di Informatica, Università di Torino, Italy**
3 **Dipartimento di Matematica, Università di Torino, Italy**

Abstract

We present a new Curry-Howard correspondence for $\mathsf{HA} + \mathsf{EM}_1$, constructive Heyting Arithmetic with the excluded middle on Σ^0_1-formulas. We add to the lambda calculus an operator $\|_a$ which represents, from the viewpoint of programming, an exception operator with a delimited scope, and from the viewpoint of logic, a restricted version of the excluded middle. We motivate the restriction of the excluded middle by its use in proof mining; we introduce new techniques to prove strong normalization for $\mathsf{HA} + \mathsf{EM}_1$ and the witness property for simply existential statements. One may consider our results as an application of the ideas of Interactive realizability, which we have adapted to the new setting and used to prove our main theorems.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Interactive realizability, classical Arithmetic, witness extraction, delimited exceptions

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.45

1 Introduction

From the beginning of proof theory many results have been obtained which clearly show that classical proofs have a constructive content. The seminal results are Hilbert's epsilon substitution method (see e.g [23]) and Gentzen's cut elimination [12]. Then, several other techniques have been introduced: among them, Gödel's double negation translation followed either by the Gödel functional interpretation [11] or Kreisel's modified realizability [18] and Friedman's translation [10]; the Curry-Howard correspondence between natural deduction and programming languages (see e.g. [27]).

In this paper we follow the Curry-Howard line of research. But what does it mean to extract constructive content from a natural deduction proof? Essentially, it means interpreting the positive connectives $\vee, \exists$ *as positively as possible*, that is, recovering information about truth as much as possible. The problem is that, even in intuitionistic Arithmetic, a disjunction $A \vee B$ can be proven without explicitly proving A or proving B; a proof of an existential statement $\exists \alpha^{\mathbb{N}} A$ may be accepted even if it does not directly provide a witness, i.e. a number n and a proof that $A[n/\alpha]$ holds. It is the very shape of the natural deduction rules that allows that: there are not only inference rules for direct arguments – *introduction* rules – but also indirect elimination rules. One can then prove a disjunction by an elimination rule, for

* This work was supported by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR)

example as a consequence of a general inductive argument for a formula $\forall\alpha^{\mathbb{N}}.\, A(\alpha) \vee B(\alpha)$ and then conclude $A(0) \vee B(0)$. It is a remarkable result of proof theory that it is possible to give a complete simple classification of the *detours* that can occur in an *intuitionistic* arithmetical proof, which are small pieces of indirect reasoning that can be readily eliminated through a simple proof transformation. Once this detours are eliminated, one obtains direct proofs of disjunctions or existential statements (see Prawitz [26]).

For classical Arithmetic, the situation may appear desperate: the double negation elimination rule $\neg\neg A \to A$ is a so indirect way of arguing, that seems impossible to be eliminated; the excluded middle $A \vee \neg A$ allows a disjunction to be asserted without having the slightest idea of which side holds. Indeed, for a long time, there has not been a set of reduction rules, nor a notion of classical detour, that worked for proofs containing *all* the logical connectives. It was Griffin [15] who gave a very elegant reduction rule for eliminating the double negation elimination. If A is concluded from $\neg\neg A$ and then used to prove $\bot$, then one can capture the part of the proof that surrounds A to obtain a proof of $\neg A$ and give it to the premiss $\neg\neg A$ in order to get a more direct proof of $\bot$. While this idea was initially applied only to negative fragments of Arithmetic, it became clear that it could be adapted even to a full set of connectives.

It was in that way that control operators entered the scene. The proof reductions for classical Arithmetic can be implemented by a Curry-Howard correspondence between proofs and functional languages enriched with operators that can capture the computational context. Several languages have been put forward for that aim. Griffin proposed the lambda calculus plus $\mathsf{call/cc}$, solution that has been developed and extended by Krivine [21, 22] with remarkable success. Parigot [25] put forward the $\lambda\mu$-calculus, which enjoys many of the nice properties of the lambda calculus that are instead lost when using $\mathsf{call/cc}$; de Groote [14] extended the $\lambda\mu$-calculus in order to interpret primitively all the logical connectives.

After these works, it became evident that enriching functional languages with other "less pure" computational constructs would allow to implement reduction rules for many mathematical axioms. For example, Krivine used the instruction quote to provide computational content to the axiom of dependent choice. Recently, Herbelin [16] has used the mechanism of delimited exceptions to give special reduction rules for Markov's principle.

The goal of this paper is to use a new combination of known computational constructs in order to interpret Heyting Arithmetic HA with the excluded middle schema EM_1, $\forall\alpha^{\mathbb{N}}\mathsf{P} \vee \exists\alpha^{\mathbb{N}}\mathsf{P}^{\bot}$, where P is any atomic decidable predicate (see [1]) and $\mathsf{P}^{\bot}$ denotes the atomic decidable predicate which is its complement. We shall give new reduction rules for $\mathsf{HA} + \mathsf{EM}_1$, and introduce a realizability semantics in order to investigate, describe and prove properties of their behavior. We shall use delimited exceptions, and permutative conversions for disjunction elimination. Permutative rules were introduced by Prawitz (see [26]) to obtain the subformula property in first-order natural deductions: in our framework, they will naturally express control operators. Delimited exceptions were used by de Groote [13] in order to interpret the excluded middle in classical propositional logic with implication; by Herbelin [16], in order to pass witnesses to some existential formula when a falsification of its negation is encountered. We shall use exceptions in a similar way, and our work may be seen as a modification and extension of some of de Groote's and Herbelin's techniques. Our reduction rules for the classical principle EM_1 are inspired by Interactive realizability [2, 3] for $\mathsf{HA} + \mathsf{EM}_1$, which describes classical programs as programs that *make hypotheses*, *test them* and *learn by refuting* the incorrect ones. The interest of EM_1 lies in the fact that this classical principle is logically simple, yet it may formalize many classical proofs: for instance, proofs of Euclidean geometry (like Sylvester conjecture, see J. von Plato [28]), of Algebra (like Dickson's Lemma,

see S. Berardi [7]) and of Analysis (those using Koenig's Lemma, see Kohlenbach [17]).

We now give an high level explanation of our contributions and of how they compare to other interpretations of classical proofs.

1.1 Excluded Middle versus Double Negation Elimination

As we have said, control operators have been mainly used to interpret primitively double negation elimination, or some related principle (as the Pierce law: $(\neg A \to A) \to A$). To interpret the excluded middle with this approach, one first proves intuitionistically $\bot$ (and thus EM) from $\neg\mathsf{EM}$ and then applies the rules of double negation elimination or Pierce law to obtain a proof term for EM. In this way, however, one does not address directly the excluded middle and sticks to an implicit negative translation which eliminates it. But what is classical logic if not the conception that formulas speak about models, and a formula is either true or false? It is also evident that the real idea behind the constructivization of classical logic is concealed in the proof of $\neg\neg\mathsf{EM}$: it is there that it is really determined *what is the use* of the continuations produced by control operators and *why* it is needed.

In this paper, we give direct reduction rules for the excluded middle EM_1. We treat it as an elimination rule, as in [13] and in the actual mathematical practice:

$$\frac{\Gamma, a : \forall\alpha^{\mathtt{N}}\mathsf{P} \vdash u : C \qquad \Gamma, a : \exists\alpha^{\mathtt{N}}\mathsf{P}^{\bot} \vdash v : C}{\Gamma \vdash u \parallel_a v : C}$$

This inference is nothing but a familiar disjunction elimination rule, where the main premise EM_1 has been cut, since, being a classical axiom, it has no computational content in itself. The proof terms u, v are both kept as possible alternatives, since one is not able to decide which branch is going to be executed at the end. A problem thus arises when C is employed as the main premise of an elimination rule to obtain some new conclusion. For example, when $C = A \to B$, and $\Gamma \vdash w : A$, one may form the proof term $(u \parallel_a v)w$ of type B. In this case, one may not be able to solve the dilemma of choosing between u and v, and the computation may not evolve further: one is stuck.

1.2 Permutation Rules for EM_1

We solve the problem as in [13] by adding permutation rules, as usual with disjunction. For example, $(u \parallel_a v)w$ reduces to $uw \parallel_a vw$. In this way, one obtains two important results: first, one may explore *both* the possibilities, $\forall\alpha^{\mathtt{N}}\mathsf{P}$ is true or $\exists\alpha^{\mathtt{N}}\mathsf{P}^{\bot}$ is true, and evaluate uw and vw; second, one duplicates the applicative context $[\,]w$, which will be needed in case of backtracking from the branch uw to vw. If $C = A \wedge B$, one may form the proof term $\pi_0(u \parallel_a v)$, which reduces to $\pi_0 u \parallel_a \pi_0 v$, and has the effect of duplicating the context $\pi_0[\,]$. Similar standard considerations hold for the other connectives. Thus permutation rules act similarly to the rules for μ in the $\lambda\mu$-calculus, but are only used to *duplicate* step-by-step the context and produce implicitly the continuation. Anyway, $\parallel$ behaves like a control-like operator.

1.3 Delimited Exceptions

The reductions that we put forward for the new proof terms $u \parallel_a v$ are inspired by the informal idea of learning by making falsifiable hypotheses. When normalizing a term $u \parallel_a v$, we shall consider u as the *active branch*. The reason is that the hypothesis $\forall\alpha^{\mathtt{N}}\mathsf{P}$ has no computational content, and it is only a certificate *serving to guarantee the correctness*

of u. Therefore, one can "run" u making the hypothesis $\forall\alpha^{\mathtt{N}}\mathsf{P}$ without the risk that the computation will be blocked; on the contrary, the branch v cannot a priori be executed without that risk, because the hypothesis $\exists\alpha^{\mathtt{N}}\mathsf{P}^{\perp}$ has a computational content (a witness) that may be requested in order to go on with the computation. That does not mean that one is not free to first perform reductions inside v, but rather that one may not expect to necessarily get useful results in that branch.

The informal idea expressed by our reductions is thus to assume $\forall\alpha^{\mathtt{N}}\mathsf{P}$ and try to produce some proof of C out of u by reducing inside u. The crucial intuition – recurring again and again in proof theory – is that when C is a concrete statement, for example a simple existential formula, one actually needs only a finite number of instances of $\forall\alpha^{\mathtt{N}}\mathsf{P}$ to prove it. Whenever u needs the truth of an instance $\mathsf{P}[n/\alpha]$ of the assumption $\forall\alpha^{\mathtt{N}}\mathsf{P}$, it checks it, and if it is true, it replaces it by its canonical proof which is just a computation. If all instances $\mathsf{P}[n/\alpha]$ of $\forall\alpha^{\mathtt{N}}\mathsf{P}$ being checked are true, *and no assumption* $\forall\alpha^{\mathtt{N}}\mathsf{P}$ *is left* (this is the non-trivial part), then the normal form u' of u is *independent* from $\forall\alpha^{\mathtt{N}}\mathsf{P}$ and we found some $u' : C$. Remark that, in this case, we do not know whether $\forall\alpha^{\mathtt{N}}\mathsf{P}$ is true or false, because u only checked finitely many instances of it: all we do know is that the full hypothesis $\forall\alpha^{\mathtt{N}}\mathsf{P}$ is unnecessary in proving C. If instead some assumption of $\forall\alpha^{\mathtt{N}}\mathsf{P}$ is left in u we are stuck. There is only one way out of this impasse and can occur at any moment: u may find some instance $\mathsf{P}[n/\alpha]$ which is false, and thus refute the assumption $\forall\alpha^{\mathtt{N}}\mathsf{P}$. In this case the attempt of proving C from $\forall\alpha^{\mathtt{N}}\mathsf{P}$ fails, we obtain $\mathsf{P}^{\perp}[n/\alpha]$ and u *raises the exception* n; from the knowledge that $\mathsf{P}^{\perp}[n/\alpha]$ holds, a canonical proof term $\exists\alpha^{\mathtt{N}}\mathsf{P}^{\perp}$ is formed and passed to v: a proof term for C has now been obtained and it can be executed.

In order to implement those reductions we shall use constant terms of the form $\mathtt{H}^{\forall\alpha\mathsf{P}}$, whose task is to take a numeral n and reduce to $\mathtt{True}$ if $\mathsf{P}[n/\alpha]$ holds, otherwise raise an exception. We shall also use a constant $\mathtt{W}^{\exists\alpha\mathsf{P}^{\perp}}$ denoting some unknown proof term for $\exists\alpha^{\mathtt{N}}\mathsf{P}^{\perp}$, whose task is to *catch* the exception raised by $\mathtt{H}^{\forall\alpha\mathsf{P}}$. Actually, these terms will occur only through typing rules of the form

$$\Gamma, a : \forall\alpha^{\mathtt{N}}\mathsf{P} \vdash [a]\mathtt{H}^{\forall\alpha\mathsf{P}} : \forall\alpha^{\mathtt{N}}\mathsf{P} \qquad \Gamma, a : \exists\alpha^{\mathtt{N}}\mathsf{P}^{\perp} \vdash [a]\mathtt{W}^{\exists\alpha\mathsf{P}^{\perp}} : \exists\alpha^{\mathtt{N}}\mathsf{P}^{\perp}$$

where a is used just as a name of a communication channel for exceptions: if in u occurs a subterm of the form $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}n$, where the closed expression $\mathsf{P}[n/\alpha]$ is false, then $u \parallel_a v$ reduces to $v[a := n]$, which denotes the result of the replacement of $[a]\mathtt{W}^{\exists\alpha\mathsf{P}^{\perp}}$ in v with the proof term $(n, \mathtt{True})$. From the viewpoint of programming, that is a *delimited exception* mechanism (see de Groote [13] and Herbelin [16] for a comparison). The scope of an exception has the form $u \parallel_a v : C$, with u the "ordinary" part of the computation and v the "exceptional" part. As pointed out to us by H. Herbelin, the whole term $u \parallel_a v$ can also be expressed in a standard way by the constructs raise and try ... with ... in the CAML programming language.

1.4 Realizability and Prawitz Validity

We now have a set of detour conversions for $\mathsf{HA} + \mathsf{EM}_1$: which notion of construction does it determine? The normalization process, even in intuitionistic logic, tends to be obscure: while the local meaning of reduction steps is clear, the global behaviour of the procedure is harder to grasp. This is the reason why it is important to define proof-theoretic semantics, in particular those who have the task of explaining what is a construction in intuitionistic or classical sense. Realizability is one of those semantics. In analogy with the discussion in Prawitz [26] about validity, one may classify realizabilities in two groups: those who give priority to introduction rules and those who rather privilege elimination rules in order to give meaning to logical connectives.

Realizabilities based on introduction rules. In this case, one explains a logical constant in term of the construction given by an introduction rule for that constant. For example, a realizer of $A \wedge B$ is a pair made by a realizer of A and a realizer B; a realizer of $A \vee B$ contains either a realizer of A or a realizer of B together with an indication of which formula is realized. Of course, this approach tends to work with constructive logics, which have the disjunction and numerical existence properties. Prawitz's notion of validity and Kreisel modified realizability are witness to that. There is one exception: Interactive realizability [3, 4], which explains positive classical connectives with introduction rules thanks to the use of the concept of state of knowledge.

Realizability based on elimination rules. In this case, one describes the meaning of a logical constant in terms of "performability of operations" or in terms of what can be obtained by the elimination rules for that constant. This approach works very well for negative connectives, and in fact is not very different from the one given by introduction rules: but since it has a semantical flavor, it is usually the preferred one. At the time of Prawitz [26], it seemed impossible that this approach could work also for positive connectives, given the circularity involved in the elimination rules (in terms of logical complexity). It was only after Girard's reducibility [9], and the work of Krivine [19, 21], that the second order definition of $A \vee B$ as $\forall X.\,(A \to X) \to (B \to X) \to X$ has been exploited for defining a realizability based on elimination rules. While remarkable, this result makes classical realizabilities based on elimination rules equivalent to some negative translation, re-proposing at the semantical level the issue which is eliminated on the syntactical one. Indeed, all realizabilities proposed for languages based on control operators are equivalent to some negative translation [24] (not surprisingly, since these operators were originally devised to interpret directly double negation elimination).

In this paper, we shall present a classical realizability borrowing ideas from both groups. The treatment of negative logical constants will be à la Kreisel, while the positive ones will be treated à la Prawitz. In particular the set of realizers of $A \vee B$ and of $\exists \alpha^{\mathtt{N}} A$ will be constructed by an inductive definition whose base case is an introduction rule; the atomic realizers will represent proofs in "extended" Post systems. This gives, first, an adaptation of Interactive realizability to a language with exceptions and control operators; second, an extension of Prawitz's notion of validity to a system with classical principles. We find these achievements interesting in their own right, because of the semantical meaning of validity given by Prawitz [26]. It seems also that our approach is not equivalent in any straightforward sense to a negative translation, in line with our desire of interpreting positive connectives *as positively as possible.*

1.5 Witness Extraction and Strong Normalization

Thanks to realizability, we shall provide a new semantical proof of a *normal form result* syntactically proven by Birolo [8], expressing that any closed normal proof term whose type is a simply existential formula $\exists \alpha^{\mathtt{N}} \mathsf{P}$ provides a witness through the process sketched above (that is, one never gets stuck with simply existential formulas); and a new *strong normalization result*, proving that all reduction paths terminate into a normal form. We anticipate that in our calculus all the reduction strategies are allowed, therefore strong normalization is not the same thing as weak normalization, as for example in Krivine's realizability [19]. This freedom is desirable, because it avoids artificial programming constraints which complicate the writing of realizers.

We remark that we cannot prove the witness property for all existential statements of $\mathsf{HA} + \mathsf{EM}_1$. Indeed, using EM_1 we may prove paradoxical statements like the drinker principle

$\exists\alpha^{\mathbb{N}}\forall\beta^{\mathbb{N}}.\,\mathsf{P}(\alpha)\to\mathsf{P}(\beta)$, for P primitive recursive, but for some P there is no map computable in the parameters of P providing some n such that $\forall\beta^{\mathbb{N}}.\,\mathsf{P}(n)\to\mathsf{P}(\beta)$. However we prove the witness property for all Π^0_2-statements of $\mathsf{HA}+\mathsf{EM}_1$, which include all statements about convergence of algorithms, therefore all statements more interesting for Computer Science. The witness property we prove is a particular case of the witness property which holds for the entire classical arithmetic by the results of Gödel: the interest of our results lies in the new reduction set we provide and in their semantics.

1.6 Non-Determinism

We anticipate that our set of reductions is non-deterministic, i.e. non-confluent. Whenever there are two false instances $\mathsf{P}[n/\alpha]$, $\mathsf{P}[m/\alpha]$ of an hypothesis $\forall\alpha^{\mathbb{N}}\mathsf{P}$ in some EM_1-rule $u\parallel_a v$, in u it may be raised either the exception n related to $\mathsf{P}[n/\alpha]$, or the exception m related to $\mathsf{P}[m/\alpha]$. The computation is converging in both cases, and the witness we get for a simple existential conclusion C is correct in both cases: however, we may obtain a *different* witness in the two cases. The interest of the non-deterministic approach is that it does not impose arbitrary restrictions ruling out potentially interesting computations: there are classical proofs whose non-deterministic interpretation is in a sense canonical (see [6], p. 40-50 for examples). Alternatively, with techniques introduced in [2], we may provide in a simple and natural way confluent evaluation rules. It is an interesting aspect of our framework that non-determinism arises just because one may generate during computation different refutations of EM_1-hypotheses, so any strategy for choosing between them re-establishes confluence. For reason of space, we shall not address this matter in the present paper.

1.7 Plan of the Paper

This is the plan of the paper. In §2 we introduce a type theoretical version of intuitionistic arithmetic HA extended with EM_1. In §3 we introduce a realizability semantics for $\mathsf{HA}+\mathsf{EM}_1$. Then in §4, 5 we prove that this semantics is sound for $\mathsf{HA}+\mathsf{EM}_1$. As a corollary, we deduce that $\mathsf{HA}+\mathsf{EM}_1$ is strongly normalizing and that any proof of a simply existential Σ^0_1-formula provides a witness.

2 The System $\mathsf{HA}+\mathsf{EM}_1$

In this section we formalize intuitionistic Arithmetic HA, and we add an operator $\parallel$ formalizing EM_1. We start with the language of formulas.

▶ **Definition 1** (Language of $\mathsf{HA}+\mathsf{EM}_1$)**.** The language $\mathcal{L}$ of $\mathsf{HA}+\mathsf{EM}_1$ is defined as follows.

1. The terms of $\mathcal{L}$ are inductively defined as either variables $\alpha,\beta,\ldots$ or 0 or $\mathsf{S}(t)$ with $t\in\mathcal{L}$. A numeral is a term of the form $\mathsf{S}\ldots\mathsf{S}0$.
2. There is one symbol $\mathcal{P}$ for every primitive recursive relation over $\mathbb{N}$; with $\mathcal{P}^{\perp}$ we denote the symbol for the complement of the relation denoted by $\mathcal{P}$. The atomic formulas of $\mathcal{L}$ are all the expressions of the form $\mathcal{P}(t_1,\ldots,t_n)$ such that $t_1,\ldots,t_n$ are terms of $\mathcal{L}$ and n is the arity of $\mathcal{P}$. Atomic formulas will also be denoted as $\mathsf{P},\mathsf{Q},\mathsf{P}_i,\ldots$.
3. The formulas of $\mathcal{L}$ are built from atomic formulas of $\mathcal{L}$ by the connectives $\vee,\wedge,\to,\forall,\exists$ as usual, with quantifiers ranging over numeric variables $\alpha^{\mathbb{N}},\beta^{\mathbb{N}},\ldots$.

From now on, if P is any *closed* atomic formula, we will write $\mathsf{P}\equiv\mathtt{True}$ ($\mathsf{P}\equiv\mathtt{False}$) if the formula is true (false) in the standard interpretation, that is, if $\mathsf{P}=\mathcal{R}(n_1,\ldots,n_k)$ and the sequence of numerals $(n_1,\ldots,n_k)$ belongs (does not belong) to the primitive recursive

relation denoted by $\mathcal{R}$. We now define in figure 1 a set of untyped proof terms, then a type assignment for them.

It is a standard natural deduction system with introduction and elimination rules for each connective and induction rules for integers, together with a term assignment in the spirit of Curry-Howard correspondence (see [27], for example).

Grammar of Untyped Proof Terms

$$t, u, v ::= x \mid tu \mid tm \mid \lambda xu \mid \lambda\alpha u \mid \langle t, u\rangle \mid \pi_0 u \mid \pi_1 u \mid \iota_0(u) \mid \iota_1(u) \mid t[x.u, y.v] \mid (m, t) \mid t[(\alpha, x).u]$$

$$\mid u \parallel_a v \mid [a]\mathtt{H}^{\forall\alpha\mathsf{P}} \mid [a]\mathtt{W}^{\exists\alpha\mathsf{P}^\perp} \mid \mathtt{True} \mid \mathsf{R}uvm \mid \mathsf{r}t_1 \dots t_n$$

where m ranges over terms of $\mathcal{L}$, x over proof terms variables and a over hypothesis variables. We also assume that in the term $u \parallel_a v$, there is some atomic formula P, such that a occurs free in u only in subterms of the form $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}$ and a occurs free in v only in subterms of the form $[a]\mathtt{W}^{\exists\alpha\mathsf{P}^\perp}$, and the occurrences of the variables in P different from α are free in both u and v.

Contexts With Γ we denote contexts of the form $e_1 : A_1, \dots, e_n : A_n$, where each e_i is either a proof-term variable $x, y, z \dots$ or a EM_1 hypothesis variable $a, b, \dots$, and $e_i \neq e_j$ for $i \neq j$.

Axioms $\Gamma, x : A \vdash x : A$ $\qquad \Gamma, a : \forall\alpha^{\mathtt{N}}\mathsf{P} \vdash [a]\mathtt{H}^{\forall\alpha\mathsf{P}} : \forall\alpha^{\mathtt{N}}\mathsf{P}$ $\qquad \Gamma, a : \exists\alpha^{\mathtt{N}}\mathsf{P}^\perp \vdash [a]\mathtt{W}^{\exists\alpha\mathsf{P}^\perp} : \exists\alpha^{\mathtt{N}}\mathsf{P}^\perp$

Conjunction $\dfrac{\Gamma \vdash u : A \quad \Gamma \vdash t : B}{\Gamma \vdash \langle u, t\rangle : A \wedge B}$ $\qquad \dfrac{\Gamma \vdash u : A \wedge B}{\Gamma \vdash \pi_0 u : A}$ $\qquad \dfrac{\Gamma \vdash u : A \wedge B}{\Gamma \vdash \pi_1 u : B}$

Implication $\dfrac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B}$ $\qquad \dfrac{\Gamma, x : A \vdash u : B}{\Gamma \vdash \lambda xu : A \to B}$

Disjunction Intro. $\dfrac{\Gamma \vdash u : A}{\Gamma \vdash \iota_0(u) : A \vee B}$ $\qquad \dfrac{\Gamma \vdash u : B}{\Gamma \vdash \iota_1(u) : A \vee B}$

Disjunction Elimination $\dfrac{\Gamma \vdash u : A \vee B \quad \Gamma, x : A \vdash w_1 : C \quad \Gamma, x : B \vdash w_2 : C}{\Gamma \vdash u[x.w_1, x.w_2] : C}$

Universal Quantification $\dfrac{\Gamma \vdash u : \forall\alpha^{\mathtt{N}}A}{\Gamma \vdash um : A[m/\alpha]}$ $\qquad \dfrac{\Gamma \vdash u : A}{\Gamma \vdash \lambda\alpha u : \forall\alpha^{\mathtt{N}}A}$

where m is any term of the language $\mathcal{L}$ and α does not occur free in any formula B occurring in Γ.

Existential Quantification $\dfrac{\Gamma \vdash u : A[m/\alpha]}{\Gamma \vdash (m, u) : \exists\alpha^{\mathtt{N}}.A}$ $\qquad \dfrac{\Gamma \vdash u : \exists\alpha^{\mathtt{N}}A \quad \Gamma, x : A \vdash t : C}{\Gamma \vdash u[(\alpha, x).t] : C}$

where α is not free in C nor in any formula B occurring in Γ.

Induction $\dfrac{\Gamma \vdash u : A(0) \quad \Gamma \vdash v : \forall\alpha^{\mathtt{N}}.\, A(\alpha) \to A(\mathsf{S}(\alpha))}{\Gamma \vdash \mathsf{R}uvt : A(t)}$

where t is any term of the language $\mathcal{L}$.

Post Rules $\dfrac{\Gamma \vdash u_1 : \mathsf{P}_1 \ \Gamma \vdash u_2 : \mathsf{P}_2 \ \cdots \ \Gamma \vdash u_n : \mathsf{P}_n}{\Gamma \vdash u : \mathsf{P}}$

where $\mathsf{P}_1, \mathsf{P}_2, \dots, \mathsf{P}_n, \mathsf{P}$ are atomic formulas and the rule is a Post rule for equality, for a Peano axiom or a primitive recursive relation and if $n > 0$, u is $\mathsf{r}u_1 \dots u_n$, otherwise u is $\mathtt{True}$.

EM1 $\dfrac{\Gamma, a : \forall\alpha^{\mathtt{N}}\mathsf{P} \vdash w_1 : C \quad \Gamma, a : \exists\alpha^{\mathtt{N}}\mathsf{P}^\perp \vdash w_2 : C}{\Gamma \vdash w_1 \parallel_a w_2 : C}$

Figure 1 Term Assignment Rules for $\mathsf{HA} + \mathsf{EM}_1$.

We replace purely universal axioms (i.e., Π^0_1-axioms) with Post rules (as in Prawitz [26]), which are inferences of the form

$$\frac{\Gamma \vdash \mathsf{P}_1 \quad \Gamma \vdash \mathsf{P}_2 \quad \cdots \quad \Gamma \vdash \mathsf{P}_n}{\Gamma \vdash \mathsf{P}}$$

where $\mathsf{P}_1, \ldots, \mathsf{P}_n, \mathsf{P}$ are atomic formulas of $\mathcal{L}$ such that for every substitution $\sigma = [t_1/\alpha_1, \ldots, t_k/\alpha_k]$ of closed terms $t_1, \ldots, t_k$ of $\mathcal{L}$, $\mathsf{P}_1\sigma = \ldots = \mathsf{P}_n\sigma \equiv \mathtt{True}$ implies $\mathsf{P}\sigma \equiv \mathtt{True}$. Let now eq be the symbol for the binary relation of equality between natural numbers ("=" will also be used). Among the Post rules, we have the Peano axioms

$$\frac{\Gamma \vdash \mathsf{eq}(\mathsf{S}t_1, \mathsf{S}t_2)}{\Gamma \vdash \mathsf{eq}(t_1, t_2)} \qquad \frac{\Gamma \vdash \mathsf{eq}(0, \mathsf{S}t)}{\Gamma \vdash \bot}$$

and axioms of equality

$$\frac{}{\Gamma \vdash \mathsf{eq}(t, t)} \qquad \frac{\Gamma \vdash \mathsf{eq}(t_1, t_2) \quad \Gamma \vdash \mathsf{eq}(t_2, t_3)}{\Gamma \vdash \mathsf{eq}(t_1, t_3)} \qquad \frac{\Gamma \vdash \mathsf{P}[t_1/\alpha] \quad \Gamma \vdash \mathsf{eq}(t_1, t_2)}{\Gamma \vdash \mathsf{P}[t_2/\alpha]}$$

We also have a Post rule for the defining axioms of each primitive recursive relation, for example the false relation $\bot$, addition, multiplication:

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash \mathsf{P}} \qquad \frac{}{\Gamma \vdash \mathsf{add}(t, 0, t)} \qquad \frac{\Gamma \vdash \mathsf{add}(t_1, t_2, t_3)}{\Gamma \vdash \mathsf{add}(t_1, \mathsf{S}t_2, \mathsf{S}t_3)}$$

$$\frac{}{\Gamma \vdash \mathsf{mult}(t, 0, 0)} \qquad \frac{\Gamma \vdash \mathsf{mult}(t_1, t_2, t_3) \quad \Gamma \vdash \mathsf{add}(t_3, t_1, t_4)}{\Gamma \vdash \mathsf{mult}(t_1, \mathsf{S}t_2, t_4)}$$

For simplifying the representation of proofs, we assume also to have a Post rule for each true closed atomic formula P:

$$\frac{}{\Gamma \vdash \mathsf{P}}$$

From the $\bot$-rule for atomic formulas we may derive the $\bot$-rule for all formulas. We assume that in the proof terms three distinct classes of variables appear: one for proof terms, denoted usually as $x, y, \ldots$; one for quantified variables of the formula language $\mathcal{L}$ of $\mathsf{HA} + \mathsf{EM}_1$, denoted usually as $\alpha, \beta, \ldots$; one for the pair of hypotheses bound by EM_1, denoted usually as $a, b, \ldots$. In the term $u \parallel_a v$, any free occurrence of a in u occurs in an expression $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}$, and denotes an assumption $\forall\alpha^{\mathtt{N}}\mathsf{P}$. Any free occurrence of a in v occurs in an expression $[a]\mathtt{W}^{\exists\alpha\mathsf{P}^{\bot}}$, and denotes an assumption $\exists\alpha^{\mathtt{N}}\mathsf{P}^{\bot}$. All the occurrences of a in u and v are bound, and we assume the usual renaming rules and alpha equivalences to avoid capture of variables in the reduction rules that we shall give. Alternatively, $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}$ is the thrower of an exception a and $[a]\mathtt{W}^{\exists\alpha\mathsf{P}^{\bot}}$ is the catcher of the same exception a. With $u \parallel v$ we denote a generic term of the form $u \parallel_a v$; we shall use this notation whenever our considerations will not depend on which is exactly the variable a. Terms of the form $((u \parallel v_1) \parallel v_2)\ldots) \parallel v_n$ for any $n \geq 0$ will be denoted as $u \parallel v_1 \parallel \ldots \parallel v_n$ or as $\mathcal{EM}[u]$. In the terms $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}$ and $[a]\mathtt{W}^{\exists\alpha\mathsf{P}^{\bot}}$ the free variables are a and those of P minus α.

Assume that Γ is a context, t an untyped proof term and A a formula, and $\Gamma \vdash t : A$: then t is said to be a typed proof term. Typing assignment satisfies Weakening, Exchange and Thinning, as usual. SN is the set of strongly normalizing untyped proof terms and NF is the set of normal untyped proof terms, as usual in lambda calculus ([27]). PNF is the inductively defined set of the Post normal forms (intuitively, normal terms representing closed proof trees made only of Post rules whose leaves are universal hypothesis followed by

Reduction Rules for HA

$(\lambda x.u)t \mapsto u[t/x] \qquad (\lambda\alpha.u)t \mapsto u[t/\alpha]$

$\pi_i\langle u_0, u_1\rangle \mapsto u_i$, for i=0,1
$\iota_i(u)[x_1.t_1, x_2.t_2] \mapsto t_i[u/x_i]$, for i=0,1
$(n, u)[(\alpha, x).v] \mapsto v[n/\alpha][u/x]$, for each numeral n
$\mathsf{R}uv0 \mapsto u$
$\mathsf{R}uv(\mathsf{S}n) \mapsto vn(\mathsf{R}uvn)$, for each numeral n

Permutation Rules for EM_1

$(u \parallel_a v)w \mapsto uw \parallel_a vw$, if a does not occur free in w

$\pi_i(u \parallel_a v) \mapsto \pi_i u \parallel_a \pi_i v$
$(u \parallel_a v)[x.w_1, y.w_2] \mapsto u[x.w_1, y.w_2] \parallel_a v[x.w_1, y.w_2]$, if a does not occur free in w_1, w_2
$(u \parallel_a v)[(\alpha, x).w] \mapsto u[(\alpha, x).w] \parallel_a v[(\alpha, x).w]$, if a does not occur free in w_1, w_2

Reduction Rules for EM_1

$([a]\mathtt{H}^{\forall\alpha\mathsf{P}})n \mapsto \mathtt{True}$, if $\mathsf{P}[n/\alpha]$ is *closed* and $\mathsf{P}[n/\alpha] \equiv \mathtt{True}$

$u \parallel_a v \mapsto u$, if a does not occur free in u
$u \parallel_a v \mapsto v[a := n]$, if $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}n$ occurs in u, $\mathsf{P}[n/\alpha]$ is *closed* and $\mathsf{P}[n/\alpha] \equiv \mathtt{False}$

Figure 2 Reduction Rules for HA + EM_1.

an elimination rule), that is: $\mathtt{True} \in \mathsf{PNF}$; for every closed term n of $\mathcal{L}$, if $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}n \in \mathsf{NF}$, then $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}n \in \mathsf{PNF}$; if $t_1, \dots, t_n \in \mathsf{PNF}$, then $\mathsf{r}t_1 \dots t_n \in \mathsf{PNF}$.

We are now going to explain the reduction rules for the proof terms of HA+EM_1, which are given in figure 2 (with $\mapsto^*$ we shall denote the reflexive and transitive closure of the one-step reduction $\mapsto$). We find among them the ordinary reductions of Intuitionistic Arithmetic for the logical connectives and induction. Permutation Rules for EM_1 are an instance of Prawitz's permutation rules for $\vee$-elimination, as explained in the introduction. Raising an exception n in $u \parallel_a v$ removes all occurrences of assumptions $[a]\mathtt{W}^{\exists\alpha\mathsf{P}^\perp}$ in v; we define first an operation removing them, and denoted $v[a := n]$.

▶ **Definition 2** (Witness Substitution)**.** Suppose v is any term and n a closed term of $\mathcal{L}$. We define

$$v[a := n]$$

as the term obtained from v by replacing each subterm $[a]\mathtt{W}^{\exists\alpha\mathsf{P}^\perp}$ corresponding to a free occurrence of a in v by $(n, \mathtt{True})$, if $\mathsf{P}[n/\alpha] \equiv \mathtt{False}$, and by $(n, [a]\mathtt{H}^{\forall\alpha\,\alpha=^0\mathsf{S}0})$, otherwise.

▶ Remark. An exception is raised only when $\mathsf{P}[n/\alpha] \equiv \mathtt{False}$. Therefore the substitution of $[a]\mathtt{W}^{\exists\alpha\mathsf{P}^\perp}$ by $(n, [a]\mathtt{H}^{\forall\alpha\,\alpha=^0\mathsf{S}0})$ will never occur in the reductions rules that we have defined. However, the general case of the substitution will be needed to define realizability, and namely because we want it to be suitable to prove strong normalization.

The rules for EM_1 translate the informal idea of learning by trial and error we sketched in the introduction, that is:

1. The first EM_1-reduction: $([a]\mathtt{H}^{\forall\alpha\mathsf{P}})n \mapsto \mathtt{True}$ if $\mathsf{P}[n/\alpha] \equiv \mathtt{True}$, says that whenever we use a closed instance $\mathsf{P}[n/\alpha]$ of the assumption $\forall\alpha^{\mathtt{N}}\mathsf{P}$, we check it, and if the instance is true we replace it with its canonical proof.
2. The second EM_1-reduction: $u \parallel_a v \mapsto u$, says that if, using the first reduction, we are able to remove all the instances of the assumption $[a]\mathtt{H}^{\forall\alpha\mathsf{P}} : \forall\alpha^{\mathtt{N}}\mathsf{P}$ in u, then the assumption is

unnecessary and the proof term $u \parallel_a v$ may be simplified to u. In this case the exceptional part v of $u \parallel_a v$ is never used.

3. The third EM_1-reduction: $u \parallel_a v \mapsto v[a := n]$, if $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}n$ occurs in u and $\mathsf{P}[n/\alpha] \equiv \mathtt{False}$, says that if we check a closed instance $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}n : \mathsf{P}[n/\alpha]$ of the assumption $\forall\alpha^{\mathbb{N}}\mathsf{P}$, and we find that the assumption is wrong, then we raise in u the exception n and we start the exceptional part $v[a := n]$ of $u \parallel_a v$. Raising an exception is a non-deterministic operation (we may have two or more exceptions to choose) and has no effect outside $u \parallel_a v$.

We claim that the reductions satisfy subject reduction: if $\Gamma \vdash t : A$ and $t \mapsto u$ then $\Gamma \vdash t : A$. The proof is by induction over t. For the reduction rule $u \parallel_a v \mapsto u$ we use the fact that a is not free in u and the Thinning rule. For the reduction rule $u \parallel_a v \mapsto v[a := n]$ we use the fact that a is not free in $v[a := n]$ and Thinning rule again.

As usual, neutral terms are terms that are not "values", and need to be further computed. We also introduce the important concept of quasi-closed term, which intuitively is a term behaving as a closed one, in the sense that it can be executed, but that contains some free hypotheses on which its correctness depends.

▶ **Definition 3** (Neutrality, Quasi-Closed terms)**.**

1. An untyped proof term is *neutral* if it is not of the form $\lambda x\, u$ or $\lambda\alpha\, u$ or $\langle u, t\rangle$ or $\iota_i(u)$ or (t, u) or $[a]\mathtt{H}^{\forall a\mathsf{P}}$ or $u \parallel_a v$.
2. If t is an untyped proof term which contains as free variables only EM_1-hypothesis variables $a_1, \ldots, a_n$, such that each occurrence of them is of the form $[a_i]\mathtt{H}^{\forall\alpha\mathsf{P}_i}$ for some P_i, then t is said to be *quasi-closed*.

3 A Realizability interpretation for $\mathsf{HA} + \mathsf{EM}_1$

In this section we define a realizability semantics for $\mathsf{HA} + \mathsf{EM}_1$, in which realizers may be interpreted as algorithms learning by trial and error a correct value. With respect to the Interactive realizability semantics in [2], the main difference is that we have no formal notion of knowledge state here. Informally, the counterpart of a knowledge state here would be the set of the free EM_1 hypothesis variables occurring in a term and the collection of all assignments $[a := n]$ produced by some reduction $u \parallel_a v \mapsto v[a := n]$ performed in the computation of the term.

Realizers will be deduced to be strongly normalizing terms, and the soundness of this realizability semantics will have strong normalization as a corollary. As in [21], realizers may be untyped terms, and also quasi-closed. With respect to the usual notion of intuitionistic realizability, there is a special case for atomic formulas, and one special case $t = u \parallel_a v$ for the connectives $\vee, \exists$.

▶ **Definition 4** (Realizability for $\mathsf{HA} + \mathsf{EM}_1$)**.** Assume t is a *quasi-closed* term in the grammar of untyped proof terms of $\mathsf{HA} + \mathsf{EM}_1$ and C is a *closed* formula. We define the relation $t \Vdash C$ by induction on C and for each fixed formula by a generalized inductive definition.

1. $t \Vdash \mathsf{P}$ if and only if one of the following holds:
   i) $t \in \mathsf{PNF}$ and $\mathsf{P} \equiv \mathtt{False}$ implies t contains a subterm $[a]\mathtt{H}^{\forall\alpha\mathsf{Q}}n$ with $\mathsf{Q}[n/\alpha] \equiv \mathtt{False}$;
   ii) $t \notin \mathsf{NF}$ and for all t', $t \mapsto t'$ implies $t' \Vdash \mathsf{P}$
2. $t \Vdash A \wedge B$ if and only if $\pi_0 t \Vdash A$ and $\pi_1 t \Vdash B$
3. $t \Vdash A \to B$ if and only if for all u, if $u \Vdash A$, then $tu \Vdash B$

4. $t \Vdash A \vee B$ if and only if one of the following holds:
   i) $t = \iota_0(u)$ and $u \Vdash A$ or $t = \iota_1(u)$ and $u \Vdash B$;
   ii) $t = u \parallel_a v$ and $u \Vdash A \vee B$ and $v[a := m] \Vdash A \vee B$ for every numeral m;
   iii) $t \notin \mathsf{NF}$ is neutral and for all t', $t \mapsto t'$ implies $t' \Vdash A \vee B$.
5. $t \Vdash \forall \alpha^{\mathtt{N}} A$ if and only if for every closed term n of $\mathcal{L}$, $tn \Vdash A[n/\alpha]$
6. $t \Vdash \exists \alpha^{\mathtt{N}} A$ if and only if one of the following holds:
   i) $t = (n, u)$ for some numeral n and $u \Vdash A[n/\alpha]$;
   ii) $t = u \parallel_a v$ and $u \Vdash \exists \alpha^{\mathtt{N}} A$ and $v[a := m] \Vdash \exists \alpha^{\mathtt{N}} A$ for every numeral m;
   iii) $t \notin \mathsf{NF}$ is neutral and for all t', $t \mapsto t'$ implies $t' \Vdash \exists \alpha^{\mathtt{N}} A$.

▶ Remark. A realizer is a quasi-closed term, which is interpreted as a program which has made hypotheses in order to decide some instances of EM_1. Its free EM_1 hypothesis variables do not influence the evolution of the term; they represent the assumptions on which the correctness of the computation depend, and they may raise an exception when the term is placed in a context of the form $u \parallel_a v$.

The definition of the realizability relation for the negative connectives $\wedge, \to, \forall$ is standard and it determines the notion of *test*, that is, the kind of input that must be provided to the realizer.

The definition of the realizability relation for the positive connectives $\vee, \exists$ determines the notion of *answer*. We shall see in the crucial Proposition 2 that indeed every realizer *does* provide an answer, under the form of *prediction* (a possibly unsafe answer): a realizer of $A \vee B$ normalizes to a term containing a realizer of A or a realizer of B and a realizer of $\exists \alpha^{\mathtt{N}} A$ normalizes to a term containing a realizer of $A[n/\alpha]$. However, these realizers are only quasi-closed, therefore their correctness depends on extra hypotheses and is not guaranteed: only in the case of closed realizers and of Σ^0_1-formulas we will prove a true disjunction property and a true witness property. The *style* of the definition of realizability for $A \vee B$, $\exists \alpha^{\mathtt{N}} A$ is inspired from Prawitz strong validity [26] and its main feature is that it depends not only on the formula, but also *on the shape of the term*; since it is an inductive definition, a term is a realizer if one can deduce it by means of a finite number of applications of the three subclauses i), ii), iii) of the definition. We observe that the base case i) of the definition is the one of intuitionistic realizability, even if we are in a classical setting: the deep reason of this phenomenon is that in the definition of $u \parallel_a v \Vdash A \vee B$, even if u may contain an hypothesis term $[a]\mathtt{H}^{\forall \alpha \mathsf{P}}$ that becomes free, this term does not "stop" the computation inside u, and u can nevertheless realize $A \vee B$, i.e. reach eventually a form $\iota_i(w)$, after steps of normalization (applications of iii)) or at the end of whatever paths one has followed by applications of ii).

In the case of an atomic formula Q, the definition is analogous to the one of Interactive realizability (see [3] for many intuitions): a proof-term should represent a proof made only of Post-rules (a calculation), possibly with the aid of some hypothesis $\forall \alpha^{\mathtt{N}} \mathsf{P}$; if the formula Q is false, than a counterexample to some hypothesis should be contained in the realizer.

▶ **Example 5** (Realizer of the Excluded Middle)**.** Any closed instance

$$\forall \alpha^{\mathtt{N}} \mathsf{P} \vee \exists \alpha^{\mathtt{N}} \mathsf{P}^{\perp}$$

of EM_1 is provable in $\mathsf{HA} + \mathsf{EM}_1$ by a straightforward application of the EM_1-rule. It shall then be a consequence of the Adequacy Theorem 7 that any instance of EM_1 is realizable. It is however instructive to construct and examine right now a realizer. We define:

$$\mathsf{E}_{\mathsf{P}} := \iota_0([a]\mathtt{H}^{\forall \alpha \mathsf{P}}) \parallel_a \iota_1([a]\mathtt{W}^{\exists \alpha \mathsf{P}^{\perp}})$$

This realizer first tries with $\forall\alpha^{\mathtt{N}}\mathsf{P}$, and if some exception is raised, switches to $\exists^{\mathtt{N}}\alpha\mathsf{P}^{\perp}$. In order to show that

$$\mathsf{E}_{\mathsf{P}} \Vdash \forall\alpha^{\mathtt{N}}\mathsf{P} \vee \exists\alpha^{\mathtt{N}}\mathsf{P}^{\perp}$$

by definition 4 of realizability, we have to prove:

1. $[a]\mathtt{H}^{\forall\alpha\mathsf{P}} \Vdash \forall\alpha^{\mathtt{N}}\mathsf{P}$, *that is, for all numerals* n, $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}n \Vdash \mathsf{P}[n/\alpha]$. $\mathsf{P}[n/\alpha]$ is closed because we assumed $\forall\alpha^{\mathtt{N}}\mathsf{P}$ closed. If $\mathsf{P}[n/\alpha] \equiv \mathtt{True}$ then $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}n \mapsto \mathtt{True}$, and $\mathtt{True} \Vdash \mathsf{P}[n/\alpha]$ by definition 4.1.(i), therefore $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}n \Vdash \mathsf{P}[n/\alpha]$ by definition 4.1.(ii). If $\mathsf{P}[n/\alpha] \equiv \mathtt{False}$ then $[a]\mathtt{H}^{\forall\alpha\mathsf{P}}n \Vdash \mathsf{P}[n/\alpha]$ by definition 4.1.(i).
2. *for all numerals* n, $[a]\mathtt{W}^{\exists\alpha\mathsf{P}^{\perp}}[a := n] \Vdash \exists\alpha^{\mathtt{N}}\mathsf{P}^{\perp}$. By definition 2, this amounts to show that $(n, \mathtt{True}) \Vdash \exists\alpha^{\mathtt{N}}\mathsf{P}^{\perp}$, when $\mathsf{P}[n/\alpha] \equiv \mathtt{False}$, that is $\mathtt{True} \Vdash \mathsf{P}^{\perp}[n/\alpha]$, and that $(n, [a]\mathtt{H}^{\forall\alpha\,\alpha=\mathtt{0}}\mathsf{S0}) \Vdash \exists\alpha^{\mathtt{N}}\mathsf{P}^{\perp}$ otherwise, that is $[a]\mathtt{H}^{\forall\alpha\,\alpha=\mathtt{0}}\mathsf{S0} \Vdash \mathsf{P}^{\perp}[n/\alpha]$. In the first case we have $\mathsf{P}^{\perp}[n/\alpha] \equiv \mathtt{True}$, in the second one the realizer contains an occurrence of $[a]\mathtt{H}^{\forall\alpha\,\alpha=\mathtt{0}}\mathsf{S0}$, having $(\alpha = 0)[\alpha/\mathsf{S0}] \equiv \mathtt{False}$. In both case we apply definition 4.1.(i).

4 Basic Properties of Realizers

In this section we prove that the set of realizers of a given formula C satisfies the usual properties for a Girard's reducibility candidate.

▶ **Definition 6.** Extending the approach of [9], we define four properties **(CR1)**, **(CR2)**, **(CR3)**, **(CR4)** of realizers t of a formula A plus an inhabitation property **(CR5)** for A:

(CR1) If $t \Vdash A$, then $t \in \mathsf{SN}$.
(CR2) If $t \Vdash A$ and $t \mapsto^* t'$, then $t' \Vdash A$.
(CR3) If $t \notin \mathsf{NF}$ is neutral and for every t', $t \mapsto t'$ implies $t' \Vdash A$, then $t \Vdash A$.
(CR4) If $t = u \parallel_a v$, $u \Vdash A$ and $v[a := m] \Vdash A$ for every numeral m, then $t \Vdash A$.
(CR5) There is a u such that $u \Vdash A$.

All properties listed above hold.

▶ Proposition 1. Every term t has the properties **(CR1)**, **(CR2)**, **(CR3)**, **(CR4)** and the inhabitation property **(CR5)** holds.

As we pointed out in the introduction, we cannot prove that any realizer of a disjunction or an existential contains a *correct witness*, but we may prove some weakening of this property: in some sense, surprisingly, also classical logic enjoys the disjunction and numerical existence properties. Namely, a realizer of $A \vee B$ contains a realizer of A or a realizer of B and a realizer of $\exists\alpha^{\mathtt{N}}A$ contains a realizer of $A[n/\alpha]$. The point is that n is not necessarily a true witness, but rather a *prediction* based on the universal assumptions contained in the realizer.

▶ Proposition 2 (Weak Disjunction and Numerical Existence Properties).
1. Suppose $t \Vdash A \vee B$. Then either $t \mapsto^* \mathcal{EM}[\iota_0(u)]$ and $u \Vdash A$ or $t \mapsto^* \mathcal{EM}[\iota_1(u)]$ and $u \Vdash B$.
2. Suppose $t \Vdash \exists\alpha^{\mathtt{N}}A$. Then $t \mapsto^* \mathcal{EM}[(n, u)]$ for some numeral n such that $u \Vdash A[n/\alpha]$.

Proof.
1. Since $t \in \mathsf{SN}$ by **(CR1)**, let t' be such that $t \mapsto^* t' \in \mathsf{NF}$. By **(CR2)**, $t' \Vdash A \vee B$. If $t' = \iota_0(u)$, we are done. The only possibility left is that $t' = v \parallel v_1 \parallel v_2 \ldots \parallel v_n$, with v not of the form $w_0 \parallel w_1$. By definition 4.4.(ii) we have $v \Vdash A \vee B$, and since v is normal and not of the form $w_0 \parallel w_1$, by definition 4.4.(i) we have either $v = \iota_0(u)$, with $u \Vdash A$, or $v = \iota_1(u)$, with $u \Vdash B$.
2. Similar to 1. ◀

We observe that in a realizer $v \parallel_{a_1} v_1 \parallel_{a_2} v_2 \dots \parallel_{a_n} v_n$ of $A \vee B$, the further we move on the left, the larger is the set of hypotheses becoming free. This is indeed the price payed to construct a realizer of A or B, which is contained in v: hypotheses have to be made.

The next task is to prove that all introduction and elimination rules of $\mathsf{HA} + \mathsf{EM}_1$ define a realizer from a list of realizers for all premises. In some case this is true by definition of realizer, we list below some non-trivial cases we have to prove.

▶ Proposition 3.

1. If for every $t \Vdash A$, $u[t/x] \Vdash B$, then $\lambda x\, u \Vdash A \to B$.
2. If for every closed term m of $\mathcal{L}$, $u[m/\alpha] \Vdash B[m/\alpha]$, then $\lambda\alpha\, u \Vdash \forall\alpha^{\mathbb{N}} B$.
3. If $u \Vdash A_0$ and $v \Vdash A_1$, then $\pi_i\langle u, v\rangle \Vdash A_i$.
4. If $w_0[x_0.u_0, x_1.u_1] \Vdash C$ and for all numerals n, $w_1[x_0.u_0, x_1.u_1][a := n] \Vdash C$, then $(w_0 \parallel_a w_1)[x_0.u_0, x_1.u_1] \Vdash C$.
5. If $t \Vdash A_0 \vee A_1$ and for every $t_i \Vdash A_i$ it holds $u_i[t_i/x_i] \Vdash C$, then $t[x_0.u_0, x_1.u_1] \Vdash C$.
6. If $t \Vdash \exists\alpha^{\mathbb{N}} A$ and for every term n of $\mathcal{L}$ and $v \Vdash A[n/\alpha]$ it holds $u[n/\alpha][v/x] \Vdash C$, then $t[(\alpha, x).u] \Vdash C$.

5 The Adequacy Theorem

In this section we prove that the realizability semantics we defined in §3 is sound for $\mathsf{HA}+\mathsf{EM}_1$, and we derive strong normalization as a corollary. The witness property for Σ^0_1-formulas, instead, may be derived directly from the basic properties of realizers (§4).

▶ **Theorem 7** (Adequacy Theorem). *Suppose that* $\Gamma \vdash w : A$ *in the system* $\mathsf{HA} + \mathsf{EM}_1$*, with*

$$\Gamma = x_1 : A_1, \dots, x_n : A_n, a_1 : \exists\alpha_1^{\mathbb{N}}\neg\mathsf{P}_1, \dots, a_m : \exists\alpha_m^{\mathbb{N}}\neg\mathsf{P}_m, b_1 : \forall\alpha_1^{\mathbb{N}}\mathsf{Q}_1, \dots, b_l : \forall\alpha_l^{\mathbb{N}}\mathsf{Q}_l$$

and that the free variables of the formulas occurring in Γ *and* A *are among* $\alpha_1, \dots, \alpha_k$*. For all closed terms* $r_1, \dots, r_k$ *of* $\mathcal{L}$*, if there are terms* $t_1, \dots, t_n$ *such that*

$$\text{for } i = 1, \dots, n,\ t_i \Vdash A_i[r_1/\alpha_1 \cdots r_k/\alpha_k]$$

then

$$w[t_1/x_1 \cdots t_n/x_n\ r_1/\alpha_1 \cdots r_k/\alpha_k\ a_1 := i_1 \cdots a_m := i_m] \Vdash A[r_1/\alpha_1 \cdots r_k/\alpha_k]$$

for every numerals $i_1, \dots, i_m$*.*

▶ **Corollary 8** (Strong Normalization of HA+EM1). *All terms of* $\mathsf{HA} + \mathsf{EM}_1$ *are strongly normalizing.*

Proof. From Theorem 7 and **(CR5)** we derive that for all proof-terms $t : A$ we have some substitution t' such that $t' \Vdash A$. From **(CR1)** we conclude that t' is strongly normalizing: as a corollary, t itself is strongly normalizing. ◀

Our last task is to prove that all proofs of simply existential statements include a witness.

▶ **Theorem 9** (Normal Form Property and Existential Witness Extraction). *Suppose* t *is closed,* $t \Vdash \exists\alpha^{\mathbb{N}}\mathsf{P}$ *and* $t \mapsto^* t' \in \mathsf{NF}$*. Then* $t' = (n, u)$ *for some numeral* n *such that* $\mathsf{P}[n/\alpha] \equiv \mathtt{True}$*.*

Proof. By proposition 2, there is some numeral n such that $t' = \mathcal{EM}[(n, u)]$ and $u \Vdash \mathsf{P}[n/\alpha]$. So

$$t' = (n, u) \parallel_{a_1} v_1 \parallel_{a_2} v_2 \dots \parallel_{a_m} v_m$$

Since t' is closed, u is quasi-closed and all its free variables are among $a_1, a_2, \dots, a_m$. We observe that u must be closed. Otherwise, by definition 4.1.(i) and $u \Vdash \mathsf{P}[n/\alpha]$ we deduce that $u \in \mathsf{PNF}$, and thus u should contain a subterm $[a_i]\mathtt{H}^{\forall\alpha\mathsf{Q}}n$; moreover, $\mathsf{Q}[n/\alpha] \equiv \mathtt{False}$ otherwise u would not be normal; but then we would have either $m \neq 0$ and $t' \notin \mathsf{NF}$ because $t' \mapsto v_1[a_1 := n] \parallel_{a_2} v_2 \dots \parallel_{a_m} v_m$, or $m = 0$ and t' non-closed. Since u is closed, we obtain $t' = (n, u)$, for otherwise $t' \mapsto (n, u) \parallel_{a_2} v_2 \dots \parallel_{a_m} v_m$ and $t' \notin \mathsf{NF}$. Since $u \Vdash \mathsf{P}[n/\alpha]$, by definition 4.1.(i) it must be $\mathsf{P}[n/\alpha] \equiv \mathtt{True}$. ◀

By the Adequacy Theorem 7 and Theorem 9, whenever $\mathsf{HA} + \mathsf{EM}_1$ proves a closed formula of the shape $\forall\alpha_1^{\mathtt{N}} \dots \forall\alpha_k^{\mathtt{N}} \exists\beta^{\mathtt{N}} \mathsf{P}$, one can extract a realizer t with the property that, for every numerals $n_1, \dots, n_k$, there is some numeral n such that $tn_1 \dots n_k \mapsto^* (n, \mathtt{True})$ and $\mathsf{P}[n_1/\alpha_1 \cdots n_k/\alpha_k\, n/\beta] \equiv \mathtt{True}$. For example, from a proof of $\forall\alpha_1^{\mathtt{N}} \forall\alpha_2^{\mathtt{N}} \exists\beta^{\mathtt{N}} \mathsf{add}(\alpha_1, \alpha_2, \beta)$, one can extract a term computing the sum of natural numbers, even if the proposition has been proved classically.

6 Conclusions

From the point of view of classical Curry-Howard correspondence, the main contribution of this paper is a new decomposition of the EM_1 reduction rules in terms of delimited exceptions and permutation rules. The expert may at this point have noticed that some deterministic restriction of our conversions may be quite directly simulated in $\lambda\mu$-calculus and, less directly, in Krivine's λ_c-calculus. However, as it is quite often the case in proof theory, a variation in the rules of a system may be crucial to gain better results and understanding. In our case, with our approach we obtain several new results.

- *Markov's Principle and Restricted* EM_1. The mechanism of delimited exceptions allows to obtain quite refined results about systems containing Markov's principle, showing directly that its addition on top of intuitionistic logic preserves the disjunction and numerical existence properties [16]. Of course, Markov's principle is provable in $\mathsf{HA} + \mathsf{EM}_1$, by the most restricted version of the EM_1 rules, where the conclusion of the rule must be a Σ^1_0-formula. We shall show in a future paper that also our system enjoys the disjunction and numerical existence properties, when it is only allowed to use the restricted excluded middle sufficient to prove Markov's principle.
- *Extension of Prawitz validity to classical proofs.* The double negation is in some sense hardwired in the $\lambda\mu$ and in the λ_c calculi. As the cognoscenti know, this forces Krivine's realizability of a formula A for these calculi to have the form $\neg A \to \bot$, where $\neg A$ is the type of stacks and $\bot$ is interpreted by $\perp\!\!\!\perp$. Loosely speaking, in this way double negation elimination becomes a tautology: $(\neg\neg A) \to \neg A \to \bot$. Our priority is instead given to EM_1, and our reduction rules allow to extend an introductions-based Prawitz validity to a classical system. Such a result would not have been possible in the context of $\lambda\mu$ or λ_c.
- *Weak disjunction and existence properties for realizability.* Thanks to the essentially positive flavor of our realizability definition for positive connectives, we have shown (Proposition 2) that our notion of realizability satisfies a remarkable property: a realizer of a disjunction contains a realizer of one of the disjuncts, and a realizer of an existential statement contains a realizer of an instance of it. Similar insights seem not possible to be easily expressed in the framework of $\lambda\mu$-calculus or Krivine's realizability (or at least, similar properties have never been noticed). It is instead the explanation of classical programs as making hypotheses, testing them and learning, that has led to our results: our realizers behave like they do *precisely* because they want to achieve the disjunction and numerical existence properties during computations.

References

1 Akama, Y. and Berardi, S. and Hayashi S. and Kohlenbach, U., *An Arithmetical Hierarchy of the Law of Excluded Middle and Related Principles*. LICS 2004, pages 192–201.

2 F. Aschieri, S. Berardi, *Interactive Learning-Based Realizability for Heyting Arithmetic with EM1*, Logical Methods in Computer Science, 2010.

3 F. Aschieri, S. Berardi, *A New Use of Friedman's Translation: Interactive Realizability*, in: Logic, Construction, Computation, Berger et al. eds, Ontos-Verlag Series in Mathematical Logic, 2012.

4 F. Aschieri, *Interactive Realizability for Classical Peano Arithmetic with Skolem Axioms*. Proceedings of Computer Science Logic 2012, Leibniz International Proceedings in Informatics (LIPIcs), vol. 16, Schloss Dagstuhl, 2012.

5 F. Aschieri, *Interactive Realizability for Second-Order Heyting Arithmetic with $EM1$ and $SK1$*, Technical Report, http://hal.inria.fr/hal-00657054.

6 S. Berardi, *An Interactive Realizability Semantics for non-constructive proofs*, Chambery Summer school of Realization, Chambery, 14–17 June, 2011.
`http://www.di.unito.it/~stefano/Berardi-RealizationChambery-13Giugno2011.pdf`

7 S. Berardi, *Some intuitionistic equivalents of classical principles for degree 2 formulas*. Ann. Pure Appl. Logic 139(1–3): 185–200 (2006)

8 G. Birolo: *Interactive Realizability, Monads and Witness Extraction*, Ph.D. thesis, April, 15, 2013, Università di Torino (`http://arxiv.org/abs/1304.4091`)

9 J.-Y. Girard and Y. Lafont and P. Taylor.: *Proofs and Types*. Cambridge University Press (1989).

10 H. Friedman, *Classically and Intuitionistically Provable Recursive Functions*, Lecture Notes in Mathematics, 1978, Volume 669/1978, 21–27.

11 K. Gödel, *Uber eine bisher noch nicht benutzte Erweiterung des finiten Standpunktes*, Dialectica 12, pp. 280–287 (1958).

12 G. Gentzen, *Die Widerspruchsfreiheit der reinen Zahlentheorie*. Mathematische Annalen, 1935.

13 P. de Groote, *A Simple Calculus of Exception Handling*, Proc. of TLCA 1995: 201–215.

14 P. de Groote, *Strong Normalization for Classical Natural Deduction with Disjunction*, Proceedings of TLCA 2001: 182–196.

15 T. Griffin, *A Formulae-as-Type Notion of Control*, Proc. of POPL, 1990.

16 H. Herbelin, *An Intuitionistic Logic that Proves Markov's Principle*, Proceedings of LICS 2010: 50-56.

17 U. Kohlenbach, *On uniform weak König's lemma*. Annals of Pure and Applied Logic, 114(1–3) (2002).

18 G. Kreisel, *On Weak Completeness of Intuitionistic Predicate Logic*, Journal of Symbolic Logic, vol. 27, 1962.

19 J.-L. Krivine, *Lambda-calcul types et modèles*, Studies in Logic and Foundations of Mathematics (1990) 1–176. Masson, Paris.

20 J.-L. Krivine, *Dependent Choiche, "Quote" and the Clock*, Theoretical Computer Science 308(1-3), 2003, 259–276.

21 J.-L. Krivine, *Classical Realizability*. In Interactive models of computation and program behavior. Panoramas et synthèses , 2009, 197–229. Société Mathématique de France.

22 J.-L. Krivine, *Realizability Algebras II: new models of ZF + DC*, Logical Methods in Computer Science, 2012.

23 G. Mints, S. Tupailo, W. Bucholz, *Epsilon Substitution Method for Elementary Analysis*, Archive for Mathematical Logic, volume 35, 1996

24 A. Miquel, *Existential witness extraction in classical realizability and via a negative translation*. Logical Methods in Computer Science 7(2) (2011)

25 M. Parigot, *Lambda-My-Calculus: An Algorithmic Interpretation of Classical Natural Deduction*. LPAR 1992: 190–201.

26 D. Prawitz: *Ideas and Results in Proof Theory*. In Proceedings of the Second Scandinavian Logic Symposiuum (1971).

27 M. H. Sorensen, P. Urzyczyn, *Lectures on the Curry-Howard isomorphism*, Studies in Logic and the Foundations of Mathematics, vol. 149, Elsevier, 2006.

28 J. von Plato: *A Constructive Approach to Sylvester's Conjecture*. J. UCS 11(12): 2165–2178 (2005)

Bounds for the quantifier depth in finite-variable logics: Alternation hierarchy

Christoph Berkholz[1], Andreas Krebs[2], and Oleg Verbitsky*[3]

1 **RWTH Aachen University, Institut für Informatik**
D-52056 Aachen, Germany
berkholz@informatik.rwth-aachen.de

2 **Wilhelm-Schickard-Institut, Universität Tübingen**
Sand 13, 72076 Tübingen, Germany
mail@krebs-net.de

3 **Humboldt-Universität zu Berlin, Institut für Informatik**
Unter den Linden 6, D-10099 Berlin, Germany
verbitsk@informatik.hu-berlin.de

Abstract

Given two structures G and H distinguishable in FO^k (first-order logic with k variables), let $A^k(G,H)$ denote the minimum alternation depth of a FO^k formula distinguishing G from H. Let $A^k(n)$ be the maximum value of $A^k(G,H)$ over n-element structures. We prove the strictness of the quantifier alternation hierarchy of FO^2 in a strong quantitative form, namely $A^2(n) \geq n/8-2$, which is tight up to a constant factor. For each $k \geq 2$, it holds that $A^k(n) > \log_{k+1} n - 2$ even over colored trees, which is also tight up to a constant factor if $k \geq 3$. For $k \geq 3$ the last lower bound holds also over uncolored trees, while the alternation hierarchy of FO^2 collapses even over all uncolored graphs.

We also show examples of colored graphs G and H on n vertices that can be distinguished in FO^2 much more succinctly if the alternation number is increased just by one: while in Σ_i it is possible to distinguish G from H with bounded quantifier depth, in Π_i this requires quantifier depth $\Omega(n^2)$. The quadratic lower bound is best possible here because, if G and H can be distinguished in FO^k with i quantifier alternations, this can be done with quantifier depth n^{2k-2}.

1998 ACM Subject Classification F.4.1 Finite model theory

Keywords and phrases Alternation hierarchy, finite-variable logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.61

1 Introduction

Given structures G and H over vocabulary σ and a first-order formula Φ over the same vocabulary, we say that Φ *distinguishes* G from H if Φ is true on G but false on H. By *alternation depth* of Φ we mean the maximum length of a sequence of nested alternating quantifiers in Φ. Obviously, this parameter is bounded from above by the *quantifier depth* of Φ. We will examine the maximum alternation depth and quantifier depth needed to distinguish two structures for restrictions of first-order logic and particular classes of structures.

For a fragment $\mathcal{L}$ of first-order logic, by $A_{\mathcal{L}}(G,H)$ we denote the minimum alternation depth of a formula $\Phi \in \mathcal{L}$ distinguishing G from H. Similarly, we let $D_{\mathcal{L}}(G,H)$ denote the

* Supported by DFG grant VE 652/1–1. On leave from the Institute for Applied Problems of Mechanics and Mathematics, Lviv, Ukraine.

minimum quantifier depth of such Φ. Obviously, $A_{\mathcal{L}}(G,H) \leq D_{\mathcal{L}}(G,H)$. We define the *alternation function* $A_{\mathcal{L}}(n)$ to be equal to the maximum value of $A_{\mathcal{L}}(G,H)$ taken over all pairs of n-element structures G and H distinguishable in $\mathcal{L}$.

Our interest in this function is motivated by the observation that if the quantifier alternation hierarchy of $\mathcal{L}$ collapses, then $A_{\mathcal{L}}(n) = O(1)$. More specifically, $A_{\mathcal{L}}(n) \leq a$ if the alternation hierarchy collapses to its a-th level $\Sigma_a \cup \Pi_a$. Thus, showing that

$$\lim_{n\to\infty} A_{\mathcal{L}}(n) = \infty \tag{1}$$

is a way of proving that the hierarchy is strict.

Note that Condition (1) is, in general, formally stronger than a hierarchy result. For example, while the alternation hierarchy of first-order logic FO is strict over colored directed trees by Chandra and Harel [3], we have $A_{\mathrm{FO}}(n) = 1$ for any class of structures over a fixed vocabulary.

An example of this nature also exists when we restrict our logic to two variables: While the alternation hierarchy of $\mathrm{FO}^2[<]$ is strict over words in an infinite alphabet by Immerman and Weis [10], we have $A_{\mathrm{FO}^2}(n) = 1$ for words in any alphabet.

Moreover, the rate of growth of $A_{\mathcal{L}}(n)$ can be naturally regarded as a quality of the strictness of the alternation hierarchy. Note that any pair of structures G and H with $A_{\mathcal{L}}(G,H) = a$ can serve as a certificate that the first a levels of the alternation hierarchy of $\mathcal{L}$ are distinct. Indeed, if G is distinguished from H by a formula $\Phi \in \mathcal{L}$ of the minimum alternation depth a, then the set of structures $L = \{\, S : S \models \Phi\}$ is not definable in $\mathcal{L}$ with less than a quantifier alternations. Thus, the larger the value of $A_{\mathcal{L}}(n)$ is, the more levels of the alternation hierarchy can be separated by a certificate of size n.

Results that we now know about the function $A_{\mathcal{L}}(n)$ are displayed in Figure 1. The upper bound $A_{\mathrm{FO}^k}(n) \leq n^{k-1} + 1$ holds true even for the quantifier depth. It follows from the relationship of the distinguishability in FO^k to the $(k-1)$-dimensional color refinement (Weisfeiler-Lehman) procedure discovered in [6, 2] and the standard color stabilization argument; see [8]. The logarithmic upper bound for trees (Theorem 3.4) holds true also for the quantifier depth.

Class of structures	Logic	Bounds for $A_{\mathcal{L}}(n)$	
uncolored trees	$\mathcal{L} = \mathrm{FO}^2$	≤ 2	Theorem 3.3
	$\mathcal{L} = \mathrm{FO}^k$, $k \geq 3$	$> \log_{k+1} n - 2$	Theorem 3.2
		$< (k+3)\log_2 n$	Theorem 3.4
colored trees	$\mathcal{L} = \mathrm{FO}^k$, $k \geq 2$	$> \log_{k+1} n - 2$	Theorems 3.1 and 3.2
	$\mathcal{L} = \mathrm{FO}^k$, $k \geq 3$	$< (k+3)\log_2 n$	Theorem 3.4
uncolored graphs	$\mathcal{L} = \mathrm{FO}^2$	≤ 2	Theorem 3.3
	$\mathcal{L} = \mathrm{FO}^k$, $k \geq 3$	$> \log_{k+1} n - 2$	Theorem 3.2
		$\leq n^{k-1} + 1$	cf. [8]
colored graphs	$\mathcal{L} = \mathrm{FO}^2$	$> n/8 - 2$	Theorem 4.1
		$\leq n + 1$	cf. [6]
	$\mathcal{L} = \mathrm{FO}^k$, $k \geq 3$	$> \log_{k+1} n - 2$	Theorem 3.2
		$\leq n^{k-1} + 1$	cf. [8]

Figure 1 Results about $A_{\mathcal{L}}(n)$.

Additionally, in Section 5 we show that the Σ_i fragment of FO^2 is not only strictly more expressive than the Σ_{i-1} fragment but also more succinct in the following sense: There are colored graphs G and H on n vertices such that they can be distinguished in $\Sigma_{i-1} \cap \mathrm{FO}^2$ and, moreover, this is possible with bounded quantifier depth in $\Sigma_i \cap \mathrm{FO}^2$ while in $\Pi_i \cap \mathrm{FO}^2$ this requires quantifier depth $\Omega(n^2)$. The quadratic lower bound is best possible here because, if G and H can be distinguished in FO^k with i quantifier alternations, this can be done with quantifier depth n^{2k-2}.

2 Preliminaries

We consider first-order formulas only in the negation normal form (i.e., any negation stands in front of a relation symbol and otherwise only monotone Boolean connectives are used). For each $i \geq 1$, let Σ_i (resp. Π_i) denote the set of (not necessary prenex) formulas where any sequence of nested quantifiers has at most $i-1$ quantifier alternations and begins with $\exists$ (resp. $\forall$). In particular, *existential logic* Σ_1 consists of formulas without universal quantification. Up to logical equivalence, $\Sigma_i \cup \Pi_i \subset \Sigma_{i+1} \cap \Pi_{i+1}$. By the *quantifier alternation hierarchy* we mean the interlacing chains $\Sigma_1 \subset \Sigma_2 \subset \ldots$ and $\Pi_1 \subset \Pi_2 \subset \ldots$. We are interested in the corresponding fragments of a finite-variable logic.

As a short notation we use $D^k_{\mathcal{L}}(G,H) = D_{\mathcal{L}\cap\mathrm{FO}^k}(G,H)$ and $A^k_{\mathcal{L}}(G,H) = A_{\mathcal{L}\cap\mathrm{FO}^k}(G,H)$. The subscript FO can be dropped; for example, $D^k(G,H) = D^k_{\mathrm{FO}}(G,H)$ and $A^k(n) = A^k_{\mathrm{FO}}(n)$. Sometimes we will write $D^k_{\exists}(G,H)$ in place of $D^k_{\Sigma_1}(G,H)$.

The universe of a structure G will be denoted by $V(G)$, and the number of elements in $V(G)$ will be denoted by $v(G)$. Since binary structures can be regarded as vertex- and edge-colored directed graphs, the elements of $V(G)$ will also be called vertices. A vertex in a simple undirected graph is *universal* if it is adjacent to all other vertices.

The *k-pebble Ehrenfeucht-Fraïssé game on structures G and H*, is played by two players, Spoiler and Duplicator, to whom we will refer as he and she respectively. The players have equal sets of k pairwise different pebbles. A *round* consists of a move of Spoiler followed by a move of Duplicator. Spoiler takes a pebble and puts it on a vertex in G or in H. Then Duplicator has to put her copy of this pebble on a vertex of the other graph. Duplicator's objective is to keep the following condition true after each round: the pebbling should determine a partial isomorphism between G and H. The variant of the game where Spoiler starts playing in G and is allowed to jump from one graph to the other less than i times during the game will be referred to as the *Σ_i game*. In the *Π_i game* Spoiler starts in H.

For each positive integer r, the r-round Ehrenfeucht-Fraïssé game (as well as its Σ_i and Π_i variants) is a two-person game of perfect information with a finite number of positions. Therefore, either Spoiler or Duplicator has a *winning strategy* in this game, that is, a strategy winning against every strategy of the opponent.

▶ **Lemma 2.1** (e.g., [10]). *$D^k_{\Sigma_i}(G,H) \leq r$ if and only if Spoiler has a winning strategy in the r-round k-pebble Σ_i game on G and H.*

The lifting construction

Note that separation of the ground floor of the alternation hierarchy for FO^2 costs nothing. We can take graphs G and H with three isolated vertices each, color one vertex of G in red, and color the other vertices of G and all vertices of H in blue. Obviously, $D^2_{\exists}(G,H) = 1$ while $D^2_{\forall}(G,H) = \infty$. It turns out that any separation example can be lifted to higher floors in a rather general way.

The lifting gadget provided by Lemma 2.2 below is a reminiscence of the classical construction designed by Chandra and Harel to prove the strictness of the first-order alternation hierarchy. The Chandra-Harel construction is applicable to other logics (see, e.g., [5, Section 8.6.3]) and can be used as a general scheme for obtaining hierarchy results. This approach was also used by Oleg Pikhurko (personal communication, 2007) to construct, for each i, a sequence of pairs of trees G_n and H_n such that $D_{\Sigma_i}(G_n, H_n) = O(1)$ while $D_{\Pi_i}(G_n, H_n) \to \infty$ as $n \to \infty$.

Given colored graphs G_0 and H_0, we recursively construct graphs G_i and H_i as shown in Fig. 2. H_1 consists of three disjoint copies of H_0 and an extra universal vertex, that will be referred to as the root vertex of H_1. The root vertex is colored in a new color absent in G_0 and H_0, say, in gray. The graph G_1 is constructed similarly but, instead of three H_0-branches, it has two H_0-branches and one G_0-branch. Suppose that $i \geq 1$ and the rooted graphs G_i and H_i are already constructed. The graph H_{i+1} consists of three disjoint copies of G_i and the gray root vertex adjacent to the root of each G_i-part. The graph G_{i+1} is constructed similarly but, instead of three G_i-branches, it has two G_i-branches and one H_i-branch.

We will say that Spoiler plays *continuously* if, after each of his moves, the two pebbled vertices are adjacent.

▶ **Lemma 2.2.** *Assume that Spoiler has a continuous strategy allowing him to win the 2-pebble Σ_1 game on G_0 and H_0 in r moves. Then, for each $i \geq 1$,*

1. $D^2_{\Sigma_i}(G_i, H_i) < r + i$;
2. $D^2_{\Sigma_i}(G_i, H_i) \geq D^2_{\Pi_{i+1}}(G_i, H_i) \geq D^2_{\exists}(G_0, H_0)$;
3. $D^2_{\Pi_i}(G_i, H_i) = \infty$;
4. *If, moreover, Spoiler has a continuous strategy allowing him to win the 2-pebble Σ_2 game on G_0 and H_0 in s moves, then $D^2_{\Sigma_{i+1}}(G_i, H_i) < s + i$.*

Proof. **1.** In the base case of $i = 1$ Spoiler is able to win the Σ_1 game on G_1 and H_1 in r moves. He forces the Σ_1 game on G_0 and H_0 by playing continuously inside the G_0-part of G_1 and wins by assumption. Furthermore, we recursively describe a strategy for Spoiler in the Σ_{i+1} game on G_{i+1} and H_{i+1} and inductively prove that it is winning. For each i, the strategy will be continuous, and the vertex pebbled in the first round will be adjacent to the root. Note that this is true in the base case.

In the first round Spoiler pebbles the root of the H_i-branch of G_{i+1}. Duplicator is forced to pebble the root of one of the G_i-branches of H_{i+1}. Indeed, if she pebbles a gray vertex at the different distance from the root of H_{i+1}, then Spoiler pebbles a shortest possible path upwards in G_{i+1} or H_{i+1} and wins once he reaches a non-gray vertex. In the second round

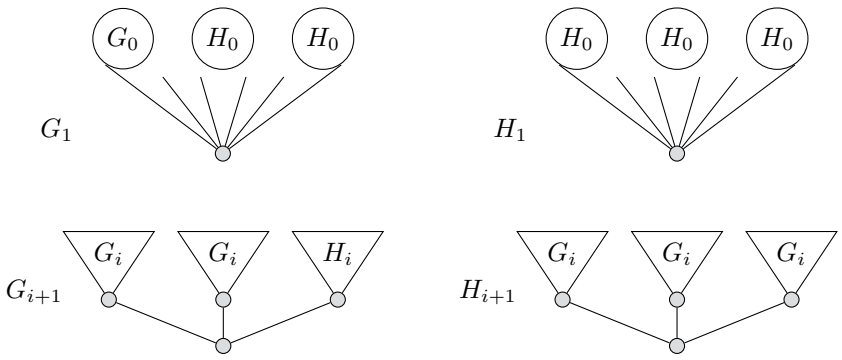


Figure 2 The lifting construction.

Spoiler jumps to this G_i-branch and, starting from this point, forces the Σ_i game on G_i and H_i by playing recursively and, hence, continuously. The only possibility for Duplicator to avoid the recursive play and not to lose immediately is to pebble a gray vertex below. In this case Spoiler wins in altogether $i+1$ moves by pebbling a path upwards in the graph where he stays, as already explained. If the game goes recursively, then by the induction assumption Spoiler needs less than $1+r+i$ moves to win.

2. In the base case of $i=1$ we have to design a strategy for Duplicator in the Π_2 game on G_1 and H_1. First of all, Duplicator pebbles the gray vertex always when Spoiler does so. Furthermore, whenever Spoiler pebbles a vertex in an H_0-branch of G_1 or H_1, Duplicator pebbles the same vertex in an H_0-branch of the other graph. It is important that, if the pebbles are in two different H_0-branches of G_1 or H_1, Duplicator has a possibility to pebble different H_0-branches in the other graph. It remains to describe Duplicator's strategy in the case that Spoiler moves in the G_0-branch of G_1. Note that once Spoiler does so, he cannot change the graph any more. In this case, Duplicator chooses a free H_0-branch in H_1 and follows her optimal strategy in the Σ_1 game on G_0 and H_0. Since the gray vertex is universal in both graphs and the G_0- and H_0-branches are isolated from each other, Spoiler wins only when he wins the Σ_1 game on G_0 and H_0, which is possible in $D^2_\exists(G_0,H_0)$ moves at the earliest.

In the Π_{i+2} game on G_{i+1} and H_{i+1} Duplicator plays similarly. She always respects the root vertex, the G_i-branches, and takes care that the pebbled vertices are either in the same or in distinct G_i-branches in both graphs. Once Spoiler moves in the H_i-branch of G_{i+1}, Duplicator invokes her optimal strategy in the Σ_{i+1} game on H_i and G_i, what is the same as the Π_{i+1} game on G_i and H_i. There is no other way for Spoiler to win than to win this subgame. By the induction assumption, this takes at least $D^2_\exists(G_0,H_0)$ moves.

3. By induction on i, we show that Duplicator has a strategy allowing her to resist arbitrarily long in the Π_i game on G_i and H_i. An important feature of the strategy is that Duplicator will always respect the distance of a pebbled gray vertex from the root. In the base case of $i=1$, such a strategy exists because in G_1 there are two copies of H_0, where Duplicator can mirror Spoiler's moves. In the Π_{i+1} game on G_{i+1} and H_{i+1}, Duplicator makes use of the existence of two copies of G_i in both graphs. Whenever Spoiler pebbles the root vertex or moves in a G_i-part in any of G_{i+1} and H_{i+1}, Duplicator mirrors this move in the other graph. Whenever Spoiler moves for the first time in the H_i-part of G_{i+1}, Duplicator responds in a free G_i-part of H_{i+1} according to her level-preserving strategy for the Π_i game on G_i and H_i, that exists by the induction assumption. When Spoiler moves in the H_i-part also with the other pebble, Duplicator continues playing in the same G_i-part of H_{i+1} following the same strategy.

4. Spoiler has a recursive winning strategy for the Σ_{i+1} game on G_i and H_i similarly to the proof of part 1. ◀

3 Alternation function for FO^k over trees

▶ **Theorem 3.1.** *$A^2(n) > \log_3 n - 2$ over colored trees.*

Proof. Applying the lifting construction described in Section 2 to a pair of single-vertex, differently colored graphs G_0 and H_0, we obtain the sequence of pairs of colored trees G_i and H_i with $v(G_i)=v(H_i)$ as shown in Fig. 3. For $i\geq 1$, we have $D^2_{\Sigma_i}(G_i,H_i)\leq i$ by part 1 of Lemma 2.2 and $D^2_{\Pi_i}(G_i,H_i)=\infty$ by part 3 of this lemma. It follows that $A^2(n_i)\geq i$ for $n_i=v(G_i)$. Note that $n_i=3n_{i-1}+1$, where $n_0=1$. Therefore $n_i=3^i+\frac{3^i-1}{2}$, which implies that $A^2(n_i)>\log_3 n_i-1$.

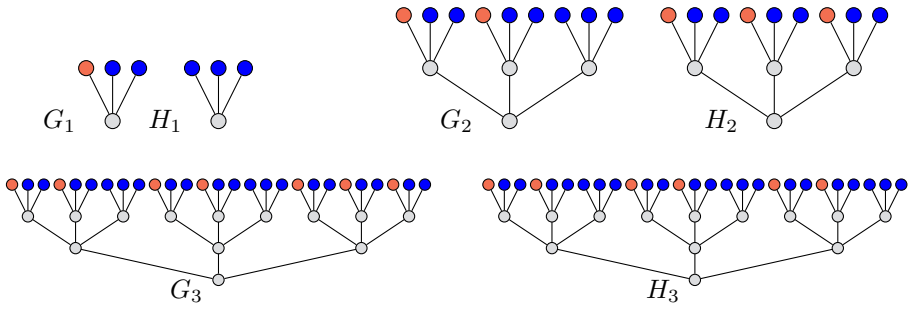


Figure 3 Proof of Theorem 3.1.

Consider now an arbitrary n and suppose that $n_i \le n < n_{i+1}$, i.e., $n_i \le n \le 3n_i$. We can increase the number of vertices in G_i and H_i to n by attaching $n - n_i$ new gray leaves at the root. Since this does not change the parameters $D^2_{\Sigma_i}(G_i, H_i)$ and $D^2_{\Pi_i}(G_i, H_i)$, we get $A^2(n) \ge A^2(n_i) > \log_3 n - 2$. ◀

Theorem 3.1 generalizes to any k-variable logic and, if $k > 2$, then no vertex coloring is needed any more.

▶ **Theorem 3.2.** *If $k \ge 3$, then $A^k(n) > \log_{k+1} n - 2$ over uncolored trees.*

Proof. Notice that the lifting construction of Lemma 2.2 generalizes to $k \ge 3$ variables by adding $k-2$ extra copies of H_0 in G_1 and H_1 and $k-2$ extra copies of G_i in G_{i+1} and H_{i+1}. Similarly to Theorem 3.1, this immediately gives us colored trees G_{i+1} and H_{i+1} such that $D^2_{\Sigma_i}(G_i, H_i) \le i$ and $D^k_{\Pi_i}(G_i, H_i) = \infty$ for all $i \ge 1$.

In order to remove colors from G_i and H_i, we construct these graphs recursively in the same way but now, instead of red and blue one-vertex graphs, we start with $G_0 =$  and $H_0 =$; see Fig. 4. Note that in the course of construction G_0 and H_0 will be handled as rooted trees (otherwise they are isomorphic).

We now claim that for the uncolored trees G_i and H_i it holds $D^3_{\Sigma_i}(G_i, H_i) \le i + 5$ and $D^k_{\Pi_i}(G_i, H_i) = \infty$. The latter claim is true exactly by the same reasons as in the colored case: since the number of Spoiler's jumps is bounded, Duplicator is always able to ensure playing on isomorphic branches. To prove the former bound, we will show that Spoiler can win similarly to the colored case playing with 3 pebbles.

Note that in the uncolored version of G_i and H_i, all formerly gray vertices have degree $k+1$, red vertices have degree 3, and blue vertices have degree 2. A typical ending of the game on the colored trees was that Spoiler pebbles a red vertex while Duplicator is forced to pebble a blue one. Now this corresponds to pebbling a vertex u of degree 3 by Spoiler and a vertex v of degree 2 by Duplicator. Having 4 pebbles, Spoiler would win by pebbling the three neighbors of u. Having only 3 pebbles, Spoiler first pebbles two neighbors u_1 and

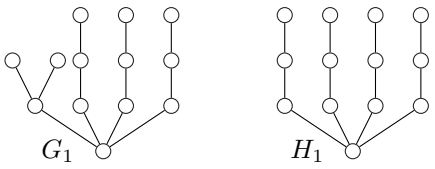


Figure 4 Proof of Theorem 3.2. The uncolored versions of G_1 and H_1 for 3-variable logic.

u_2 of u (in fact, one neighbor is already pebbled immediately before u). Duplicator must respond with the two neighbors v_1 and v_2 of v. In the next round Spoiler moves the pebble from u to its third neighbor u_3. Duplicator must remove the pebble from v and place it on some vertex v_3 non-adjacent to both v_1 and v_2. Note that, while the distance between any two vertices of u_1, u_2, and u_3 equals 2, there is a pair of indices s and t such that v_s and v_t are at the distance more than 2. Spoiler now wins by moving the pebble from u_q to u, where $\{q\} = \{1,2,3\} \setminus \{s,t\}$.

It remains to note that with 3 pebbles Spoiler is able to force climbing upwards in the trees and, hence, he can follow essentially the same winning strategy as in the colored case. Duplicator can deviate from this scenario only in the first round. Recall that in this round Spoiler pebbles a vertex u at the distance 1 from the root level, having degree at least 3. Suppose that Duplicator responds with pebbling a vertex v at the distance more than 1 from the root level. If $i = 1$, then v is of degree at most 2, and Spoiler wins as explained above. If $i \geq 2$, then v can have degree 3 or $k+1$. In this case Spoiler forces climbing up and wins by pebbling a leaf above a formerly blue vertex because by this point Duplicator has already reached the highest possible level. Suppose now that in the first round Duplicator pebbles the root vertex. Then Spoiler puts a second pebble on the root of his graph, Duplicator is forced to pebble a vertex one level higher, and Spoiler again wins by forcing climbing up from the root to the highest leaf level.

Thus, we have shown that $A^k(n_i) \geq i$ for $n_i = v(G_i)$. Since $n_i = (k+1)n_{i-1} + 1$ and $n_0 = 3$, we have $n_i = 3(k+1)^i + \frac{(k+1)^i - 1}{k}$, which implies that $A^2(n_i) > \log_{k+1} n_i - 1$. Like to the proof of Theorem 3.1, this bound extends to all n at the cost of decreasing it by 1. ◀

Theorems 3.1 and 3.2 are optimal in the sense that they cannot be extended to FO^2 over uncolored trees. The reason is that the quantifier alternation hierarchy of FO^2 over uncolored graphs collapses to the second level.

▶ **Theorem 3.3.** *If a class of uncolored graphs is definable by a first-order formula with two variables, then it is definable by a first-order formula with two variables and one quantifier alternation.*

We now show that the bound of Theorem 3.2 is tight up to a constant factor. The following theorem implies that, if $k \geq 3$, then $A^k(n) < (k+3)\log_2 n$ over colored trees. The proof easily extends to the class of all binary structures whose Gaifman graph is a tree.

▶ **Theorem 3.4.** *Let $k \geq 3$. If $D^k(T,T') < \infty$ for colored trees T and T', then*

$$D^k(T,T') < (k+3)\log_2 n \tag{2}$$

where n denotes the number of vertices in T.

Proof. Let $T - v$ denote the result of removal of a vertex v from the tree T. The component of $T - v$ containing a neighbor u of v will be denoted by T_{vu} and considered a rooted tree with the root at u. A similar notation will apply also to T'. The rooted trees T_{vu} will be called *branches of T at the vertex v*. Let $\tau(v)$ denote the maximum number of pairwise isomorphic branches at v. We define the *branching index* of T by $\tau(T) = \max_v \tau(v)$. In order to prove the theorem, we will show that the bound (2) is true for any non-isomorphic colored trees with branching index at most k and that $D^k(T,T') = D^k(T \bmod k, T' \bmod k)$ for $T \bmod k$ and $T' \bmod k$ being "truncated" versions of T and T' whose branching index is bounded by k. We first handle the latter task.

The following fact easily follows from the trivial observation that k pebbles can be placed on at most k isomorphic branches.

Claim A. Let T be a colored tree. Suppose that T has more than k isomorphic branches at a vertex v. Remove all but k of them from T and denote the resulting tree by $\hat{T}$. Then $D^k(T,G) = D^k(\hat{T},G)$ for any colored graph G. $\triangleleft$

The truncated tree T mod k is obtained from T by a series of truncations as in Claim A. The truncations steps should be done from the top to the bottom in order to exclude appearance of new isomorphic branches in the course of the procedure. In order to define the "top and bottom" formally, recall that the *eccentricity* of a vertex v in a graph G is defined by $e(v) = \max_u dist(v,u)$, where $dist(v,u)$ denotes the distance between the two vertices. The *diameter* and the *radius* of G are defined by $d(G) = \max_v e(v)$ and $r(G) = \min_v e(v)$ respectively. A vertex v is *central* if $e(v) = r(G)$. For trees it is well known (e.g., [7, Chapter 4.2]) that if $d(T)$ is even, then T has a unique central vertex c. If $d(T)$ is odd, then T has exactly two central vertices c_1 and c_2, that are adjacent. Let us regard the central vertices as lying on the bottom level and the tree T as growing upwards. The height of a vertex is then its distance to the nearest central vertex. Starting from the highest level and going downwards, for each vertex v we cut off extra branches at v if their number exceeds k. Note that this operation can increase the number of isomorphic branches from vertices in lower levels but cannot do this for vertices in higher levels. Therefore, the resulting tree T mod k has branching index at most k.

Applying repeatedly Claim A, we arrive at the equality $D^k(T,T') = D^k(T \bmod k, T' \bmod k)$. Note that $T \bmod k \ncong T' \bmod k$ because it is assumed that $D^k(T,T') < \infty$. Thus, we have reduced proving the bound (2) to the case that T and T' are non-isomorphic and both have branching index at most k. Therefore, below we make this assumption.

We have to show that Spoiler is able to win the k-pebble game on such T and T' in less than $(k+3)\log_2 n$ moves. Below we will actively exploit the following fact ensured by a standard halving strategy for Spoiler.

Claim B. Suppose that in the 3-pebble Ehrenfeucht-Fraïssé game on graphs G and H some two vertices $x, y \in V(G)$ at distance n are pebbled so that their counterparts $x', y' \in V(H)$ are at a strictly larger distance. Then Spoiler can win in at most $\lceil \log n \rceil$ extra moves. $\triangleleft$

Every tree T has a single-vertex *separator*, that is, a vertex v such that no branch of T at v has more than $n/2$ vertices; see, e.g., [7, Chapter 4.2]. The idea of Spoiler's strategy is to pebble such a vertex and to force further play on some non-isomorphic branches of T and T', where the same strategy can be applied recursively. This scenario was realized in [8, Theorem 5.2] for first-order logic with counting quantifiers. Without counting, we have to use some additional tricks that are based on boundedness of the branching index. Below, by $N(v)$ we will denote the neighborhood of a vertex v.

Thus, in the first round Spoiler pebbles a separator v in T and Duplicator responds with a vertex v' somewhere in T'. Since $T \ncong T'$, there is an isomorphism type $\mathcal{B}$ of a branch of T at v that appears with different multiplicity among the branches of T' at v'. Spoiler can use this fact to force pebbling vertices $u \in N(v)$ and $u' \in N(v')$ so that the rooted trees T_{vu} and $T'_{v'u'}$ are non-isomorphic (the pebbles on v and v' can be reused but, finally, v and v' have to remain pebbled as well). This is easy to do if the multiplicity of $\mathcal{B}$ in one of the trees is at most $k-2$. If this multiplicity is $k-1$ in one tree and k in the other, then Spoiler can do it still with k pebbles like as in the proof of Theorem 3.2. This phase of the game can take $k+2$ rounds.

The next goal of Spoiler is to force pebbling adjacent vertices v_1 and u_1 in T_{vu} and adjacent vertices v'_1 and u'_1 in $T'_{v'u'}$ so that $T_{v_1u_1} \ncong T'_{v'_1u'_1}$ and

$$v(T_{v_1u_1}) \leq v(T_{vu})/2 \text{ or } v(T'_{v'_1u'_1}) \leq v(T_{vu})/2. \tag{3}$$

Once this is done, the same will be repeated recursively (with the roles of T and T' swapped if only the second inequality in (3) is true).

To make the transition from T_{vu} to $T_{v_1u_1}$, Spoiler first pebbles a separator w of T_{vu}. Note that Duplicator is forced to respond with a vertex w' in $T'_{v'u'}$. Otherwise we would have $dist(w,u) = dist(w,v) - 1$ while $dist(w',u') = dist(w',v') + 1$. Therefore, some distances among the three pebbled vertices would be different in T and in T' and Spoiler could win in less than $\log v(T_{vu}) + 1$ moves by Claim B.

Let $T_{w\setminus u}$ denote the rooted tree obtained by removing from T the branch at w containing u and rooting the resulting tree at w. Note that $V(T_{w\setminus u}) \subset V(T_{vu})$. We consider a few cases.

Case 1: $T_{w\setminus u} \ncong T'_{w'\setminus u'}$. In the trees $T_{w\setminus u}$ and $T'_{w'\setminus u'}$ we will consider branches at their roots w and w'.

Subcase 1-a: $T_{w\setminus u}$ contains a branch of isomorphism type $\mathcal{B}$ that has different multiplicity in $T'_{w'\setminus u'}$. As above, Spoiler can use k pebbles and $k+1$ moves to force pebbling vertices $x \in N(w)$ and $x' \in N(w')$ such that $T_{wx} \ncong T'_{w'x'}$ and

$$T_{wx} \in \mathcal{B} \text{ or } T'_{w'x'} \in \mathcal{B}. \tag{4}$$

The pebbles occupying v, v' and u, u' can be released. The pebbles on w and w' can also be reused but, finally, w and w' have to remain pebbled. The branches T_{wx} and $T'_{w'x'}$ will now serve as $T_{v_1u_1}$ and $T'_{v'_1u'_1}$. Condition (3) follows from (4) because w is a separator of T_{vu}.

Subcase 1-b: $T_{w\setminus u}$ does not contain any branch as in Subcase 1-a. In this subcase there is a vertex $x' \in N(w')$ such that $T'_{w'x'}$ is a branch of $T'_{w'\setminus u'}$ and the isomorphism type of $T'_{w'x'}$ does not appear in $T_{w\setminus u}$. Spoiler moves the pebble from v' to x'. Suppose that Duplicator responds with $x \in N(w)$. If x lies on the path between u and w (while x' does not lie on the path between u' and w'), then equality of distances among the pebbled vertices cannot be preserved, and Spoiler wins by Claim B. If x does not lie between u and w, then T_{wx} is a branch of T_{vu} at the vertex w. The first equality in Condition (3) is then true because w is a separator of T_{vu}. In this case, T_{wx} and $T'_{w'x'}$ can serve as $T_{v_1u_1}$ and $T'_{v'_1u'_1}$.

Case 2: $T_{w\setminus u} \cong T'_{w'\setminus u'}$. We assume that $dist(u,w) = dist(u',w')$ because otherwise Spoiler wins by Claim B. For a vertex y on the path between u and w, let $T_{y\setminus u,w}$ denote the rooted tree obtained by removing from T the branches at y containing u and w and rooting the resulting tree at y. The rooted tree $T_{u\setminus v,w}$ is defined similarly. Note that $T_{u\setminus v,w}$ and each $T_{y\setminus u,w}$ are parts of a branch of T_{vu} at the vertex w and, therefore, have at most $v(T_{vu})/2$ vertices. Given y between u and w, by y' we will denote the vertex lying between u' and w' at the same distance to these vertices as y to u and w. Since $T_{vu} \ncong T'_{v'u'}$, we must have

$$T_{y\setminus u,w} \ncong T'_{y'\setminus u',w'} \quad \text{for some } y \text{ or} \tag{5}$$

$$T_{u\setminus v,w} \ncong T'_{u'\setminus v',w'}. \tag{6}$$

Assume that Condition (5) is true and fix such y.

Subcase 2-a: $T_{y\setminus u,w}$ contains a branch of isomorphism type $\mathcal{B}$ that has different multiplicity in $T'_{y'\setminus u',w'}$. Spoiler moves the pebble from v to y. Duplicator is forced to move the pebble from v' to y'. The pebbles occupying u, u' and w, w' can now be released. Spoiler proceeds

similarly to Subcase 1-a and forces pebbling vertices $z \in N(y)$ and $z' \in N(y')$ such that $T_{yz} \ncong T'_{y'z'}$ and one of these trees has isomorphism type $\mathcal{B}$ and, hence, is as small as desired.

Subcase 2-b: $T_{y\setminus u,w}$ does not contain any branch as in Subcase 2-a. In this subcase there is a vertex $z' \in N(y')$ such that $T'_{y'z'}$ is a branch of $T'_{y'\setminus u',w'}$ whose isomorphism type does not appear in $T_{y\setminus u,w}$. Similarly to Subcase 1-b, Spoiler aims to pebble y' and z' while forcing Duplicator to respond with y and $z \in N(y)$ such that T_{yz} is a part of $T_{y\setminus u,w}$. This will ensure that $T_{yz} \ncong T'_{y'z'}$ and that T_{yz} is small enough. Now Spoiler's task is more complicated because he has to prevent Duplicator from pebbling z on the path between u and w. Since this requires keeping the pebbles on u, u' and w, w', Spoiler cannot pebble both y' and z' if there are only $k = 3$ pebbles. In this case he first pebbles the vertex z' by the pebble released from v. Let z be Duplicator's response. If z is in $N(y)$ and does not lie between u and w, Spoiler succeeds by moving the pebble from u' to y'. Duplicator is forced to move the pebble from u to y because w' remains pebbled and, therefore, the position of y is determined by the distances to z and w. If z is not in $N(y)$ or lies between u and w, then Spoiler wins because $dist(z, u) \neq dist(z', u')$ or $dist(z, w) \neq dist(z', w')$

An analysis of the case (6) is quite similar. The role of the triple (u, y, w) is now played by the triple (v, u, w).

Note that the transition from T_{vu} to $T_{v_1u_1}$ takes at most $k + 3$ rounds. Also, 2 rounds suffice to win the game once the current subtree T_{vu} has at most 2 vertices. The number of transitions from the initial branch of order at most $n/2$ to one with at most 2 vertices is bounded by $\log_2 n - 1$ because $v(T_{vu})$ becomes twice smaller each time. It follows that Spoiler wins the game on T and T' in less than $k + 2 + (\log_2 n - 1)(k + 3) + 2 \leq (k + 3)\log_2 n + 1$ moves. The additive term of 1 can be dropped because if pebbling the initial branch takes no less than $k + 2$ moves, then the size of this branch will actually not exceed n/k. ◀

4 Alternation function for FO^2 over colored graphs

Theorem 3.1 gives us a logarithmic lower bound on the alternation function $A^2(n)$, which is true even for trees. Over all colored graphs, we now prove a linear lower bound. Along with the general upper bound $A^2(n) \leq n + 1$, it shows that $A^2(n)$ has a linear growth.

▶ **Theorem 4.1.** *$A^2(n) > n/8 - 2$.*

Proof. For each integer $m \geq 2$, we will construct colored graphs G and H, both with $n = 8m - 4$ vertices, that can be distinguished in FO^2 with $m - 2$, but no less than that, alternations. The graph $G = 2G_m$ is the union of two disjoint copies of the same graph G_m and, similarly, $H = 2H_m$ where G_m and H_m are defined as follows. Each of G_m and H_m is obtained by merging two building blocks A_m and B_m shown in Fig. 5. The colored graph A_m is a "ladder" with m horizontal rungs, each having 2 vertices. The vertices on the bottom rung are colored in green, the vertices on the top rung are colored one in red and the other in blue, the remaining $2m - 4$ vertices are white (uncolored). The graph B_m is obtained from A_m by recoloring red in apricot and blue in cyan. A_m and B_m are glued together at the green vertices. There are two ways to do this, and the resulting graphs G_m and H_m are non-isomorphic. Let α^+ (resp. α^-) denote the partial isomorphism from G_m to H_m identifying the A_m-parts (resp. the B_m-parts) of these graphs.

We will design a strategy allowing Spoiler to win the $(m - 2)$-alternation (i.e., Σ_{m-1} or Π_{m-1}) 2-pebble Ehrenfeucht-Fraïssé game on G and H and a strategy allowing Duplicator to win the $(m - 3)$-alternation game. Before playing on G and H, we analyse the 2-pebble game on G_m and H_m. Spoiler can win this game as follows. In the first round he pebbles the

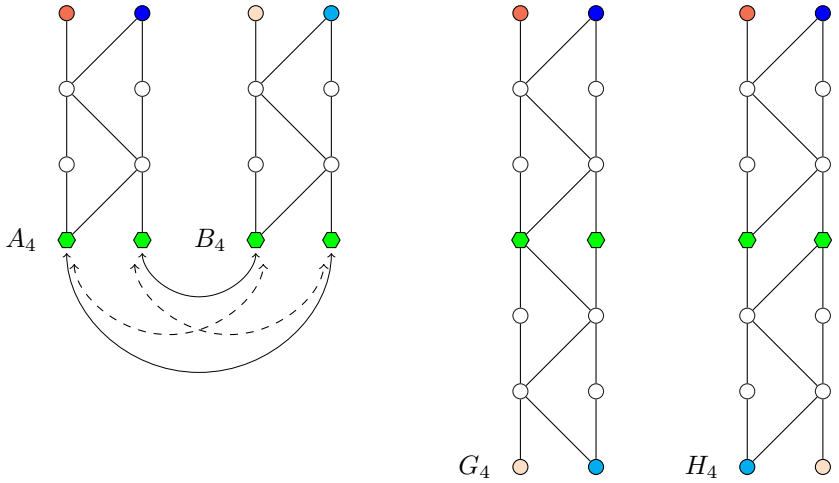


Figure 5 Proof of Theorem 4.1.

left green vertex in G_m; see Fig. 5. Not to lose immediately, Duplicator responds either with the left or with the right green vertex in H_m. The corresponding partial isomorphism can be extended to α^+ in the former case and to α^- in the latter case (but not to both α^+ and α^-). These two cases are similar, and we consider the latter of them, where there is no extension to α^+ and hence Spoiler has a chance to win playing in the A_m-parts of G_m and H_m.

In the second round Spoiler pebbles the upright neighbor of the left green vertex in G_m. His goal in subsequent rounds is to force pebbling, one by one, edges along the upright paths to the red vertex in G_m and to the blue vertex in H_m. If Duplicator makes a step down, Spoiler wins by reaching the top rung sooner than Duplicator. If Duplicator moves all the time upward, starting from the third round of the game she has a possibility to slant. Spoiler prevents this by changing the graph. Note that in one of the graphs there is only one way upstairs, and Spoiler always leaves this graph for Duplicator. In this way Spoiler wins by making m moves and alternating between the graphs $m-2$ times.

The strategy we just described is inoptimal with respect to the alternation number. In fact, Spoiler can win the game on G_m and H_m with no alternation at all by pebbling in the first round the right green vertex in G_m. If Duplicator responds with the left green vertex in H_m, Spoiler puts the second pebble on the non-adjacent vertex in the next upper rung. Duplicator is forced to play in a different rung of H_m because otherwise she would violate the non-adjacency relation. If in the first round Duplicator responds with the right green vertex, Spoiler plays similarly, but in the lower rung of G_m. In any case, the second pebble is closer to the red or to the apricot vertex in G_m than in H_m, which makes Spoiler's win easy.

Nevertheless, the former, $(m-2)$-alternation strategy has an advantage: Spoiler ensures that the two pebbled vertices are always adjacent. By this reason, the same strategy can be used by Spoiler to win also the game on $G = 2G_m$ and $H = 2H_m$. Once Duplicator steps aside to another copy of G_m or H_m, she immediately loses.

The partial isomorphism α^+ from G_m to H_m determines two partial isomorphisms α_0^+ and α_1^+ from $G = 2G_m$ to $H = 2H_m$ identifying the two A_m-parts of G with the two A_m-parts of H. Similarly, α^- gives rise to two partial isomorphisms α_0^- and α_1^-.

We now show that the number of alternations $m-2$ is optimal for the game on G and H. Fix an integer a such that Spoiler has a winning strategy in the a-alternation 2-pebble game on G and H. For this game, let us fix an arbitrary winning strategy for Spoiler and a strategy for Duplicator satisfying the following conditions.

- Duplicator always respects vertex rungs.
- Additionally, Duplicator respects adjacency.
- Duplicator respects also non-adjacency. Moreover, whenever Spoiler violates adjacency of the vertices pebbled in one graph, Duplicator responds so that the vertices pebbled in the other graph are not only non-adjacent but even lie in different G_m- or H_m-components.
- If Spoiler pebbles a vertex above the green rung and the three preceding rules still do not determine Duplicator's response uniquely, then she responds according to α_0^+ or α_1^+; in a similar situation below the green rung, she plays according to α_0^- or α_1^-.

Note that these rules uniquely determine Duplicator's moves on non-green vertices provided one pebble is already on the board. In particular, the choice of α_0^+ or α_1^+ in the last rule depends on the component where this pebble is placed.

Let $u_i \in V(G)$ and $v_i \in V(H)$ denote the vertices pebbled in the i-th round of the game. We now highlight a crucial property of Duplicator's strategy. Suppose that u_i, v_i and u_{i+1}, v_{i+1} are in the A_m-parts of G and H and that u_{i+1} and v_{i+1} are non-green. Then the following conditions are met.

- If u_i and u_{i+1} are non-adjacent, then $\alpha_s^+(u_{i+1}) = v_{i+1}$ for $s = 0$ or $s = 1$.
- If u_i and u_{i+1} (as well as v_i and v_{i+1}) are adjacent and $\alpha_s^+(u_i) = v_i$ for $s = 0$ or $s = 1$, then $\alpha_s^+(u_{i+1}) = v_{i+1}$ for the same s.

The similar property holds if the pebbles are in the B_m parts.

Suppose that Spoiler wins in the r-th round. Note that Duplicator's strategy allows Spoiler to win only when u_r and v_r are on the top or on the bottom rungs and have different colors. Since the two cases are similar, assume that Spoiler wins on the top.

Let p be the smallest index such that all vertices in the sequence $u_p, v_p, \ldots, u_r, v_r$ are above the green level. By assumption, $\alpha_s^+(u_r) \neq v_r$ for both $s = 0, 1$. The aforementioned property of Duplicator's strategy implies that, furthermore,

$$\alpha_0^+(u_i) \neq v_i \text{ and } \alpha_1^+(u_i) \neq v_i \text{ for all } i \geq p. \tag{7}$$

Therefore, u_{i+1} and u_i as well as v_{i+1} and v_i are adjacent for all $i \geq p$ (for else Duplicator plays so that $\alpha_s^+(u_{i+1}) = v_{i+1}$ for $s = 0$ or $s = 1$). By the same reason, $p > 1$ and u_{p-1} and u_p are also adjacent. It follows that u_{p-1} and v_{p-1} are green and $\alpha_s^+(u_{p-1}) \neq v_{p-1}$ for both $s = 0, 1$.

Another consequence of (7) is that both vertex sequences $u_{p-1}, u_p, \ldots, u_r$ and $v_{p-1}, v_p, \ldots, v_r$ lie on upright paths. This follows from the fact that either from u_i or from v_i there is only one edge emanating upstairs (also downstairs), and it is upright.

It remains to notice that after each transition to the adjoining rung (i.e., from u_i, v_i to u_{i+1}, v_{i+1} for $i \geq p-1$) Spoiler has to jump to the other graph because otherwise Duplicator will choose the neighbor that ensures $\alpha_s^+(u_{i+2}) = v_{i+2}$ for some value of $s = 0, 1$. This observation readily implies that the number of alternations a cannot be smaller than $m - 2$.

We have shown that $A^2(n) \geq m - 1$ if $n = 8m - 4$. Adding up to seven isolated vertices to both G and H, we get the same bound also for $n = 8m - 3, \ldots, 8m + 3$. Therefore, $A^2(n) \geq (n - 11)/8$ for all n. ◀

5 Succinctness results

Since $D^k_{\Sigma_i}(G, H) = D^k_{\Pi_i}(H, G)$, the following result holds true as well for $\Pi_i \cap \mathrm{FO}^k$.

▶ **Theorem 5.1.** *Let G and H be structures over the same vocabulary. If G is distinguishable from H in $\Sigma_i \cap \mathrm{FO}^k$, then $D^k_{\Sigma_i}(G, H) \leq (v(G)v(H))^{k-1} + 1$.*

In particular, if binary structures G and H have n elements each and G is distinguishable from H in existential two-variable logic, then $D^2_\exists(G,H) \leq n^2+1$. We now show that this bound is tight up to a constant factor. For the existential-positive fragment of FO^2, a quadratic lower bound can be obtained from the benchmark instances for the arc consistency problem going back to [4, 9]; see [1] where also an alternative approach is suggested. We here elaborate on the construction presented in [1]. To implement this idea for existential two-variable logic, we need to undertake a more delicate analysis as the existential-positive fragment is more restricted and simpler.

▶ **Theorem 5.2.** *There are infinitely many colored graphs G and H, both on n vertices, such that G is distinguishable from H in existential two-variable logic and $D^2_\exists(G,H) > n^2/11$.*

Proof. Our construction will depend on an integer parameter $m \geq 2$. We construct a pair of colored graphs G_m and H_m such that G_m is distinguishable from H_m in the existential two-variable logic, both $v(G_m) = O(m)$ and $v(H_m) = O(m)$, and $D^2_\exists(G_m,H_m) = \Omega(m^2)$. Though $v(G_m) < v(H_m)$, later we will be able to increase the number of vertices in G_m to $v(H_m)$.

The graphs have vertices of 4 colors, namely apricot, blue, cyan, and dandelion. G_m contains a cycle of length $3(2m-1)$ where apricot, blue, and cyan alternate in this order; see Fig. 6. H_m contains a similar cycle of length $3 \cdot 2m$. Successive apricot, blue, and cyan vertices will be denoted by a_i, b_i, and c_i in G_m, where $0 \leq i < 2m-1$, and by a'_i, b'_i, and c'_i in H_m, where $0 \leq i \leq 2m-1$. Furthermore, the vertex a_0 is adjacent to a dandelion vertex d_0, and every a'_i except for $i = m$ is adjacent to a dandelion vertex d'_i. This completes the description of the graphs.

By Lemma 2.1, we have to show that Spoiler is able to win the 2-pebble Σ_1 game on G_m and H_m and that Duplicator is able to prevent losing the game for $\Omega(m^2)$ rounds.

Note that, once the pair (a_0, a'_m) is pebbled, Spoiler wins in the next move by pebbling d_0. He is able to force pebbling (a_0, a'_m) as follows. In the first round he pebbles a_0. Suppose that Duplicator responds with a'_s, where $0 \leq s < m$. In a series of subsequent moves, Spoiler goes around the whole circle in G_m, visiting $c_{2m-2}, b_{2m-2}, a_{2m-2}, c_{2m-2}, \ldots$ and using the two pebbles alternately (if $m < s < 2m$, he does the same but in the other direction). As Spoiler comes back to a_0, Duplicator is forced to arrive at a'_{s+1}. The next Spoiler's tour around the circle brings Duplicator to a'_{s+2} and so forth. Thus, the most successful moves for Duplicator in the first round is a'_0. Then Spoiler needs to play $1 + m \cdot 3(2m-1) + 1 = 6m^2 - 3m + 2$ rounds in order to win.

Our next task is to design a strategy for Duplicator allowing her to survive $\Omega(m^2)$ rounds, no matter how Spoiler plays. We will show that Duplicator is able to force Spoiler to pass

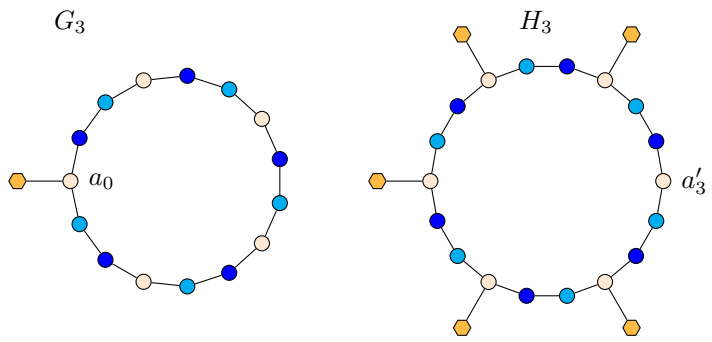


Figure 6 Proof of Theorem 5.2.

around the cycle in G_m many times. A crucial observation is that (a_0, a'_m) is the only pair whose pebbling allows Spoiler to win in one extra move.

Let us regard the additive group $\mathbb{Z}_{2m}$ as a cycle graph with i and j adjacent iff $i-j=\pm 1$. Denote the distance between vertices in this graph by Δ. The same letter will denote the following partial function $\Delta : V(G_m) \times V(H_m) \to \mathbb{Z}$. For two vertices of the same color, say, for a_i and a'_j, we set $\Delta(a_i, a'_j) = \Delta(i, j)$. Note that $\Delta(a_0, a'_m) = m$, which is the largest possible value. Duplicator' strategy will be to keep the value of the Δ-function on the pebbled pair as small as possible.

Specifically, in the first round Duplicator responds to Spoiler's move x with pebbling a vertex x' such that $\Delta(x, x') = 0$ (that is, if $x = a_i, b_i, c_i, d_0$, then $x' = a'_i, b'_i, c'_i, d'_0$ respectively). Suppose that a pair (y, y') is pebbled in the preceding round and Duplicator is still alive. If Spoiler pebbles x in the current round, Duplicator chooses her response x' by the following criteria. Below, $\sim$ denotes the adjacency relation.

- x' should have the same color as x and, moreover, $x' \sim y'$ iff $x \sim y$ (this is always possible unless $(y, y') = (a_0, a'_m)$ and $x = d_0$);
- if there is still more than one choice, x' should minimize the parameter $\Delta(x, x')$.

We do not consider the cases when $x = y$ or when x is pebbled by the pebble removed from y because, in our analysis, we can assume that Spoiler uses an *optimal* strategy, allowing him to win the 2-pebble Σ_1 game on G_m and H_m from the initial position (y, y') in the smallest possible number of rounds (if he does not play optimally, Duplicator survives even longer).

Claim C. If $x \not\sim y$ and $x \neq y$, then $\Delta(x, x') \leq 1$.

Proof of Claim C. Assume first that $x \neq d_0$ and $y \neq d_0$. W.l.o.g., suppose that y and y' are apricot and, specifically, $y' = a'_j$ (the blue and the cyan cases are symmetric to the apricot case). Not to lose immediately, Duplicator cannot pebble x' in $\{c'_{j-1}, a'_j, b'_j\}$, where $j-1$ is supposed to be an element of $\mathbb{Z}_{2m}$. This can obstruct attaining $\Delta(x, x') = 0$ (if $x \in \{c_{j-1}, a_j, b_j\}$), but then there is a choice of x' with $\Delta(x, x') = 1$.

Assume now that $x = d_0$. Then $x' = d'_0$ if $y' \neq a'_0$ and $x' = d'_1$ otherwise. In both cases $\Delta(x, x') \leq 1$. Finally, let $y = d_0$ and $y' = d'_j$. Then the value $x' = a'_j$ is forbidden and, if this prevents $\Delta(x, x') = 0$, then we have $\Delta(x, x') = 1$. $\triangleleft$

Consider now the dynamical behaviour of $\Delta(x, x')$, assuming that Duplicator uses the above strategy and Spoiler follows an optimal winning strategy. We have $\Delta(x, x') = 0$ at the beginning of the game and $\Delta(x, x') = m$ at the end (that is, in the round immediately before Spoiler wins). Consider the last round of the game where $\Delta(x, x') \leq 1$. By Claim C, starting from the next round Spoiler always moves along an edge in G_m. Note that, from now on, visiting d_0 earlier than in the very last round would be inoptimal. Therefore, Spoiler walks along the circle. Another consequence of optimality is that he moves always in the same direction.

W.l.o.g., we can suppose that Spoiler moves in the ascending order of indices. Note that $\Delta(x, x')$ increases by 1 only under the transition from $x = a_{2m-2}$ to $x = a_0$ (at this point, the index of x makes a jump in $\mathbb{Z}_{2m}$, while the index of x' moves along $\mathbb{Z}_{2m}$ always continuously). In order to increase $\Delta(x, x')$ from 1 to m, the edge $a_{2m-2}a_0$ must be passed $m-1$ times. It follows that, before Spoiler wins, the game lasts at least $2+(m-2)\cdot 3(2m-1) = 6m^2-15m+8$ rounds.

Note that $v(G_m) = 6m-2$ and $v(H_m) = 8m-1$. In order to make the number of vertices in both graphs $n = 8m-1$, let m be multiple of 3 and add two new connected components to G_m, namely the cycle of length $2m$ with alternating colors apricot, blue, and cyan and one isolated vertex of any color. Spoiler can still win by playing in the old component. Since

playing in the new components does not help him, the game on the modified G_m and the same H_m lasts at least $6m^2 - 15m + 8 = \frac{3}{32}\,n^2 - O(n)$ rounds. ◀

Lifting it higher

Since $D^2_{\Sigma_i}(G,H) = D^2_{\Pi_i}(H,G)$, the following results hold true as well for $\Pi_i \cap \mathrm{FO}^2$.

▶ **Theorem 5.3.** *Let $i \geq 1$. There are infinitely many colored graphs G and H, both on n vertices, such that G is distinguishable from H in $\Sigma_i \cap \mathrm{FO}^2$ and $D^2_{\Sigma_i}(G;H) > \frac{1}{11\cdot 9^i}\,n^2 - \frac{1}{11\cdot 3^i}n$.*

Proof. For infinitely many values of an integer parameter n_0, Theorem 5.2 provides us with colored graphs G_0 and H_0 on n_0 vertices each such that Spoiler has a continuous winning strategy in the 2-pebble Σ_1 game on G_0 and H_0, and $D^2_{\exists}(G_0,H_0) > \frac{1}{11}\,n_0^2$. Let G_i and H_i be now the graphs obtained from G_0 and H_0 by the lifting construction described in Section 2. Note that $v(G_i) = 3\,v(G_{i-1}) + 1$, where $G_0 = G$. It follows that $n = v(G_i) = 3^i n_0 + \frac{3^i-1}{2}$. The graph G_i is distinguishable from H_i in $\Sigma_i \cap \mathrm{FO}^2$ by part 1 of Lemma 2.2. By part 2 of this lemma, we have $D^2_{\Sigma_i}(G_i,H_i) > \frac{1}{11}\,n_0^2$, which implies the bound stated in terms of n. ◀

Using a similar sequence of graphs, we can also show that $\Sigma_i \cap \mathrm{FO}^2$ is more succinct than $\Sigma_{i-1} \cap \mathrm{FO}^2$. Given i, let us construct G_i and H_i starting from the same G_0 as in the proof of Theorem 5.3 and a slightly modified H_0. Specifically, we make all dandelion vertices in H_0 adjacent; see Fig. 6 for G_0 and H_0, where H_0 is still unmodified. This makes part 4 of Lemma 2.2 applicable, which along with part 2 gives us the following result.

▶ **Theorem 5.4.** *For each $i \geq 2$ there are infinitely many colored graphs G and H, both on n vertices, such that $D^2_{\Sigma_i}(G,H) = O(1)$ while $D^2_{\Sigma_{i-1}}(G,H) < \infty$ and $D^2_{\Pi_i}(G,H) = \Omega(n^2)$.*

References

1 Christoph Berkholz and Oleg Verbitsky. On the speed of constraint propagation and the time complexity of arc consistency testing. In K. Chatterjee and J. Sgall, editors, *MFCS'13*, volume 8087 of the Lecture Notes in Computer Science, pages 159–170. Springer, 2013.

2 Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, 12(4):389–410, 1992.

3 Ashok K. Chandra and David Harel. Structure and complexity of relational queries. *J. Comput. Syst. Sci.*, 25(1):99–128, 1982.

4 Rina Dechter and Judea Pearl. A problem simplification approach that generates heuristics for constraint-satisfaction problems. Technical report, Cognitive Systems Laboratory, Computer Science Department, University of California, Los Angeles, 1985.

5 Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite model theory*. Springer, 1995.

6 Neil Immerman and Eric Lander. Describing graphs: A first-order approach to graph canonization. In A. Selman, editor, *Complexity Theory Retrospective*, pages 59–81. Springer, 1990.

7 Øystein Ore. *Theory of graphs*, volume 38 of *Colloquium Publications*. AMS, Providence, R.I., 1962.

8 Oleg Pikhurko and Oleg Verbitsky. Logical complexity of graphs: a survey. In M. Grohe and J. Makowsky, editors, *Model theoretic methods in finite combinatorics*, volume 558 of *Contemporary Mathematics*, pages 129–179. AMS, Providence, RI, 2011.

9 Ashok Samal and Tom Henderson. Parallel consistent labeling algorithms. *International Journal of Parallel Programming*, 16:341–364, 1987.

10 Philipp Weis and Neil Immerman. Structure theorem and strict alternation hierarchy for FO^2 on words. *Logical Methods in Computer Science*, 5(3), 2009.

A The proof of Theorem 3.3

The *complement* of a graph G is the graph on the same vertex set $V(G)$ with any two vertices adjacent if and only if they are not adjacent in G. We call a graph *normal* if it has neither isolated nor universal vertex. Note that a graph is normal iff its complement is normal. For every graph G with at least 2 vertices we inductively define its *rank* $\operatorname{rk} G$.

- Graphs of rank 1 are exactly the empty, the complete, and the normal graphs.
- Graphs of rank 2 are exactly the graphs obtained by adding universal vertices to empty graphs, or isolated vertices to complete graphs, or either universal or isolated vertices to normal graphs.
- If $i \geq 2$, disconnected graphs of rank $i+1$ are obtained from connected graphs of rank i by adding a number of isolated vertices.
- For every i, connected graphs of rank i are exactly complements of disconnected graphs of rank i.

A simple inductive argument on the number of vertices shows that all graphs with at least two vertices get ranked. Indeed, if a graph G is normal, complete, or empty, it receives rank 1. This includes the case that G has two vertices. If G does not belong to any of these three classes, it has either isolated or universal vertices. Since graphs with universal vertices are connected and are the complements of graphs with isolated vertices, it suffices to consider the case that G has isolated vertices. Remove all of them from G and denote the result by G'. Note that G' has less vertices than G but still more than one vertex. By the induction assumption, G' is ranked. If $\operatorname{rk} G' = 1$, then $\operatorname{rk} G = 2$ by definition. If $\operatorname{rk} G' > 1$, then G' must be connected (for else it would be normal). Therefore, $\operatorname{rk} G = \operatorname{rk} G' + 1$ by definition.

We now introduce a ranking of vertices in a graph G. If $\operatorname{rk} G = 1$, then all vertices of G get *rank* 1. Suppose that $\operatorname{rk} G > 1$. If G is disconnected, it has at least one isolated vertex; if G is connected, there is at least one universal vertex. Denote the set of such vertices by ∂G. Every vertex in ∂G is assigned *rank* 1. If $u \notin \partial G$, then it is assigned *rank* one greater than the rank of u in the graph $G - \partial G$. The rank of a vertex u in G will be denoted by $\operatorname{rk} u$. It ranges from the lowest value 1 to the highest value $\operatorname{rk} G$. Note that a vertex u with $\operatorname{rk} u < \operatorname{rk} G$ has the same adjacency to all other vertices of equal or higher rank.

Given an integer $m \geq 1$ and a graph G with $\operatorname{rk} G > m$, we define the *m-tail type* of G to be the sequence $(t_0, t_1, \ldots, t_m)$ where $t_0 \in \{\mathit{conn}, \mathit{disc}\}$ depending on whether G is connected or disconnected and, for $i \geq 1$, $t_i \in \{\mathit{thin}, \mathit{thick}\}$ depending on whether G has one or more vertices of rank i.

Furthermore, we define the *kernel* of a graph G to be its subgraph induced on the vertices of rank $\operatorname{rk} G$. Note that the kernel of any G is a graph of rank 1. We define the *head type* of G to be *empty*, *compl*, or *norma* depending on the kernel. We say that graphs G and H are of the same *type* if $\operatorname{rk} G = \operatorname{rk} H$, G and H have the same head type, and if $\operatorname{rk} G > 1$, then they also have the same m-tail type for $m = \operatorname{rk} G - 1$. The single-vertex graph has its own type.

▶ **Lemma 1.1.**

1. *If G and H are of the same type, then $D^2(G, H) = \infty$.*
2. *If G and H have the same m-tail type, then $D^2(G, H) \geq m$.*

▶ **Lemma 1.2.**

1. *For each m-tail type, the class of graphs of this type is definable by a first-order formula with two variables and one quantifier alternation.*
2. *For each G, the class of graphs of the same type as G is definable by a first-order formula with two variables and one quantifier alternation.*

Let C be a class of graphs definable by a formula with two variables of quantifier depth less than m. By Lemma 1.1, C is the union of finitely many classes of graphs of the same type (each of rank at most m) and finitely many classes of graphs of the same m-tail type. By Lemma 1.2, C is therefore definable by a first-order formula with two variables and one quantifier alternation. To complete the proof of Theorem 3.3, it remains to prove the lemmas.

Proof of Lemma 1.1. 1. Let $\operatorname{rk} G = \operatorname{rk} H = m+1$. Let $V(G) = U_1 \cup \dots U_{m+1}$ and $V(H) = V_1 \cup \dots V_{m+1}$ be the partitions of the vertex sets of G and H according to the ranking of vertices. We will describe a winning strategy for Duplicator in the two-pebble game on G and H. We will call a pair of pebbled vertices $(u, v) \in V(G) \times V(H)$ *straight* if $u \in U_i$ and $v \in V_i$ for the same i. Note that both the kernels U_{m+1} and V_{m+1} contain at least 2 vertices and, since G and H are of the same type, $|U_i| = 1$ iff $|V_i| = 1$. This allows Duplicator to play so that the vertices pebbled in each round form a straight pair and the equality relation is never violated. If the head type of G and H is *empty* or *compl*, this strategy is winning because the adjacency of vertices $u \in U_i$ and $u' \in U_j$ depends only on the indices i and j and is the same as the adjacency of any vertices $v \in V_i$ and $v' \in V_j$. It remains to notice that Duplicator can resist also when the game is played inside the normal kernels U_{m+1} and V_{m+1}. In this case she never loses because, for every vertex in a normal graph, she can find another adjacent or non-adjacent vertex, as she desires.

2. We have to show that Duplicator can survive in at least $m-1$ rounds. Note that both $\operatorname{rk} G \geq m+1$ and $\operatorname{rk} H \geq m+1$. Similarly to part 1, consider partitions $V(G) = U_1 \cup \dots U_{m+1}$ and $V(H) = V_1 \cup \dots V_{m+1}$, where U_{m+1} and V_{m+1} now consist of the vertices whose rank is higher than m. In the first round Duplicator plays so that the pebbled vertices form a straight pair. However, starting from the second round it can be for her no more possible to keep the pebbled pairs straight. Call a pair of pebbled vertices $(u, v) \in V(G) \times V(H)$ *skew* if $u \in U_i$ and $v \in V_j$ for different i and j. Assume that Spoiler uses his two pebbles alternatingly (playing with the same pebble in two successive rounds gives him no advantage). Let (u_r, v_r) denote the pair of vertices pebbled the r-th round. If (u_r, v_r) is skew, let S_r denote the minimum s such that $u_r \in U_s$ or $v_r \in V_s$. If (u_r, v_r) is straight, we set $S_r = m+1$. Our goal is to show that, if $S_r = m+1$, then Duplicator has a non-losing move in the next round such that $S_{r+1} \geq m-1$ and that, as long as $1 < S_r \leq m$, she has a non-losing move such that $S_{r+1} \geq S_r - 1$. This readily implies that Duplicator does not lose the first $m-1$ rounds.

To avoid multiple treatment of symmetric cases, we use the following notation. Let $\{G_1, G_2\} = \{G, H\}$. Let $y_1 \in G_1$ and $y_2 \in G_2$ denote the vertices being pebbled in the round $r+1$, and let $x_1 \in G_1$ and $x_2 \in G_2$ be the vertices pebbled in the round r (in the previous notation, $\{x_1, x_2\} = \{u_r, v_r\}$ and $\{y_1, y_2\} = \{u_{r+1}, v_{r+1}\}$).

Suppose first that $\{x_1, x_2\}$ is a straight pair contained in the slice $U_i \cup V_i$. If $i \leq m$, it makes no problem for Duplicator to move so that the pair $\{y_1, y_2\}$ is also straight. This holds true also if $i = m+1$ and Spoiler pebbles $y_a \in U_j \cup V_j$ with $j \leq m$. Thus, in these cases $S_{r+1} = S_r = m+1$. However, if $i = j = m+1$, moving straight can be always Duplicator's loss. In this case she survives by pebbling a vertex y_{3-a} of rank m or $m-1$, depending on the adjacency relation between x_a and y_a. In this case $S_{r+1} \geq m-1$.

Let us accentuate the property of the vertex ranking that is beneficial to Duplicator in the last case. Recall that, if a vertex u is not in the graph kernel, it has the same adjacency to all other vertices of equal or higher rank. If u is adjacent to all such vertices, we say that u is of *universal type*; otherwise we say that it is of *isolated type*. Duplicator uses the fact that the type of a vertex gets flipped when its rank increases by one.

Suppose now that $\{x_1, x_2\}$ is a skew pair. Let $x_1 \in U_i \cup V_i$ and $x_2 \in U_j \cup V_j$ and, w.l.o.g.,

assume that $i > j$. Since $j = S_r$, it is supposed that $j > 1$. We consider three cases depending on Spoiler's move y_a. In the most favorable for Duplicator case, $\operatorname{rk} y_a < j$. Then Duplicator responds with a vertex y_{3-a} of the same rank, resetting S_{r+1} back to the initial value $m+1$. If Spoiler pebbles a vertex y_2 of $\operatorname{rk} y_2 \geq j$, then Duplicator responds with a vertex y_1 of $\operatorname{rk} y_1 = j$, keeping $S_{r+1} \geq j = S_r$ (unchanged or reset to $m+1$). Finally, consider the case when Spoiler pebbles a vertex y_1 of $\operatorname{rk} y_1 \geq j$. Assume that x_2 is of universal type (the other case is symmetric). If y_1 and x_1 are adjacent, then Duplicator responds with a vertex y_2 of $\operatorname{rk} y_2 = i$, keeping $S_{r+1} \geq S_r$. If y_1 and x_1 are not adjacent, then Duplicator responds with y_2 of $\operatorname{rk} y_2 = j-1$, which is of isolated type. This is the only case when $S_{r+1} = S_r - 1$ decreases. ◀

Proof of Lemma 1.2. 1. Consider an m-tail type $(t_0, t_1, \ldots, t_m)$. Assume that $t_0 = \mathit{conn}$ (the case of $t_0 = \mathit{disc}$ is similar). Let $\sim$ denote the adjacency relation. We inductively define a sequence of formulas $\Phi_s(x)$ with occurrences of two variables x and y and with one free variable:

$$\begin{aligned}
\Phi_1(x) &\stackrel{\text{def}}{=} \forall y\,(y \sim x \vee y = x),\\
\Phi_{2k}(x) &\stackrel{\text{def}}{=} \forall y\,(\Phi_{2k-1}(y) \vee y \not\sim x),\\
\Phi_{2k+1}(x) &\stackrel{\text{def}}{=} \forall y\,(\Phi_{2k}(y) \vee y \sim x \vee y = x).
\end{aligned}$$

Here $\Phi_{2k-1}(y)$ is obtained from $\Phi_{2k-1}(x)$ by swapping x and y. A simple inductive argument shows that, if G is a connected graph and $\operatorname{rk} G$ is greater than an odd (resp. even) integer s, then $G, v \models \Phi_s(x)$ exactly when the vertex v is of universal (resp. isolated) type and $\operatorname{rk} v \leq s$.

Furthermore, we define a sequence of closed formulas Ψ_s with alternation number 1:

$$\begin{aligned}
\Psi_1 &\stackrel{\text{def}}{=} \exists x \Phi_1(x) \wedge \exists x \neg \Phi_1(x),\\
\Psi_2 &\stackrel{\text{def}}{=} \exists x \Phi_1(x) \wedge \exists x \Phi_2(x) \wedge \exists x (\neg \Phi_1(x) \wedge \neg \Phi_2(x)),\\
\Psi_s &\stackrel{\text{def}}{=} \exists x \Phi_1(x) \wedge \exists x \Phi_2(x) \wedge \bigwedge_{i=3}^{s} (\Phi_i(x) \wedge \neg \Phi_{i-2}(x)) \wedge \exists x (\neg \Phi_{s-1}(x) \wedge \neg \Phi_s(x)), \quad s \geq 3.
\end{aligned}$$

Note that a graph G satisfies Ψ_s if and only if G is connected and $\operatorname{rk} G > s$.

We are now able to define the class of graphs of m-tail type $(t_0, t_1, \ldots, t_m)$ by the conjunction

$$\Psi_m \wedge \bigwedge_{i=1}^{m} T_i,$$

where

$$T_i \stackrel{\text{def}}{=} \exists x \exists y\,(x \neq y \wedge \Phi_i(x) \wedge \neg \Phi_{i-2}(x) \wedge \Phi_i(y) \wedge \neg \Phi_{i-2}(y))$$

if $t_i = \mathit{thick}$ and

$$T_i \stackrel{\text{def}}{=} \forall x \forall y\,(\neg \Phi_i(x) \vee \neg \Phi_i(y) \vee \Phi_{i-2}(x) \vee \Phi_{i-2}(y) \vee x = y)$$

if $t_i = \mathit{thin}$ (if $i \leq 2$, the subformulas with non-positive indices should be ignored).

2. The single-vertex graph is defined by a formula $\forall x \forall y\,(x = y)$. The three classes of graphs of rank 1 are defined by the following three formulas:

$$\begin{aligned}
&\exists x \exists y\,(x \neq y) \wedge \forall x \forall y\,(x \not\sim y),\\
&\exists x \exists y\,(x \neq y) \wedge \forall x \forall y\,(x = y \vee x \sim y),\\
&\forall x \exists y\,(x \sim y) \wedge \forall x \exists y\,(x \neq y \vee x \not\sim y).
\end{aligned}$$

Suppose that $\operatorname{rk} G = m+1$ and $m \geq 1$. Let $(t_0, t_1, \ldots, t_m)$ be the m-tail type of G. Assume that G is connected, that is, $t_0 = \mathit{conn}$ (the disconnected case is similar). We use the formulas $\Phi_s(x)$, Ψ_s, and T_i constructed in the first part. If the head type of G is *compl* or *empty* (the former is possible if m is even and the latter if m is odd), then the type of G is defined by

$$\Psi_m \wedge \bigwedge_{i=1}^{m} T_i \wedge \forall x \Phi_{m+1}(x).$$

If the head type of G is *norma*, then the type of G is defined by

$$\Psi_m \wedge \bigwedge_{i=1}^{m} T_i \wedge \neg\exists x \Phi_{m+1}(x).$$

Indeed, $\Psi_m \wedge \bigwedge_{i=1}^{m} T_i$ is true on a graph H if and only if H has the m-tail type $(t_0, t_1, \ldots, t_m)$ and $\operatorname{rk} H \geq m+1$. Let $Q \subset V(H)$ denote the set of vertices not in the tail part. Then Q is a homogeneous set exactly when H satisfies $\forall x \Phi_{m+1}(x)$, and Q spans a normal subgraph exactly when H satisfies $\neg\exists x \Phi_{m+1}(x)$. ◀

B Proof of Theorem 5.1

By Lemma 2.1, we have to prove that, if Spoiler has a winning strategy in the r-round k-pebble Σ_i game on G and H for some r, then he has a winning strategy in the game with $v(G)v(H)+1$ rounds.

The proof is based on a general game-theoretic argument. Consider a two-person game, where the players follow some fixed strategies and one of them wins. Then the length of the game cannot exceed the total number of all possible positions because once a position occurs twice, the play falls into an endless loop. Here it is assumed that the players' strategies are *positional*, that is, that a strategy of a player maps a current *position* (rather than the sequence of all previous positions) to one of the moves available for the player.

Implementing this scenario for the Σ_i game, we have to overcome two complications. First, we have to "reduce" the space $V(G)^k \times V(H)^k$ of all possible positions in the game, which has size $(v(G)v(H))^k$. Second, we have take care of the fact that, if $i > 1$, then Spoiler's play can hardly be absolutely memoryless in the sense that he apparently has to remember the number of jumps left to him or, at least, the graph in which he moved in the preceding round.

We begin with some notation. Let $\bar{u}$ and $\bar{v}$ be tuples of vertices in G and H, respectively, having the same length no more than k. Given $\Xi \in \{\Sigma, \Pi\}$ and $a \geq 1$, let $R(\Xi, a, \bar{u}, \bar{v})$ be the minimum r such that Spoiler has a winning strategy in the Ξ_a game on G and H starting from the initial position $(\bar{u}, \bar{v})$. Given a k-tuple $\bar{w}$ and $j \leq k$, let $\sigma_j \bar{w}$ denote the $(k-1)$-tuple obtained from $\bar{w}$ by removal of the j-th coordinate. Note that, if $\bar{u} \in V(G)^k$ and $\bar{v} \in V(H)^k$, then

$$R(\Xi, a, \bar{u}, \bar{v}) = \min_{1 \leq j \leq k} R(\Xi, a, \sigma_j \bar{u}, \sigma_j \bar{v}). \tag{8}$$

In order to estimate the length of the k-pebble Σ_i game on G and H, we fix a strategy for Duplicator arbitrarily and consider the strategy for Spoiler as described below. For $i \geq 1$, we will say that $\bar{C}_s = (\Xi_s, a_s, \bar{u}_s, \bar{v}_s)$ is the *position after the s-th round* if

- $\Xi_s = \Sigma$ if in the s-th round Spoiler moved in G and $\Xi_s = \Pi$ if he moved in H;
- during the first s rounds Spoiler jumped from one graph to another $i - a_s$ times;

- after the s-th round the pebbles $p_1, \ldots, p_k$ are placed on the vertices $\bar{u} \in V(G)^k$ and $\bar{v} \in V(H)^k$ (we suppose that in the first round Spoiler puts all k pebbles on one vertex).

Furthermore, we will say that $\tilde{C}_s = (\Xi_s, a_s, \tilde{u}_s, \tilde{v}_s)$ is the *position before the* $(s+1)$*-th move* if in the $(s+1)$-th round Spoiler moves the pebble p_j and $\tilde{u}_s = \sigma_j \bar{u}_s$ and $\tilde{v}_s = \sigma_j \bar{v}_s$.

Let us describe Spoiler's strategy. He makes the first move according to an arbitrarily prescribed strategy that is winning for him in the $D^k_{\Sigma_i}(G,H)$-round k-pebble Σ_i game on G and H. If this move is in G, let $\Xi_1 = \Sigma$ and $a_1 = i$; otherwise $\Xi_1 = \Pi$ and $a_1 = i-1$. After Duplicator responses, the position $\bar{C}_1$ is specified. Note that $R(\bar{C}_1) < D^k_{\Sigma_i}(G,H)$.

Suppose that the s-th round has been played and after this we have the position $\bar{C}_s = (\Xi_s, a_s, \bar{u}_s, \bar{v}_s)$. In the next round Spoiler plays with the pebble p_j for the smallest value of j such that

$$R(\tilde{C}_s) = R(\bar{C}_s). \tag{9}$$

Such index j exists by (8). Spoiler makes his move according to a prescribed strategy that is winning for him in the $R(\bar{C}_s)$-round k-pebble $(\Xi_s)_{a_s}$ game on G and H with the initial position $(\tilde{u}_s, \tilde{v}_s)$. If he moves in the same graph as in the s-th round, then $\Xi_{s+1} = \Xi_s$ and $a_{s+1} = a_s$; otherwise Ξ_{s+1} gets flipped and $a_{s+1} = a_s - 1$.

Note that $a_{s+1} \le a_s$ and, if $\Xi_{s+1} \ne \Xi_s$, then $a_{s+1} < a_s$. Since Spoiler in each round uses a strategy optimal for the rest of the game,

$$R(\bar{C}_{s+1}) < R(\bar{C}_s). \tag{10}$$

It follows that the described strategy allows Spoiler to win the Σ_i game on G and H in at most $D^k_{\Sigma_i}(G,H)$ moves.

We now estimate the length of the game from above. Suppose that after the t-th round Duplicator is still alive. Due to (9) and (10),

$$R(\tilde{C}_1) > R(\tilde{C}_2) > \ldots > R(\tilde{C}_t).$$

It follows that the elements of the sequence $\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_t$ are pairwise distinct. We conclude from here that the elements of the sequence $(\tilde{u}_1, \tilde{v}_1), (\tilde{u}_2, \tilde{v}_2), \ldots, (\tilde{u}_t, \tilde{v}_t)$ are pairwise distinct too. Indeed, let $s' > s$. If $a_{s'} = a_s$, then $\Xi_s = \Xi_{s'}$. Since $\tilde{C}_s \ne \tilde{C}_{s'}$, we have $(\tilde{u}_s, \tilde{v}_s) \ne (\tilde{u}_{s'}, \tilde{v}_{s'})$. If $a_{s'} < a_s$, the same inequality follows from the fact that $R(\Xi, a, \tilde{u}, \tilde{v}) \le R(\Xi', a', \tilde{u}, \tilde{v})$ whenever $a' < a$.

Since $(\tilde{u}_s, \tilde{v}_s)$ ranges over $V(G)^{k-1} \times V(H)^{k-1}$, we conclude that $t \le (v(G)v(H))^{k-1}$ and, therefore, Spoiler wins in the round $(v(G)v(H))^{k-1} + 1$ at latest.

Unambiguity and uniformization problems on infinite trees

Marcin Bilkowski* **and Michał Skrzypczak†**

University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
{m.bilkowski,mskrzypczak}@mimuw.edu.pl

Abstract

A nondeterministic automaton is called unambiguous if it has at most one accepting run on every input. A regular language is called unambiguous if there exists an unambiguous automaton recognizing this language. Currently, the class of unambiguous languages of infinite trees is not well-understood. In particular, there is no known decision procedure verifying if a given regular tree language is unambiguous. In this work we study the self-dual class of bi-unambiguous languages — languages that are unambiguous and their complement is also unambiguous. It turns out that thin trees (trees with only countably many branches) emerge naturally in this context.

We propose a procedure P designed to decide if a given tree automaton recognizes a bi-unambiguous language. The procedure is sound for every input. It is also complete for languages recognisable by deterministic automata. We conjecture that P is complete for all inputs but this depends on a new conjecture stating that there is no MSO-definable choice function on thin trees. This would extend a result by Gurevich and Shelah on the undefinability of choice on the binary tree.

We provide a couple of equivalent statements to our conjecture, we also give several related results about uniformizability on thin trees. In particular, we provide a new example of a language that is not unambiguous, namely the language of all thin trees. The main tool in our studies are algebras that can be seen as an adaptation of Wilke algebras to the case of infinite trees.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases nondeterministic automata, infinite trees, MSO logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.81

1 Introduction

Infinite trees form a rich class of models, one infinite tree may encode whole set of finite words or a strategy in an infinite duration game. Therefore, the decidability of Monadic Second-Order (MSO) logic over infinite trees [19] is often called the *mother of all decidability results*. The proof of this decidability result follows a similar line as in the case of finite words [27] — we find a model of automata that are equivalent in expressive power with MSO logic and have decidable emptiness problem.

The proof of Rabin's theorem deals with nondeterministic automata as deterministic ones have strictly smaller expressive power. It is one of the main reasons why many problems about regular languages of infinite trees are very hard. For example, no algorithm is known

* This paper has been partially supported by the Polish Ministry of Science grant no. N206 567840.
† Author supported by ERC Starting Grant "Sosna" no. 239850.

to decide the parity index in the class of all regular tree languages. On the other hand, there are many results for the restricted class of deterministic languages [11, 15, 16, 17, 13]. Unambiguous automata can be seen as a natural intermediate class between deterministic and nondeterministic ones. An automaton is unambiguous if it has at most one accepting run on every input. In some settings [25, 3] unambiguous automata admit faster algorithms than general nondeterministic automata.

The unambiguous automata do not capture the class of all regular languages of infinite trees. As shown in [5], the language L_b of trees containing at least one letter b cannot be recognised by any unambiguous automaton. The proof uses a result by Gurevich and Shelah [8] stating that there is no MSO-definable choice function on the full binary tree (see [5] for a simpler proof of this result). To the authors' best knowledge, the non-definability of choice has been so far the only method to show that a tree language is ambiguous (i.e. not unambiguous).

The class of unambiguous languages of infinite trees is not well-understood. In particular, there is no effective procedure known that decides whether a given nondeterministic automaton recognises an unambiguous language. Additionally, unambiguous languages lack some natural properties. As witnessed by the language L_b, a complement of an unambiguous (and even deterministic) language may be ambiguous. Also, as shown in Proposition 2 of this work, a sum of two deterministic languages may be ambiguous.

Due to the above reasons we concentrate on the class of languages such that both the language and its complement are unambiguous. We call these languages *bi-unambiguous*. An easy argument shows that this class is effectively closed under boolean operations. Moreover, the class is rich enough to contain languages beyond the σ-algebra generated by $\mathbf{\Pi}^1_1$ sets (see [9]). In particular, there are bi-unambiguous languages that are topologically harder than all deterministic languages.

Our motivating problem is to find an effective procedure that verifies if a given regular tree language is bi-unambiguous. Unfortunately, we are unable to solve this problem in full generality. We have a candidate P for such a procedure and we prove that P is sound — if P returns YES then the language is bi-unambiguous. Also, P is complete for deterministic languages — if L is deterministic and bi-unambiguous then P returns YES. The completeness of P in the general case relies on a new conjecture (Conjecture 1 below).

Interestingly, the class of thin trees (trees containing only countably many branches, see [12, 21, 2]) emerges naturally in this context. The crucial technical tool of the procedure P can be seen as an application of the algebra designed for thin trees [10, 2] in the setting of all trees. For this purpose a class of *prophetic thin algebras* is introduced. Basing on algebraic observations we show that P is complete if the following conjecture holds.

▶ Conjecture 1 (Undefinability of a choice function on thin trees). There is no MSO formula in the language of trees $\varphi(x, X)$ such that for every non-empty set $X \subseteq \{l, r\}^*$ that is contained in a thin tree, $\varphi(x, X)$ holds for exactly one vertex x and such a vertex x belongs to X.

To the authors' best knowledge the above conjecture is new. It is a strengthening of the result of Gurevich and Shelah [8] as we restrict the class of allowed sets X.

We find this conjecture interesting in its own right. A number of equivalent statements is provided. Also, it turns out that, assuming Conjecture 1, the class of finite prophetic thin algebras has many good properties (e.g. it is a pseudo-variety of algebras corresponding exactly to the class of bi-unambiguous languages).

Conjecture 1 can be seen as an instance of a more general problem of uniformization. We provide some related results on uniformizability on thin trees. In particular, we show that there exists some non-uniformizable formula on thin trees. It can be seen as an alternative

to [8] answer to Rabin's Uniformization Problem. Also, we show that the language of all thin trees is ambiguous, thus providing an essentially new example of an ambiguous language.

We begin by introducing some basic definitions and notions. In Section 3 we define the procedure P and show its properties. Section 4 is devoted to the analysis of the choice problem on thin trees. In Section 5 we study related uniformization problems on thin trees.

2 Basic notions

2.1 Trees

For technical reasons we work with ranked alphabets $A = (N, L)$ where N (like nodes) contains binary symbols and L (like leafs) contains nullary symbols. We assume that both sets N and L are finite and nonempty. We say that t is a *tree over the alphabet* (N, L) if t is a function from its nonempty domain $\mathrm{dom}(t) \subseteq \{l, r\}^*$ into $N \cup L$ in such a way that $\mathrm{dom}(t)$ is prefix-closed and for every vertex $w \in \mathrm{dom}(t)$ either:

- w is an *(internal) node of* t (i.e. $wl, wr \in \mathrm{dom}(t)$) and $t(w) \in N$, or
- w is a *leaf of* t (i.e. $wl, wr \notin \mathrm{dom}(t)$) and $t(w) \in L$.

The set of all trees over an alphabet A is denoted as Tr_A. A tree containing no leaf is *full*. If $t \in \mathrm{Tr}_A$ is a tree and $w \in \mathrm{dom}(t)$ is a vertex of t then by $t\restriction_w \in \mathrm{Tr}_A$ we denote the subtree of t rooted in w. By $\preceq$ we denote the prefix-order on elements of $\{l, r\}^{\leq\omega}$.

A sequence $\pi \in \{l, r\}^\omega$ is an *infinite branch of a tree* t if for every $w \prec \pi$ we have that $w \in \mathrm{dom}(t)$. An element $d \in \{l, r\}$ is called a *direction*, the opposite direction is denoted as $\bar{d}$. The empty sequence ϵ is called the *root* of a tree t. If π is an infinite branch of a tree t and $w \not\prec \pi$ but w is a child of a vertex on π then we say that w is *off* π.

A tree $t \in \mathrm{Tr}_A$ is *thin* if there are only countably many infinite branches of t. The set of all thin trees is denoted by Th_A. A tree that is not thin is *thick*. A tree is *regular* if it has only finitely many different subtrees. For $a \in N$ by $a(t_l, t_r) \in \mathrm{Tr}_A$ we denote the tree consisting of the root ϵ labelled by the letter a and two subtrees $t_l, t_r \in \mathrm{Tr}_A$ respectively.

A *multi-context* over an alphabet $A = (N, L)$ is a tree $c \in \mathrm{Tr}_{(N, L \sqcup \{\Box\})}$. A vertex $w \in \mathrm{dom}(c)$ such that $c(w) = \Box$ is called a *hole of* c. For every tree $t \in \mathrm{Tr}_A$ the *composition of* c *and* t, denoted $c \cdot t \in \mathrm{Tr}_A$, is obtained by plugging copies of t in all the holes of c.

If a multi-context c has exactly one hole not in the root then it is called a *context*. The set of all contexts over the alphabet A is denoted as Con_A. The set of all contexts over A that are thin as trees is denoted by ThCon_A. For $t \in \mathrm{Tr}_A$ and $w \in \mathrm{dom}(t)$, by $t[\Box/w]$ we denote the context obtained from t by putting the hole in w.

Let $t_A \in \mathrm{Tr}_A$ and M be a ranked alphabet. We say that $t_M \in \mathrm{Tr}_M$ is a *labelling of* t_A if $\mathrm{dom}(t_M) = \mathrm{dom}(t_A)$. In that case we define the tree $(t_A, t_M) \in \mathrm{Tr}_{A \times M}$ in the natural way.

2.2 Automata

A nondeterministic parity tree automaton is a tuple $\mathcal{A} = (Q, A, \delta, I, \Omega)$ where

- Q is a finite set of *states*,
- $A = (N, L)$ is a ranked alphabet,
- $\delta = \delta_2 \sqcup \delta_0$ is a *transition relation*: $\delta_2 \subseteq Q \times Q \times N \times Q$ contains *transitions for nodes* (q, q_l, a, q_r) and $\delta_0 \subseteq Q \times L$ contains *transitions for leafs* (q, b),
- $I \subseteq Q$ is a set of *initial states*,
- $\Omega\colon Q \to \mathbb{N}$ is a *priority function*.

A run of an automaton $\mathcal{A}$ on a tree $t \in \mathrm{Tr}_A$ is a labelling ρ of t over the ranked alphabet (Q, Q) such that for every vertex w of t

- if w is a node of t then $(\rho(w), \rho(wl), t(w), \rho(wr)) \in \delta_2$,
- if w is a leaf of t then $(\rho(w), t(w)) \in \delta_0$.

A run ρ is *consistent* if for every infinite branch π of t the lim sup of values of Ω on states along π is even: $\limsup_{n\to\infty} \Omega(\rho(\pi \restriction_n)) \equiv 0 \mod 2$. The state $\rho(\epsilon)$ is called the *value of* ρ.

Similarly one can define a run ρ on a context c with the hole w, the only difference is that there is no constraint on the value $\rho(w)$ in the hole of c.

A run ρ is *accepting* if it is consistent and $\rho(\epsilon) \in I$. A tree $t \in \mathrm{Tr}_A$ is *accepted by* $\mathcal{A}$ if there exists an accepting run ρ of $\mathcal{A}$ on t. The set of trees accepted by $\mathcal{A}$ is called the *language recognised by* $\mathcal{A}$ and is denoted by $\mathrm{L}(\mathcal{A})$. A language $L \subseteq \mathrm{Tr}_A$ is *regular* if there exists an automaton recognising L.

We say that an automaton $\mathcal{A}$ is *deterministic* if $I = \{q_I\}$ and for every state $q \in Q$ and letter $a \in N$ there is at most one transition $(q, q_l, a, q_r) \in \delta_2$. An automaton $\mathcal{A}$ is *unambiguous* if for every tree $t \in \mathrm{L}(\mathcal{A})$ there is exactly one accepting run of $\mathcal{A}$ on t. A language $L \subseteq \mathrm{Tr}_A$ is *deterministic* (resp. *unambiguous*) if there exists a deterministic (resp. unambiguous) automaton recognising L. A language that is not unambiguous is called *ambiguous*. A deterministic language is, by the definition, unambiguous. A language L is *bi-unambiguous* if both L and $\mathrm{Tr}_A \setminus L$ are unambiguous.

We finish this section with an observation showing that unambiguous languages are not closed under finite union.

▶ Proposition 2. There exist deterministic languages L_1, L_2 such that $L_1 \cup L_2$ is ambiguous.

2.3 Logic

We use the standard notion of Monadic Second-Order (MSO) logic (see [26]). The syntax of this logic allows quantification over elements and subsets of the domain, boolean connectives, predicates for the letters in a given alphabet, and two relations l-child, r-child.

For a given MSO formula $\varphi(\vec{P})$ over an alphabet $A = (N, L)$ with n parameters $P_1, \ldots, P_n$ by $\mathrm{L}(\varphi(\vec{P}))$ we denote the set of trees over the alphabet $(N \times \{0,1\}^n, L \times \{0,1\}^n)$ that satisfy φ when parameters P are decoded from their characteristic functions.

The crucial property of MSO logic is expressed by the following theorem.

▶ **Theorem 3** (Rabin [19]). *A language $L \subseteq \mathrm{Tr}_A$ is regular if and only if there exists an MSO formula φ such that $L = \mathrm{L}(\varphi)$. There are effective procedures translating MSO formulas into equivalent automata and vice versa.*

3 Bi-unambiguous languages

In this section we concentrate on the following decision problem.

▶ Problem 4. The input is a nondeterministic parity tree automaton $\mathcal{A}$. The output should be YES if the language $\mathrm{L}(\mathcal{A})$ is bi-unambiguous. Otherwise, the output should be NO.

We construct a procedure P with the following properties.

▶ **Theorem 5.** *Let $\mathcal{A}$ be a nondeterministic tree automaton.*

1. *$P(\mathcal{A})$ terminates.*
2. *If $P(\mathcal{A}) = \mathrm{YES}$ then $\mathrm{L}(\mathcal{A})$ is bi-unambiguous.*

3. *If* $\mathrm{L}(\mathcal{A})$ *is deterministic and* $P(\mathcal{A}) = \mathrm{NO}$ *then* $\mathrm{L}(\mathcal{A})$ *is not bi-unambiguous*[1].
4. *If Conjecture 1 is true and* $P(\mathcal{A}) = \mathrm{NO}$ *then* $\mathrm{L}(\mathcal{A})$ *is not bi-unambiguous.*

Recall that it is decidable whether a given regular tree language is recognisable by a deterministic tree automaton (see [17]). Therefore, the above assumption that $\mathrm{L}(\mathcal{A})$ is deterministic can be effectively checked given some representation of $\mathrm{L}(\mathcal{A})$. The rest of this section is devoted to defining P and showing the above theorem.

3.1 Thin algebra

The crucial tool in the construction of the procedure P is a variant of thin forest algebra [2], called *thin algebra*. Thin algebra can be seen as a natural extension of Wilke algebra [28, 30] and Wilke tree algebra [29] to the case of infinite trees.

Let us fix a ranked alphabet $A = (N, L)$. A *thin algebra over* A is a two-sorted algebra (H, V) with a number of operations:

- $u \cdot v \in V$ for $u, v \in V$,
- $v \cdot h \in H$ for $v \in V, h \in H$,
- $v^\infty \in H$ for $v \in V$,
- $\mathrm{Node}(a, d, h) \in V$ for $a \in N$, $d \in \{l, r\}$, and $h \in H$,
- $\mathrm{Leaf}(b) \in H$ for $b \in L$.

Note that the first three operations are the same as in the case of Wilke algebras. The last two operations allow to operate on trees. For simplicity, we write $a(\Box, h)$ instead of $\mathrm{Node}(a, l, h)$ and $a(h, \Box)$ instead of $\mathrm{Node}(a, r, h)$. Similarly, $b()$ stands for $\mathrm{Leaf}(b)$ and $a(h_l, h_r) \in H$ denotes the result of $a(h_l, \Box) \cdot h_r$.

The axioms of thin algebra are axioms of Wilke algebra and one additional axiom: $a(\Box, h_r) \cdot h_l = a(h_l, \Box) \cdot h_r$.

▶ Fact 6. Let (H, V) be a thin algebra and let $(v_i)_{i \in \mathbb{N}}$ be any sequence of elements of V. There exists a unique value $\prod_i v_i \in H$ for which: if $j_0 < j_1 < \ldots$ is a sequence of numbers and $s, e \in V$ are types such that:

- $v_0 \cdot \ldots \cdot v_{j_0} = s$,
- for all $i \geq 0$ $v_{j_i+1} \cdot \ldots \cdot v_{j_{i+1}} = e$

then $s \cdot e^\infty = \prod_i v_i$. Also, the following holds $\prod_{i \geq 0} v_i = v_0 \cdot \prod_{i \geq 1} v_i$.

Proof. The same as in the case of Wilke algebra, see [18]. ◀

It is easy to verify that the pair $(\mathrm{Tr}_A, \mathrm{Con}_A)$ has a natural structure of a thin algebra. In particular, the operation c^∞ constructs the tree c^∞ from a context c by *looping* the hole of c to the root of c. The subalgebra of $(\mathrm{Tr}_A, \mathrm{Con}_A)$ consisting of thin regular trees and thin regular contexts is free in the class of thin algebras over the alphabet A. The algebra $(\mathrm{Tr}_A, \mathrm{Con}_A)$ is not free.

A homomorphism $\alpha \colon (H, V) \to (H', V')$ between two thin algebras over the same alphabet A is defined in the usual way: α should be a function mapping elements of H into H' and elements of V into V' that preserves all the operations of thin algebra. Such a homomorphism is *surjective* if $\alpha(H) = H'$ and $\alpha(V) = V'$.

Since $(\mathrm{Tr}_A, \mathrm{Con}_A)$ is not free in the class of thin algebras, we need to define one additional requirement for homomorphisms $\alpha \colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$. Let $A = (N, L)$ and put $A \sqcup H =$

[1] What is equivalent to ambiguity of the complement of $\mathrm{L}(\mathcal{A})$.

$(N, L \sqcup H)$. Consider any tree $c \in \mathrm{Tr}_{A \sqcup H}$ and $t \in \mathrm{Tr}_A$. We say that t *is an extension of* c if $\mathrm{dom}(c) \subseteq \mathrm{dom}(t)$ and for every $w \in \mathrm{dom}(c)$ either:

- $c(w) \in N \cup L$ and $c(w) = t(w)$,
- $c(w) \in H$ and $c(w) = \alpha(t\restriction_w)$.

That is, t is supposed to agree with c on all the letters in $N \cup L$ and whenever c declared some type $h \in H$ in a leaf w then the subtree $t\restriction_w$ has α-type h (i.e. $\alpha(t\restriction_w) = h$).

▶ **Definition 7.** We say that $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$ *is compositional* if there exists a function $\bar{\alpha}\colon \mathrm{Tr}_{A \sqcup H} \to H$ such that if $t \in \mathrm{Tr}_A$ is an extension of $c \in \mathrm{Tr}_{A \sqcup H}$ then $\bar{\alpha}(c) = \alpha(t)$.

Let $L \subseteq \mathrm{Tr}_A$ be a language of trees. We say that a homomorphism $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$ *recognises* L if α is compositional and there is a set $F \subseteq H$ such that $L = \alpha^{-1}(F)$.

▶ Fact 8. Since every context $c \in \mathrm{Con}_A$ can be obtained as a finite combination of trees $t \in \mathrm{Tr}_A$ using the operation Node, if $\alpha_1, \alpha_2\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$ are two homomorphisms that agree on Tr_A then $\alpha_1 = \alpha_2$.

The following theorem introduces the notion of *syntactic morphism* for a given language. It is an adaptation of a similar theorem for the case of thin forest algebras, see [10] for a deeper explanation. For the sake of completeness, a sketch of a proof is given in Appendix A.

▶ **Theorem 9.** *For every regular tree language L there exists a* syntactic morphism for L*: a finite thin algebra $S_L = (H, V)$ (called a* syntactic algebra of L*) and a homomorphism $\alpha_L\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to S_L$ such that:*

- *α_L is surjective, compositional, and recognises L,*
- *for every $h \in H$ the language $L_h := \alpha_L^{-1}(\{h\})$ is regular,*
- *if $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to S$ is surjective and recognises L then there is exactly one homomorphism $\beta\colon S \to S_L$ such that $\beta \circ \alpha = \alpha_L$.*

A syntactic algebra S_L and languages L_h can be effectively computed basing on a nondeterministic automaton recognising L.

Note that by the last bullet, all syntactic morphisms for a given language are *isomorphic* — there are homomorphisms β that make the respective diagrams commute. Therefore, a syntactic morphism can be seen as a unique representation of a language.

An intermediate step in this proof requires a definition of some finite thin algebra $S_{\mathcal{A}} = (H_{\mathcal{A}}, V_{\mathcal{A}})$ that recognises the language $\mathrm{L}(\mathcal{A})$ for a given automaton $\mathcal{A}$. The constructed algebra is called the *automaton algebra for* $\mathcal{A}$. The definition of $S_{\mathcal{A}}$ is the same as in [10]. The homomorphism into $S_{\mathcal{A}}$ that recognises $\mathrm{L}(\mathcal{A})$ is based on the following operation that will be used later:

$$Q_{\mathcal{A}}(t) = \{q \in Q :\ \exists_\rho\ \rho \text{ is a consistent run of } \mathcal{A} \text{ on } t \text{ with value } q\} \subseteq 2^Q. \tag{1}$$

If $\mathcal{A}$ is known from the context, we write just $Q(t)$. By $\tau_{\mathcal{A}}(t)$ we denote the labelling of t defined $\tau_{\mathcal{A}}(t)(w) = Q_{\mathcal{A}}(t\restriction_w)$.

What is important in Theorem 9 is that we explicitly fix the homomorphism α_L. Usually (e.g. in the case of monoids) there is a unique such homomorphism for a fixed interpretation of the alphabet. It turns out that this is not the case for thin algebras and all binary trees. Therefore, to fully describe a given language we need an algebra S_L, a set $F \subseteq H$, and a homomorphism α_L (it can be represented by the languages L_h).

3.2 Prophetic algebras

The situation when there are multiple homomorphisms from all trees into a given thin algebra comes from the fact that the algebra may not be *prophetic*. In this section we formally introduce this notion.

Let (H, V) be a thin algebra over an alphabet $A = (N, L)$. Let $t \in \mathrm{Tr}_A$ be a tree. A labelling $\tau \in \mathrm{Tr}_{(H,H)}$ of t is a *marking of t by types in H* if:

- for every node w of t we have $\tau(w) = t(w)(\tau(wl), \tau(wr))$,
- for every leaf w of t we have $\tau(w) = t(w)()$.

A marking τ is *consistent* if it is consistent on every infinite branch π of t. Let $\pi = d_0 d_1 \dots$ and let $w_0 \prec w_1 \prec \dots$ be the sequence of vertices of t along π. The sequence of types of contexts $v_i = \mathrm{Node}(t(w_i), d_i, \tau(w_i \bar{d}_i))$ is called the *decomposition of τ along π*. Now, τ is *consistent on π* if for every $i \in \mathbb{N}$ we have

$$\tau(w_i) = \prod_{j \geq i} v_j. \tag{2}$$

Informally speaking, a marking τ is consistent along π if the types of τ along π agree with the types that can be computed using $\prod$ basing on the types of vertices that are off π. By the definition of a marking, it is enough to require (2) for infinitely many $i \in \mathbb{N}$ in the definition of consistency.

Note that if a homomorphism $\alpha \colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to S$ is fixed, for every tree $t \in \mathrm{Tr}_A$ the marking $\tau_\alpha(t)(w) := \alpha(t \restriction_w)$ (called the *marking induced by α on t*) is consistent.

▶ **Example 10.** Fix the alphabet $A_b = (\{n\}, \{b\})$. Let $L_b \subseteq \mathrm{Tr}_{A_b}$ contain exactly these trees which have at least one leaf. One may verify that the syntactic morphism for L_b can be defined as follows: $H_{L_b} = \{h_a, h_b\}$, $V_{L_b} = \{v_a, v_b\}$, and $\alpha_{L_b}(t) = h_b$ (resp. $\alpha_{L_b}(c) = v_b$) if and only if a tree t (resp. a context c) contains any leaf (not counting the hole of c).

Let t_n be the full binary tree equal everywhere n. Observe that t_n does not belong to L_b and the marking $\tau_{\alpha_{L_b}}(t_n)$ induced by α_{L_b} on t_n equals h_a in every vertex. Consider another marking τ' of t_n that equals h_b everywhere. Note that τ' is consistent — it looks like a consistent marking along every branch. Therefore, t has two consistent markings.

Going further, one can construct a compositional homomorphism $\alpha' \colon (\mathrm{Tr}_{A_b}, \mathrm{Con}_{A_b}) \to (H_{L_b}, V_{L_b})$ that assigns h_b to the tree t_n. Therefore, there are two distinct compositional homomorphisms from $(\mathrm{Tr}_{A_b}, \mathrm{Con}_{A_b})$ to (H_{L_b}, V_{L_b}).

Recall that the language L_b used above is known to be ambiguous, see [14].

The following fact follows from [2], the proof goes via induction on *rank* of thin trees.

▶ Fact 11. If $t \in \mathrm{Tr}_A$ is a thin tree and (H, V) is a finite thin algebra over the alphabet A then there exists exactly one consistent marking τ of t.

The following definition is crucial for the procedure P. The term *prophetic* is motivated by [6].

▶ **Definition 12.** We say that a thin algebra (H, V) over an alphabet A is *prophetic* if for every tree $t \in \mathrm{Tr}_A$ there exists at most one consistent marking of t by types in H.

Note that if $\alpha \colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to S$ is a homomorphism and S is prophetic then, for every tree $t \in \mathrm{Tr}_A$, the only consistent marking of t is the marking induced by α. In particular, there is at most one homomorphism from $(\mathrm{Tr}_A, \mathrm{Con}_A)$ into S, see Fact 8.

Since the property that a given finite thin algebra is prophetic can be expressed in MSO over the full binary tree, so we obtain the following fact.

▶ Fact 13. It is decidable whether a given finite thin algebra (H, V) is prophetic.

▶ Fact 14. By the definition, a subalgebra of a prophetic thin algebra is also prophetic. Similarly, a product of two prophetic thin algebras is also prophetic.

3.3 Semi-characterisation

The following theorem gives a connection between bi-unambiguous languages and prophetic algebras.

▶ **Theorem 15.** *A language $L \subseteq \mathrm{Tr}_A$ is bi-unambiguous if and only if there exists a surjective homomorphism $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$ that recognises L such that (H, V) is a finite prophetic thin algebra over the alphabet A.*

First assume that L is a bi-unambiguous language. Let $\mathcal{A}, \mathcal{B}$ be a pair of unambiguous automata recognising L and $\mathrm{Tr}_A \setminus L$ respectively. We describe how to construct a finite prophetic thin algebra (H_U, V_U) recognising L.

The first step can be expressed as the following fact.

▶ Fact 16. Assume that $\mathcal{A}$ is an unambiguous automaton over an alphabet A and $t \in \mathrm{Tr}_A$. Assume that τ is a consistent marking of t by types in the automaton algebra $S_{\mathcal{A}}$. Then there is at most one run ρ of $\mathcal{A}$ on t such that $\rho(\epsilon) \in I^{\mathcal{A}}$ and $\forall_{w \in \mathrm{dom}(t)}\ \rho(w) \in \tau(w)$.

Using the above observation and properties of the automaton algebra, we can entail that for every consistent marking τ of a given tree t and for every $q \in \tau_{\mathcal{A}}(t)(\epsilon)$ there is a consistent run of $\mathcal{A}$ on t with value q. Therefore, for every consistent marking τ of t we have $\forall_{w \in \mathrm{dom}(t)}\ \tau(w) \subseteq \tau_{\mathcal{A}}(t)(w)$. Our aim is to put some additional constraints on τ that imply equality in the above formula. This is obtained by the second step of the reasoning, as expressed in the following lemma. The idea to use pairs of sets of states in this context was suggested by Igor Walukiewicz.

▶ **Lemma 17.** *Let $\mathcal{A}, \mathcal{B}$ be a pair of unambiguous automata recognising L and $\mathrm{Tr}_A \setminus L$ respectively. Let $R = \{(Q_{\mathcal{A}}(t), Q_{\mathcal{B}}(t)) :\ t \in \mathrm{Tr}_A\}$. Then the set R ordered coordinate-wise by inclusion is an antichain.*

Now let $t \in \mathrm{Tr}_A$ and assume that we have consistent markings τ_1, τ_2 of t with respect to algebras $S_{\mathcal{A}}, S_{\mathcal{B}}$ respectively. Assume that for every $w \in \mathrm{dom}(t)$ we have $(\tau_1(w), \tau_2(w)) \in R$. Then $\tau_1(w) \subseteq \tau_{\mathcal{A}}(t)(w)$, $\tau_2(w) \subseteq \tau_{\mathcal{B}}(t)(w)$, and by the above lemma $\tau = \tau_{\mathcal{A}}(t), \tau' = \tau_{\mathcal{B}}(t)$. This shows that the product of algebras $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$ is prophetic.

The following lemma implies the opposite direction of Theorem 15.

▶ **Lemma 18.** *Let $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$ be a compositional homomorphism into a finite prophetic thin algebra (H, V) and $h_0 \in H$. The language $L_{h_0} = \alpha^{-1}(h_0)$ is unambiguous.*

Using this lemma, if α recognises a language L then L and $\mathrm{Tr}_A \setminus L$ are finite disjoint unions of unambiguous languages L_{h_0} so L is bi-unambiguous.

The construction of an unambiguous automaton $\mathcal{C}$ recognising L goes as follows: let $\mathcal{C}$ guess some marking τ of a given tree t by types in H. Then, $\mathcal{C}$ verifies that the root is labelled by h_0 and the marking τ is consistent. Since consistency of a marking is a branch-wise ω-regular condition, so it can be verified by a deterministic top-down automaton. Since (H, V) is prophetic, so the only possible consistent marking of t is the marking induced by α. So $\mathcal{C}$ has at most one accepting run on t and it accepts t if and only if $\alpha(t) = h_0$.

Theorem 15 implies the following lemma, that can also be proved without use of algebra.

▶ Remark. The class of bi-unambiguous languages is closed under boolean operations and language quotients $c^{-1} \cdot L = \{t :\ c \cdot t \in L\}$ for contexts c.

3.4 The procedure P

Now we can formally define our procedure P. This procedure consists of three steps:
1. Read a nondeterministic automaton $\mathcal{A}$ recognising a regular tree language L.
2. Compute the syntactic thin algebra S_L of L.
3. Answer YES if S_L is prophetic, otherwise answer NO.

By Theorem 9 and Fact 13 both operations undertaken by P are effective. Therefore, P is well-defined and always terminates. Note that if S_L is prophetic then, by Theorem 15, the language L is bi-unambiguous. Therefore, Item 5 of Theorem 5 holds. The only remaining possibility of failure of the procedure P is when L is bi-unambiguous but the syntactic algebra S_L is not prophetic. Our aim is to exclude this possibility. In general, Conjecture 1 implies that the syntactic algebra of a bi-unambiguous language is prophetic, see Remark 4.1. This shows that Item 5 of Theorem 5 holds. The following theorem implies Item 5 of Theorem 5.

▶ **Theorem 19.** *If L is deterministic and bi-unambiguous then the algebra S_L is prophetic.*

The rest of this section is devoted to proving this theorem. Let $\mathcal{D}$ be a deterministic tree automaton recognising $L \subseteq \mathrm{Tr}_A$. A state $q \in Q_\mathcal{D}$ is *nontrivial* if there is a tree t not accepted by $\mathcal{D}$ from q (i.e. there is no consistent run of $\mathcal{D}$ on t with value q). Let $t \in L$ be a tree and ρ be the accepting run of $\mathcal{D}$ on t. Let $T_\mathcal{D}(t) \subseteq \{l, r\}^*$ be the set of vertices $w \in \mathrm{dom}(t)$ such that $\rho(w)$ is a nontrivial state of $\mathcal{D}$. Note that $T_\mathcal{D}(t)$ is a prefix-closed subset of $\mathrm{dom}(t)$. We start with the following lemma.

▶ **Lemma 20.** *If $\mathcal{D}$ is a deterministic tree automaton and $\mathrm{Tr}_A \setminus \mathrm{L}(\mathcal{D})$ is unambiguous then for every tree $t \in \mathrm{L}(\mathcal{D})$ the set $T_\mathcal{D}(t)$ is thin.*

Proof. Assume contrary and fix a regular tree $t \in L$ such that $T = T_\mathcal{D}(t)$ is thick. Let ρ be the run of $\mathcal{D}$ on t. Let $\mathcal{A}$ by an unambiguous automaton recognising $\mathrm{Tr}_A \setminus \mathrm{L}(\mathcal{D})$. Now observe that for every $w \in T$ there exists a tree t_w not accepted by $\mathcal{D}$ from the state $\rho(w)$. Let $X \subseteq T$ be any prefix-free set. Let $t(X)$ be the tree obtained from t by plugging simultaneously subtrees t_w under w for every $w \in X$. Note that if $X \neq \emptyset$ then $t(X) \notin \mathrm{L}(\mathcal{D})$ — the run ρ does not extend to accepting run under any $w \in X$. Therefore, we obtain

$$t(\emptyset) \notin \mathrm{L}(\mathcal{A}) \text{ and } \forall_{X \subseteq T} \ (X \text{ is prefix-free and nonempty } \Rightarrow \ t(X) \in \mathrm{L}(\mathcal{A}))\,. \tag{3}$$

Now we construct an automaton $\bar{\mathcal{A}}$ for the language L_b (see Example 10). The transitions of $\bar{\mathcal{A}}$ simulate transitions of $\mathcal{A}$ on T. Whenever $\bar{\mathcal{A}}$ reaches a leaf, it simulates the behaviour of $\mathcal{A}$ on the respective tree t_w. Since $\mathcal{A}$ is unambiguous, so is $\bar{\mathcal{A}}$. And, by (3) $\mathrm{L}(\bar{\mathcal{A}}) = L_b$. This gives us a contradiction with the fact that L_b is ambiguous. ◀

▶ Fact 21. Let $\mathcal{D}$ be a deterministic automaton and $t \in \mathrm{L}(\mathcal{D}) \subseteq \mathrm{Tr}_A$. Assume that $t' \in \mathrm{Tr}_A$ is a tree satisfying $w \in \mathrm{dom}(t')$ and $t'(w) = t(w)$ for every $w \in T_\mathcal{D}(t)$. Then $t' \in \mathrm{L}(\mathcal{D})$.

Proof. The accepting run of $\mathcal{D}$ on vertices in $T_\mathcal{D}(t)$ can be extended to t' by triviality of the states outside $T_\mathcal{D}(t)$. ◀

Now we can finish the proof of Theorem 19.

Proof. Assume contrary that the syntactic algebra S_L of L is not prophetic. Let t be a tree and τ, τ' be a pair of distinct consistent markings of t. Let $h = \tau(\epsilon)$ and $h' = \tau'(\epsilon)$. We can assume that $h \neq h'$ (otherwise instead of t we take $t\restriction_w$ where w is a node for which $\tau(w) \neq \tau'(w)$). Since $h \neq h'$ so there exists a multi-context c such that (by symmetry)

$c \cdot t \in L$ and $c \cdot t' \notin L$. Let $w_0, w_1, \ldots$ be the list of holes of c. Since $c \cdot t \in L$ so we can consider the set $T = T_{\mathcal{D}}(c \cdot t) \subseteq \{l, r\}^*$.

By Lemma 20 we know that T is thin, in particular $T_i := T \restriction_{w_i}$ is thin for every i. Let $\bar{t}_i$ be the tree obtained from t by substituting some tree of α_L-type $\tau'(w)$ instead of $t \restriction_w$ for every minimal $w \notin T_i$. Since T_i is thin and α_L-types of subtrees of $\bar{t}_i$ agree with τ' outside T_i so $\alpha_L(\bar{t}_i) = h'$ — we use the fact that T_i is thin. Let $\bar{t}$ be the tree obtained from c by putting $\bar{t}_i$ instead of the hole w_i. Then, by compositionality of α_L we obtain that $\alpha_L(\bar{t}) = \alpha_L(c \cdot t')$, so $\bar{t} \notin L$. But $c \cdot t$ and $\bar{t}$ agree on $T_{\mathcal{D}}(t)$ so by Fact 21 $\bar{t} \in L$, a contradiction. ◀

4 (Un)definability of choice on thin trees

In this section we study Conjecture 1, we show a couple of equivalent statements and prove some of its consequences (in particular Item 5 of Theorem 5). We start by formulating the choice problem as a instance of a more general question.

▶ **Definition 22.** Let $\varphi(X, \vec{P})$ be a MSO formula on A-labelled trees with monadic parameters X and $\vec{P} = P_1, \ldots, P_n$. We say that $\psi(X, \vec{P})$ is an *uniformization of* φ if the following conditions are satisfied for every tree t, values of parameters $\vec{P}$, and sets $X_1, X_2 \subseteq \mathrm{dom}(t)$:

$$\begin{aligned} \Big(\exists_X\ \varphi(X, \vec{P})\Big) &\Leftrightarrow \Big(\exists_X\ \psi(X, \vec{P})\Big) \\ \psi(X_1, \vec{P}) &\Rightarrow \varphi(X_1, \vec{P}) \\ \Big(\psi(X_1, \vec{P}) \wedge \psi(X_2, \vec{P})\Big) &\Rightarrow X_1 = X_2 \end{aligned}$$

That is, whenever it is possible to pick some X satisfying $\varphi(X, \vec{P})$ then ψ chooses exactly one such X. For simplicity, we allow a (possible empty) list of additional parameters $\vec{P}$ and we assume that the first variable is the one that is supposed to be uniformized.

Now, Conjecture 1 says that the following formula does not have uniformization:

CHOICE(x, X) := the given tree is thin and $x \in X$.

A simple interpretation argument shows that Conjecture 1 is equivalent to the non-uniformizability of the following simpler formula.

LEAF $-$ CHOICE(x) := the given tree is thin and x is a leaf.

The following proposition expresses the crucial technical condition, allowing to entail properties of thin algebras using Conjecture 1.

▶ Proposition 23 (assuming Conjecture 1). Assume that $\alpha\colon (H, V) \to (H', V')$ is a surjective homomorphism between two finite thin algebras. Let t be a tree and τ' be a consistent marking of t by H'. Then there exists a consistent marking τ of t by H such that $\forall_{w \in \mathrm{dom}(t)}\ \alpha(\tau(w)) = \tau'(w)$.

Sketch of the proof: assume contrary and fix a regular pair (t_0, τ') such that there is no marking τ as above. Consider the standard automaton-pathfinder game, where the automaton proposes a marking τ and the pathfinder picks directions to show that τ does not satisfy the above conditions. Since there is no such τ, so pathfinder has a finite memory winning strategy σ. Now, given a thin tree t we can define the unique consistent marking τ that satisfies $\alpha(\tau) = \tau'$ on t. The play resulting in pathfinder playing σ and automaton playing τ must end in a leaf of t. ◀

The second important tool in our analysis enables to make a connection between uniformized relations and induced markings. A formal definition of a transducer and a proof of the following theorem are given in Appendix B.

▶ **Theorem 24.** *Assume that $L_A \subseteq \mathrm{Tr}_A, L_M \subseteq \mathrm{Tr}_{A\times M}$ are regular languages of trees for two ranked alphabets A, M such that L_A is a projection of L_M onto A. Assume that $\forall_{t_A\in L_A}\ \exists!_{t_M\in \mathrm{Tr}_M}\ (t_A, t_M) \in L_M$. Then, there exist:*

- *a compositional homomorphism $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to S$ into a finite thin algebra S,*
- *a deterministic finite state transducer that reads the marking induced by α on a given tree t_A and outputs the labelling t_M such that $(t_A, t_M) \in L_M$, whenever such t_M exists.*

Now we can present two algebraic statements that are equivalent to Conjecture 1.

▶ **Theorem 25.** *The following conditions are equivalent:*

1. *Conjecture 1 holds.*
2. *For every finite thin algebra (H, V) over an alphabet $A = (N, L)$ and every tree $t \in \mathrm{Tr}_A$ there exists a consistent marking of t by types in H.*
3. *For every finite thin algebra (H, V) over the alphabet $A_b = (\{n\}, \{b\})$ there exists a consistent marking of the full tree $t_n \in \mathrm{Tr}_{A_b}$ by types in H.*

Note that in the above theorem algebras (H, V) come without any homomorphism from $(\mathrm{Tr}_A, \mathrm{Con}_A)$, so there is no notion of the induced marking.

Proof. First we show $1 \Rightarrow 2$. Let (H, V) be a finite thin algebra over an alphabet $A = (N, L)$. Let $(H', V') = (\{h_0\}, \{v_0\})$ be the singleton thin algebra with $b() = h_0$ for every $b \in L$. There is a unique homomorphism $\alpha\colon (H, V) \to (H', V')$. Take any tree $t \in \mathrm{Tr}_A$. Let τ' be the consistent marking of t that is constant equal h_0 on $\mathrm{dom}(t)$. By Proposition 23 there exists a consistent marking of t by types in H.

Of course Item 3 follows from Item 2.

For $3 \Rightarrow 1$ we assume that $\psi(x)$ is an MSO formula uniformizing LEAF − CHOICE. Using Theorem 24 we find a deterministic transducer $\mathcal{T}$ that reads types of subtrees of a given thin tree (with respect to some homomorphism α into a finite thin algebra (H, V)) and outputs directions towards the chosen leaf. Let (H', V') be the subalgebra of (H, V) containing α-types of $(\mathrm{Th}_A, \mathrm{ThCon}_A)$. By Item 3 there is a consistent marking τ of the full tree t_n by types in H'. We can consider the sequence of directions π given by $\mathcal{T}$ on (t_n, τ). Since t does not have any leaf, so π is infinite. Now, we can substitute all subtrees that are not on π by thin trees of the respective types given by τ. The result is a thin tree t' such that the directions produced by $\mathcal{T}$ do not reach any leaf of t' — a contradiction. ◀

4.1 Prophetic thin algebras

It turns out that (assuming Conjecture 1) the class of finite prophetic thin algebras has a number of nice properties. Most of them can be read as properties of the class of bi-unambiguous languages. To emphasise that we work under the assumption of Conjecture 1, we explicitly put it as a pre-assumption in the statements.

▶ **Theorem 26** (Conjecture 1)**.** *Let (H, V) be a prophetic thin algebra over an alphabet A. There exists a unique homomorphism $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$. Additionally, α is compositional.*

Proof. The uniqueness of the homomorphism was observed in Section 3.2. By Theorem 25 and the fact that (H, V) is prophetic, every tree $t \in \mathrm{Tr}_A$ has exactly one consistent marking τ_t by types in H. Let us define $\alpha(t) = \tau_t(\epsilon)$. Clearly α is a compositional homomorphism — if t is an extension of c then the consistent marking τ_t must agree with the types in the leafs of c. ◀

▶ **Theorem 27** (Conjecture 1). *Let $\beta\colon S \to S'$ be a surjective homomorphism between two finite thin algebras. If S is prophetic then S' is also prophetic.*

Proof. First fix the homomorphism $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to S$ given by Theorem 26. Note that $\beta \circ \alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$ is a compositional homomorphism. Assume that S' is not prophetic, so there exists a tree t with two consistent markings σ, σ' by types of S'. Without loss of generality we can assume that σ is the marking induced by $\beta \circ \alpha$ and $\sigma'(\epsilon) \neq \sigma(\epsilon)$. Let τ be the marking by types in S induced by α on t. Observe that pointwise $\beta(\tau) = \sigma$. By Proposition 23 there exists a consistent marking τ' of t such that pointwise $\beta(\tau') = \sigma'$. Therefore, τ, τ' are two distinct consistent markings of t by types in H — a contradiction. ◀

The following remark ends the proof of Item 5 of Theorem 5.

▶ Remark (Conjecture 1). If $L \subseteq \mathrm{Tr}_A$ is bi-unambiguous then S_L is prophetic.

Proof. Since L is bi-unambiguous so by Theorem 15 there exists a surjective homomorphism $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$ that recognises L such that (H, V) is a finite prophetic thin algebra. Since S_L is a syntactic algebra of L so there exists a surjective homomorphism $\beta\colon (H, V) \to S_L$. By Theorem 27 we obtain that S_L is also prophetic. ◀

The next statement shows that prophetic thin algebras form a robust class from the point of view of universal algebra (see [4] for an introduction to this field). The proof follows directly from Theorem 27 and Fact 14.

▶ Remark (Conjecture 1). The class of finite prophetic thin algebras is a pseudo-variety: it is closed under homomorphic images, subalgebras, and finite direct products.

5 Uniformizability results on thin trees

In this section we study Conjecture 1 in the context of related uniformization problems on thin trees. One of the notions we concentrate on are *skeletons* of thin trees, introduced in [2].

▶ **Definition 28.** Let $t \in \mathrm{Tr}_A$ be a tree. We say that $\sigma \subseteq \mathrm{dom}(t)$ is a *skeleton of* t if $\epsilon \notin \sigma$ and the following conditions are satisfied:

- if $w \in \mathrm{dom}(t)$ is an internal node of t then σ contains exactly one of the vertices wl, wr,
- if π is an infinite branch of t then all but finitely many vertices on π belong to σ.

We identify a set $\sigma \subseteq \mathrm{dom}(t)$ with its characteristic function $\sigma \in \mathrm{Tr}_{(\{0,1\},\{0,1\})}$. By $\mathrm{SKEL}(\sigma)$ we denote the MSO formula expressing the above properties.

The following proposition expresses the crucial property of skeletons, see [2].

▶ Proposition 29 ([2]). A tree t is thin if and only if there exists a skeleton of t.

Note that a thin tree may have multiple skeletons. The main idea behind skeletons is that they provide decompositions of thin trees: every skeleton σ of a thin tree t defines *the main branch of* σ that follows σ from the root of t and along this branch *simpler thin trees* are plugged. The second bullet in the definition of skeletons means that such a decomposition is well-founded — we can go off the main branch only finitely many times.

5.1 Non-uniformizability

In this section we give the following two negative results.

▶ **Theorem 30.** *There is no MSO formula uniformizing* $\mathrm{SKEL}(\sigma)$.

▶ **Theorem 31.** *The language* $\mathrm{Th}_{A_b} \subset \mathrm{Tr}_{A_b}$ *of thin trees over the alphabet* A_b *is ambiguous.*

The above theorem can be seen as complementing the following theorem from [2] (adjusted to the case of trees instead of forests).

▶ **Theorem 32** (Theorem 12 from [2])**.** *For every regular language* $L \subseteq \mathrm{Tr}_A$ *that contains only thin trees there exists a nondeterministic automaton* $\mathcal{A}$ *such that* $\mathrm{L}(\mathcal{A}) \cap \mathrm{Th}_A = L$ *and* $\mathcal{A}$ *has at most one accepting run on every thin tree.*

Therefore, every regular tree language containing only thin trees is unambiguous *relatively to thin trees*. But, by Theorem 31, it is the best we can get: even the language of all thin trees is ambiguous among all trees.

The proofs base on two observations, first of them is the existence of transducers, see Theorem 24. The second ingredient is a weaker version of Item 2 in Theorem 25. It is motivated by a similar result on preclones, see [1].

▶ **Theorem 33.** *For every finite thin algebra* (H, V) *over an alphabet* $A = (N, L)$ *there exists a thick tree* $t \in \mathrm{Tr}_A$ *and a consistent marking* τ *of* t *by types in* H.

The proof uses Green's relations [7] in the monoid V of a given thin algebra to find an appropriate idempotent $e \in V$ that enables to construct a tree t . The constructed tree is thick but it is not full — many subtrees of t are thin and contain leafs.

Now we can present a sketch of the proof of Theorem 30.

Proof. Assume that $\psi(\sigma)$ is a uniformization of $\mathrm{SKEL}(\sigma)$. Using Theorem 24 we find: a homomorphism $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$ info a finite thin algebra and a transducer $\mathcal{T}$. Let (H', V') be the subalgebra of (H, V) that is the image of $(\mathrm{Th}_A, \mathrm{ThCon}_A)$ under α.

Using Theorem 33 we construct a thick tree t with a consistent marking τ by types in H'. We run the transducer $\mathcal{T}$ on (t, τ) what results in a labelling t_M of t. Since t is not thin so it has no skeleton. Therefore, one of the conditions for skeletons is not satisfied by t_M. Assume that there exists a branch π of t such that t_M labels infinitely many vertices on π by 0. The other possibility is similar but simpler. Now we can plug thin trees of types given by τ along π obtaining t'. By the construction, t' is thin and τ equals along π the marking of t' induced by α. Therefore, we can run $\mathcal{T}$ on $(t', \tau_\alpha(t))$ obtaining a result t'_M that agrees with t_M on π. It is a contradiction since $\mathcal{T}$ is supposed to produce a correct skeleton for every thin tree and t'_M violates assumptions of skeleton on π. ◀

5.2 Degrees of uniformization

In this section we study relationships between uniformization problems on thin trees. The results of this section were found as answers to questions posed by Alexander Rabinovich.

The following definition is motivated by *degrees of selection* studied in [22].

▶ **Definition 34.** We say that a formula $\varphi(X, \vec{P})$ has *higher degree of uniformization* than $\varphi'(Y, \vec{R})$ (denoted $\varphi'(Y, \vec{R}) \preceq_{uni} \varphi(X, \vec{P})$) if there exists a formula $\psi(Y, \vec{R})$ that is defined in MSO extended by an additional predicate $U(X, \vec{P})$ and $\psi(Y, \vec{R})$ uniformizes $\varphi(Y, \vec{R})$ whenever U is interpreted as any relation uniformizing $\varphi(X, \vec{P})$.

▶ Fact 35. The relation $\preceq_{uni}$ is transitive and reflexive. If $\varphi'(X, \vec{P}) \preceq_{uni} \varphi(Y, \vec{R})$ and $\varphi(Y, \vec{R})$ is uniformizable then so is $\varphi'(X, \vec{P})$.

We say that φ is *on thin trees* if φ is satisfied only on thin trees. The following theorem implies that $\mathrm{SKEL}(\sigma)$ is maximal with respect to $\preceq_{uni}$ among MSO formulas on thin trees.

▶ **Theorem 36.** *For every formula $\varphi(X, \vec{P})$ on thin trees there exists a formula $\varphi'(X, \vec{P}, \sigma)$ that uniformizes $\bar{\varphi}(X, \vec{P}, \sigma) := \varphi(X, \vec{P}) \wedge \mathrm{SKEL}(\sigma)$.*

The proof is based on the fact that every MSO-definable relation on ω-words is uniformizable, see [24, 12, 20]. Since every skeleton gives a decomposition of a given tree as disjoint branches, so we can uniformize the given formula φ independently on these branches. By well-foundedness of skeletons the result is well-defined. The above theorem can also be derived from the proof of Theorem 6.7 in [12] but in a less explicit way.

It turns out that uniformization of $\mathrm{SKEL}(\sigma)$ is connected with definability of well-orderings on thin trees. We say that a formula $\psi(x, y)$ *defines well-order on thin trees* if for every thin tree $t \in \mathrm{Tr}_{A_b}$ the relation $<_\psi$ defined as $(x <_\psi y \Leftrightarrow \psi(x, y))$ is a linear order on $\mathrm{dom}(t)$ and there is no infinite descending sequence of $<_\psi$. In the rest of this section we show that uniformizations of skeletons and definable well-orderings are equivalent — it is possible to define one of them basing on the other.

One direction is simple : the structure of a skeleton gives a natural lexicographic well-order of vertices of a given thin tree. The other direction is a bit more involved: given any definable well-order of a given thin tree t we need to define a skeleton of t.

▶ **Theorem 37.** *If there exists an MSO-definable well-order on thin trees then there exists a uniformization of* $\mathrm{SKEL}(\sigma)$.

The following remark follows from Theorem 30 and Theorem 37. It should be connected with a result of [5] stating that the MSO theory of the full binary tree extended with any well-order is undecidable.

▶ Remark. There is no MSO formula $\psi(x, y)$ that defines well-order on thin trees.

Acknowledgements. The authors would like to thank Mikołaj Bojańczyk, Henryk Michalewski, Damian Niwiński, Alexander Rabinovich, and Igor Walukiewicz for fruitful discussions on the subject.

References

1 Mikołaj Bojańczyk. Algebra for trees. A draft version of a chapter that will appear in the AutomathA handbook, 2010.
2 Mikołaj Bojańczyk, Tomasz Idziaszek, and Michał Skrzypczak. Regular languages of thin trees. In *STACS 2013*, volume 20 of *LIPIcs*, pages 562–573, 2013.
3 Nicolas Bousquet and Christof Löding. Equivalence and inclusion problem for strongly unambiguous Büchi automata. In *LATA*, pages 118–129, 2010.
4 Stanley Burris and H. P. Sankappanavar. *A Course in Universal Algebra.* Number 78 in Graduate Texts in Mathematics. Springer-Verlag, 1981.
5 Arnaud Carayol, Christof Löding, Damian Niwiński, and Igor Walukiewicz. Choice functions and well-orderings over the infinite binary tree. *Cent. Europ. J. of Math.*, 8:662–682, 2010.
6 Olivier Carton, Dominique Perrin, and Jean Éric Pin. Automata and semigroups recognizing infinite words. In *Logic and Automata, History and Perspectives*, pages 133–167, 2007.

7 James Alexander Green. On the structure of semigroups. *Annals of Mathematics*, 54(1):163–172, 1951.

8 Yuri Gurevich and Saharon Shelah. Rabin's uniformization problem. *J. Symb. Log.*, 48(4):1105–1119, 1983.

9 Szczepan Hummel. Unambiguous tree languages are topologically harder than deterministic ones. In *GandALF*, pages 247–260, 2012.

10 Tomasz Idziaszek. *Algebraic methods in the theory of infinite trees*. PhD thesis, University of Warsaw, 2012. unpublished.

11 Orna Kupferman, Shmuel Safra, and Moshe Y. Vardi. Relating word and tree automata. In *LICS*, pages 322–332. IEEE Computer Society, 1996.

12 Shmuel Lifsches and Saharon Shelah. Uniformization and skolem functions in the class of trees. *J. Symb. Log.*, 63(1):103–127, 1998.

13 Filip Murlak. The Wadge hierarchy of deterministic tree languages. *Logical Methods in Computer Science*, 4(4), 2008.

14 Damian Niwiński and Igor Walukiewicz. Ambiguity problem for automata on infinite trees. unpublished, 1996.

15 Damian Niwiński and Igor Walukiewicz. Relating hierarchies of word and tree automata. In *STACS*, pages 320–331, 1998.

16 Damian Niwiński and Igor Walukiewicz. A gap property of deterministic tree languages. *Theor. Comput. Sci.*, 1(303):215–231, 2003.

17 Damian Niwiński and Igor Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Electr. Notes Theor. Comput. Sci.*, 123:195–208, 2005.

18 Dominique Perrin and Jean Éric Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. Elsevier, 2004.

19 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. of the American Math. Soc.*, 141:1–35, 1969.

20 Alexander Rabinovich. On decidability of monadic logic of order over the naturals extended by monadic predicates. *Information and Computation*, 205(6):870–889, 2007.

21 Alexander Rabinovich and Sasha Rubin. Interpretations in trees with countably many branches. In *LICS*, pages 551–560. IEEE, 2012.

22 Alexander Rabinovich and Amit Shomrat. Selection in the monadic theory of a countable ordinal. *J. Symb. Log.*, 73(3):783–816, 2008.

23 Saharon Shelah. The monadic theory of order. *The Annals of Mathematics*, 102(3):379–419, 1975.

24 Dirk Siefkes. The recursive sets in certain monadic second order fragments of arithmetic. *Arch. Math. Logik*, 17(1–2):71–80, 1975.

25 Richard Edwin Stearns and Harry B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM J. Comput.*, 14(3):598–611, 1985.

26 Wolfgang Thomas. Languages, automata and logics. Technical Report 9607, Institut für Informatik und Praktische Mathematik, Christian-Albsechts-Universität, Kiel, Germany, 1996.

27 Boris A. Trakhtenbrot. Finite automata and the monadic predicate calculus. *Siberian Mathematical Journal*, 3(1):103–131, 1962.

28 Thomas Wilke. An algebraic theory for regular languages of finite and infinite words. *Int. J. Alg. Comput.*, 3:447–489, 1993.

29 Thomas Wilke. An algebraic characterization of frontier testable tree languages. *Theor. Comput. Sci.*, 154(1):85–106, 1996.

30 Thomas Wilke. Classifying discrete temporal properties. Habilitationsschrift, Universitat Kiel, apr. 1998.

A Thin algebra

First, let us write explicitly all the axioms of thin algebra (we assume that $h, h_l, h_r \in H$ and $u, v, w \in V$):

1. $(u \cdot v) \cdot w = u \cdot (v \cdot w)$,
2. $(u \cdot v) \cdot h = u \cdot (v \cdot h)$,
3. $(uv)^\infty = u(vu)^\infty$,
4. $(v^n)^\infty = v^\infty$ for every $n \geq 1$,
5. $a(h_l, \Box) \cdot h_r = a(\Box, h_r) \cdot h_l$.

Let R_A be the set of all regular thin trees over a ranked alphabet $A = (N, L)$. Let C_A be the set of all regular thin contexts over A. Note that (R_A, C_A) has the natural structure of a thin algebra over A.

▶ Fact 38. (R_A, C_A) is the free algebra in the class of thin algebras over the alphabet A.

Proof. See [10] for the proof of this fact in the context of forests. ◀

The rest of this section is devoted to showing the following theorem.

▶ **Theorem 9.** *For every regular tree language L there exists a* syntactic morphism for L*: a finite thin algebra $S_L = (H, V)$ (called a* syntactic algebra of L*) and a homomorphism $\alpha_L \colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to S_L$ such that:*

- *α_L is surjective, compositional, and recognises L,*
- *for every $h \in H$ the language $L_h := \alpha_L^{-1}(\{h\})$ is regular,*
- *if $\alpha \colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to S$ is surjective and recognises L then there is exactly one homomorphism $\beta \colon S \to S_L$ such that $\beta \circ \alpha = \alpha_L$.*

A syntactic algebra S_L and languages L_h can be effectively computed basing on a nondeterministic automaton recognising L.

A syntactic algebra S_L of a given language L can be constructed using standard tools of universal algebra (namely the congruence $\sim_L$). What remains is to show that the constructed algebra is finite. For this purpose we provide some homomorphism $\alpha \colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$ that recognises L (see Theorem 41 of [10]) and such that (H, V) is a finite thin algebra. Then, by the universal property of the syntactic algebra, S_L is a surjective image of (H, V), thus S_L is finite.

Let us define a relation $\sim_L$ on the sets Tr_A and Con_A. We assume that $t, t' \in \mathrm{Tr}_A$, $c, c' \in \mathrm{Con}_A$, and D denotes the set of all multi-contexts over the alphabet A.

$$t \sim_L t' \iff \text{for every } d \in D \text{ we have } (d \cdot t \in L \Leftrightarrow d \cdot t' \in L)$$
$$c \sim_L c' \iff \text{for every } d \in D \text{ and } s \in \mathrm{Tr}_A \text{ we have } (d \cdot (c \cdot s) \in L \Leftrightarrow d \cdot (c' \cdot s) \in L)$$

▶ Fact 39. The relation $\sim_L$ is a congruence on $(\mathrm{Tr}_A, \mathrm{Con}_A)$ with respect to the operations of thin algebra. Moreover, if $\alpha \colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$ recognises L then (by compositionality of α)

$$\alpha(t) = \alpha(t') \implies t \sim_L t' \text{ and } \alpha(c) = \alpha(c') \implies c \sim_L c'. \tag{4}$$

We define $S_L = (H_L, V_L)$ as the quotient of $(\mathrm{Tr}_A, \mathrm{Con}_A)$ by the relation $\sim_L$ defined above. Since $\sim_L$ is a congruence, so S_L has a structure of thin algebra. We define α_L as the quotient morphism of $\sim_L$.

Now we construct some finite thin algebra recognising L. Let $\mathcal{A}$ be a nondeterministic automaton over an alphabet A with states Q such that $\mathcal{A}$ recognises L. Assume that $\mathcal{A}$ uses priorities $\{0,\dots,k\}$. First, recall the definition of $Q_{\mathcal{A}}(t)$ from (1):

$$Q_{\mathcal{A}}(t) = \{q \in Q : \exists_\rho\ \rho \text{ is a consistent run of } \mathcal{A} \text{ on } t \text{ with value } q\} \subseteq 2^Q.$$

Similarly, if c is a context over A then let $\Delta_{\mathcal{A}}(c)$ contain those pairs $(q,i,p) \in Q \times \{0,\dots,k\} \times Q$ such that there exists a consistent run ρ of $\mathcal{A}$ on c with the value q, the value in the hole p, and the maximal priority on the path from the root to the hole equal i.

Now consider the function

$$\alpha_{\mathcal{A}} \colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (2^Q, 2^{Q\times\{0,\dots,k\}\times Q})$$

that assigns to a tree $t \in \mathrm{Tr}_A$ the set $Q_{\mathcal{A}}(t)$ and assigns to a context $c \in \mathrm{Con}_A$ the set $\Delta_{\mathcal{A}}(c)$.

▶ Fact 40. The function $\alpha_{\mathcal{A}}$ induces uniquely the structure of thin algebra on its image $S_{\mathcal{A}} := (H_{\mathcal{A}}, V_{\mathcal{A}}) \subseteq (2^Q, 2^{Q\times\{0,\dots,k\}\times Q})$ in such a way that $\alpha_{\mathcal{A}}$ becomes a compositional homomorphism of thin algebras. Moreover, $\alpha_{\mathcal{A}}$ recognises $\mathrm{L}(\mathcal{A})$, since

$$\mathrm{L}(\mathcal{A}) = \alpha_{\mathcal{A}}^{-1}\left(\{h \in H_{\mathcal{A}} : \ h \cap I^{\mathcal{A}} \neq \emptyset\}\right).$$

The algebra $S_{\mathcal{A}}$ along with the homomorphism $\alpha_{\mathcal{A}}$ defined above is called the *automaton algebra for* $\mathcal{A}$. The following lemma presents an important feature of this algebra.

▶ **Lemma 41.** *Assume that $\mathcal{A}$ is a nondeterministic tree automaton over an alphabet A, $t \in \mathrm{Tr}_A$ is a tree, and τ is a consistent marking of t by types in $H_{\mathcal{A}}$. Let $q \in Q^{\mathcal{A}}$ be a state of $\mathcal{A}$. The following conditions are equivalent:*

- $q \in \tau(\epsilon)$
- *There exists a run (possibly not consistent) ρ of $\mathcal{A}$ on t with value q such that for every vertex $w \in \mathrm{dom}(t)$ we have $\rho(w) \in \tau(w)$. Additionally, for every infinite branch π of t there exists a run ρ_π as above that is consistent on π.*

Proof. First assume that $q \in \tau(\epsilon)$. We inductively show that there exists a run of $\mathcal{A}$ on t satisfying $\rho(w) \in \tau(w)$. Assume that $t = a(t_l, t_r)$ for a pair of trees t_l, t_r. Let $h = \tau(\epsilon)$, $h_l = \tau(l)$, and $h_r = \tau(r)$. We need to find a transition $(q, q_l, a, q_r) \in \delta_2^{\mathcal{A}}$ such that $q_l \in h_l$ and $q_r \in h_r$. Let t_l', t_r' be trees that are mapped by $\alpha_{\mathcal{A}}$ to h_l, h_r respectively. Observe that

$$q \in h = a(h_l, h_r) = \alpha_{\mathcal{A}}\left(a(t_l', t_r')\right),$$

therefore there exists a consistent run with value q on $a(t_l', t_r')$. The first transition used by this run gives us the states $q_l \in h_l, q_r \in h_r$. Note that if w is a leaf of t and $q \in \tau(w)$ then $(q, t(w)) \in \delta_0$, so the constructed run is also consistent in leafs.

Using the above observation, it is enough to construct a run ρ along π that satisfies $\rho(w) \in \tau(w)$ for every w that is off π — it will extend to a run on the subtree $t\restriction_w$. The existence of such a run follows from the definition of operations of thin algebra, see Section 4.4.1 of [10] — the fact that $q \in \tau(\epsilon)$ comes from the fact that for every Ramsey decomposition $s \cdot e^\infty$ of the contexts along the branch π, there is a loop of transitions in $s \cdot e^\infty$ starting in q and satisfying the parity condition.

Now assume that the second bullet of the statement is satisfied. We want to show that $q \in \tau(\epsilon)$. If the tree t is finite then $q \in \tau(\epsilon)$ by induction on the height of t. Otherwise, there exists an infinite branch π of t and similarly as above, any run ρ_π consistent on π is a witness that $q \in h$. ◀

▶ **Lemma 42.** *The automaton morphism* $\alpha_{\mathcal{A}}\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H_{\mathcal{A}}, V_{\mathcal{A}})$ *can be computed effectively basing on* $\mathcal{A}$*. The syntactic algebra* S_L *for* $L = \mathrm{L}(\mathcal{A})$ *and the unique homomorphism* $\beta\colon (H_{\mathcal{A}}, V_{\mathcal{A}}) \to S_L$ *are computable effectively basing on* α_L*.*

Proof. The homomorphism $\alpha_{\mathcal{A}}$ and the structure of thin algebra of $(H_{\mathcal{A}}, V_{\mathcal{A}})$ can be written *by hand*, see Section 4.4.1 from [10].

The homomorphism β can be computed using Moore's algorithm, see Lemma 23 of the cited thesis. The construction is similar to the minimisation of a finite deterministic automaton: we mark pairs of elements of $H_{\mathcal{A}}$ and $V_{\mathcal{A}}$ as non-equivalent. We start with all the pairs in $F \times (H_{\mathcal{A}} \setminus F)$ where $\alpha_{\mathcal{A}}^{-1}(F) = L$. Then we iteratively add a pair (s, s') whenever there is an operation of thin algebra (with some parameters fixed) that maps s, s' into r, r' respectively and (r, r') is a marked pair. After a finite number of steps no new pair can be marked and the set of non-marked pairs is a congruence $\sim$ on $(H_{\mathcal{A}}, V_{\mathcal{A}})$. β can be defined as the quotient morphism induced by $\sim$. ◀

B Transducer for an uniformized relation

Let $A = (N, L), M = (M_2, M_0)$ be a pair of ranked alphabets. Let $B = N \sqcup L$. A *transducer from* A *to* M is a deterministic device $\mathcal{T} = (Q, \delta, q_I)$ such that:

1. Q is a finite set of states,
2. $q_I \in Q$ is an initial state,
3. δ is a pair of functions δ_2, δ_0,
4. $\delta_2\colon Q \times B \times N \times B \to Q \times M_2 \times Q$ determines transitions in internal nodes,
5. $\delta_0\colon Q \times L \to M_0$ determines transitions in leafs.

Note that a transition in an internal node w takes three letters as the input: the letter in wl, w, and wr.

For every tree $t \in \mathrm{Tr}_A$ a transducer $\mathcal{T}$ defines inductively the labelling $\mathcal{T}(t)$ of t by letters in M. The definition is inductive. We start in $w = \epsilon$ in the state q_I. Assume that the transducer reached a vertex $w \in \mathrm{dom}(t)$ in a state q. If w is a leaf then we put $\mathcal{T}(t)(w) = \delta_0(q, t(w))$. Otherwise, let a_l, a, a_r be letters of t in wl, w, wr respectively. Then let $\delta_2(q, a_l, a, a_r) = (q_l, m, q_r)$, put $\mathcal{T}(t)(w) = m$, and continue in wl, wr in states q_l, q_r respectively.

▶ Fact 43. The value $\mathcal{T}(t)(w)$ in a vertex $w \in \mathrm{dom}(t)$ depends on the letters of t in vertices of the form v, vl, vr for $v \prec w$. That is, if t, t' agree on all vertices v, vl, vr for $v \prec w$ then $\mathcal{T}(t)(w) = \mathcal{T}(t')(w)$.

▶ **Theorem 24.** *Assume that* $L_A \subseteq \mathrm{Tr}_A, L_M \subseteq \mathrm{Tr}_{A \times M}$ *are regular languages of trees for two ranked alphabets* A, M *such that* L_A *is a projection of* L_M *onto* A*. Assume that* $\forall_{t_A \in L_A}\ \exists!_{t_M \in \mathrm{Tr}_M}\ (t_A, t_M) \in L_M$*. Then, there exist:*

- *a compositional homomorphism* $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to S$ *into a finite thin algebra* S*,*
- *a deterministic finite state transducer that reads the marking induced by* α *on a given tree* t_A *and outputs the labelling* t_M *such that* $(t_A, t_M) \in L_M$*, whenever such* t_M *exists.*

▶ **Example 44.** Let $\mathcal{A}$ be an unambiguous tree automaton. Let $L_A = \mathrm{L}(\mathcal{A})$ and L_M contain pairs (t, ρ) where ρ is an accepting run of $\mathcal{A}$ on $t \in \mathrm{Tr}_A$. Then, the above theorem states that there exists a transducer that reads the marking induced by some homomorphism α on a given tree $t \in \mathrm{L}(\mathcal{A})$ and produces the accepting run of $\mathcal{A}$ on t.

A simple proof of the above theorem can be given using the composition method (see [23]). This proof was suggested by Mikołaj Bojańczyk as a simplification of an earlier proof given by the authors. However, since we are focused on automata, we only sketch it here and give a longer self-contained proof below. Assume that there is an MSO formula defining language L_M that has quantifier depth n. Let $|M| = k$ and let $\alpha\colon (\mathrm{Tr}_A, \mathrm{Con}_A) \to (H, V)$ be a homomorphism that recognises all the $(n{+}k{+}1)$-types of MSO over A. In a vertex w the transducer $\mathcal{T}$ can store in its memory the $(n{+}m{+}1)$-type of the currently read context. Then, given $(n{+}k{+}1)$-types of both subtrees under w, it can compute the $(n{+}k)$-type of the tree $t[x := w]$ with the current vertex w denoted by an additional variable x. The $(n{+}k)$-type of $t[x := w]$ is enough to ask about the truth value of the following formulas (for every $a \in M_2$):

there is a labelling $t_M \in L_M$ of $t[x := w]$ such that $t_M(x) = a$.

If there is any such labelling t_M, then the above formula is true for exactly one letter $a \in M_2$. The transducer $\mathcal{T}$ outputs this letter in w and proceeds in wl, wr updating the type of the context.

The rest of this section is devoted to an automata-based proof of Theorem 24.

Let $\mathcal{A}$ be some nondeterministic tree automaton recognising the language L_M. Let Q be the set of states of $\mathcal{A}$. Consider a modification $\bar{\mathcal{A}}$ of the automaton $\mathcal{A}$ where letters of M used in transitions are removed. Formally, $\bar{\mathcal{A}}$ is a projection of $\mathcal{A}$ from the alphabet $A \times M$ to A. Note that $\mathrm{L}(\bar{\mathcal{A}}) = L_A$. Let us fix the alphabet $G = (2^Q, 2^Q)$.

Let $\alpha_{\bar{\mathcal{A}}}$ be the automaton morphism into the automaton algebra $(H_{\bar{\mathcal{A}}}, V_{\bar{\mathcal{A}}})$ for $\bar{\mathcal{A}}$. Let $t_A \in \mathrm{Tr}_A$ be a tree. Let $\tau(t_A) = \tau_{\bar{\mathcal{A}}}(t_A)$ be the marking induced by the automaton morphism $\alpha_{\bar{\mathcal{A}}}$ on t_A, that is $\tau(t_A)(w) = Q_{\mathcal{A}}(t_A \restriction_w)$.

The construction goes as follows. The input alphabet is $A \times G$. The set of states $Q^{\mathcal{T}}$ of $\mathcal{T}$ is 2^Q. The state $\emptyset \in Q^{\mathcal{T}}$ is a sink state reached if the given tree does not belong to L_A.

The invariant for non-sink states is: if $\mathcal{T}$ is in a vertex w and it have assigned letters $m_v \in M$ to all vertices $v \prec w$ then the state S_w of $\mathcal{T}$ in w satisfies:

$$S_w = \{q \in Q : \text{exists an accepting run of } \bar{\mathcal{A}} \text{ on } t_A \text{ using letters } m_v \text{ in vertices } v \prec w\}. \quad (5)$$

We will show that the invariant can be preserved. Let us fix a moment during the computation of $\mathcal{T}$: we are in a vertex $w \in \mathrm{dom}(t_A)$. We can assume that w is an internal node of t_A. We have already assigned letters $m_v \in M$ to all nodes $v \prec w$. The marking $\tau(t_A)$ gives us sets $Q_{wl}, Q_{wr} \subseteq Q$ in nodes wl, wr respectively. The current state of $\mathcal{T}$ is a set of states $S_w \subseteq Q$.

Consider the following set of letters:

$$P_w = \left\{ m \in M_2 : \exists_{(q,q_l,(t_A(w),m),q_r)\in\delta_2^{\mathcal{A}}}\ q \in S_w \wedge q_l \in Q_{wl} \wedge q_r \in Q_{wr} \right\}.$$

If $P_w = \emptyset$ then let $\mathcal{T}$ fall in a sink state $\emptyset \in 2^Q$ and from that point on output some fixed letters (of arity 2 and 0 respectively) $(m_2, m_0) \in M$. We will show that during the run of $\mathcal{T}$ on any tree $t_A \in L_A$ the sets P_w are nonempty. But first we show the following lemma.

▶ **Lemma 45.** *The set P_w contains at most one letter.*

Proof. Let $t(w) = a$. Assume contrary that there are two letters $m, m' \in P_w$. Consider the respective transitions $(q, q_l, (a, m), q_r)$ and $(q, q'_l, (a, m'), q'_r)$. Since $q, q' \in S_w$ so by (5) there are two accepting runs ρ, ρ' of $\bar{\mathcal{A}}$ on $t_A[\Box/w]$ that assign letters m_v to $v \prec w$ and have values q, q' respectively in the hole w.

For $d \in \{l, r\}$ let $t_d, t'_d \in \mathrm{Tr}_M$ be trees and ρ_d, ρ'_d be consistent runs of $\mathcal{A}$ that witness that $q_d, q'_d \in Q_{wd}$, i.e. ρ_d is a consistent run of $\mathcal{A}$ on $(t_A \restriction_{wd}, t_d)$ with value q_d, similarly for t'_d, ρ'_d, q'_d.

Consider now two trees over the alphabet $A \times M \times Q$:

$$t = (t_A[\Box/w], \rho) \cdot (a, m, q)((t_A \restriction_{wl}, t_l, \rho_l), (t_A \restriction_{wr}, t_r, \rho_r)),$$
$$t' = (t_A[\Box/w], \rho') \cdot (a, m', q')((t_A \restriction_{wl}, t'_l, \rho'_l), (t_A \restriction_{wr}, t'_r, \rho'_r)).$$

Note that:

- both t, t' equal t_A on the A'th coordinate,
- they differ in vertex w on the M'th coordinate,
- the Q'th coordinate of t, t' denotes an accepting run of $\mathcal{A}$ on the $A \times M$ coordinates.

Therefore, we have a contradiction: t_A has two different labellings t_M, t'_M (one with m and the other with m' in w) such that $(t_A, t_M) \in L_M$ and $(t_A, t'_M) \in L_M$. ◀

Let $\mathcal{T}$ select as the letter m_w the only element of P_w whenever $P_w \neq \emptyset$. By the definition of P_w, the invariant (5) holds in the vertices wl, wr.

Now take any tree $t_A \in L_A$ and consider the result $t_R = \mathcal{T}(t_A, \tau(t_A))$. Let t_M be the unique labelling of t_A such that $(t_A, t_M) \in L_M$. Let ρ be an accepting run of $\mathcal{A}$ on (t_A, t_M). We show inductively that $t_R = t_M$ what finishes the proof. Let w be a node of t_A and assume that for all $v \prec w$ we have $t_R(v) = t_M(v)$. Let $(q, q_l, (a, m), q_r)$ be the transition used by ρ in w. By the definition of P_w this transition is a witness that $m \in P_w$. Therefore, P_w is not empty and $t_R(w) = m = t_M(w)$.

A characterization of the Taylor expansion of λ-terms*

Pierre Boudes, Fanny He, and Michele Pagani

LIPN – University Paris 13
Villetaneuse, France
{boudes, he, pagani}@lipn.univ-paris13.fr

Abstract

The Taylor expansion of λ-terms, as introduced by Ehrhard and Regnier, expresses a λ-term as a series of multi-linear terms, called simple terms, which capture bounded computations. Normal forms of Taylor expansions give a notion of infinitary normal forms, refining the notion of Böhm trees in a quantitative setting.

We give the algebraic conditions over a set of normal simple terms which characterize the property of being the normal form of the Taylor expansion of a λ-term. From this full completeness result, we give further conditions which semantically describe normalizable and total λ-terms.

1998 ACM Subject Classification F.4.1 Lambda calculus and related systems, F.3.2 Denotational semantics

Keywords and phrases Lambda-Calculus, Böhm trees, Differential Lambda-Calculus, Linear Logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.101

1 Introduction

Recently various semantics of linear logic and λ-calculus have been proposed (e.g. [5, 6]), where morphisms are infinitely differentiable functions between vector spaces (or more generally, modules). In fact, one can define the Taylor expansion of a function as an infinite sum of terms that are calculated from the values of the function's n-th derivatives at a given point. These models subsequently provide an intriguing way to describe the regularity of the behavior of a λ-term — the functions interpreting λ-terms are analytic, i.e. they are equal to their Taylor expansion at any point of their domain.

The main interest of Taylor expansion for the λ-calculus lies in the analogy that can be drawn between the usual notion of differentiation and its computational meaning. In fact, applying the derivative at 0 of the λ-term M on the argument N corresponds to passing the input N to M *exactly once*. This can be formalized in the setting of the *differential λ-calculus* [7] — an extension of the λ-calculus with a syntactic derivative operator, allowing to compute the optimal approximation of a program when applied to depletable arguments. The Taylor formula yields a natural notion of linear approximation of the ordinary application of λ-calculus. Let M and N be two λ-terms and assume $(\mathtt{D}^n M \cdot N^n)0$ denotes the n-th derivative of M at 0 applied to N (as in the usual notation $f^{(n)}(0) \cdot x^n$ with M as function f, N as argument x, and 0 as non-linear argument). Then the application of the term M to

* This work was supported by the ANR 12 JS02 006 01 project COQUAS: Computing with quantitative semantics.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 101–115

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the term N becomes:

$$(M)N = \sum_{n=0}^{\infty} \frac{1}{n!} (\mathtt{D}^n M \cdot N^n)0 \; . \tag{1}$$

Basically, $(\mathtt{D}^n M \cdot N^n)0$ expresses an evaluation of the application $(M)N$ using exactly n linear copies of N. The coefficient $\frac{1}{n!}$ is there to take care of the number of permutations on the argument of $\mathtt{D}^n M$.

More generally, if one fully develops each application occurring in a λ-term M into its corresponding Taylor expansion, one expresses the term as an infinite sum M^* of purely "differential programs", all of which containing only multi-linear applications and applications to 0. Ehrhard and Regnier develop a convenient notation for expressing such "differential programs", calling them *simple terms* [8, 9]. Also, the authors define a rewriting system inspired by the standard rules for computing derivatives of polynomials which allow to give a normal form $\mathrm{NF}(M^*)$ to a Taylor series M^*.

We denote by Δ^{NF} the set of normal simple terms (Equation (10)) and by $\mathbb{Q}^+[\![\Delta^{\mathrm{NF}}]\!]$ the formal linear combinations of terms in Δ^{NF} taking coefficients in the semi-ring of positive rational numbers. The combinations of $\mathbb{Q}^+[\![\Delta^{\mathrm{NF}}]\!]$ allow to express the normal forms of the Taylor expansion of λ-terms, however not all formal combinations can be associated with a λ-term. We propose here a characterization of the image of the λ-calculus into $\mathbb{Q}^+[\![\Delta^{\mathrm{NF}}]\!]$.

Let us underline that the space $\mathbb{Q}^+[\![\Delta^{\mathrm{NF}}]\!]$ is an example of syntax-based semantics of the untyped λ-calculus (as well as of its differential extension) — the interpretation being given by the function $M \mapsto \mathrm{NF}(M^*)$, and the composition being assured by a notion of linear substitution (see Equation (5)). Roughly speaking, such a model is a quantitative refinement of the model provided by Böhm trees [1, §10] and it is crucial to the study of vectors based semantics, exactly as Böhm trees are at the core of domain-based denotational semantics. Our result basically amounts to defining a sub-model of $\mathbb{Q}^+[\![\Delta^{\mathrm{NF}}]\!]$ which is fully complete for the untyped λ-calculus.

Fortunately, the problem is significantly simplified by Ehrhard and Regnier's result. In particular, they prove [9, Corollary 35] that the normal form of the Taylor expansion of a λ-term M can be defined as:

$$\mathrm{NF}(M^*) = \sum_{t \in \mathrm{NF}(\tau(M))} \frac{1}{\mathrm{m}(t)} t \tag{2}$$

where: $\mathrm{m}(t)$ is a natural number (called *multiplicity coefficient*) univocally defined by t, and $\mathrm{NF}(\tau(M))$ is the *support* of the normal form of the Taylor expansion of M, i.e. the set of the simple terms appearing with a non-zero coefficient in $\mathrm{NF}(M^*)$. The open issue is then to characterize the sets of simple terms which are of the form $\mathrm{NF}(\tau(M))$, for some λ-term M.

We give three conditions which are necessary and sufficient for a subset $\mathcal{T}$ of Δ^{NF} to be equal to $\mathrm{NF}(\tau(M))$, for some λ-term M (Theorem 25): (i) the set of free variables occurring in $\mathcal{T}$ must be finite; (ii) $\mathcal{T}$ must be recursively enumerable; (iii) $\mathcal{T}$ must be an ideal with respect to a definedness relation (Definition 9).

Our characterization is based on two previous results: a theorem by Ehrhard and Regnier stating that Taylor expansion and normalization (λ-terms via Böhm trees; and sets of simple terms via the NF operator) commute (here Theorem 8), and Barendregt's characterization of the Böhm-like trees which are Böhm trees of λ-terms (here Theorem 5). Conditions (i) and (ii) are in fact an adaptation of Barendregt's characterization, but they are not sufficient to characterize Taylor expansions since the Taylor expansion gives a more atomic decomposition of λ-terms than that obtained from Böhm trees. Condition (iii) is a completeness condition,

assuring that there is no hole in the description of the support of a Taylor expansion. This condition looks like the usual one characterizing the set of finite approximants of a tree as an ideal with respect to the subtree order relation. However, the fine grained notion of approximant given by the simple terms makes our definedness relation a bit subtler than a subtree relation, in fact it is not even a preorder relation.

Our result concerns only the supports of the formal combinations in $\mathbb{Q}^+[\![\Delta^{\mathrm{NF}}]\!]$, i.e. the subsets of Δ^{NF}, the issue about coefficients being completely accomplished by Equation 2. Actually, the powerset of Δ^{NF} has an interest by its own. In fact, the powerset of Δ^{NF} can be formally seen as the space $\mathbf{Bool}[\![\Delta^{\mathrm{NF}}]\!]$ of the formal combinations with coefficients in the boolean ring $\mathbf{Bool} = (\{0,1\}, \max, \min, 0, 1)$. The Taylor expansion of a λ-term M into the space $\mathbf{Bool}[\![\Delta^{\mathrm{NF}}]\!]$ is the support $\tau(M)$ of the Taylor expansion of M into $\mathbb{Q}^+[\![\Delta^{\mathrm{NF}}]\!]$. Let us remark that $\mathbf{Bool}[\![\Delta^{\mathrm{NF}}]\!]$ gives another syntax-based quantitative semantics of the λ-calculus, and our result characterizes the image of the λ-calculus into $\mathbf{Bool}[\![\Delta^{\mathrm{NF}}]\!]$.

Related works. The question of characterizing the support of Ehrhard and Regnier's Taylor expansion has already been addressed by Pagani and Tasson in the setting of the simply typed linear logic proof-nets [12]. In that paper, the authors define a rewriting algorithm taking as input a finite set of cut-free differential nets (corresponding here to the normal simple terms) and either returning a cut-free proof-net or falling in a deadlock. Although related, our approach is different from that of [12] in various points. First, we are considering the untyped λ-calculus, while [12] deals with a simply typed (hence strongly normalizing) framework. Second, the criterion proposed by [12] is the termination of a rewriting relation in a proof-net. In the present setting, this will amount to refer to the termination of an algorithm which starts with a set $\mathcal{T}$ of simple terms and tries to compute a λ-term M such that $\mathcal{T} \subseteq \mathrm{NF}(\tau(M))$. We are giving more abstract conditions without referring to the termination of a computation. Finally, [12] characterizes the property of being a *finite subset* of the support of the Taylor expansion, while we are capturing here the property of being the whole support ($\mathcal{T} = \mathrm{NF}(\tau(M))$).

Structure of the paper. Section 2 recalls the standard notions on λ-calculus and Böhm trees needed in the sequel. In particular, Definition 2 introduces Böhm-like trees and Theorem 5 states Barendregt's characterization of the set of Böhm-like trees corresponding to λ-terms. Section 3 defines Ehrhard and Regnier's resource λ-calculus, the normal form operator NF and the support $\tau(M)$ of the Taylor expansion of a λ-term M (Equation 8). Theorem 8 recalls the commutation of the Taylor expansion with Böhm trees. Finally, Section 4 contains the original results of the paper. Our characterization of the normal forms of the Taylor expansions of λ-terms is given in Theorem 25. Corollary 28 gives an answer to what the maximal cliques of simple terms correspond to, a question addressed in [9]. Finally, Corollary 29 states the conditions characterizing the property of being a normalizable λ-term.

2 Λ-Calculus and Böhm trees

We denote by Λ the set of λ-terms, written using Krivine's convention [11]:

$$\Lambda : M, N ::= x \mid (M)N \mid \lambda x.M,$$

where x ranges over a countable set Var of variables. As usual, we suppose that application associates to the left and λ-abstraction to the right. The *α-conversion* and the set $\mathrm{FV}(M)$ of *free variables of* M are defined following [11]. A term M is *closed* whenever $\mathrm{FV}(M) = \emptyset$.

Given two terms M, N, we denote by $M\{N/x\}$ the term obtained by simultaneously substituting N to all free occurrences of x in M, subject to the usual proviso about renaming bound variables in M to avoid capture of free variables in N.

Hereafter terms are considered up to α-conversion. We define the following terms:

$$\mathbf{I} := \lambda x.x, \qquad \mathbf{S} := \lambda xyz.((x)z)(y)z, \qquad \mathbf{\Omega} := (\lambda x.(x)x)\lambda x.(x)x,$$
$$\mathbf{\Theta}_f := \lambda x.(f)(x)x, \qquad \mathbf{\Theta} := \lambda f.(\mathbf{\Theta}_f)\mathbf{\Theta}_f.$$

The β-reduction $\xrightarrow{\beta}$ is the smallest relation over Λ containing the β-step $(\lambda x.M)N \xrightarrow{\beta} M\{N/x\}$ and closed under the following context rules (supposing $M \xrightarrow{\beta} M'$):

$$\text{(abs)}: \ \lambda x.M \xrightarrow{\beta} \lambda x.M', \qquad \text{(fun)}: \ (M)N \xrightarrow{\beta} (M')N, \qquad \text{(arg)}: \ (N)M \xrightarrow{\beta} (N)M'. \tag{3}$$

We denote by $\xrightarrow{\beta*}$ the reflexive and transitive closure of $\xrightarrow{\beta}$. A *β-normal form* is a normal form for $\xrightarrow{\beta}$. The reduction $\xrightarrow{\beta}$ is confluent which implies that the β-normal form of a λ-term is unique, whenever it exists. However, $\xrightarrow{\beta}$ is not normalizing, i.e. there are λ-terms M without a β-normal form, e.g. $\mathbf{\Omega}$ and $\mathbf{\Theta}$. Indeed, $\mathbf{\Omega}$ reduces only to itself, i.e. $\mathbf{\Omega} \xrightarrow{\beta} \mathbf{\Omega}$, while $\mathbf{\Theta}$ yields the infinite reduction sequence $\mathbf{\Theta} \xrightarrow{\beta} \lambda f.(f)(\mathbf{\Theta}_f)\mathbf{\Theta}_f \xrightarrow{\beta} \lambda f.(f)(f)(\mathbf{\Theta}_f)\mathbf{\Theta}_f \xrightarrow{\beta} \dots$.

Notice however that $\mathbf{\Omega}$ and $\mathbf{\Theta}$ are quite different. Any application $(\mathbf{\Omega})M_0 \dots M_{n-1}$ gives a non-normalizing λ-term, independently from $M_0, \dots, M_{n-1}$. In that sense $\mathbf{\Omega}$ represents the everywhere undefined function. On the contrary, $\mathbf{\Theta}$ is a fundamental term producing a potentially infinite iteration of its first argument, i.e. $(\mathbf{\Theta})M_0 \dots M_{n-1} \xrightarrow{\beta*} (M_0)(\mathbf{\Theta})M_0M_1 \dots M_{n-1} \xrightarrow{\beta*} (M_0)(M_0)(\mathbf{\Theta})M_0M_1 \dots M_{n-1} \xrightarrow{\beta*} \dots$, and so it converges for certain $M_0, \dots, M_{n-1}$ for appropriate reduction strategies. Böhm trees are introduced in [1, §10] and provide a notion of infinitary normal form allowing to distinguish between totally undefined terms, like $\mathbf{\Omega}$, and terms like $\mathbf{\Theta}$, which are not normalizing but can interact with their arguments giving convergent computations.

The Böhm tree of a term is defined by using the *head reduction* $\xrightarrow{h}$, a deterministic restriction of $\xrightarrow{\beta}$ obtained by forbidding (arg) in the set of rules (3) and by restricting the (fun) rule to the case where M is not an abstraction. A *head normal form* is a normal form for the reduction $\xrightarrow{h}$, and it always has the shape: $\lambda x_0 \dots x_{m-1}.(y)M_0 \dots M_{n-1}$, for $n, m \geq 0$, $x_0, \dots, x_{m-1}$, y variables, and $M_0, \dots, M_{n-1}$ λ-terms. The variable y is called the *head variable*, and $M_0, \dots, M_{n-1}$ its arguments.

Notice that $\mathbf{\Omega}$ has no head normal form, while $\lambda f.(f)(\mathbf{\Theta}_f)\mathbf{\Theta}_f$ is a head normal form of $\mathbf{\Theta}$. Basically, the Böhm tree $\mathrm{BT}(M)$ of a λ-term M is a (possibly infinite) nesting of head normal forms: if M is β-normalizable, then $\mathrm{BT}(M)$ is just the applicative tree of its β-normal form (e.g. Figure 1a, 1c), otherwise it can be either infinite in depth (e.g. Figure 1d) or with "holes" denoting totally undefined sub-terms (e.g. Figure 1b) or both (e.g. Figure 1e).

We recall some definitions of [1, §10]. We introduce the set of Böhm-like trees (Definition 2), which is the codomain of the function $\mathrm{BT}(\,)$. In general, a labelled tree can be represented as a partial function from the set $\mathbb{N}^*$ of finite sequences of integers to the set of possible labels. In the case of Böhm-like trees, a label is a pair $(\lambda x_0 \dots x_{m-1}.y, n)$, where $\lambda x_0 \dots x_{m-1}.y$ represents the λ-prefix and the head variable of a head normal form, and n the number of its arguments, which is also a bound to the number of children of the node labelled by $(\lambda x_0 \dots x_{m-1}.y, n)$. Definition 4 associates with any λ-term M a Böhm-like tree $\mathrm{BT}(M)$. Not every Böhm-like tree is the Böhm tree of a λ-term, and we recall in Theorem 5 Barendregt's characterization of the image set of $\mathrm{BT}(\,)$.

▶ Notation 1. Greek letters α, β, γ will vary over $\mathbb{N}^*$, the set of finite sequences of natural numbers. We denote by @ the concatenation operator, by $<$ the strict prefix order, by $\text{length}(\alpha)$ the length of a sequence $\alpha \in \mathbb{N}^*$, and by $\langle n_1, \dots, n_k \rangle$ the sequence of $n_1, \dots, n_k$. In particular, $\langle\,\rangle$ is the empty sequence. For example $\langle 1,2\rangle < \langle 1,2,3\rangle = \langle 1\rangle @ \langle 2,3\rangle$.

We recall that, when f is a partial function, $f(x)\downarrow$ means that $f(x)$ is defined and that $f(x)\uparrow$ means that $f(x)$ is undefined.

▶ **Definition 2.** Let $\Sigma \stackrel{\triangle}{=} \{\lambda x_0 \dots x_{m-1}.y \;;\; m \in \mathbb{N}, x_i, y \in \text{Var}\}$. A *Böhm-like tree*[1] is a partial function $\mathcal{B}$ from $\mathbb{N}^*$ to $\Sigma \times \mathbb{N}$ such that:

- if $\mathcal{B}(\alpha)\downarrow$ and $\beta < \alpha$, then $\mathcal{B}(\beta)\downarrow$,
- if $\mathcal{B}(\alpha) = (a, n)$, then $\forall k \geq n$, $\mathcal{B}(\alpha @ \langle k\rangle)\uparrow$.

Actually, a sequence α in the domain of $\mathcal{B}$ describes a path from the root to a node labelled by $\mathcal{B}(\alpha)$. Notice that the enumeration of the children of a node allows some holes, representing the presence of totally undefined arguments in the head normal form associated with such a node. Figure 1 gives examples of a convenient graphical representation of Böhm-like trees, where the number n of a label (a, n) is encoded by using pending edges. More precisely,

Figure 1a is the function $\langle\,\rangle \mapsto (\lambda x.x, 0)$,

Figure 1b is the function $\langle\,\rangle \mapsto (\lambda x.x, 1)$,

Figure 1c is the function $\langle\,\rangle \mapsto (\lambda xyz.x, 2), \langle 0\rangle \mapsto (z, 0), \langle 1\rangle \mapsto (y, 1), \langle 10\rangle \mapsto (z, 0)$,

Figure 1d is the function $\langle\,\rangle \mapsto (\lambda f.f, 1), \langle 0\rangle \mapsto (f, 1), \langle 00\rangle \mapsto (f, 1), \langle 000\rangle \mapsto (f, 1) \dots$,

Figure 1e is the function $\langle\,\rangle \mapsto (f, 2), \langle 1\rangle \mapsto (f, 2), \langle 11\rangle \mapsto (f, 2), \langle 111\rangle \mapsto (f, 2) \dots$.

The *height* of a Böhm-like tree $\mathcal{B}$ is defined by: $\text{height}(\mathcal{B}) \stackrel{\triangle}{=} \sup\{1 + \text{length}(\alpha) \mid \mathcal{B}(\alpha)\downarrow\}$. Remark that the domain of $\mathcal{B}$ can be empty and in that case it has a zero height, otherwise the height is positive or infinite. Given $h \in \mathbb{N}$, the *restriction of* $\mathcal{B}$ *to* h is the Böhm-like tree $\mathcal{B}|_h$ defined by cutting off subtrees at height h: $\mathcal{B}|_h(\alpha) = \mathcal{B}(\alpha)$ if $\text{length}(\alpha) < h$, otherwise $\mathcal{B}|_h(\alpha)\uparrow$. Moreover, if $\mathcal{B}(\langle\,\rangle) = (a, n)$ and $i < n$, we define the *i-th subtree* $\mathcal{B}_i$ of $\mathcal{B}$ as: $\mathcal{B}_i(\alpha) = \mathcal{B}(\langle i\rangle @ \alpha)$.

Trees can be seen as ordered by the set-theoretical inclusion on the graph of their functions. With respect to this order $\{\mathcal{B}|_h\}_{h\in\mathbb{N}}$ is a chain whose limit is $\mathcal{B}$, i.e. $\mathcal{B} = \bigcup_{h\in\mathbb{N}} \mathcal{B}|_h$. Such a remark is useful for proofs and definitions on infinite Böhm-like trees. For example, the set $\text{FV}(\mathcal{B})$ of *free variables* of a Böhm-like tree is defined as follows: if $\text{height}(\mathcal{B}) = 0$, then $\text{FV}(\mathcal{B}) \stackrel{\triangle}{=} \emptyset$; if $\text{height}(\mathcal{B}) = h + 1$, let $\mathcal{B}(\langle\,\rangle) = (\lambda x_0 \dots x_{m-1}.y, n)$, then $\text{FV}(\mathcal{B}) \stackrel{\triangle}{=} (\{y\} \cup \bigcup_{i=0}^{n-1} \text{FV}(\mathcal{B}_i)) \setminus \{x_0, \dots, x_{m-1}\}$; finally, if $\text{height}(\mathcal{B}) = \infty$, then $\text{FV}(\mathcal{B}) \stackrel{\triangle}{=} \bigcup_{h\in\mathbb{N}} \text{FV}(\mathcal{B}|_h)$.

▶ **Definition 3.** A Böhm-like tree $\mathcal{B}$ is *recursively enumerable*, *r.e.* for short, if $\mathcal{B}$ is a partial recursive function (after some coding of Σ).

▶ **Definition 4.** The *Böhm tree* $\text{BT}(M)$ of a λ-term M is defined as follows:

- if $M \stackrel{h*}{\to} \lambda x_0 \dots x_{m-1}.(y) M_0 \dots M_{n-1}$, then,

$$\begin{aligned} \text{BT}(M)(\langle\,\rangle) &\stackrel{\triangle}{=} (\lambda x_0 \dots x_{m-1}.y, n) \\ \text{BT}(M)(\langle i\rangle @ \alpha) &\stackrel{\triangle}{=} \text{BT}(M_i)(\alpha) && \text{if } i < n, \\ \text{BT}(M)(\langle i\rangle @ \alpha) &\uparrow && \text{otherwise.} \end{aligned}$$

- Otherwise, $\text{BT}(M)$ is the totally undefined function.

[1] This is called an *effective Böhm-like tree* in [1, Def. 10.1.9].

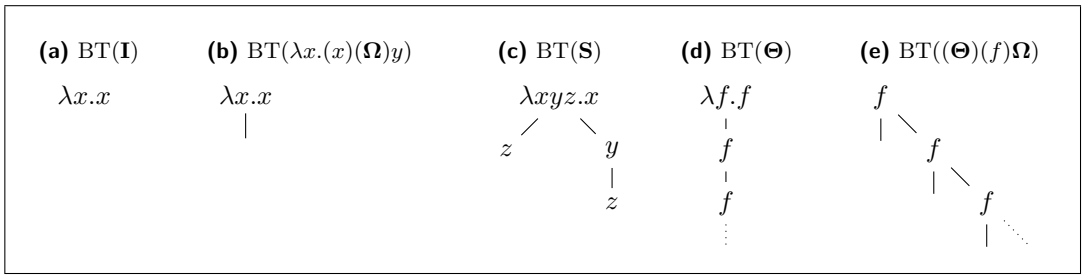


Figure 1 Some examples of Böhm trees.

Figure 1 gives examples of BT( ) values. A Böhm-like tree which is not in the image of a λ-term is the infinite branch

$$\epsilon \mapsto (x_0, 1), \quad \langle 0\rangle \mapsto (x_1, 1), \quad \langle 00\rangle \mapsto (x_2, 1), \quad \langle 000\rangle \mapsto (x_3, 1)\dots$$

for $x_0, x_1, x_2, x_3, \dots$ pairwise distinct variables.

▶ **Theorem 5** ([1, Theorem 10.1.23]). *Let $\mathcal{B}$ be a Böhm-like tree. There is a λ-term M such that $\mathrm{BT}(M) = \mathcal{B}$ if, and only if, $\mathrm{FV}(\mathcal{B})$ is finite and $\mathcal{B}$ is r.e..*

The left-to-right direction is a straight consequence of the remark that $\mathrm{FV}(\mathrm{BT}(M)) \subseteq \mathrm{FV}(M)$ and of the fact that the definition of $\mathrm{BT}(M)$ is effective (hence by Church's Thesis it is r.e.). The proof of the right-to-left direction is more involved. Essentially, it uses a theorem stating that the set Λ^0 of closed λ-terms can be enumerated in such a way that there are two λ-terms $\mathbf{E}$ and $\mathbf{F}$ such that $\forall M \in \Lambda^0$, denoting by $\underline{M}$ the Church numeral associated with M by the enumeration, we have $(\mathbf{E})M =_\beta \underline{M}$ and, vice versa, $(\mathbf{F})\underline{M} =_\beta M$. We refer to [1, 10] for more details.

3 Taylor expansion

We recall the resource calculus as presented in [8, 9]. Let us warn the reader that the name "resource calculus" also refers in the literature to slightly different calculi. In particular, we have the calculus by Boudol et al.'s in [3], which is a resource-sensitive extension of the lazy call-by-value λ-calculus, and Tranquilli's resource λ-calculus [13] which is basically a different notation for Ehrhard and Regnier's differential λ-calculus. The resource calculus of [8, 9], which we briefly recall here, is a fragment of Tranquilli's calculus.

We define the set Δ of the simple terms (Grammar (4)), a rewriting relation $\xrightarrow{r}$ over the finite powerset of Δ (rules (5) and (6)), and a normal form operator NF over the (finite and infinite) powerset of Δ (Equation 7). Concerning the discussion in the Introduction, the simple terms in Δ are a notation that express the terms in a Taylor series (Equation 2) and the subsets of Δ are the supports of such series.

Equation (8) defines the Taylor expansion of the λ-calculus and Equation (9) extends it to Böhm-like trees. Theorem 8 states Ehrhard and Regnier's correspondence between the resource reduction on the Taylor expansion of λ-terms and their Böhm trees.

Resource calculus. The set Δ of *simple terms* is defined by:

$$\Delta : t ::= x \mid \lambda x.t \mid \langle t\rangle\mu, \tag{4}$$

where x ranges over the set Var of variables, and μ is a finite multiset of simple terms, called *bag*. Small Latin letters like s, t, u will vary on simple terms, and small Greek letters μ, ν, ρ

on bags. We recall that we adopt Krivine's notation [11], so, in particular, $\langle t\rangle\mu_1\dots\mu_n$ is a shortcut for $\langle\dots\langle\langle t\rangle\mu_1\rangle\dots\rangle\mu_n$.

▶ Notation 6. We recall that for any set E, a *multiset* on E is a function $\mu : E \to \mathbb{N}$. The *support* $|\mu|$ of μ is the set of all elements e of E such that $\mu(e) \neq 0$, and μ is called a finite multiset when $|\mu|$ is a finite set. We denote by $\mathcal{M}(E)$ the set of multisets on E and by $\mathcal{M}_f(E)$ the set of finite multisets on E. A multiplicative notation is used for bags, 1 is the empty bag and $\mu\cdot\nu$ is the disjoint union of μ and ν. The bag $[t]$ is the singleton containing exactly one occurrence of the simple term t. More occurrences of t can be written as a power: $[t^3] = [t,t,t] = [t]\cdot[t]\cdot[t]$.

Free and bound variables and the α-equivalence $\equiv_\alpha$ are defined as in the λ-calculus.

The symbols $\mathcal{S},\mathcal{T},\mathcal{U}$ will vary over the powerset $\mathcal{P}(\Delta^{\mathrm{NF}})$. By notational convention, we extend all the constructs of the grammar of Δ as point-wise operations on (possibly infinite) sets of simple terms, like for example

$$\lambda x.\mathcal{T} \triangleq \{\lambda x.t \mid t\in\mathcal{T}\},\qquad \langle\mathcal{S}\rangle[\mathcal{T}^2,\mathcal{U}] \triangleq \{\langle s\rangle[t_1,t_2,u] \mid s\in\mathcal{S}, t_1,t_2\in\mathcal{T}, u\in\mathcal{U}\}.$$

The number of free occurrences of x in t, called *degree* of x in t, is written $\deg_x(t)$. A *redex* is a simple term of the shape $\langle\lambda x.t\rangle[s_1,\dots,s_n]$. Its reduction gives a finite set of simple terms, which is empty whenever $\deg_x(t)\neq n$, otherwise it is the set of all possible simple terms obtained by linearly replacing each free occurrence of x with exactly one s_i, for $i=1,\dots,n$. Formally,

$$\langle\lambda x.t\rangle[s_1,\dots,s_n] \xrightarrow{r} \begin{cases} \{t\{s_{\sigma(1)}/x_1,\dots,s_{\sigma(n)}/x_n\} \mid \sigma\in\mathcal{S}_n\} & \text{if } \deg_x(t)=n,\\ \emptyset & \text{otherwise.}\end{cases} \tag{5}$$

where $\mathcal{S}_n$ is the group of permutations over $n=\{1,\dots,n\}$ and $x_1,\dots,x_n$ is any enumeration of the free occurrences of x in t, so that $t\{s_{\sigma(i)}/x_i\}$ denotes the term obtained from t by replacing the i-th free occurrence of x with the term $s_{\sigma(i)}$.

The relation $\xrightarrow{r}\subseteq \Delta\times\mathcal{P}_f(\Delta)$ is extended to $\mathcal{P}_f(\Delta)\times\mathcal{P}_f(\Delta)$ by context closure, i.e. $\xrightarrow{r}$ is the smallest relation on $\mathcal{P}_f(\Delta)\times\mathcal{P}_f(\Delta)$ satisfying Equation 5 and such that, whenever $t\xrightarrow{r}\mathcal{T}$ and $\mathcal{S}\in\mathcal{P}_f(\Delta)$, we have:

$$\lambda x.t\xrightarrow{r}\lambda x.\mathcal{T},\qquad \langle t\rangle\mu\xrightarrow{r}\langle\mathcal{T}\rangle\mu,\qquad \langle s\rangle[t]\cdot\mu\xrightarrow{r}\langle s\rangle[\mathcal{T}]\cdot\mu,\qquad \{t\}\cup\mathcal{S}\xrightarrow{r}\mathcal{T}\cup\mathcal{S}. \tag{6}$$

Notice that the size (*i.e.* the number of symbols) of each simple term in $\mathcal{T}$ is strictly less than the size of t, whenever $t\xrightarrow{r}\mathcal{T}$. Therefore, $\xrightarrow{r}$ is strongly normalizing on $\mathcal{P}_f(\Delta)$. Moreover, one can easily check that $\xrightarrow{r}$ is weakly confluent, thus confluent by Newman's lemma.

▶ **Proposition 7.** *The relation $\xrightarrow{r}$ is strongly normalizing and confluent on $\mathcal{P}_f(\Delta)$.*

In particular, given any simple term t, its unique *normal form* $\mathrm{NF}(t)$ is always well-defined. Such an operator is extended over $\mathcal{P}(\Delta^{\mathrm{NF}})$ by:

$$\mathrm{NF}(\mathcal{T}) \triangleq \bigcup_{t\in\mathcal{T}} \mathrm{NF}(t). \tag{7}$$

We also extend over the notation for free variables by setting $\mathrm{FV}(\mathcal{T}) \triangleq \bigcup_{t\in\mathcal{T}}\{\mathrm{FV}(t)\}$.

Taylor expansion. We define the map τ from Λ to $\mathcal{P}_f(\Delta)$ by structural induction on the λ-terms. As said in the Introduction, τ is the support of the Taylor expansion described in [7, 9], from the λ-calculus to the infinite formal linear combinations in $\mathbb{Q}^+[\![\Delta]\!]$, as well as the Taylor expansion into $\mathbf{Bool}[\![\Delta]\!]$.

$$\tau(x) \triangleq x, \qquad \tau(\lambda x.M) \triangleq \lambda x.\tau(M), \qquad \tau((M)N) \triangleq \bigcup_{n=0}^{\infty} \langle \tau(M)\rangle[\tau(N)^n]. \tag{8}$$

For example, we have

$$\begin{aligned}
\tau(\mathbf{I}) &\triangleq \{\lambda x.x\},\\
\tau(\mathbf{S}) &\triangleq \{\lambda xyz.\langle x\rangle[z^{n_0}][\langle y\rangle[z^{n_1}],\dots,\langle y\rangle[z^{n_k}]] \;;\; k,n_0,\dots,n_k \in \mathbb{N}\},\\
\tau(\mathbf{\Omega}) &\triangleq \{\langle \lambda x.\langle x\rangle[x^{n_0}]\rangle[\lambda x.\langle x\rangle[x^{n_1}],\dots,\lambda x.\langle x\rangle[x^{n_k}]] \;;\; k,n_0,\dots,n_k \in \mathbb{N}\},\\
\tau(\mathbf{\Theta}) &\triangleq \{\lambda f.\langle \lambda x.\langle f\rangle[\langle x\rangle[x^{n_1}],\dots,\langle x\rangle[x^{n_k}]]\rangle[\lambda x.\langle f\rangle[\langle x\rangle[x^{n_{1,1}}],\dots,\langle x\rangle[x^{n_{1,k_1}}]],\dots,\\
&\qquad\qquad \lambda x.\langle f\rangle[\langle x\rangle[x^{n_{h,1}}],\dots,\langle x\rangle[x^{n_{h,k_h}}]]] \;;\; k,n_i,h,n_{i,j} \in \mathbb{N}\}
\end{aligned}$$

The Taylor expansion of a β-redex contains resource redexes. If one looks for an invariant under β-reduction, one should consider the normal forms of the Taylor expansions. By induction on the size of the simple terms, one can for example prove that $\mathrm{NF}(\tau(\mathbf{\Omega})) = \emptyset$, while

$$\mathrm{NF}(\tau(\mathbf{\Theta})) = \{\lambda f.\langle f\rangle 1,\ \lambda f.\langle f\rangle[(\langle f\rangle 1)^n],\ \lambda f.\langle f\rangle[\langle f\rangle[(\langle f\rangle 1)^{n_1}],\dots,\langle f\rangle[(\langle f\rangle 1)^{n_k}]],\ \dots\}.$$

Indeed, notice that these examples can be seen as a *thick* version [2] of the finite approximants of the Böhm tree of the corresponding λ-terms, where each sub-tree has been recursively replaced by a finite multiset of copies of it. We recall the following Theorem 8, stating a commutation between the resource reduction and the Taylor expansion through the Böhm tree operator. To do this, we need to extend the notion of Taylor expansion to Böhm-like trees:

$$\tau(\mathcal{B}) \triangleq \begin{cases} \emptyset & \text{if } \mathrm{height}(\mathcal{B}) = 0,\\ \{\lambda x_0\dots x_{m-1}.\langle y\rangle \mu_0\dots\mu_{n-1} \;;\; \mu_i \in \mathcal{M}_F(\tau(\mathcal{B}_i))\} & \text{if } \mathrm{height}(\mathcal{B}) = h+1 \text{ and}\\ & \mathcal{B}(\langle\ \rangle) = (\lambda x_0\dots x_{m-1}.y, n),\\ \bigcup_h \tau(\mathcal{B}|_h) & \text{if } \mathrm{height}(\mathcal{B}) = \infty. \end{cases} \tag{9}$$

▶ **Theorem 8** ([8, Theorem 2]). *For every λ-term M, $\mathrm{NF}(\tau(M)) = \tau(\mathrm{BT}(M))$.*

4 Characterizing the Taylor expansion

In this section, we prove our main result (Theorem 25), an algebraic characterization of the normal forms of the Taylor expansion of Λ. Definitions 9 and 10 give the two crucial notions (a notion of ideal and a notion of effective element) for such a characterization.

Let Δ^{NF} be the set of the simple terms which are normal forms with respect to $\xrightarrow{r}$. They have the following shape:

$$\Delta^{\mathrm{NF}} : t \ ::= \lambda x_0\dots x_{m-1}.\langle y\rangle \mu_0\dots\mu_{n-1} \tag{10}$$

where $m, n \geq 0$ and each μ_i is a bag of simple terms in normal form.

▶ **Definition 9** (Definedness). The *definedness relation* $\preceq$ on Δ^{NF} is given as follows:

$$\lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0 \dots \mu_{n-1} \preceq t \text{ iff } \begin{cases} t = \lambda x_0 \dots x_{m-1}.\langle y\rangle \nu_0 \dots \nu_{n-1} \text{ and} \\ \forall i < n, |\mu_i| \neq \emptyset \implies \exists v \in |\nu_i|, \forall u \in |\mu_i|, u \preceq v. \end{cases}$$

Given $\mathcal{T} \in \mathcal{P}(\Delta^{\mathrm{NF}})$ we say that:

- $\mathcal{T}$ is *downward closed* whenever $\forall t$ s.t. $\exists t' \in \mathcal{T}, t \preceq t'$, we have $t \in \mathcal{T}$;
- $\mathcal{T}$ is *directed*, whenever $\forall t, t' \in \mathcal{T}$, we have $\exists t'' \in \mathcal{T}$, s.t. $t, t' \preceq t''$;
- $\mathcal{T}$ is an *ideal* whenever it is downward closed and directed.

Notice that the relation $\preceq$ is transitive but not reflexive (i.e. it is not a preorder). For instance, $\langle y\rangle[\langle y\rangle 1[y], \langle y\rangle[y]1] \not\preceq \langle y\rangle[\langle y\rangle 1[y], \langle y\rangle[y]1]$, as $\langle y\rangle 1[y] \not\preceq \langle y\rangle[y]1$ as well as $\langle y\rangle[y]1 \not\preceq \langle y\rangle 1[y]$. It is not antireflexive either, for example $\langle y\rangle[x] \preceq \langle y\rangle[x]$. Moreover, it is neither symmetric nor antisymmetric. We have $\langle y\rangle[x]1 \preceq \langle y\rangle[x][x]$, but $\langle y\rangle[x][x] \not\preceq \langle y\rangle[x]1$, and $\langle y\rangle[t] \preceq \langle y\rangle[t,t]$, $\langle y\rangle[t,t] \preceq \langle y\rangle[t]$ but $\langle y\rangle[t] \neq \langle y\rangle[t,t]$. However, on the simple terms having bags of at most one element, $\preceq$ is an order relation.

We need to formalize what is an "effective element" of $\mathcal{P}(\Delta^{\mathrm{NF}})$. In order to do that, we use the notion of a recursively enumerable subset of $\mathbb{N}$ and an encoding of Δ^{NF} into $\mathbb{N}$. Let G be any effective bijection between $\Delta^{\mathrm{NF}}/\equiv_\alpha$ and $\mathbb{N}$. One way of defining G is by using the De Bruijn notation [4], which gives a system of canonical representatives for the α-equivalence. We will omit such details, so we fix once and for all a recursive Gödel numbering $\mathrm{G} : \Delta^{\mathrm{NF}} \mapsto \mathbb{N}$.

▶ **Definition 10** (Effectiveness). An element $\mathcal{T}$ of $\mathcal{P}(\Delta^{\mathrm{NF}}])$ is *recursively enumerable*, *r.e.* for short, whenever the set $\mathrm{G}(\mathcal{T}) \stackrel{\triangle}{=} \{\mathrm{G}(t)\ ;\ t \in \mathcal{T}\} \subseteq \mathbb{N}$ is recursively enumerable, i.e. either $\mathcal{T} = \emptyset$, or there exists a total recursive function $\phi : \mathbb{N} \mapsto \mathbb{N}$ such that $\mathrm{G}(\mathcal{T}) = \{\phi(n)\ ;\ n \in \mathbb{N}\}$.

Notice that the notion of a r.e. element of $\mathcal{P}(\Delta^{\mathrm{NF}})$ does not depend on the chosen Gödel enumeration of Δ^{NF}.

▶ **Definition 11.** We say that a set $\mathcal{T} \in \mathcal{P}(\Delta^{\mathrm{NF}})$ is *single-headed with width* n, whenever there exist $m \in \mathbb{N}$, $x_0, \dots, x_{m-1}, y \in \mathrm{Var}$, $J \subseteq \mathbb{N}$ and, for each $i < n$, a family $(\mu_i^j)_{j\in J}$ of bags such that :

$$\mathcal{T} = \{\lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0^j \dots \mu_{n-1}^j \mid j \in J\}. \tag{11}$$

For each i, we denote by $\widehat{\mathcal{T}_i}$ the set $\{\mu_i^j \mid j \in J\}$ and simply by $\mathcal{T}_i$ the set $\bigcup_{j\in J} |\mu_i^j|$. Notice that $\widehat{\mathcal{T}_i} \subseteq \mathcal{M}_F(\mathcal{T}_i)$.

▶ **Lemma 12.** *If $\mathcal{T}$ is directed then $\mathcal{T}$ is single-headed and $\mathcal{T}_i$ is directed (for every i less than the width of $\mathcal{T}$). Let $\mathcal{T} \in \mathcal{P}(\Delta^{\mathrm{NF}})$ be non-empty. If $\mathcal{T}$ is single-headed with width $n \in \mathbb{N}$ and downward closed then $\mathcal{T}_i$ is downward closed (for every $i < n$).*

Proof. It is straightforward to check that $\mathcal{T}$ directed implies $\mathcal{T}$ single-headed.

From now on assume $\mathcal{T}$ is single-headed of width n and let us prove that $\mathcal{T}_i$ is downward closed (resp. directed) whenever $\mathcal{T}$ is downward closed (resp. directed).

Let $u' \preceq u \in \mathcal{T}_i$. This means that there is $t = \lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0 \dots \mu_{n-1} \in \mathcal{T}$, such that $u \in |\mu_i|$. Define $t' \stackrel{\triangle}{=} \lambda x_0 \dots x_{m-1}.\langle y\rangle 1 \dots 1[u']1 \dots 1$ and notice that $t' \preceq t$. By the downward closedness of $\mathcal{T}$ we conclude $t' \in \mathcal{T}$, so $u' \in \mathcal{T}_i$ and hence $\mathcal{T}_i$ is downward closed.

Let $u_1, u_2 \in \mathcal{T}_i$. We have in $\mathcal{T}$ two elements $t_1 = \lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0^1 \dots \mu_{n-1}^1$ and $t_2 = \lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0^2 \dots \mu_{n-1}^2$, such that $u_1 \in |\mu_i^1|$ and $u_2 \in |\mu_i^2|$. By directedness of $\mathcal{T}$,

there is an element $t_3 = \lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0^3 \dots \mu_{n-1}^3 \in \mathcal{T}$ such that $t_1, t_2 \preceq t_3$. So, there is $v_1, v_2 \in |\mu_i^3|$, such that $u_1 \preceq v_1$ and $u_2 \preceq v_2$. Again by directedness of $\mathcal{T}$, there exists $t_4 = \lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0^4 \dots \mu_{n-1}^4 \in \mathcal{T}$ such that $t_3 \preceq t_4$, so there exists $w \in |\mu_i^4|$, such that $v_1, v_2 \preceq w$. We have $w \in \mathcal{T}_i$ and by transitivity, $u_1, u_2 \preceq w$. Hence $\mathcal{T}_i$ is directed. ◀

We now define a coherence relation on normal simple terms, originally introduced in [9], which will mainly be used to approximate the notion of ideal. Indeed, we will show that every ideal is a clique (Proposition 16) and every clique is included in an ideal (Proposition 20).

▶ **Definition 13** (Coherence, [9, §3]). The coherence relation $\frown$ on Δ^{NF} is defined by:

$$\lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0 \dots \mu_{n-1} \frown t \text{ iff } \begin{cases} t = \lambda x_0 \dots x_{m-1}.\langle y\rangle \nu_0 \dots \nu_{n-1} \text{ and} \\ \forall i < n, \forall u, u' \in |\mu_i \cdot \nu_i|, u \frown u'. \end{cases}$$

We call $\mathcal{T} \in \mathcal{P}(\Delta^{\mathrm{NF}})$ a *clique* whenever $\forall t, t' \in \mathcal{T}, t \frown t'$.

Notice that $\frown$ is symmetric, but not reflexive. For example, $\langle y\rangle[x, z] \not\frown \langle y\rangle[x, z]$. Let us stress, however, that the previous example $\langle y\rangle[\langle y\rangle 1[y], \langle y\rangle[y]1]$ showing the non-reflexivity of $\preceq$, is indeed coherent with itself. Furthermore, $\frown$ is not transitive. We have $\langle y\rangle[x][z] \frown \langle y\rangle[x]1$ and $\langle y\rangle[x][y] \frown \langle y\rangle[x]1$ but $\langle y\rangle[x][z] \not\frown \langle y\rangle[x][y]$.

As a direct consequence of the definition of coherence we get the following lemma.

▶ **Lemma 14** (Hereditary characterization of cliques). *Let $\mathcal{C} \in \mathcal{P}(\Delta^{\mathrm{NF}})$, $\mathcal{C}$ is a clique iff $\mathcal{C}$ is single-headed of a certain width n and for each $i < n$, $\mathcal{C}_i$ is a clique.*

Hereafter, we will often use the induction of the following notion of height, analogous to the definition of height of a Böhm-like tree.

▶ **Definition 15** (Height). Given a simple normal form $t = \lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0 \dots \mu_{n-1}$, we define $\mathrm{height}(t) = 1 + \max_i(\max_{t_i \in |\mu_i|} \mathrm{height}(t_i))$. The height of a set $\mathcal{T} \in \mathcal{P}(\Delta^{\mathrm{NF}})$ is $\mathrm{height}(\mathcal{T}) = \sup_{t \in \mathcal{T}}(\mathrm{height}(t))$. We define $\mathcal{T}|_h = \{t \in \mathcal{T} \mid \mathrm{height}(t) < h\}$.

Notice that $\mathrm{height}(t) = 1$ if and only if t has only empty bags. Of course, the height of a set of simple normal forms $\mathcal{T}$ can be infinite. For example, $\mathrm{height}(\mathrm{NF}(\tau(\mathbf{\Theta}))) = \infty$, while $\mathrm{height}(\mathrm{NF}(\tau(\mathbf{S}))) = 3$.

▶ **Proposition 16** (Ideals are cliques). *If $\mathcal{D} \in \mathcal{P}(\Delta^{\mathrm{NF}})$ is an ideal then $\mathcal{D}$ is a clique.*

Proof. It is enough to check that, for every two terms $t_1, t_2 \in \Delta^{\mathrm{NF}}$, if there exists an ideal $\mathcal{D}'$ such that $t_1, t_2 \in \mathcal{D}'$ then $t_1 \frown t_2$. Let us prove this claim by induction on the maximal height among t_1 and t_2. Since $\mathcal{D}'$ is an ideal there exists $t_3 \in \mathcal{D}'$ such that $t_1, t_2 \preceq t_3$ and, by Lemma 12, $\mathcal{D}'$ is single-headed of a certain width n, so there exist bags μ_i^j such that

$$t_j = \lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0^j \dots \mu_{n-1}^j (\text{for each } j = 1, 2, 3),$$

moreover each $\mathcal{D}'_i$ is directed.

If $\max(\mathrm{height}(t_1), \mathrm{height}(t_2)) = 1$ then $t_1 = t_2 = \lambda x_0 \dots x_{m-1}.\langle y\rangle 1 \dots 1$ and this term is coherent with itself. Now let $h = \max(\mathrm{height}(t_1), \mathrm{height}(t_2))$. In order to prove $t_1 \frown t_2$ we have to prove that for each $i < n$, for every $u_1, u_2 \in |\mu_i^1 \cdot \mu_i^2|$, we have $u_1 \frown u_2$. But $\max(\mathrm{height}(u_1), \mathrm{height}(u_2)) \leq h - 1$ and u_1, u_2 belong to $\mathcal{D}'_i$ which is directed. We conclude by induction hypothesis. ◀

In fact, every subset of an ideal is a clique (just check that the proof above does not use the downward closure).

▶ **Definition 17** (Linearization)**.** Let $\mathcal{C}$ be a non-empty clique of finite height. The *linearization* of $\mathcal{C}$, defined by induction on height($\mathcal{C}$), is the simple term $\mathcal{L}(\mathcal{C}) \stackrel{\triangle}{=} \lambda x_0 \dots x_{m-1}.\langle y\rangle \xi_0 \dots \xi_{n-1}$ where, for every $i < n$, $\xi_i = [\mathcal{L}(\mathcal{C}_i)]$ if $\mathcal{C}_i$ non-empty, otherwise $\xi_i = 1$.

▶ **Lemma 18.** *Let $\mathcal{C} \subseteq \mathcal{D}$ be two non-empty cliques of finite height. We have:* height($\mathcal{L}(\mathcal{C})$) = height($\mathcal{C}$)*, $\mathcal{L}(\mathcal{C}) \preceq \mathcal{L}(\mathcal{D})$ and, finally, $\forall t \in \mathcal{C}$, $t \preceq \mathcal{L}(\mathcal{C})$.*

Proof. Let $\mathcal{C}$ and $\mathcal{D}$ satisfying the hypothesis. By Lemma 14, $\mathcal{C}$ and $\mathcal{D}$ are single headed with same width n. For any $i < n$, notice that $\mathcal{C}_i \subseteq \mathcal{D}_i$. We proceed by induction on height($\mathcal{C}$). If height($\mathcal{C}$) = 1, then $\mathcal{C} = \{\mathcal{L}(\mathcal{C}) = \lambda x_0 \dots x_{m-1}.\langle y\rangle 1 \dots 1\}$. Clearly, one gets the statement. Otherwise, let height($\mathcal{C}$) > 1. Then $\mathcal{L}(\mathcal{C})$ is defined as in the equation of Definition 17. By Lemma 14, $\mathcal{D}_i$ and $\mathcal{C}_i$ are cliques. Hence, whenever $\mathcal{C}_i$ is non-empty, we have by induction hypothesis that $\mathcal{L}(\mathcal{C}_i)$ satisfies the statement of the lemma for $\mathcal{C}_i$ and $\mathcal{D}_i$. One can then easily conclude that height($\mathcal{L}(\mathcal{C})$) = height($\mathcal{C}$), and $\mathcal{L}(\mathcal{C}) \preceq \mathcal{L}(\mathcal{D})$ and, finally, $\forall t \in \mathcal{C}$, $t \preceq \mathcal{L}(\mathcal{C})$. ◀

▶ **Lemma 19.** *Let $\mathcal{D}$ be an ideal and let $\mathcal{C} \subseteq \mathcal{D}$ be a non-empty clique of finite height. Then, $\mathcal{L}(\mathcal{C}) \in \mathcal{D}$.*

Proof. Let $\mathcal{D}$ and $\mathcal{C}$ satisfying the hypothesis. By Lemma 12, $\mathcal{D}$ (and hence $\mathcal{C}$) are single headed and of same width n. The proof of $\mathcal{L}(\mathcal{C}) \in \mathcal{D}$ is by induction on height($\mathcal{C}$).

Let I be the set of indices $i < n$ such that $\mathcal{C}_i$ is non-empty. If I is empty, then $\mathcal{L}(\mathcal{C}) = \lambda x_0 \dots x_{m-1}.\langle y\rangle 1 \dots 1$, and one can deduce that $\mathcal{L}(\mathcal{C}) \in \mathcal{D}$ since $\mathcal{D}$ is downward closed and $\mathcal{L}(\mathcal{C})$ is $\preceq$ to any element of $\mathcal{D}$. If otherwise I is non-empty, then for every $i \in I$, we have that $\mathcal{C}_i \subseteq \mathcal{D}_i$, as well as that $\mathcal{C}_i$ is a clique (Lemma 14). By induction hypothesis we can suppose $\mathcal{L}(\mathcal{C}_i) \in \mathcal{D}_i$. This means that there is $w_i \in \mathcal{D}, w_i = \lambda x_0 \dots x_{m-1}.\langle y\rangle \rho_0 \dots \rho_{n-1}$, and $\mathcal{L}(\mathcal{C}_i) \in |\rho_i|$. As $\mathcal{D}$ is directed and $\preceq$ is transitive, we can construct $w \in \mathcal{D}$ such that $w_i \preceq w$, for every $i \in I$. We remark that $\mathcal{L}(\mathcal{C}) \preceq w \in \mathcal{D}$, therefore $\mathcal{L}(\mathcal{C}) \in \mathcal{D}$. ◀

▶ **Proposition 20** (Cliques can be ideals)**.** *Let $\mathcal{C}$ be a clique. Then:*

1. *the set $\mathcal{I}(\mathcal{C}) \stackrel{\triangle}{=} \{t \mid \exists h > 0, t \preceq \mathcal{L}(\mathcal{C}|_h)\}$ is an ideal;*
2. *This set $\mathcal{I}(\mathcal{C})$ is the smallest ideal containing $\mathcal{C}$.*

Proof. Obviously $\mathcal{I}(\mathcal{C})$ is downward closed. Moreover, by Lemma 18, $\{\mathcal{L}(\mathcal{C}|_h)\}_{h\in\mathbb{N}}$ is a chain (in fact $\mathcal{C}|_h \subseteq \mathcal{C}|_{h+1}$), hence directed. We conclude that $\mathcal{I}(\mathcal{C})$ is directed because the downward closure of a directed set is directed ($\preceq$ being a transitive relation).

As for item 2, first, we prove that $\mathcal{C} \subseteq \mathcal{I}(\mathcal{C})$. By Lemma 18, $\mathcal{C}|_h \subseteq \mathcal{I}(\mathcal{C})$, for every h. We conclude because $\mathcal{C} = \bigcup_h \mathcal{C}|_h$. Second, let $\mathcal{D}$ be an ideal containing $\mathcal{C}$, it is enough to prove $\mathcal{L}(\mathcal{C}|_h) \in \mathcal{D}$ for every h for concluding $\mathcal{C} \subseteq \mathcal{D}$. This is a consequence of Lemma 19. ◀

▶ **Corollary 21.** *Let $\mathcal{C}$ be a maximal clique. Then, $\mathcal{C}$ is an ideal.*

Proof. Let $\mathcal{C}$ be a maximal clique. By Proposition 20, $\mathcal{I}(\mathcal{C})$ is an ideal and $\mathcal{C} \subseteq \mathcal{I}(\mathcal{C})$. And by Proposition 16, $\mathcal{I}(\mathcal{C})$ is a clique. So we conclude by maximality of $\mathcal{C}$, that $\mathcal{C}$ is equal to the ideal $\mathcal{I}(\mathcal{C})$. ◀

Recall from Section 2 that we consider Böhm-like trees ordered by the set-theoretical inclusion on the graph of their functions. Indeed, such an order is reflected in their Taylor expansion.

▶ **Lemma 22.** *If $\mathcal{B}, \mathcal{B}'$ are Böhm-like trees, $\tau(\mathcal{B}) \subseteq \tau(\mathcal{B}')$ iff $\mathcal{B} \subseteq \mathcal{B}'$. Moreover,* height($\mathcal{B}$) = height($\tau(\mathcal{B})$)*.*

Proof. $\Longrightarrow$ We suppose that $\tau(\mathcal{B}) \subseteq \tau(\mathcal{B}')$. By induction on length$(\alpha)$, for any $\alpha \in \mathbb{N}^*$, we show that $\mathcal{B}(\alpha) = \mathcal{B}'(\alpha)$, whenever $\mathcal{B}(\alpha)$ is defined.

If $\mathcal{B}(\langle\,\rangle) = (\lambda x_0 \dots x_{m-1}.y, n)$, then $\lambda x_0 \dots x_{m-1}.\langle y\rangle 1 \dots 1 \in \tau(\mathcal{B}) \subseteq \tau(\mathcal{B}')$, therefore by definition $\mathcal{B}'(\langle\,\rangle) = (\lambda x_0 \dots x_{m-1}.y, n)$. The induction case follows because $\tau(\mathcal{B}) \subseteq \tau(\mathcal{B}')$ implies $\tau(\mathcal{B}_i) \subseteq \tau(\mathcal{B}'_i)$ for every $i < n$. Then, $\mathcal{B}(\langle i\rangle @ \alpha) = \mathcal{B}_i(\alpha)$ which, by induction hypothesis, is equal to $\mathcal{B}'_i(\alpha) = \mathcal{B}'(\langle i\rangle @ \alpha)$.

$\Longleftarrow$ We suppose that $\mathcal{B} \subseteq \mathcal{B}'$. Let $t \in \tau(\mathcal{B})$, by induction on height(t), one shows that $t \in \tau(\mathcal{B}')$. The reasoning is similar to the left-to-right implication.

The last statement regarding the height is trivial. ◀

▶ **Lemma 23.** *Let $\mathcal{T} \in \mathcal{P}(\Delta^{\mathrm{NF}})$. There is a Böhm-like tree $\mathcal{B}$ such that $\tau(\mathcal{B}) = \mathcal{T}$ iff $\mathcal{T}$ is an ideal.*

Proof. $\Longrightarrow$ The proof depends whether height$(\mathcal{B})$ is finite or infinite.
If height$(\mathcal{B})$ is finite, we proceed by induction on height$(\mathcal{B})$. If $\mathcal{B}(\langle\,\rangle) = (\lambda x_0 \dots x_{m-1}.y, n)$, then $\tau(\mathcal{B}) = \{\lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0 \dots \mu_{n-1} \mid i < n, \mu_i \in \mathcal{M}_F(\tau(\mathcal{B}_i))\}$.

Downward closure. Let $t \preceq t' \in \tau(\mathcal{B})$. Since $t' \in \tau(\mathcal{B})$ we have t' of the shape

$$t' = \lambda x_0 \dots x_{m-1}.\langle y\rangle \mu'_0 \dots \mu'_{n-1}. \tag{12}$$

Then, since $t \preceq t'$, t is of the form

$$t = \lambda x_0 \dots x_{m-1}.\langle y\rangle \mu_0 \dots \mu_{n-1}, \tag{13}$$

such that, for every $i < n$, either $\mu_i = 1 \in \mathcal{M}_F(\tau(\mathcal{B}_i))$, or there exists $u' \in \mu'_i$ such that for any $u \in \mu_i$, $u \preceq u'$. Notice that $u' \in \tau(\mathcal{B}_i)$, so by induction hypothesis $u \in \tau(\mathcal{B}_i)$. Since this is the case for every $u \in \mu_i$, we get $\mu_i \in \mathcal{M}_F(\tau(\mathcal{B}_i))$. We conclude $t \in \tau(\mathcal{B})$.

Directedness. Let $t, t' \in \tau(\mathcal{B})$, and let us find t'' such that $t, t' \preceq t''$. The terms t and t' are as in Equation (13) and (12), respectively.
For $i < n$, $\mu_i \cdot \mu'_i \in \mathcal{M}_F(\tau(\mathcal{B}_i))$. By induction hypothesis, $\tau(\mathcal{B}_i)$ is directed. Since $\mu_i \cdot \mu'_i$ is finite (and $\preceq$ is transitive), if $\mu_i \cdot \mu'_i$ is non-empty, then there exists $t_i \in \tau(\mathcal{B}_i)$, such that for any $u \in \mu_i \cdot \mu'_i$, $u \preceq t_i$. Then, define $\xi_i = [t_i]$ if $\mu_i \cdot \mu'_i$ non-empty, otherwise $\xi_i = 1$. We set $t'' = \lambda x_0 \dots x_{m-1}.\langle y\rangle \xi_0 \dots \xi_{n-1}$. Notice that $t'' \in \tau(\mathcal{B})$, since $\xi_i \in \mathcal{M}_F(\tau(\mathcal{B}_i))$, and $t, t' \preceq t''$.

Now let us consider the case height$(\mathcal{B}) = \infty$. By definition $\tau(\mathcal{B}) = \bigcup_{h\in\mathbb{N}} \tau(\mathcal{B}|_h)$. Notice that $\tau(\mathcal{B}|_h)$ is a Böhm-like tree of finite height, so, by the previous reasoning, we can conclude that $\tau(\mathcal{B}|_h)$ is an ideal. Since $\{\mathcal{B}|_h\}_{h\in\mathbb{N}}$ is a chain, we can easily conclude that the whole $\tau(\mathcal{B})$ is an ideal.

$\Longleftarrow$ In this case, we also split in two subcases, depending whether height$(\mathcal{T})$ is finite or infinite. In the first case our induction is on height$(\mathcal{T})$.

If height$(\mathcal{T}) = 0$, then $\mathcal{T} = \emptyset$ and we choose the Böhm-like tree undefined everywhere. Otherwise, $\mathcal{T}$ is non-empty and we can write it as in Equation (11), by Lemma 12. Moreover, for any $i < n$, $\mathcal{T}_i$ is an ideal. By induction hypothesis, we can assume that there exists a Böhm-like tree $\mathcal{B}^i$ such that $\tau(\mathcal{B}^i) = \mathcal{T}_i$ (notice that $\mathcal{B}^i$ is the everywhere undefined function whenever $\mathcal{T}_i = \emptyset$). Define then $\mathcal{B}(\langle\rangle) \stackrel{\triangle}{=} (\lambda x_0 \dots x_{m-1}.y, n)$ and $\mathcal{B}(\langle i\rangle @ \alpha) = \mathcal{B}^i(\alpha)$. Note that in particular $\mathcal{B}_i = \mathcal{B}^i$.

In order to prove $\tau(\mathcal{B}) = \mathcal{T}$, we must show that $\widehat{\mathcal{T}_i} = \mathcal{M}_F(\tau(\mathcal{B}_i))$, for every $i < n$. By Lemma 12, we only know that $\widehat{\mathcal{T}_i} \subseteq \mathcal{M}_F(\mathcal{T}_i) = \mathcal{M}_F(\tau(\mathcal{B}^i)) = \mathcal{M}_F(\tau(\mathcal{B}_i))$. Let $\mu_i \in \mathcal{M}_F(\mathcal{T}_i)$, we prove that $\mu_i \in \widehat{\mathcal{T}_i}$. Actually, we prove that the term $t_{\mu_i} \stackrel{\triangle}{=} \lambda x_0 \dots x_{m-1}.\langle y\rangle 1 \dots 1 \mu_i 1 \dots 1 \in \mathcal{T}$, which is enough to conclude.

If $\mu_i = 1$ then the bags in t_{μ_i} are all empty and so one concludes by the downward closure of $\mathcal{T}$, remarking that t_{μ_i} is $\preceq$ to any element in $\mathcal{T}$. Otherwise, we have that $|\mu_i|$ is a non-empty sub-set of $\mathcal{T}_i$ of finite height (since it has finite cardinality). Moreover, since $\mathcal{T}_i$ is an ideal, $\mathcal{T}_i$, and hence $|\mu_i|$, are cliques (Proposition 16). We first apply Lemma 19 to $\mathcal{C} = |\mu_i| \subseteq \mathcal{T}_i$. We then obtain $\mathcal{L}(|\mu_i|) \in \mathcal{T}_i$, so there exists $t \in \mathcal{T}$ whose i-th bag contains $\mathcal{L}(|\mu_i|)$. We further apply Lemma 18, arguing that $\forall u \in |\mu_i|$, $u \preceq \mathcal{L}(|\mu_i|)$. This induces that $t_{\mu_i} \preceq t$, so finally, $t_{\mu_i} \in \mathcal{T}$.

We now suppose that $\text{height}(\mathcal{T}) = \infty$. Recall Definition 15, and notice that $\mathcal{T}|_h \subseteq \mathcal{T}$ is downward closed because $t \preceq t'$ implies $\text{height}(t) \leq \text{height}(t')$. We now show that $\mathcal{T}|_h$ is directed. In fact, take any two $t_1, t_2 \in \mathcal{T}|_h$. Set $\mathcal{C} = \{t_1, t_2\} \subseteq \mathcal{T}$. By Proposition 16, $\mathcal{T}$ and $\mathcal{C}$ are cliques, so by Lemma 18 and 19, $\mathcal{L}(\{t_1, t_2\}) \in \mathcal{T}$, $t_1, t_2 \preceq \mathcal{L}(\{t_1, t_2\})$ and $\text{height}(\mathcal{L}(\{t_1, t_2\})) \leq h$. We conclude $\mathcal{L}(\{t_1, t_2\}) \in \mathcal{T}|_h$.

We can then apply the reasoning for sets of finite heights and conclude that there exists $\mathcal{B}^h$ such that $\tau(\mathcal{B}^h) = \mathcal{T}|_h$. As $\{\mathcal{T}|_h\}_{h\in\mathbb{N}}$ is by definition a chain, we conclude with Lemma 22 that $\{\mathcal{B}^h\}_{h\in\mathbb{N}}$ is a chain. Let $\mathcal{B} = \bigcup_{h\in\mathbb{N}} \mathcal{B}^h$, we have that $\mathcal{B}$ is a Böhm-like tree. Remark that $\mathcal{B}|_h = \mathcal{B}^h$, as for any $h \in \mathbb{N}$, the difference between $\mathcal{B}^h$ and $\mathcal{B}^{h+1}$ lies only on the sequences of length h, which are undefined for $\mathcal{B}^h$. Finally $\tau(\mathcal{B}) \overset{\triangle}{=} \bigcup_{h\in\mathbb{N}} \tau(\mathcal{B}|_h) = \bigcup_{h\in\mathbb{N}} \tau(\mathcal{B}^h) = \bigcup_{h\in\mathbb{N}} \mathcal{T}|_h = \mathcal{T}$. ◀

▶ **Lemma 24.** *Let $\mathcal{B}$ be a Böhm-like tree, then*

- $\text{FV}(\mathcal{B}) = \text{FV}(\tau(\mathcal{B}))$,
- *$\mathcal{B}$ is r.e. iff $\tau(\mathcal{B})$ is r.e.*

Proof. The only difficulty is to prove that $\tau(\mathcal{B})$ r.e. implies $\mathcal{B}$ r.e. Observe that an element t of $\tau(\mathcal{B})$ defines a kind of subtree of $\mathcal{B}$ (in fact, a rooted *thick* subtree [2]) which can be used to define a partial function $f_t : \mathbb{N}^* \to \Sigma \times \mathbb{N}$ such that, whenever $f_t(\alpha)$ is defined, $f_t(\alpha)$ equals $\mathcal{B}(\alpha)$. Moreover for every $\alpha \in \mathbb{N}^*$, such that $\mathcal{B}(\alpha)$ is defined, there exists $t \in \tau(\mathcal{B})$ such that $f_t(\alpha)$ is defined. Hence, given an effective enumeration of $\tau(\mathcal{B})$, one can compute $\mathcal{B}(\alpha)$ by iterating over $\tau(\mathcal{B})$ with t until $f_t(\alpha)$ is defined (if $\mathcal{B}(\alpha)$ is undefined this will never end). ◀

▶ **Theorem 25.** *Let $\mathcal{T} \in \mathcal{P}(\Delta^{\text{NF}})$. There is a λ-term M such that $\text{NF}(\tau(M)) = \mathcal{T}$ iff the following conditions hold:*

1. *$\text{FV}(\mathcal{T})$ is finite,*
2. *$\mathcal{T}$ is r.e.,*
3. *$\mathcal{T}$ is an ideal wrt $\preceq$.*

Proof. By Theorem 8, the equality $\text{NF}(\tau(M)) = \mathcal{T}$ can be replaced by $\tau(\text{BT}(M)) = \mathcal{T}$. Then, by Theorem 5 the condition at the left-hand side of the iff can be replaced by "there is a Böhm-like tree $\mathcal{B}$ such that (i) $\text{FV}(\mathcal{B})$ is finite, (ii) $\mathcal{B}$ is r.e., (iii) $\tau(\mathcal{B}) = \mathcal{T}$". The equivalence is then achieved by Lemma 23 and 24. ◀

In [9], Ehrhard and Regnier noticed that the Taylor expansion of normal forms are maximal cliques with respect to the set-theoretical inclusion. However, not every maximal clique $\mathcal{T}$ represents a normalizable λ-term, even if $\mathcal{T}$ enjoys the conditions of Theorem 25. For example, $\text{NF}(\tau(\mathbf{\Theta}))$ is a maximal clique. We define the total λ-terms (called $\perp$-free in [1, Definition 10.1.12]) and prove that they correspond to the property of being a maximal clique in $\mathcal{P}(\Delta^{\text{NF}})$.

▶ **Definition 26** (Totality). A Böhm-like tree $\mathcal{B}$ is *total* whenever $\mathcal{B}(\langle\rangle)\downarrow$ and $\mathcal{B}(\alpha) = (a,n)$ implies that for all $i < n$, $\mathcal{B}(\alpha@\langle i\rangle)\downarrow$. A λ-term M is *total* whenever $M \overset{h*}{\to} \lambda x_0 \dots x_{m-1}.(y)M_0 \dots M_{n-1}$ and $M_0, \dots M_{n-1}$ are total, i.e. $\mathrm{BT}(M)$ is total.

▶ **Lemma 27.** *Let $\mathcal{B}$ be a Böhm-like tree. The set $\tau(\mathcal{B})$ is a maximal clique (with respect to the set-theoretical inclusion) iff $\mathcal{B}$ is total.*

Proof. One proves the equivalent statement: there exists $t \in \Delta^{\mathrm{NF}}$, $t \notin \tau(\mathcal{B})$, coherent with any element of $\tau(\mathcal{B})$, iff $\mathcal{B}$ is not total. The left-to-right direction is by induction on $\mathrm{height}(t)$, noticing that if a subtree $\mathcal{B}_i$ is not total, then $\mathcal{B}$ is not total. The converse direction is by induction on $\mathrm{length}(\alpha)$ for a sequence α such that $\mathcal{B}(\alpha) = (a,n)$ but there is $i < n$, $\mathcal{B}(\alpha@\langle i\rangle)\uparrow$. ◀

▶ **Corollary 28.** *Let $\mathcal{T} \in \mathcal{P}(\Delta^{\mathrm{NF}})$. There is a total λ-term M such that $\mathrm{NF}(\tau(M)) = \mathcal{T}$ iff*

1. $\mathrm{FV}(\mathcal{T})$ *is finite,*
2. $\mathcal{T}$ *is r.e.,*
3. $\mathcal{T}$ *is a maximal clique.*

Proof. $\Longrightarrow$ By Theorem 25, we get **1** and **2**. Condition **3** is a consequence of Theorem 8 and Lemma 27.

$\Longleftarrow$ By Lemma 21 and Theorem 25 we get a λ-term M such that $\mathrm{NF}(\tau(M)) = \mathcal{T}$. By Theorem 8 and Lemma 27 we conclude that M is total. ◀

Of course, one would like to characterize the property of having a β-normal form.

▶ **Corollary 29.** *Let $\mathcal{T} \in \mathcal{P}(\Delta^{\mathrm{NF}})$. There is a normalizable λ-term M such that $\mathrm{NF}(\tau(M)) = \mathcal{T}$ iff*

1. $\mathrm{height}(\mathcal{T})$ *is finite,*
2. $\mathcal{T}$ *is a maximal clique.*

Proof. $\Longrightarrow$ Remark that if M is normalizable, then $\mathrm{BT}(M)$ is total and of finite height. In fact, $\mathrm{BT}(M)$ is the applicative tree of the normal form of M. By Corollary 28, $\mathrm{NF}(\tau(M))$ is a maximal clique. By Theorem 8 and Lemma 22 we get that $\mathrm{height}(\mathrm{NF}(\tau(M)))$ is finite.

$\Longleftarrow$ Consider the simple term $\mathcal{L}(\mathcal{T})$. One can easily prove that $\mathcal{L}(\mathcal{T})$ does not contain empty bags, otherwise $\mathcal{T}$ would not be maximal. Moreover, by induction of $\mathrm{height}(\mathcal{L}(\mathcal{T}))$, one also proves that the λ-term M obtained from $\mathcal{L}(\mathcal{T})$ by replacing any resource application $\langle t\rangle[u]$ with a λ-calculus application $(t)u$ is such that $\mathcal{L}(\mathcal{T}) \in \tau(M)$. Let us prove that $\mathcal{T} = \tau(M)$. By Proposition 20, $\mathcal{I}(\{\mathcal{L}(\mathcal{T})\})$ is the smallest ideal containing $\mathcal{L}(\mathcal{T})$, so $\mathcal{I}(\{\mathcal{L}(\mathcal{T})\}) \subseteq \tau(M)$. By Lemma 18, for every $t \in \mathcal{T}$, $t \preceq \mathcal{L}(\mathcal{T})$, hence $\mathcal{T} \subseteq \mathcal{I}(\{\mathcal{L}(\mathcal{T})\}) \subseteq \tau(M)$. By the maximality of $\mathcal{T}$ we conclude $\mathcal{T} = \tau(M)$. ◀

Let us notice that the simple term $\mathcal{L}(\mathcal{T})$ used in the proof of Corollary 29 for proving the existence of a β-normal form M s.t. $\tau(M) = \mathcal{T}$ is called the *linearization* of M in [9].

5 Conclusion

The main Theorem 25, combined with Ehrhard and Regnier's Equation 2, gives a characterization of the support $\mathrm{NF}(\tau(M))$ of $\mathrm{NF}(M^*)$ that can be extended to a characterization of $\mathrm{NF}(M^*)$ itself (with positive rational coefficients instead of booleans). This refinement can be done by requiring in addition that the coefficient of each non-zero element t of the formal combination should exactly be the so-called *multiplicity coefficient* $\mathrm{m}(t)$ of t, a number defined by induction on t through the use of a generalization of binomial coefficients to

multisets (see [9]). Roughly speaking the multiplicity coefficient of a simple term t expresses the various ways t can be recombined into itself by varying enumeration of multisets.

Such a strict correspondence between supports and coefficients is lost in a non-deterministic setting. Ehrhard and Regnier's Equation 2 is due to the fact that the supports of the Taylor expansions of the λ-terms are cliques, and this is false as soon as one allows to superpose programs (e.g. by adding an `or` constructor, or a random operator). An open issue is then to capture the convergence of formal combinations of simple terms to non-deterministic, or probabilistic λ-terms.

Acknowledgement. We thank the reviewers for their useful and detailed comments on improving the quality of this publication.

References

1. Henk Barendregt. *The lambda calculus: its syntax and semantics*. North-Holland, Amsterdam, 1984.
2. Pierre Boudes. Thick subtrees, games and experiments. In Pierre-Louis Curien, editor, *Proceedings of TLCA 2009*, number 5608 in Lecture Notes in Computer Sciences, pages 65–79. Springer Verlag, 2009.
3. Gérard Boudol, Pierre-Louis Curien, and Carolina Lavatelli. A semantics for lambda calculi with resources. *Mathematical Structures in Computer Science*, 9(4):437–482, 1999.
4. N.G. de Bruijn. A survey of the project automath. In J.H. Geuvers R.P. Nederpelt and R.C. de Vrijer, editors, *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*, pages 141 – 161. Elsevier, 1994.
5. Thomas Ehrhard. On Köthe sequence spaces and linear logic. *Mathematical Structures in Computer Science*, 12(5):579–623, 2003.
6. Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15(4):615–646, 2005.
7. Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1):1–41, 2003.
8. Thomas Ehrhard and Laurent Regnier. Böhm trees, Krivine's machine and the Taylor expansion of lambda-terms. In Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, editors, *CiE*, volume 3988 of *Lecture Notes in Computer Science*, pages 186–197. Springer, 2006.
9. Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor Expansion of Ordinary Lambda-Terms. *Theor. Comput. Sci.*, 403(2-3):347–372, 2008.
10. Fanny He. On the Characterization of the Taylor Expansion of λ-terms. Master's thesis, LMFI – Paris VII, France, October 2012.
11. Jean-Louis Krivine. *Lambda-calcul: types et modèles*. Études et recherches en informatique. Masson, 1990.
12. Michele Pagani and Christine Tasson. The Taylor Expansion Inverse Problem in Linear Logic. In Andrew Pitts, editor, *Proceedings of the Twenty-Fourth Annual IEEE Symposium on Logic in Computer Science (LICS 2009)*, pages 222–231. IEEE Computer Society Press, 2009.
13. Paolo Tranquilli. Intuitionistic differential nets and lambda-calculus. *Theorical Computer Science*, 412(20):1979–1997, 2011.

Team building in dependence

Julian Bradfield

Laboratory for Foundations of Computer Science, University of Edinburgh,
10 Crichton St, Edinburgh, EH8 9AB, U.K.
jcb@inf.ed.ac.uk

Abstract

Hintikka and Sandu's Independence-Friendly Logic was introduced as a logic for partially ordered quantification, in which the independence of (existential) quantifiers from previous (universal) quantifiers is written by explicit syntax. It was originally given a semantics by games of imperfect information; Hodges then gave a (necessarily) second-order Tarskian semantics. More recently, Väänänen (2007) has proposed that the many curious features of IF logic can be better understood in his Dependence Logic, in which the (in)dependence of variables is stated in atomic formula, rather than by changing the definition of quantifier; he gives semantics in Tarskian form, via imperfect information games, and via a routine second-order perfect information game. He then defines Team Logic, where classical negation is added to the mix, resulting in a full second-order expressive logic. He remarks that no game semantics appears possible (other than by playing at second order). In this article, we explore an alternative approach to game semantics for DL, where we avoid imperfect information, yet stay locally apparently first-order, by sweeping the second-order information into longer games (infinite games in the case of countable models). Extending the game to Team Logic is not possible in standard games, but we conjecture a move to transfinite games may achieve a 'natural' game for Team Logic.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases partially ordered quantification, independence-friendly logic, game semantics

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.116

1 Introduction

In the 60s, Henkin [9] introduced partially ordered quantifiers, which extend first-order logic (FOL) by quantifiers such as $\begin{smallmatrix}\forall x & \exists y \\ \forall u & \exists v\end{smallmatrix}$, where the existential choice of y is to be made without knowing the value of u, and similarly v is chosen independently of x. Subsequent work [8, 17] establish some basic properties, such as the equi-expressiveness of such quantifiers with existential second-order logic (ESOL), but little more was done. Then Hintikka and Sandu [10] gave new life to the topic with a provocative paper, which introduced a new linear syntax with the independence explicitly noted ($\forall x.\forall u.\exists y/u.\exists v/x$), gave it a semantics by extending Hintikka's celebrated games for FOL to be games of imperfect information, so that truth and falsity are defined to be the existence of a winning strategy one or other of the two players, demonstrated a number of surprising properties of the logic, and argued that it should displace FOL as the foundation of mathematics. Although the last claim has been generally politely ignored, a community has grown up exploring the ramifications of independence, and the properties of Independence-Friendly Logic (IFL) have continued to surprise us – the recent textbook [13] provides a comprehensive survey of the mainstream of IFL research. Branching off from the IFL river are a number of tributary streams, including logics for agent-based systems [14], and work by myself and colleagues on the relationship of IFL to logics for concurrency [6, 4]. One extension of IFL studied by Kreutzer and myself

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 116–128

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

[5] was the IF version of Least Fixpoint Logic (IF-LFP), in which the least fixpoint operator on relations is added. At that time, we were able to show, by roundabout means, that this extension is very powerful, and on finite models includes all of SOL. Our semantics there was the fixpoint extension of Hodges' [11] Tarskian semantics for IFL, a semantics which is second-order in nature, as it must be to describe the power of IF.

More recently, Väänänen has been developing the thesis that IFL is perhaps not the best notation for understanding the problems posed by independent quantification. This work is collected, up to 2007, in the textbook [16]. Väänänen proposes that, instead of annotating quantifiers with (in)dependence requirements, one should leave them alone, and instead use, as atomic formulae, *dependence atoms* of the form $\mathscr{D}(\vec{x}, y)$, which means that the value of y is functional in the values of $\vec{x}$ (and is therefore independent of any other variables that may be in scope). So instead of $\forall x.\forall u.\exists y/u.\exists v/x.\ldots$, we write $\forall x.\forall u.\exists y.\exists v.\mathscr{D}(x,y) \wedge \mathscr{D}(u,v) \wedge \ldots$. This he calls Dependence Logic (DL). The primary semantics for DL is a Tarskian semantics, necessarily second-order, which is essentially the semantics given by Hodges for IFL; however, as he does not need to deal with the slashed quantifiers of IFL, the semantics is considerably easier to use in proving results about DL (such as the extremely high undecidability of its validity problem). There are two derived semantics: an imperfect information game which reconstructs the Hintikka game, though it appears less natural, and a perfect information game, which of course has second order moves and is basically the standard Hintikka SOL game for the semantics.

One of the bugbears throughout the history of research on IF and its descendants is negation. There are two ways of understanding negation: in the game-theoretic understanding of IFL, it is natural to think of negation via the game-theoretic duality familiar from the FOL game: negation corresponds to swapping the roles of the two players. However, this is not the same as negation in the classical sense. As the IF games are of imperfect information, they are not determined, and so a formula may be neither true nor false in the usual understanding (for example, $\forall x.\exists y/x.x = y$ is not true on a 2-element domain, but neither is its dual $\exists x.\forall y/x.x \neq y$). Hintikka considered forming the boolean closure of IFL, and Hodges [11] introduced a technique for dealing with classical negation in a limited sense.

True classical negation corresponds to saying (in the game view) that a player 'does not have a winning strategy'. It is therefore apparently a firmly second order operator. In the DL framework, Väänänen's primary semantics is already second-order, and it is therefore immediate to add classical negation: a team satisfies 'not ϕ' if it does not satisfy ϕ. The result is a logic with full second order power, and the ability easily to say highly complex properties.

Väänänen considered that as Team Logic expresses SOL, it was hard to imagine a game theoretic semantics (other than the trivial semantics by playing games directly at second order in the meta-language). In this article, we introduce a novel game semantics for DL, and discuss how one can imagine extending it to TL, although imagination has yet to be made real. Of course, such games have very high complexity to solve, but they fit conceptually within the game semantics of IF logic.

We will start the paper by introducing, completely but very concisely, the notations and concepts of IFL and DL, and also explain the intuitions which underly them – a long-standing feature of IF research has been that it is truly philosophical logic, not just formal symbol shuffling. We will then describe the intuition behind our alternative game semantics, and then proceed to its formal definition and proof of correctness.

2 Preliminaries

Notation and terminology is not entirely standardized. Where there are several options, I shall generally follow Väänänen, but sometimes Hodges. I shall use standard concepts such as free and bound variables without further explanation. Formulae are always considered as abstract syntax trees, and words such as 'earlier' and 'above' should be so interpreted.

An important convention is that 'subformula' always means 'node in the abstract syntax tree', and includes the formula itself; unlike FOL, in IF logics one may need to treat different occurrences of the same textual subformula differently.

Throughout we assume some structure $\mathscr{M}$ with a domain M of values which may be bound to variables of the logics. All the logics may be defined to include constants, function symbols and terms built thereby, and interpreted by operations in $\mathscr{M}$, but to reduce notation, we shall generally state definitions without them.

We will start by, as Hintikka did originally, ignoring negation, and working entirely in negation normal form with both $\wedge$ and $\vee$, and both $\forall$ and $\exists$. After introducing IF and DL in this form, we will discuss negation.

2.1 IF logic syntax and semantics

The syntax is that of FOL in negation normal form (where, as usual, we shall let $x, y, \dots \in Var$ denote variables, and $P, Q, \dots \in Prop$ denote relational atoms) – we write $\not P$ for negated atoms. The equality symbol is usually included, but for our purposes may be treated as a relational atom rather than a distinct primitive. In addition, there are 'slashed quantifiers' $\exists x/\vec{y}$ and $\forall x/\vec{y}$, where $\vec{y}$ is a subset of the variables bound earlier in the formula (or, in general, of those variables and the free variables of the formula). '$/\vec{y}$' does not bind the $\vec{y}$.

The semantics of a sentence ϕ (the original presentation does not account for open formulae) is given by a game between two players, Eloise and Abelard. The game is played just as for the standard Hintikka game for FOL: at $\exists x.\psi$ ($\forall x.\psi$), Eloise (Abelard) chooses a value for x; at $\psi_0 \vee \psi_1$ ($\psi_0 \wedge \psi_1$), Eloise (Abelard) chooses to proceed to ψ_0 or ψ_1. At atoms and negated atoms, Eloise (Abelard) wins iff the formula is true (false) with the current binding of variables to values.

At a slashed quantifier $\exists x/\vec{y}$ ($\forall x/\vec{y}$), Eloise (Abelard) is required to make her (his) choice of y without knowing the currently bound values of $\vec{y}$.

ϕ is said to be *true* iff Eloise has a winning strategy in this game of imperfect information; *false* iff Abelard does; and *undetermined* otherwise.

An alternative, and more mathematically tractable, way of phrasing the semantics is to drop any reference to 'imperfect information' in the moves of the game, and instead say that the players are subject to constraints in the strategies they are allowed to use: their strategies at slashed quantifiers must be *uniform* in the $\vec{y}$.

2.2 DL syntax and semantics

DL has the syntax of FOL (in negation normal form for the present), with addition of *dependence atoms* $\mathscr{D}(\vec{x}, y)$ and negated dependence atoms $\not\mathscr{D}(\vec{x}, y)$

An *assignment* s is a mapping from a set $\mathrm{dom}(s)$ of variables to values in M. $s(v/y)$ denotes the assignment s extended (or updated) to map y to v.

A *team* is a set of assignments with a common domain. If X is a team, $X(M/y)$ denotes the team $\{s(v/y) : s \in X, v \in M\}$, i.e. X extended by all possible choices for y. If $F\colon X \to M$

is a function choosing one value for each member of X, $X(F/y)$ denotes the team $\{s(F(s)/y) : s \in X\}$, i.e. X extended by one choice of y for each member of the team.

A *triple* is a tuple (ϕ, X, d) where d is 0 or 1. The intended interpretation is that $(\phi, X, 1)$ means that the team X makes the formula ϕ true, and $(\phi, X, 0)$ means that X makes ϕ false – as with IF logic, it may be that neither holds.

The semantics of DL is given by defining the set $\mathscr{T}$ of triples that hold in $\mathscr{M}$. $\mathscr{T}$ is defined inductively as follows, where 'dually' means 'by exchanging 0 and 1 in the definition'. For relational atoms $P(\vec{x})$, $(P(\vec{x}), X, 1) \in \mathscr{T}$ iff $P(s(\vec{x}))$ holds in $\mathscr{M}$ for every $s \in X$, and $(P(\vec{x}), X, 0) \in \mathscr{T}$ iff $P(s(\vec{x}))$ fails in $\mathscr{M}$ for every $s \in X$; and dually for $\not P$.

For the 'boolean' operators, the rules are $(\psi_0 \wedge \psi_1, X, 1) \in \mathscr{T}$ iff $(\psi_0, X, 1) \in \mathscr{T}$) and $(\psi_1, X, 1) \in \mathscr{T}$); $(\psi_0 \wedge \psi_1, X, 0) \in \mathscr{T}$ iff there are X_i such that $X = X_0 \cup X_1$ and $(\psi_i, X_i, 0) \in \mathscr{T}$ for both i; and dually for $\vee$.

For the quantifiers, $(\exists x.\psi, X, 1) \in \mathscr{T}$ iff $(\psi, X(F/x), 1) \in \mathscr{T}$ for some choice function F; and $(\exists x.\psi, X, 0) \in \mathscr{T}$ iff $(\psi, X(M/x), 0) \in \mathscr{T}$; and dually for $\forall$.

So far, the semantics looks like the natural lifting of FOL to teams, and indeed the semantics is, so far, equivalent to the FOL semantics.

The extension comes with the dependence atoms. The rule for positive dependence atoms is: $(\mathscr{D}(\vec{x}, y), X, 1) \in \mathscr{T}$ iff X makes y functional in $\vec{x}$: that is, if $s, s' \in X$ agree on $\vec{x}$, they must also agree on y; and $(\mathscr{D}(\vec{x}, y), \varnothing, 0) \in \mathscr{T}$; and dually for $\not\mathscr{D}$.

It is hard to give a good intuition for the semantics of $\not\mathscr{D}$; it is never true, except in the artifical vacuous case of the empty team (a purely technical requirement). Conversely $\mathscr{D}$ is never false (except vacuously), which one can perhaps best understand by considering that a non-functional team can always be made functional by ejecting some of its members, and it is a core property of the logic that if a team makes a formula true (or false), so does any subteam.

Note that a consequence of these rules is that the empty team makes every formula both true and false; but since it is impossible to give any intuitive meaning to a formula in the absence of any assigment, this technicality is not overly obtrusive.

On the other hand, the team consisting solely of the empty assignment, i.e. the unique non-empty team on no variables, $X = \{\varnothing\}$, plays a major role. A sentence ϕ is *true* in $\mathscr{M}$ if $(\phi, \{\varnothing\}, 1) \in \mathscr{T}$.

In Väänänen's terminology, we say X *has type* ψ iff $(\psi, X, 1) \in \mathscr{T}$.

2.3 Negation

As adumbrated in the introduction, the problem is that there are two natural understandings of negation, and though they coincide on FOL, they differ on IF. The first understanding is *game negation*: this corresponds to exchanging the roles of the players. Thus, for example, the game negation of an undetermined sentence is also undetermined, while the game negation of a true sentence is false, and vice versa. Game negation, in both the IF game semantics and the DL semantics, obeys the familiar De Morgan dualities.

The second understanding is *classical negation*: a classically negated sentence is true iff it is not the case that the unnegated sentence is true. Hence the negation of an undetermined sentence is true, thereby introducing an asymmetry.

The first careful analysis of these issues was done by Hodges [11], who also showed that classical negation could be defined in terms of game negation and an independently justifiable 'flattening' operation. In that article, Hodges used $\sim$ for game negation, and $\neg$ for classical negation. Rather unfortunately, Väänänen [16] reversed this notation. I shall

follow Hodges' notation, as it is more mnemonic ($\sim$ is smooth and symmetrical, like game negation, and $\neg$ is flat and asymmetrical, like classical negation).

As will be apparent from our description of the semantics in negation normal form, the 'standard' negation in IF and DL is the game negation. If it is included as a primitive, then the rule is $(\sim\psi, X, d) \in \mathscr{T}$ iff $(\psi, X, 1-d) \in \mathscr{T}$, and then $\wedge$ can be defined in terms of $\vee$ and $\forall$ in terms of $\exists$ by the usual dualities, and our negated atoms $\not R$ and $\not\mathscr{D}$ are just $\sim P$ and $\sim\mathscr{D}$.

We defer further discussion of classical negation until after we have introduced team-building games for the simpler case of DL.

3 Team-building games for DL

The idea is simple: although IFL and DL require either imperfect information, restrictions to uniform strategies, or games with explicit second order moves, one can consider building finite approximations to these second-order objects incrementally, essentially expressing the construction as a fixpoint operator, which we have previously studied in the context of IF; thereby, although the players build actual teams in the limit, each move in the game looks like a first-order game move.

Henceforth, we consider only **countable** models.

3.1 Game definition

As with the games for IFL, we shall define our games only for sentences. Throughout the rest of this section, let ϕ be a sentence of DL in negation normal form. Let Φ be the set of subformulas of ϕ; let *Var* be restricted to mean the variables occurring in ϕ. For a subformula ψ, let $Var(\psi)$ be the variables in scope in ψ (regardless of whether they occur in ψ). Let $\psi' \preccurlyeq \psi$ mean ψ' is a subformula of ψ.

An *annotated assignment*, or aa for short, is an assignment s together with a subformula ψ. It serves to record how the assignment was used.

A *crowd* is a set of annotated assignments, not necessarily with the same domains.

A position in the game comprises a subformula ψ, a crowd X, and an assignment s. s behaves exactly as in a first-order Hintikka game, and the crowd handles the dependency aspects.

The initial position is $(\phi, \varnothing, \varnothing)$.

The game looks like repeated plays of the first-order game, but with the addition that Eloise remembers how she played last time, and is required to play consistently with her earlier choices. The rules are:

- At position $(\psi_0 \vee \psi_1, X, s)$, Eloise checks to see whether there is $(s', \psi') \in X$ such that $\psi' \preccurlyeq \psi_i$ for some i, and $s' \restriction Var(\psi_0 \vee \psi_1) = s$. If so, there will be only one such i (which will follow from the rules), and she must choose it. Otherwise, she chooses freely $i = 0, 1$. Play moves to (ψ_i, X, s).
- For $(\psi_0 \wedge \psi_1, X, s)$, Abelard chooses freely whether to move to ψ_0 or ψ_1.
- At position $(\exists x.\psi, X, s)$, Eloise checks to see whether there is $(s', \psi') \in X$ such that $\psi' \preccurlyeq \psi$ and $s' \restriction Var(\exists x.\psi) = s$. If so, she chooses $v = s'(x)$; otherwise she has a free choice of v. Play moves to $(\psi, X, s(v/x))$.
- For $(\forall x.\psi, X, s)$, Abelard has a free choice of value for x.
- At a (negated) relational atom R ($\not R$), play stops, and Eloise wins the play iff s satisfies (fails) R.

- At a dependency atom $\psi = \mathscr{D}(\vec{x}, y)$, we check whether team built so far for ψ (recorded in the crowd X) is functional. We take the crowd $X' = X \cup \{ (s, \psi) \}$, and consider the team $Y = \{s' : (s', \psi) \in X'\}$. If Y does not satisfy ψ – i.e. y is not functional in $\vec{x}$ – then Abelard wins. Otherwise, if $(s, \psi) \in X$, then Eloise wins. Otherwise, Abelard challenges by moving to $(\phi, X', \varnothing)$.
- At a negated dependency atom $\psi = \not\mathscr{D}(\vec{x}, y)$, Abelard wins.

The intuition for the $\mathscr{D}$ rule is that during the current play, Eloise builds up a series of choices which together make a team satisfying the dependency atom. However, Abelard should have challenged her with all his possibilities; if he has exhausted his choices (in the finite case), or unnecessarily repeated something he tried earlier, he loses. (This feature is not necessary; it serves merely to force the game to be finite on a finite domain.)

The game as defined so far may, in the case of an infinite domain, not terminate, as play may pass indefinitely through dependency atoms. On such a play, Eloise wins.

This completes the definition of the game.

3.2 Remarks

The positions of the game are all finite objects, and hence the moves and finite-winning conditions are all recursive (relative to the intepretations of relational atoms), even if the domain is infinite.

As noted, if ϕ is a formula without dependence atoms (i.e. is FOL), then the game is exactly the usual Hintikka game with some additional book-keeping that is not used.

For non-FOL formulae, if the sentence ϕ is true, Eloise builds up incrementally on each play a team required to satisfy the dependency atoms, which in the DL semantics are constructed at one swoop by the quantifier semantics. In the case of a finite domain, Eloise's strategy in this game amounts to building her full strategy for the second-order DL game – on every play of this game. The consequence is that a winning strategy for Eloise in this game gives a winning strategy for DL (or for the IF game), but the same is not true for Abelard.

The game has perfect information, has simple (Büchi) winning conditions, and is therefore determined. Consequently it is clear that this game does not match the IF game. In fact, our asymmetrical treatment of the players in the rules and winning conditions amounts to making this the game simulate the second order game for the skolemized sentence, rather than the IF imperfect information game.

Because the winning conditions are Büchi, it also follows that if a player has a winning strategy, they have a history-free winning strategy, i.e. one that depends only on the current position in the game. In particular, the order in which aas are added to the crowd need not be remembered.

Since we have included a repetition detection in the winning conditions, which is not actually necessary, the game always terminates on finite domains.

3.3 Examples

3.3.1 A simple 'Snap' game

First, consider the (IF non-determined) sentence mentioned in the introduction, representing the game where Eloise and Abelard independently choose a boolean value in $M = \{0, 1\}$, and Eloise wins iff the two values are the same. In IFL, this is $\forall x.\exists y/x.x = y$; in DL, it is $\forall x.\exists y.\mathscr{D}(y) \wedge x = y$.

A sample play of the game is: A chooses 0; E chooses 0; A chooses the dependency atom, X is functional, so play continues: A chooses 1, E chooses 1, and then A will again challenge D, and E will lose because the crowd is now non-functional. This strategy of repeatedly challenging until either the accumulated crowd is non-functional or E chooses a $y \neq x$ is winning for Abelard; contrast with the IF game, where Abelard has no winning strategy.

3.3.2 An infinite game

A more interesting example arises by borrowing a well known trick for expressing the infinitude of the domain in IFL (and so incidentally demonstrating the non-FOL expressivity of IFL). In DL, the formula is

$$\exists c.\forall x.\forall u.\exists y.\exists v.\mathscr{D}(x,y) \wedge \mathscr{D}(u,v) \wedge (y = v \vee x \neq u) \wedge (x = u \vee y \neq v) \wedge (y \neq c)$$

To understand this formula, see that it asserts the existence of y that is $f(x)$, and v that is $g(u)$, and moreover $f = g$, by the first FOL clause (which says $x = u \to y = v$), and f is injective, by the second FOL clause (which says $y = v \to x = u$), and moreover f never takes the value c.

In the IFL version, the formula is true on an infinite domain, but undetermined on a finite domain of size > 2 – although Eloise can't win, she may, by blind chance, escape Abelard's attempts to detect a failure of the main clause.

Consider the team-building game in the case of an infinite domain. Eloise has a winning strategy – indeed, uncountably many winning strategies – just as in the IF game, as follows. After choosing c, she keeps in her head a suitable function f – for example, she enumerates the domain starting at c, and maps each element to the next in the enumeration. After Abelard chooses x and u, she chooses $y = f(x)$ and $v = f(u)$. Now Abelard has no chance to win in the boolean clauses, so his only hope is to challenge the dependency atoms. But whichever one he challenges, the past choices of y or v, as recorded in the crowd, are functional; so either he repeats himself and loses, or he plays for ever, and loses.

Now consider a finite domain, say $M = (0, 1, 2)$. Now Eloise cannot win; but Abelard can win, because by repeating the challenge at dependency atoms, he can force Eloise into violating either functionality, injectivity, or not hitting c.

This illustrates the theorem we shall shortly prove: Eloise wins the team-building game iff she wins the IF game, and thus iff the DL sentence is true.

3.3.3 Team-building and game negation

If our game were perfectly symmetric, we would have a contradiction (since the IF game is not determined) – how does the asymmetry we introduced solve the problem?

To see this, consider some examples involving game negation. First, consider the 'Snap' formula. If we negate the formula in DL, and push negations through to the atoms, we get $\exists x.\forall y.\not\mathscr{D}(y) \vee x \neq y$. Clearly Eloise cannot win this game: she can't win at $\not\mathscr{D}$, and Abelard will choose y to equal x. This is, indeed, a winning strategy for Abelard, although the formula is not false in IFL or DL.

Now consider the infinity example. The game negation of the formula is

$$\forall c.\exists x.\exists u.\forall y.\forall v.\not\mathscr{D}(x,y) \vee \not\mathscr{D}(u,v) \vee (y \neq v \wedge x = u) \vee (x \neq u \wedge y = v) \vee (y = c)$$

Suppose the domain is infinite. Eloise cannot win at the negated dependency atoms. If she chooses $u = x$, Abelard chooses his $y = v \neq c$ and wins; if she chooses $u \neq x$, Abelard chooses $y \neq v$ and $y \neq c$ and wins. Here we have Abelard winning a false sentence.

In the case of a finite domain, the same strategy suffices for Abelard.

To sum up, by the introduction of the asymmetry, we have arranged that Eloise wins ϕ iff ϕ is true (in the DL or IFL sense); and Eloise wins $\sim\phi$ iff ϕ is false; and if ϕ is undetermined, Abelard wins both ϕ and $\sim\phi$.

3.4 Correctness

The underlying core of the correctness theorem for the team-building game is (the dual of) Kleene's theorem that Σ^0_1-$IND = \Pi^1_1$. However, the details require some attention.

▶ **Theorem 1.** *If Eloise has a winning strategy in the team-building game for ϕ, then ϕ is DL-true, and vice versa.*

Proof. Suppose then that Eloise has a winning strategy in the team-building game for ϕ. For each subformula ψ we will construct a team $X(\psi)$ that has the type of ψ, and so that $\varnothing$ is in the team for ϕ. To do this, we will construct choice functions for Eloise, essentially giving her strategy in the second-order game for DL. The team-building game strategy does not necessarily define a unique winning strategy in the second-order game; we shall build one particular strategy.

We first make an auxiliary definition. Let X be a crowd, and ψ a formula. $X\restriction\psi$ denotes the team given as

$$\{s\restriction Var(\psi) : (s,\psi') \in X \text{and } \psi' \text{ is a subformula of } \psi\}$$

Now let T be the game tree that results from playing all Abelard's choices against Eloise's strategy. We will need to traverse this tree in a particular well-behaved order. Wlog we may assume that $M = \mathbb{N}$ (the finite case is strictly easier). A node in the game tree can be uniquely defined by a sequence of integers, giving the choices made by the players: the value of v at quantifiers, and 0 or 1 at boolean operators (although, of course, there is no Eloise branching in this tree). Order the nodes lexicographically according to this sequence (an ordering which may have length ω^ω). Call this ordering $\zeta\colon \omega^\omega \to T$.

We could define suitable teams for the first-order part directly from the semantics, so our concern is with the dependency atoms. By the rules of the game, every crowd X that appears on a dependency atom node satisfies the atom; but any such crowd has dealt with only a finite number of Abelard's possible plays. Since the union of functional teams is not necessarily functional, we need to take care when combining crowds. We will achieve this by the ordered traversal of the tree.

We will proceed by building up a crowd that contains the information required to produce teams for the DL semantics of ϕ, starting with the empty crowd.

Consider the last Abelard choice in the node labels of T (as defined above). We will explore all the possible choices, for fixed values of the earlier choices, by following an 'almost leftmost' path through T. Specifically, starting from the root, we follow a path in T by taking at each Abelard choice the leftmost unexplored choice. After making the last Abelard choice and following any subsequent Eloise choices, we are at an atomic node (ψ, X, s). If ψ is a relational atom, then s satisfies it; we backtrack to the point of the last Abelard choice, and explore the next choice. ψ cannot be a negated dependency atom, as no such nodes occur in the tree. If ψ is a dependency atom, we add s to the crowd (per the game rules), and follow on down T, repeating the earlier choices, and then exploring the next possibility for the final Abelard choice. Note that the game rules mean that Eloise cannot change her mind about her response to any of the pre-final Abelard choices, as they will be recorded in the crowd.

In the case of a finite domain, we reach a leaf, and take the crowd X_1 at that leaf. In the case of an infinite domain, this procedure may involve traversing an infinite path through T. In that case, we take X_1 to be the union of all the crowds at atomic nodes on the path. The resulting crowd does satisfy all the dependency atoms occurring on path: suppose not, then there is a dependency atom $\psi = \mathscr{D}(\vec{x}, y)$ on the path, and two assignments s, s' annotated with ψ, such that s and s' agree on $\vec{x}$ but differ on y. But since X_1 is a union of a increasing chain of crowds, there is some ψ-node on the path at which both s and s' already occur, and is therefore false, which is a contradiction.

Having generated X_1, we now restart the game with crowd X_1, and explore all the final Abelard choices for the next value of the pre-final Abelard choice. It is not immediate that Eloise's strategy can still win starting with an infinite crowd (that therefore does not appear anywhere in the actual team-building game); however, we can show that she can. Suppose not. Then Abelard has a winning play. However, all Abelard-winning plays are finite; therefore his winning play uses only a finite amount of information about the crowd, and so he can also win with this play against the finite crowd, which does appear in T.

Thus we obtain $X_2 \supseteq X_1$. We then repeat the procedure until we have exhausted the pre-final Abelard choices; then back up to consider the pre-pre-final choice for Abelard, and so on until we have exhausted all the Abelard choices. This gives a crowd X, from which Eloise can win the team-building game for ϕ; but as the crowd is exhausted, all her choices are pre-made, and the game will terminate after one round. This then gives the choices for the DL semantics: at $\exists x.\psi$, the choice function F is extracted from the crowd, and at disjunctions, the choice of disjunct is extracted from the crowd.

This completes the proof of the interesting direction.

The other direction is almost immediate: if ϕ is true, then the choice functions F at existential nodes, and the split $X = X_0 \cup X_1$ (with an arbitrary choice to make the split disjoint) give a strategy for Eloise in the team-building game. ◀

▶ **Corollary 2.** *Eloise wins the team-building game for $\sim\phi$ iff ϕ is false.*

Proof. Game negation in the DL semantics is exactly the swapping of 1 triples and 0 triples. ◀

▶ **Corollary 3.** *If ϕ is undetermined, then Abelard wins the team-building game for ϕ and also for $\sim\phi$.*

3.5 Further examples and remarks

3.5.1 The infinite game again

The infinite game of subsection 3.3.2 illustrates the proof in both directions. If we know the DL formula is true, then Eloise has choice functions defined by her 'secret' function f, and the obvious choice function at the disjunctions. On the other hand, if she is playing the team-building game, she does not even have to have the function f in her head: all she needs to do is, in each round, is to choose any y and v which will satisfy the relational part, and maintain consistency with her previous choices, thus building up the function f. Note that there is no *a priori* need for her to build the same function along different branches of the game tree; most of the work in the correctness proof was in showing that even if she doesn't, we can extract a single choice function from a modified tree.

3.5.2 Constraints on Abelard

In the original IFL, only $\exists$ quantifiers could be slashed, and only on $\forall$ variables: Abelard had no memory loss, and Eloise always remembered her own choices. Later work adopted the symmetrical approach, but many people (including myself) found it hard to hard to understand constrained Abelard choices intuitively as a logical property. The intuition that Abelard also has restricted knowledge at choices is simple enough, but one (or at least I) naturally imagines such a restriction should make it easier for Eloise to win; for, in the formal game semantics, she surely now only has to win against Abelard's uniform strategies, which should be easier than winning against all strategies.

However, this intuition is misleading: consider $\exists x.\forall y/x.\psi$. Eloise only has to win against uniform Abelard strategies – but since she doesn't know which uniform strategy Abelard is playing, that's the same as winning against any strategy. In other words, slashing Abelard variables makes it harder for Abelard to win, but not easier for Eloise to win. (As a referee put it, one needs to lose the 'truth bias' and consider truth and falsity on equal terms.) This is perhaps more easily seen in DL, thus:

In dependence logic, the dependence atoms know nothing about whether variables belong to Eloise or Abelard, and one can perfectly well write, e.g., $\exists x.\forall y.\mathscr{D}(x,y) \vee x \neq y$. This sentence is true: in the DL semantics, E's choice function at $\exists$ selects an arbitrary element v for x, and then at the $\vee$ she splits the team with (v,v) on the left and all other (v,v') on the right. In the team-building game, she chooses v (which will thus be fixed in the crowd for the rest of the play), and plays left or right at $\vee$ similarly. In this case, the game will terminate after at most two rounds, owing to our optimizing termination criterion for repeated Abelard challenges.

Here we have the alternative explanation of why Abelard slashing has been found confusing in IFL. Consider again the Snap formula: $\forall x.\exists y/x.x = y$ in IFL; $\forall x.\exists y.\mathscr{D}(y) \wedge x = y$ in DL. One naturally defines that the negation of the IF formula will be $\exists x.\forall y/x.x \neq y$. However, DL negation shows us that this means $\exists x.\forall y.\not\mathscr{D}(y) \vee x \neq y$, and since negated dependency atoms are never true, the Abelard slashing has no effect on truth, only on falsity.

3.5.3 Team-building for IFL

Since there is a simple translation from IFL (assuming that variables are not re-used) into DL, by $\exists y/\vec{x}.\psi$ mapping to $\exists y.\mathscr{D}(\mathit{Var}(\psi)\backslash\vec{x}, y) \wedge \psi$, and dually for $\forall$, the team-building game for DL immediately gives one for IFL. We can avoid the explicit translation by saying that immediately after a slashed existential quantifier, Abelard has the choice to proceed into ψ, or to start the next round. (Nothing new happens after a slashed universal quantifier, for the reasons just explained.)

4 Negation and Team Logic

4.1 Team Logic

When Väänänen defined DL, he did not use the nnf formulation that we have been using. Rather, he took game negation $\sim$ as a primitive, along with $\exists$ and $\vee$, and defined $\forall$ and $\wedge$ via De Morgan dualities with $\sim$. The semantics with fundamental triples is designed to maintain both the 'truth' and 'falsity' semantics at the same time, via the flag $d = 0, 1$.

However, when he extended DL to *Team Logic* (TL) by adding classical negation, he reverted to a traditional asymmetrical semantics, in which the denotation of a formula is a

set of teams, rather than a set of fundamental triples. Consequently, the game duals are no longer maintained automatically, and so in TL game negation does not appear explicitly, and the game dual connectives are defined as primitives, while classical $\neg$ is added as a primitive with its usual semantics.

To add to the confusion, TL changes the notation used for connectives, owing to a (slightly distorted) analogy with linear logic, and because he wishes to maintain the usual De Morgan dualities when considering *classical* rather than game negation. So the DL and IFL $\vee$ is instead written $\otimes$, and the IFL and DL $\forall$ is instead written ! (by analogy with the linear logic 'of course' operator). Then $\vee$ is reused for the classical dual of $\wedge$, and $\forall$ for the classical dual of $\exists$. (The real link between TL and linear logic is explained in [2] – it is partly linear, and partly intuitionistic.) The justification for this is perhaps that TL is really a first-order logic about teams, which involves statements about their members, whereas DL and IFL intend to be logics about members, which need teams to express certain properties.

We hope to avoid the confusion within this article by leaving our existing DL notations alone, so as not to have to reformulate the game rules, and then when required we will use subscripted versions for the new TL classical operators.

4.2 Negation in the team-building game

The team building game has the property that Eloise winning characterises truth, and Abelard winning characterises non-truth, so that at the level of entire sentences, swapping players corresponds to classical negation. However, we know from [16] that adding classical negation as a primitive to DL greatly increases its complexity, taking it to full second-order power. It is therefore interesting to explore what happens to the team-building game when we apply the usual methods of incorporating negation into a model-checking game.

Before doing so, we make one simplification to the team-building game: we drop the condition at dependence atoms that says "if $s \in X$ and X is functional, then Eloise wins". As already noted, this condition is just an optimization for the case of finite domains; its job can also be done by the "Eloise wins infinite plays" condition.

The usual way to incorporate negation into a model-checking game for a logic (e.g. in modal mu-calculus games, see e.g. [7]) is to add the De Morgan duals (w.r.t. the negation in question) of all the operators in the logic (if not there already), dualize the rules for the new operators, and dualize the winning conditions, and then deal with formulae in nnf (w.r.t. the negation in question). If we apply this methodology:

- We need the classical duals of the 'boolean' operators (where we already have both game duals in the logic). The classical dual of $\wedge$ we write as $\vee_T$ ([16] writes $\vee$); of $\vee$ we write $\wedge_T$ ([16] writes $\oplus$).
- In the semantics of TL, the DL $\forall$ and $\exists$ are self-dual under classical negation, and so no new operators are needed.
- For the relational atoms, the interpretation of $\neg$ is the same as that of $\sim$, so we need no new symbols.
- For dependency atoms, $\mathscr{D}$ under an odd number of negations is written $\bar{\mathscr{D}}$, and a $\not\mathscr{D}$ is written $\bar{\not\mathscr{D}}$.

The rules for the operators are dualized by exchanging 'Eloise' and 'Abelard'.

Dualizing the winning conditions raises a number of obstacles in the classically negated dependencies $\bar{\mathscr{D}}$. The natural dualized rule would have Eloise winning as soon as a functional crowd (i.e. a crowd not satisfying the atom) is constructed. This, of course, is wrong – the first time through the atom, there will be one aa in the crowd, which is necessarily functional.

Instead, we need to allow Eloise to keep trying to construct a non-functional team; if she fails forever, then Abelard will win. So the rule for $\bar{\mathscr{D}}$ simply moves the game to the start of the next round.

The real problem comes from the infinite plays. The team-building game has Eloise winning on infinite plays involving $\mathscr{D}$. We did not need to worry about which particular $\mathscr{D}$ occurs infinitely often, because if Abelard can break any allegedly functional Eloise team, he can do it in finite time, and his best strategy is to do so. However, if Abelard needs to win infinite plays involving $\bar{\mathscr{D}}$, we are faced with the problem of plays involving both infinitely many $\mathscr{D}$s and infinitely many $\bar{\mathscr{D}}$s. While one might think that a sufficiently clever intertwining might solve the problem, it is in fact possible to prove that there is no (reasonable) solution to this problem.

▶ **Theorem 4.** *Given an extended team-building game as indicated, then there is no first-order, or even uniform second-order, way to define winning infinite plays such that the game captures Team Logic.*

Proof. It is a standard theorem (see, e.g. [12]) that if the winning conditions of a (standard) perfect information game are Σ^1_n, then the winning sets for the game are at most Π^1_{n+1}. Since TL expresses all second-order properties, no second-order winning condition of fixed quantifier depth suffices. ◀

However, while it is impossible to reach SOL with standard games, it may be possible with richer games (other than by going back to imperfect information). We

▶ Conjecture 5. The team-building game can be extended to a game with transfinitely (but still countably) long plays, with first-order winning conditions, so as to capture Team Logic.

5 Conclusion

In this article, we have shown how it is possible to design a game semantics for DL (and hence IFL) that preserves the intuition of first-order games, but does not resort to imperfect information. The game is thus determined, and so characterizes truth and non-truth, rather than truth and falsehood.

We discussed why the natural attempts to extend such games to full Team Logic must fail, and conjectured that nonetheless they can be extended with the (already studied) idea of transfinite plays.

Future work is to investigate the conjecture further, and also explore the use of similar team-building games in the logic IF-LFP (independence logic with fixpoints), a logic which is also second-order expressive, and has not received any game characterization previously.

Acknowledgements. This paper grew out of the Dagstuhl Seminar 13071 'Dependence Logic: Theory and Applications'.

I thank an anonymous referee for helpful corrections and suggestions.

References

1 S. Abramsky, J. Kontinen, J Väänanen, H. Vollmer, Dependence Logic: Theory and Applications (Dagstuhl Seminar 13071), *Dagstuhl Reports*, **3**(2) 45–54, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, `http://dx.doi.org/10.4230/DagRep.3.2.45` (2013).

2 S. Abramsky, J. A. Väänanen, From IF to BI: a tale of dependence and separation. CoRR abs/1102.1388 (2011)

3 J. C. Bradfield, The modal mu-calculus alternation hierarchy is strict, *Theor. Comput. Sci.* **195** 133–153 (1997).

4 J. C. Bradfield, Independence: logics and concurrency. In: Tuomo Aho and Ahti-Veikko Pietarinen (eds). Truth and Games: Essays in Honour of Gabriel Sandu (Acta Philosophica Fennica 78) 47–70. Helsinki: Societas Philosophica Fennica. (2006)

5 J. C. Bradfield and S. Kreutzer, The complexity of independence-friendly fixpoint logic, in: Stefan Bold, Benedikt Löwe, Thoralf Räsch, Johan van Benthem (eds.), Foundations of the Formal Sciences V, Infinite Games 39-62. [Studies in Logic 11]. London: College Publications (2007).

6 J. C. Bradfield and S. B. Fröschle, Independence-friendly modal logic and true concurrency, *Nordic J. Computing* **9** 102–117 (2002).

7 J. C. Bradfield and C. Stirling, Modal Mu-Calculi, in P. Blackburn, J. van Benthem and F. Wolter, *Handbook of Modal Logic*,721–756, Elsevier (2007).

8 H. B. Enderton, Finite partially ordered quantifiers, *Z. für Math. Logik u. Grundl. Math.* **16** 393–397 (1970).

9 L. Henkin, Some remarks on infinitely long formulas, *Infinitistic Methods*, Pergamon Press, Oxford and PAN, Warsaw, 167–183 (1961).

10 J. Hintikka and G. Sandu, A revolution in logic?, *Nordic J. Philos. Logic* **1**(2) 169–183 (1996).

11 W. Hodges, Compositional semantics for a language of imperfect information, *Int. J. IGPL* **5**(4), 539–563.

12 Y. N. Moschovakis, *Descriptive set theory*. North-Holland, Amsterdam & New York (1980).

13 A. L. Mann, G. Sandu and M. Sevenster, Independence-Friendly Logic – A Game Theoretic Approach, Cambridge University Press (2011).

14 Marc Pauly, Logic for Social Software. Ph.D. Thesis, Universiteit van Amsterdam (2001).

15 G. Sandu, On the logic of information independence and its applications, *J. Philos. Logic* **22** 361-372 (1993).

16 J. Väänänen, *Dependence Logic*. Cambridge University Press (2007).

17 W. J. Walkoe, Jr, Finite partially-ordered quantification. *J. Symbolic Logic* **35** 535–555 (1970).

Saturation-Based Model Checking of Higher-Order Recursion Schemes

Christopher Broadbent[1] **and Naoki Kobayashi**[2]

1 **The Technische Universität München**
broadben@in.tum.de
2 **The University of Tokyo**
koba@is.s.u-tokyo.ac.jp

Abstract

Model checking of higher-order recursion schemes (HORS) has recently been studied extensively and applied to higher-order program verification. Despite recent efforts, obtaining a scalable model checker for HORS remains a big challenge. We propose a new model checking algorithm for HORS, which combines two previous, independent approaches to higher-order model checking. Like previous type-based algorithms for HORS, it directly analyzes HORS and outputs intersection types as a certificate, but like Broadbent et al.'s saturation algorithm for collapsible pushdown systems (CPDS), it propagates information backward, in the sense that it starts with target configurations and iteratively computes their pre-images. We have implemented the new algorithm and confirmed that the prototype often outperforms TRecS and `CSHORe`, the state-of-the-art model checkers for HORS.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Model checking, higher-order recursion schemes, intersection types

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.129

1 Introduction

The model checking of higher-order recursion schemes (higher-order model checking for short) [10, 22] is a generalization of finite-state and pushdown model checking, and has recently been applied to automated verification of higher-order programs [13, 23, 19]. Higher-order recursion schemes (HORS) are higher-order grammars describing possibly infinite trees, and can also be seen as simply-typed functional programs with recursion and tree constructors. Thus they serve as natural models for higher-order programs, and various verification problems for functional programs can be easily reduced to higher-order model checking [13, 11, 23, 26].

Despite the very bad worst-case complexity of higher-order model checking (k-EXPTIME complete for order-k HORS [22, 18]), several *practical* model checking algorithms [11, 14, 21, 4] have been developed, which do not immediately suffer from the hyper-exponential bottleneck. The state-of-the-art model checker TRecS can handle a few hundred lines of HORS generated from various program verification problems. It is, however, not scalable enough to support automated verification of thousands or millions of lines of code. Thus, obtaining a better higher-order model checker is a grand challenge in the field, and that is also the general goal of the present work.

The previous algorithms for higher-order model checking can be roughly classified into two radically different approaches, as shown in Table 1. The algorithms of Kobayashi [11, 14] and Neatherway et al. [21] directly work on HORS and check properties expressed by trivial

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 129–148

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Table 1 Classification of higher-order model checking algorithms.

algorithms	models	properties	state representation	propagation
Type-based approach [11, 14, 21]	HORS	trivial	types	forward
Saturation-based approach [4, 5]	CPDS	co-trivial	stack automata	backward
HorSat	HORS	co-trivial	types/automata	backward
HorSatT	HORS	trivial	types/automata	backward

tree automata [1] (Büchi tree automata where all the states are accepting, which can be used for describing safety properties stating that certain (bad) states are unreachable). They use intersection types [13, 17] to finitely represent infinite states. Information is propagated in the *forward* direction, in the sense that they start from the requirement that the start symbol must have the initial state of the trivial automaton as its type, and compute the "post-image" to collect type information needed to conclude that bad states are unreachable. On the other hand, the recent algorithm of Broadbent et al. [4, 5] works on collapsible pushdown systems (CPDS) (which are equi-expressive with HORS although the mutual translations are rather complex [9, 6]) and deals with properties expressed by *co-trivial* tree automata (Büchi tree automata where no states are accepting, which can be used for describing the complement of a safety property, stating that certain (bad) states are reachable). Their algorithm generalizes the saturation algorithm for pushdown system model checking [2, 7]. It finitely represents infinite states of CPDS by using stack automata, and propagates information in the *backward* direction, in the sense that it starts with final (bad) states, computes the "pre-image", and checks whether the start state is in the pre-image. Broadbent et al. [5] recently report that with an optimization based on forward static analysis, a saturation-based model checker `CSHORe` can compete with TRecS [11, 12]. Due to the huge gap as summarized in Table 1, however, the two approaches have been of independent use, and it was difficult to transfer or integrate the techniques.

The present paper fills the gap between the two approaches, and proposes a new model-checking algorithm HorSat and its variation HorSatT for HORS. As indicated in Table 1, the new algorithms are classified somewhere between the two approaches. Like the previous algorithms of Kobayashi and Neatherway et al., HorSat works directly on HORS. Like Broadbent et al.'s saturation algorithm for CPDS [4, 5], however, HorSat deals with *co-trivial* automata, and propagates information *backwards*, starting from final states and iteratively computing the pre-image. We use intersection types to represent the pre-image, but they can equally be interpreted as alternating tree automata accepting (tree representations of) the terms in the pre-image, which are somehow related to the stack automata representation used in Broadbent et al.'s saturation algorithm. We have implemented the new algorithms and obtained promising experimental results.

Besides filling the gap, the main advantage of the new algorithms over the previous algorithms for HORS model checking is the efficiency (both in theory and in practice). Unlike TRecS [11] or TravMC [21], they satisfy the fixed-parameter polynomial time complexity; thus it is expected to scale to large inputs better than TRecS. The previous fixed-parameter polynomial time algorithms GTRecS [14] for HORS did not scale well due to a large constant factor. According to the experiments, the new algorithm clearly outperforms GTRecS.

Compared with the saturation algorithm for CPDS [4, 5], the new algorithms have the following advantages:

– The new algorithms are much easier to understand, implement, and optimize. For example, the saturation algorithm for CPDS [4] uses non-standard automata called stack automata to represent a set of CPDS states, while we can use standard tree automata (or equivalently intersection types) to represent a set of states (which are just applicative terms).

– It is much easier to verify the result of model checking HORS. It is partly because our algorithm works directly on HORS without a detour to CPDS, but also because the variant HorSatT of our algorithm can generate intersection types as a certificate, which is completely compatible with the certificate used by other model checking algorithms [11, 14], and whose validity can be easily checked (by an independent tool) based on the type-based characterization of trivial automata model checking [13].

In Section 2 we recall the nomenclature and foundational results of model checking with intersection types. Section 3 introduces the HorSat algorithm and its application to co-trivial automata model-checking. We then proceed in Section 4 to the variant HorSatT, which can be used in the trivial case and generates a certificate when the HORS is safe. We follow up with some experimental results in Section 5.

2 Preliminaries

In this section, we review HORS, and tree automata (for infinite trees). We then define trivial/co-trivial automata model checking of HORS, and provide their characterizations in terms of intersection types.

We write $dom(f)$ and $codom(f)$ for the domain and co-domain of a map f. A *ranked alphabet*, denoted often by Σ, is a mapping from symbols to their arities. An (unlabeled) *tree* D is a subset of $\{1,\dots,m\}^*$ such that $\epsilon \in D$, and for every $\pi \in \{1,\dots,m\}^*$ and $k \in \{1,\dots,m\}$, $\pi k \in D$ implies $\{\pi\} \cup \{\pi i \mid 1 \le i \le k\} \subseteq D$. For a set S of symbols, an *S-labeled tree* is a map from a tree to S. For a ranked alphabet Σ, a *Σ-labeled ranked tree* T is a $dom(\Sigma)$-labeled tree such that for every $\pi \in dom(T)$, $\{k \mid \pi k \in dom(T)\} = \{k \mid 1 \le k \le \Sigma(T(\pi))\}$.

Next we define *applicative terms*. The set of *sorts* is:

$$\kappa \text{ (sorts)} \ ::= \mathtt{o} \mid \kappa_1 \to \kappa_2$$

Here, the sort $\mathtt{o}$ describes ranked trees. The *order* and *arity* of a sort κ, written $\mathtt{ord}(\kappa)$ and $\mathtt{ar}(\kappa)$ respectively, are:

$$\mathtt{ord}(\mathtt{o}) = 0 \qquad \mathtt{ord}(\kappa_1 \to \kappa_2) = max(\mathtt{ord}(\kappa_1)+1, \mathtt{ord}(\kappa_2))$$
$$\mathtt{ar}(\mathtt{o}) = 0 \qquad \mathtt{ar}(\kappa_1 \to \kappa_2) = \mathtt{ar}(\kappa_2)+1$$

A *sort environment* is a finite map from variables to sorts. The set $\mathbf{ATerms}_{\Gamma,\Sigma,\kappa}$ of *applicative terms* having sort κ under a sort environment $\mathcal{K}$ is defined inductively by: (i) $a \in \mathbf{ATerms}_{\mathcal{K},\Sigma,\underbrace{\mathtt{o}\to\cdots\to\mathtt{o}}_{\Sigma(a)}\to\mathtt{o}}$, (ii) $x \in \mathbf{ATerms}_{\mathcal{K},\Sigma,\kappa}$ if $\mathcal{K}(x) = \kappa$, and (iii) $t_1 t_2 \in \mathbf{ATerms}_{\mathcal{K},\Sigma,\kappa}$ if $t_1 \in \mathbf{ATerms}_{\mathcal{K},\Sigma,\kappa'\to\kappa}$ and $t_2 \in \mathbf{ATerms}_{\mathcal{K},\Sigma,\kappa'}$ for some κ'.

▶ **Definition 1** (HORS). A (deterministic) higher-order recursion scheme (HORS), written $\mathcal{G}$, is a quadruple $(\Sigma, \mathcal{N}, \mathcal{R}, S)$, where

1. Σ is a *ranked alphabet*. The elements of $dom(\Sigma)$ are called *terminals*.
2. $\mathcal{N}$ is a map from a finite set of symbols called *non-terminals* to sorts. $dom(\Sigma)$ and $dom(\mathcal{N})$ must be disjoint.

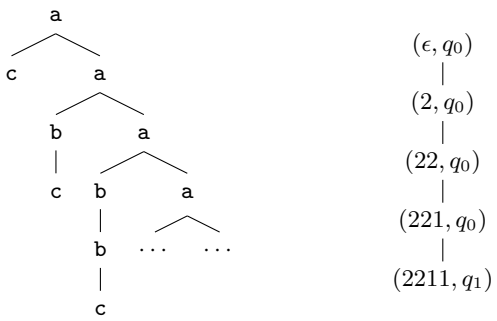


Figure 1 $\mathbf{Tree}(\mathcal{G}_1)$ (left) and a run-tree of $\mathcal{A}_1$ over $\mathbf{Tree}(\mathcal{G}_1)$ (right).

3. $\mathcal{R}$ is a map from the set of non-terminals (i.e. $dom(\mathcal{N})$) to terms of the form $\lambda x_1.\cdots\lambda x_\ell.t$, where t is an applicative term. For each $F \in dom(\mathcal{N})$, if $\mathcal{R}(F) = \lambda x_1.\cdots\lambda x_\ell.t$, then $\mathcal{N}(F)$ must be of the form $\kappa_1 \to \cdots \to \kappa_\ell \to \mathtt{o}$, and $t \in \mathbf{ATerms}_{\mathcal{N}\cup\{x_1\mapsto\kappa_1,\ldots,x_\ell\mapsto\kappa_\ell\},\Sigma,\mathtt{o}}$.
4. S is a non-terminal called the *start symbol*. We require that $S \in dom(\mathcal{N})$ and $\mathcal{N}(S) = \mathtt{o}$.

The *order* of a non-terminal F, written $\mathtt{ord}(F)$, is the order of its sort, i.e. $\mathtt{ord}(\mathcal{N}(F))$. The *order* of a HORS $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$, written $\mathtt{ord}(\mathcal{G})$, is the highest order of its non-terminals.

Intuitively, a HORS $\mathcal{G} = (\Sigma, \mathcal{N}, \{F_1 \mapsto t_1, \ldots, F_k \mapsto t_k\}, F_1)$ is a tree-generating, simply-typed call-by-name functional program, given by the recursive function definitions $F_1 = t_1, \ldots, F_k = t_k$ with the main function F_1 and the tree constructors Σ. We sometimes write $\Sigma_\mathcal{G}, \mathcal{N}_\mathcal{G}, \mathcal{R}_\mathcal{G}, S_\mathcal{G}$ for the four components of $\mathcal{G}$. The reduction relation $\longrightarrow_\mathcal{R}$ on terms of sort $\mathtt{o}$ is defined by:

$$\begin{array}{ll} F\,t_1\,\cdots\,t_k \longrightarrow_\mathcal{R} [t_1/x_1,\ldots,t_k/x_k]t & \text{if } \mathcal{R}(F) = \lambda x_1.\cdots\lambda x_k.t \\ a\,t_1,\cdots\,t_i\,\cdots\,t_k \longrightarrow_\mathcal{R} a\,t_1,\cdots\,t_i'\,\cdots\,t_k & \text{if } t_i \longrightarrow_\mathcal{R} t_i' \end{array}$$

Here, $[t_1/x_1,\ldots,t_k/x_k]t$ denotes the term obtained from t by replacing all the (free) occurrences of $x_1,\ldots,x_k$ with $t_1,\ldots,t_k$ respectively. When $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$, we also write $s \longrightarrow_\mathcal{G} t$ for $s \longrightarrow_\mathcal{R} t$.

For an applicative term t of sort $\mathtt{o}$, the $(\Sigma \cup \{\bot \mapsto 0\})$-labeled tree $t^\bot$ (in the term representation) is defined by: (i) $(a\,t_1\,\cdots\,t_k)^\bot = a\,t_1^\bot\,\cdots\,t_k^\bot$ and (ii) $(F\,t_1\,\cdots\,t_k)^\bot = \bot$. The value tree [22] of $\mathcal{G}$, written $\mathbf{Tree}(\mathcal{G})$, is the $\Sigma \cup \{\bot \mapsto 0\})$-labeled ranked tree obtained as the least upper bound of the set $\{t^\bot \mid S \longrightarrow_\mathcal{G} t\}$ with respect to the least precongruence $\sqsubseteq$ that satisfies $\bot \sqsubseteq T$ for every tree T.

▶ **Example 2.** Consider $\mathcal{G}_1 = (\Sigma = \{\mathtt{a} \mapsto 2, \mathtt{b} \mapsto 1, \mathtt{c} \mapsto 0\}, \mathcal{N}, \mathcal{R}, S)$ where:

$$\mathcal{N} = \{S \mapsto \mathtt{o}, F \mapsto \mathtt{o} \to \mathtt{o}\} \quad \mathcal{R} = \{S \mapsto F\,\mathtt{c}, F \mapsto \lambda x.\mathtt{a}\,x\,(F(\mathtt{b}\,x))\}$$

S is reduced as follows, generating the tree in Figure 1.

$$S \longrightarrow_{\mathcal{G}_1} F\,\mathtt{c} \longrightarrow_{\mathcal{G}_1} \mathtt{a}\,\mathtt{c}\,(F(\mathtt{b}\,\mathtt{c})) \longrightarrow_{\mathcal{G}_1} \cdots$$ ◀

Next, we introduce tree automata, which are used for describing properties on trees.

▶ **Definition 3** (trivial/co-trivial ATA). An *alternating tree automaton* (ATA) is a quadruple $\mathcal{A} = (\Sigma, Q, \Delta, q_I)$, where Σ is a ranked alphabet, Q is a set of states, $q_I \in Q$ is the initial state, and $\Delta \subset Q \times dom(\Sigma) \times 2^{\{1,\ldots,m\}\times Q}$ is a transition function where m is the largest arity in Σ. We require that if $(q, a, S) \in \Delta$, then $S \subseteq \{1, \ldots, \Sigma(a)\} \times Q$. For a (possibly infinite) Σ-labeled ranked tree T, a $(dom(T) \times Q)$-labeled tree R is a *run-tree* of $\mathcal{A}$ over T if (i) $R(\epsilon) = (\epsilon, q_I)$, and (ii) if $R(\pi) = (\pi', q)$, then there exists $S = \{(j_1, q_{k_1}), \ldots, (j_\ell, q_{k_\ell})\}$ such that $(q, T(\pi'), S) \in \Delta$ and $\{i \mid \pi i \in dom(R)\} = \{1, \ldots, \ell\}$ with $R(\pi i) = (\pi' j_i, q_{k_i})$ for each $i \in \{1, \ldots, \ell\}$. A (possibly infinite) Σ-labeled ranked tree T is *accepted by* $\mathcal{A}$ *in trivial mode* if there is a (possibly infinite) run-tree of $\mathcal{A}$ over T, and T is accepted by $\mathcal{A}$ *in co-trivial mode* if there is a *finite* run-tree of $\mathcal{A}$ over T. An ATA is called *trivial* (resp. *co-trivial*) if it accepts input trees in trivial (resp. co-trivial) mode. For an ATA $\mathcal{A} = (\Sigma, Q, \Delta, q_I)$ (with $\perp \notin dom(\Sigma)$), we write $\mathcal{A}^\top$ for the ATA $(\Sigma \cup \{\perp \mapsto 0\}, Q, \Delta \cup \{(q, \perp, \emptyset) \mid q \in Q\}, q_I)$, and $\mathcal{A}^\perp$ for $(\Sigma \cup \{\perp \mapsto 0\}, Q, \Delta, q_I)$,

▶ **Example 4.** Let $\mathcal{A}_1 = (\Sigma, \{q_0, q_1\}, \Delta_1, q_0)$ where $\Sigma = \{\mathtt{a} \mapsto 2, \mathtt{b} \mapsto 1, \mathtt{c} \mapsto 0\}$ and

$$\Delta_1 = \{(q_0, \mathtt{a}, \{(1, q_0)\}), (q_0, \mathtt{a}, \{(2, q_0)\}), (q_0, \mathtt{b}, \{(1, q_1)\}),\\ (q_1, \mathtt{b}, \emptyset), (q_1, \mathtt{a}, \{(1, q_0)\}), (q_1, \mathtt{a}, \{(2, q_0)\})\}.$$

In the co-trivial mode, $\mathcal{A}_1$ accepts trees that have a path containing two consecutive occurrences of $\mathtt{b}$. Figure 1 shows a run-tree of $\mathcal{A}_1$ over $\mathbf{Tree}(\mathcal{G}_1)$. In the trivial mode, $\mathcal{A}_1$ additionally accepts trees having an infinite path. ◀

▶ **Remark 5.** *A trivial (co-trivial) ATA is a special case of alternating parity tree automaton [8], where all the states have priority* 0 *(resp.* 1*). Therefore, from a trivial automaton* $\mathcal{A}$*, one can construct a co-trivial automaton* $\overline{\mathcal{A}}$ *that accepts the complement of the trees accepted by* $\mathcal{A}$*, and vice versa.*

▶ **Example 6.** Recall ATA $\mathcal{A}_1$ in Example 4. A tree is accepted by $\mathcal{A}_1$ in the co-trivial (resp. trivial) mode if and only if it is not accepted by the following ATA $\overline{\mathcal{A}_1} = (\Sigma, \{\overline{q_0}, \overline{q_1}\}, \overline{\Delta_1}, \overline{q_0})$ in the trivial (resp. co-trivial) mode.

$$\overline{\Delta_1} = \{(\overline{q_0}, \mathtt{a}, \{(1, \overline{q_0}), (2, \overline{q_0})\}), (\overline{q_0}, \mathtt{b}, \{(1, \overline{q_1})\}), (\overline{q_0}, \mathtt{c}, \emptyset), (\overline{q_1}, \mathtt{a}, \{(1, \overline{q_0}), (2, \overline{q_0})\}), (\overline{q_1}, \mathtt{c}, \emptyset)\}$$

In the present paper, we are interested in the following model checking problems.

▶ **Definition 7** (trivial/co-trivial ATA model checking). A trivial (resp. co-trivial) ATA model checking problem for HORS is the decision problem: "Given a HORS $\mathcal{G}$ and an ATA $\mathcal{A}$ as input, is $\mathbf{Tree}(\mathcal{G})$ accepted by $\mathcal{A}^\top$ (resp. $\mathcal{A}^\perp$) in trivial (resp. co-trivial) mode?"

By Remark 5, the decidability of trivial/co-trivial ATA model checking follows immediately from that of alternating parity tree automata (APT) model checking for HORS [22], and a trivial ATA model checking problem can be reduced to a co-trivial ATA model checking, and vice versa.

Following Kobayashi and Ong's type systems for HORS [13, 17], we provide below a type-based characterization of trivial/co-trivial ATA model checking for HORS. Fix an ATA $\mathcal{A} = (\Sigma, Q, \Delta, q_I)$. The set of types is given by:

$$\tau \text{ (types)} ::= q \mid \sigma \to \tau \qquad \sigma \text{ (intersections)} ::= \bigwedge\{\tau_1, \ldots, \tau_k\}$$

Intuitively, the type $q \in Q$ describes a tree accepted by $\mathcal{A}$ from the state q (i.e., accepted by (Σ, Q, Δ, q)). The type $\sigma \to \tau$ describes a function that takes an element of (intersection) type

σ as input, and returns an element of type τ. The intersection $\bigwedge\{\tau_1,\ldots,\tau_k\}$ (where k may be 0) describes an element of the intersection of the sets denoted by $\tau_1,\ldots,\tau_k$. When $k=0$, we write $\top$ for $\bigwedge\{\tau_1,\ldots,\tau_k\}$. We often write $\tau_1\wedge\cdots\wedge\tau_k$ or $\bigwedge_{i\in\{1,\ldots,k\}}\tau_i$ for $\bigwedge\{\tau_1,\ldots,\tau_k\}$. We assume that $\bigwedge$ binds tighter than $\to$, so that $q_0\wedge q_1\to q_2$ means $(q_0\wedge q_1)\to q_2$.

A type τ is called a *refinement* of a sort κ, written $\tau :: \kappa$ if $\tau :: \kappa$ is derivable by the following rules:

$$\frac{q\in Q}{q :: o} \qquad \frac{\tau :: \kappa \qquad \tau_i :: \kappa' \text{ for each } i\in I}{(\bigwedge_{i\in I}\tau_i\to\tau) :: \kappa'\to\kappa}$$

A *type environment* is a *set* of type bindings of the form $x:\tau$. For a type environment Γ, We write $dom(\Gamma)$ for the set $\{x \mid x:\tau\in\Gamma\}$. Unlike ordinary type systems, a type environment may contain multiple type bindings for the variable, like $\{x:q_0, x:q_1\}$. We sometimes omit curly brackets and just write $x_1:\tau_1,\ldots,x_n:\tau_n$ for $\{x_1:\tau_1,\ldots,x_n:\tau_n\}$. When $\sigma=\bigwedge\{\tau_1,\ldots,\tau_n\}$, we also write $x:\sigma$ for $x:\tau_1,\ldots,x:\tau_n$.

The rules for a type judgment $\Gamma\vdash_{\mathcal{A},\mathcal{G}} t:\tau$ are:

$$\frac{}{\Gamma\cup\{x:\tau\}\vdash_{\mathcal{A},\mathcal{G}} x:\tau}\ (\text{T-Var}) \qquad \frac{(q,a,\{(i,q_j)\mid 1\le i\le\Sigma(a), j\in I_i\})\in\Delta_{\mathcal{A}}}{\Gamma\vdash_{\mathcal{A},\mathcal{G}} a:\bigwedge_{j\in I_1}q_j\to\cdots\to\cdots\bigwedge_{j\in I_{\Sigma(a)}}q_j\to q}\ (\text{T-Con})$$

$$\frac{\Gamma\vdash_{\mathcal{A},\mathcal{G}} t_1:\bigwedge_{i\in I}\tau_i\to\tau \quad \Gamma\vdash_{\mathcal{A},\mathcal{G}} t_2:\tau_i \text{ (for each } i\in I)}{\Gamma\vdash_{\mathcal{A},\mathcal{G}} t_1t_2:\tau}\ (\text{T-App}) \qquad \frac{\{x_i:\tau_j\mid 1\le i\le k, j\in I_i\}\vdash_{\mathcal{A},\mathcal{G}} t:q \quad \mathcal{R}_{\mathcal{G}}(F)=\lambda x_1.\cdots\lambda x_k.t \quad (\bigwedge_{j\in I_1}\tau_j\to\cdots\to\bigwedge_{j\in I_k}\tau_j\to q)::\mathcal{N}_{\mathcal{G}}(F)}{\Gamma\vdash_{\mathcal{A},\mathcal{G}} F:\bigwedge_{j\in I_1}\tau_j\to\cdots\to\bigwedge_{j\in I_k}\tau_j\to q}\ (\text{T-NT})$$

We sometimes omit the subscripts $\mathcal{A}$ and $\mathcal{G}$ when they are clear from the context.

The following are special cases of the soundness and completeness of Kobayashi and Ong's type system for APT model checking [17] (where the priorities are restricted to 0 and 1 respectively for Clauses (i) and (ii) of Theorem 8.[1]

▶ **Theorem 8.** *(i)* $\mathbf{Tree}(\mathcal{G})$ *is accepted by* $\mathcal{A}^\top$ *in the trivial mode if and only if there is a* possibly infinite *derivation tree for* $\emptyset\vdash_{\mathcal{A},\mathcal{G}} S:q_I$. *(ii)* $\mathbf{Tree}(\mathcal{G})$ *is accepted by* $\mathcal{A}^\bot$ *in the co-trivial mode if and only if there is a finite derivation tree for* $\emptyset\vdash_{\mathcal{A},\mathcal{G}} S:q_I$.

The existing practical[2] model checking algorithms for HORS [11, 14, 21] deal with trivial ATA model checking, and try to construct an infinite derivation tree for $\emptyset\vdash_{\mathcal{A},\mathcal{G}} S:q_I$ [21], or infer a set of types of non-terminals occurring in such a derivation tree [11, 14]. All of the algorithms run in the *forward* direction, in the sense that they start with $S:q_I$, and expand non-terminals to construct types/derivation trees while checking that invalid trees cannot be generated from S.

3 Co-Trivial ATA Model Checking

This section presents a co-trivial ATA model checking algorithm that runs in the backward manner. We first note the following property (see Appendix B for a proof).

[1] Kobayashi and Ong [17] do not consider HORS's that generate trees containing $\bot$; see Section A in Appendix on how to derive the result below for $\bot$-generating HORS.

[2] Since the worst-case complexity of trivial/co-trivial ATA model checking is n-EXPTIME complete [22, 18], we call a model checking algorithm *practical* if it terminates in a realistic time (say, in a few minutes, rather than in several years) for typical inputs.

▶ **Lemma 9.** *Let $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$ be a HORS. $\mathbf{Tree}(\mathcal{G})$ is accepted by a co-trivial ATA $\mathcal{A}^{\perp}$ if and only if there exists a term t such that $S \longrightarrow_{\mathcal{G}}^{*} t$ and $t^{\perp}$ is accepted by $\mathcal{A}^{\perp}$.*

By the lemma above, for co-trivial ATA model checking, it suffices to start with the set of terms $\mathcal{T}_0 = \{t \in \mathbf{ATerms}_{\mathcal{N},\Sigma,\mathsf{o}} \mid t^{\perp} \text{ is accepted by } \mathcal{A}^{\perp} \text{ in the co-trivial mode}\}$, expand it to the set $\mathcal{T}$ of all the terms that can be reduced to a term in $\mathcal{T}_0$, and check whether $S \in \mathcal{T}$. Since $\mathcal{T}$ may be *infinite* in general, we represent a (possibly infinite) set of terms by using a *finite* type environment Γ for non-terminals. We write $\mathbf{Terms}_{\Gamma}$ for the set $\{t \mid \Gamma \vdash^{-}_{\mathcal{A}} t : q_I\}$ of terms. Here, $\Gamma \vdash^{-}_{\mathcal{A}} t : \tau$ means that there is a finite derivation for $\Gamma \vdash_{\mathcal{A},\mathcal{G}} t : \tau$ that does not use rule T-NT (so that $\mathcal{G}$ does not matter), where non-terminals are treated as variables. Then, the initial set $\mathcal{T}_0$ above is represented by the empty type environment $\emptyset$, i.e., $\mathbf{Terms}_{\emptyset} = \mathcal{T}_0$, as stated by the following lemma: see Appendix B for a proof.

▶ **Lemma 10.** *$\emptyset \vdash^{-}_{\mathcal{A}} t : q_I$ if and only if $t^{\perp}$ is accepted by $\mathcal{A}^{\perp}$ in the co-trivial mode.*

We shall construct below a monotonic function $\mathcal{F}$ on type environments that satisfies the following conditions:

(I) $\{t \in \mathbf{ATerms}_{\mathcal{N},\Sigma,\mathsf{o}} \mid \exists t'.t \longrightarrow_{\mathcal{G}} t' \in \mathbf{Terms}_{\Gamma}\} \subseteq \mathbf{Terms}_{\mathcal{F}(\Gamma)}$; and
(II) $\mathbf{Terms}_{\mathcal{F}^i(\emptyset)} \subseteq \{t \mid \exists t'.t \longrightarrow_{\mathcal{G}}^{*} t' \in \mathcal{T}_0\}$ for every i.

Then, we have $\mathbf{Terms}_{\bigcup_{i\in\omega} \mathcal{F}^i(\emptyset)} = \mathcal{T}$. Since $\mathcal{F}$ is monotonic and a type environment ranges over a finite set $\{\Gamma \mid dom(\Gamma) \subseteq dom(\mathcal{N}) \text{ and } \forall F : \tau \in \Gamma.\tau :: \mathcal{N}(F)\}$, we can obtain $\bigcup_{i\in\omega} \mathcal{F}^i(\emptyset)$ by computing $\mathcal{F}(\emptyset)$, $\mathcal{F}^2(\emptyset)$, $\mathcal{F}^3(\emptyset), \ldots$ until it converges; we can then check whether $S : q_I \in \bigcup_{i\in\omega} \mathcal{F}^i(\emptyset)$ holds to decide whether $\mathbf{Tree}(\mathcal{G})$ is accepted by $\mathcal{A}^{\perp}$ in the co-trivial mode, as stated below.

▶ **Lemma 11.** *Suppose that a monotonic function $\mathcal{F}$ on type environments satisfies the two conditions above. Then, $S : q_I \in \bigcup_{i\in\omega} \mathcal{F}^i(\emptyset)$ if and only if $\mathbf{Tree}(\mathcal{G})$ is accepted by $\mathcal{A}$.*

Proof. By the first condition of $\mathcal{F}$, we have $\{t \mid \exists t'.t \longrightarrow_{\mathcal{G}}^{i} t' \in \mathbf{Terms}_{\emptyset}\} \subseteq \mathbf{Terms}_{\mathcal{F}^i(\emptyset)}$. Thus, we have $\mathcal{T} = \mathbf{Terms}_{\bigcup_{i\in\omega} \mathcal{F}^i(\emptyset)}$. The required result follows by Lemma 9. ◀

It remains to construct $\mathcal{F}$ that satisfies the conditions (I) and (II) above. The following lemma provides a clue as to how to construct $\mathcal{F}$. It states that typing is closed under the inverse of substitutions: see Section B for a proof.

▶ **Lemma 12.** *Suppose $s \in \mathbf{ATerms}_{\mathcal{K}\cup\{x_1:\kappa_1,\ldots,x_\ell:\kappa_\ell\},\Sigma,\mathsf{o}}$ and $t_i \in \mathbf{ATerms}_{\mathcal{K},\Sigma,\kappa_i}$ for each $i \in \{1,\ldots,\ell\}$. If $\Gamma \vdash^{-}_{\mathcal{A}} [t_1/x_1,\ldots,t_\ell/x_\ell]s : q$ with $\Gamma :: \mathcal{K}$, then there exist (possibly empty) sets I_i and $\{\tau_j \mid j \in I_i\}$ for each $i \in \{1,\ldots,\ell\}$ such that: (i) $\Gamma \cup \{x_i : \tau_j \mid i \in \{1,\ldots,\ell\}, j \in I_i\} \vdash^{-}_{\mathcal{A}} s : q$; (ii) $\Gamma \vdash^{-}_{\mathcal{A}} t_i : \tau_j$ for each $i \in \{1,\ldots,\ell\}$ and $j \in I_i$; and (iii) $\tau_j :: \kappa_i$ for every $j \in I_i$.*

The above lemma implies that if $F\,t_1 \cdots t_\ell \longrightarrow_{\mathcal{G}} [t_1/x_1,\ldots,t_\ell/x_\ell]s$ and $\Gamma \vdash^{-}_{\mathcal{A}} [t_1/x_1,\ldots,t_\ell/x_\ell]s : q$, then $\Gamma \cup \{F : \bigwedge_{j\in I_1} \tau_1 \to \cdots \to \bigwedge_{j\in I_\ell} \tau_j \to q\} \vdash_{\mathcal{A}} F\,t_1 \cdots t_\ell : q$ holds, where I_i and τ_j are as given by the lemma above. This motivates us to define $\mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}$ (where $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{G}}$) as follows.

$$\mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}(\Gamma) = \Gamma \cup \{F : \Gamma'(x_1) \to \cdots \to \Gamma'(x_\ell) \to q \mid \mathcal{R}(F) = \lambda x_1.\cdots\lambda x_\ell.t, \\ \mathcal{N}(F) = \kappa_1 \to \cdots \to \kappa_\ell \to \mathsf{o}, \text{ and } x_1 : \kappa_1, \ldots, x_\ell : \kappa_\ell; \Gamma \vdash_{\mathcal{A}} t : q \Rightarrow \Gamma'\}.$$

Here, $\Gamma(x)$ is an abbreviation of $\bigwedge\{\tau \mid x : \tau \in \Gamma\}$. The relation $\mathcal{K}; \Gamma_N \vdash_{\mathcal{A}} t : \tau \Rightarrow \Gamma_V$ (where Γ_N and Γ_V are meta-variables for type environments on non-terminals and variables respectively) is defined by:

Γ := $\emptyset$;
while not($\mathcal{F}_{\mathcal{G},\mathcal{A}}(\Gamma)$=$\Gamma$) do ($\Gamma$:= $\mathcal{F}_{\mathcal{G},\mathcal{A}}(\Gamma)$; if $S : q_I \in \Gamma$ then return TRUE);
return FALSE

Figure 2 Co-trivial ATA model checking algorithm HORSAT.

$$\frac{x : \tau \in \mathit{Inhabited}(\mathcal{K}, \Gamma_N)}{\mathcal{K}; \Gamma_N \vdash_{\mathcal{A}} x : \tau \Rightarrow x : \tau} \qquad \frac{\emptyset \vdash_{\mathcal{A},\mathcal{G}} a : \tau}{\mathcal{K}; \Gamma_N \vdash_{\mathcal{A}} a : \tau \Rightarrow \emptyset} \qquad \frac{}{\mathcal{K}; \Gamma_N \cup \{F : \tau\} \vdash_{\mathcal{A}} F : \tau \Rightarrow \emptyset}$$

$$\frac{\mathcal{K}; \Gamma_N \vdash_{\mathcal{A}} t_1 : \bigwedge_{i \in I} \tau_i \to \tau \Rightarrow \Gamma_0 \qquad \mathcal{K}; \Gamma_N \vdash_{\mathcal{A}} t_2 : \tau_i \Rightarrow \Gamma_i \text{ (for each } i \in I) \qquad \Gamma_0 \cup \bigcup_{i \in I} \Gamma_i \in \mathit{Inhabited}(\mathcal{K}, \Gamma)}{\mathcal{K}; \Gamma_N \vdash_{\mathcal{A}} t_1 t_2 : \tau \Rightarrow \Gamma_0 \cup \bigcup_{i \in I} \Gamma_i}$$

Here, $\mathit{Inhabited}(\mathcal{K}, \Gamma_N)$ is the set of type environments for which the corresponding term environments can be constructed from non-terminals in Γ_N, i.e.:

$$\{\Gamma \mid \Gamma :: \mathcal{K} \text{ and } \forall x \in \mathit{dom}(\Gamma). \exists s \in \mathbf{ATerms}_{\mathcal{N},\Sigma,\mathcal{K}(x)}. \forall x : \tau \in \Gamma. \Gamma_N \vdash^{-}_{\mathcal{A}} s : \tau\}.$$

The relation $\mathcal{K}; \Gamma_N \vdash_{\mathcal{A}} t : \tau \Rightarrow \Gamma_V$ intuitively means that $\Gamma_N, \Gamma_V \vdash t : \tau$ holds and that Γ_V is inhabited, as stated by the following lemma (which follows by straightforward induction).

▶ **Lemma 13.** *If $\mathcal{K}; \Gamma_N \vdash t : \tau \Rightarrow \Gamma_V$, then $\Gamma_N \cup \Gamma_V \vdash t : \tau$ and $\Gamma_V \in \mathit{Inhabited}(\mathcal{K}, \Gamma_N)$. Conversely, if $\Gamma_N \cup \Gamma_V \vdash^{-}_{\mathcal{A}} t : \tau$ and $\Gamma_V \in \mathit{Inhabited}(\mathcal{K}, \Gamma_N)$, then $\mathcal{K}; \Gamma_N \vdash t : \tau \Rightarrow \Gamma'_V$ for some Γ'_V such that $\Gamma'_V \subseteq \Gamma_V$.*

Reading the rules for $\mathcal{K}; \Gamma_N \vdash_{\mathcal{A}} t : \tau \Rightarrow \Gamma_V$ in a bottom-up manner, we can interpret them as an algorithm which, given $\mathcal{K}, \Gamma_N$ and τ as input, outputs Γ_V such that $\Gamma_N, \Gamma_V \vdash^{-}_{\mathcal{A}} t : \tau$ and $\Gamma_V \in \mathit{Inhabited}(\mathcal{K}, \Gamma_N)$. For checking the inhabitance condition $\Gamma \in \mathit{Inhabited}(\mathcal{K}, \Gamma_N)$, we can use Rehof and Urzyczyn's reduction to the emptiness problem of alternating tree automata [25]: see Appendix C.

We write $\mathcal{F}_{\mathcal{G},\mathcal{A}}$ for $\mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}_{\mathcal{G}}}$, and also omit the subscripts when they are clear from the context. The following lemma justifies the definition of $\mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}$.

▶ **Lemma 14.** *Suppose $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}', S)$ and $\mathcal{R} \subseteq \mathcal{R}'$. If $\Gamma \vdash^{-}_{\mathcal{A}} t : q$ and $s \longrightarrow_{\mathcal{R}} t$, then $\mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}(\Gamma) \vdash^{-}_{\mathcal{A}} s : q$.*

Proof. The proof proceeds by induction on the derivation of $t \longrightarrow_{\mathcal{R}} t'$. Since the induction step is trivial, we show only the base case, where $t = F\, t_1 \cdots t_\ell$ and $t' = [t_1/x_1, \ldots, t_\ell/x_\ell]s$ with $\lambda x_1. \cdots \lambda x_\ell. s = \mathcal{R}(F)$. Let $\mathcal{N}(F) = \kappa_1 \to \cdots \to \kappa_\ell \to \mathsf{o}$. By Lemma 12, we have:

$$\Gamma \cup \{x_i : \tau_j \mid i \in \{1, \ldots, \ell\}, j \in I_i\} \vdash^{-}_{\mathcal{A}} s : q \qquad \Gamma \vdash^{-}_{\mathcal{A}} t_i : \tau_j \text{ for each } i \in \{1, \ldots, \ell\}, j \in I_i$$

and $\tau_j :: \kappa_i$ for every $j \in I_i$. By Lemma 13, there exists Γ_V such that

$$\{x_1 : \kappa_1, \ldots, x_\ell : \kappa_\ell\}; \Gamma \vdash^{-}_{\mathcal{A}} s : q \Rightarrow \Gamma_V \qquad \Gamma \vdash^{-}_{\mathcal{A}} t_i : \tau \text{ for each } x_i : \tau \in \Gamma_V.$$

Thus, we have $F : \Gamma_V(x_1) \to \cdots \to \Gamma_V(x_\ell) \to q \in \mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}(\Gamma)$, which implies $\mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}(\Gamma) \vdash^{-}_{\mathcal{A}} F\, t_1 \cdots t_\ell : q$ as required. ◀

The whole algorithm is given in Figure 2. The following theorem guarantees the soundness and completeness of the algorithm.

▶ **Theorem 15.** *Let the function $\mathcal{F}_{\mathcal{G},\mathcal{A}}$ be as defined above. Then, $S : q_I \in \bigcup_{i\in\omega} \mathcal{F}^i_{\mathcal{G},\mathcal{A}}(\emptyset)$ if and only if* $\mathbf{Tree}(\mathcal{G})$ *is accepted by $\mathcal{A}^\perp$ in the co-trivial mode.*

Proof. By Lemma 11, it suffices to show that $\mathcal{F} = \mathcal{F}_{\mathcal{G},\mathcal{A}}$ satisfies the two conditions (I) and (II). Condition (I) follows immediately from Lemma 14.

To check condition (II), suppose $t \in \mathbf{Terms}_{\mathcal{F}^m(\emptyset)}$, i.e., $\mathcal{F}^m(\emptyset) \vdash^-_{\mathcal{A}} t : q_I$. We construct the following HORS $\mathcal{G}^{(m)}$ as an approximation of $\mathcal{G}$:

$$\mathcal{G}^{(m)} = (\Sigma \cup \{\perp \mapsto 0\}, \mathcal{N}^{(m)}, \mathcal{R}^{(m)}, F_1^{(m)}) \quad \mathcal{N}^{(m)} = \{F_i^{(j)} \mid i \in \{1,\dots,n\}, j \in \{0,\dots,m\}\}$$
$$\mathcal{R}^{(m)} = \{F_i^{(0)} \mapsto \lambda x_1.\cdots\lambda x_\ell.\perp \mid \mathtt{ar}(\mathcal{N}(F_i)) = \ell\}$$
$$\cup\{F_i^{(j)} \mapsto [F_1^{(j-1)}/F_1,\dots,F_n^{(j-1)}/F_n]\mathcal{R}(F_i) \mid j \geq 1\}$$

$\mathcal{G}^{(m)}$ is a HORS without recursion, obtained by unfolding non-terminals of $\mathcal{G}$. We write $t^{(m)}$ for the term obtained by replacing each non-terminal F in t with $F^{(m)}$. By the assumption $\mathcal{F}^m(\emptyset) \vdash^-_{\mathcal{A}} t : q_I$, we have $\emptyset \vdash_{\mathcal{A}^\perp,\mathcal{G}^{(m)}} t^{(m)} : q_I$: see Lemma 27 in Appendix B. By the strong normalization of the simply-typed λ-calculus, we have $t^{(m)} \longrightarrow^*_{\mathcal{G}^{(m)}} u$ for some $(\Sigma\cup\{\perp \mapsto 0\})$-labeled tree (in the term representation) u. By the type preservation property (Lemma 29 in Appendix B), we have $\emptyset \vdash_{\mathcal{A}^\perp,\mathcal{G}^{(m)}} u : q_I$. By Lemma 10, u is accepted by $\mathcal{A}^\perp$. By the construction of $t^{(m)}$, there exists a term t' such that $t \longrightarrow^*_{\mathcal{G}} t'$ and $t'^\perp = u$. Thus, we have $t \in \{s \mid \exists t'.s \longrightarrow^*_{\mathcal{G}} t' \in \mathcal{T}_0\}$ as required. ◀

▶ **Example 16.** Let $\mathcal{G}$ be $\mathcal{G}_1$ in Example 2 and $\mathcal{A}$ be $\mathcal{A}_1$ in Example 4. Then, since $\emptyset \vdash_{\mathcal{A}} \mathtt{a}\,x\,(F(\mathtt{b}\,x)) : q_0 \Rightarrow x : q_0$ and $\emptyset \vdash_{\mathcal{A}} \mathtt{a}\,x\,(F(\mathtt{b}\,x)) : q_1 \Rightarrow x : q_0$, we have:

$$\mathcal{F}(\emptyset) = \{F : q_0 \to q_0, F : q_0 \to q_1\}.$$

Similarly, $\mathcal{F}^2(\emptyset), \mathcal{F}^3(\emptyset), \dots$ are computed as follows.

$$\mathcal{F}^2(\emptyset) = \mathcal{F}(\emptyset) \cup \{F : q_1 \to q_0, F : q_1 \to q_1\} \qquad \mathcal{F}^3(\emptyset) = \mathcal{F}^2(\emptyset) \cup \{F : \top \to q_0, F : \top \to q_1\}$$
$$\mathcal{F}^4(\emptyset) = \mathcal{F}^3(\emptyset) \cup \{S : q_0, S : q_1\} = \mathcal{F}^5(\emptyset)$$

Thus, $\bigcup_{i\in\omega} \mathcal{F}^i(\emptyset) = \mathcal{F}^4(\emptyset)$ contains $S : q_0$, which implies that $\mathbf{Tree}(\mathcal{G}_1)$ is accepted by $\mathcal{A}_1$ in the co-trivial mode. ◀

▶ **Remark 17.** *The inhabitance check needed for computing $\mathcal{F}(\Gamma)$ can be quite costly: in fact, the inhabitance problem is EXPTIME-complete [25]. In order to avoid the drawback, we can actually replace $\mathit{Inhabited}(\mathcal{K}, \Gamma_N)$ in the rules for the relation $\mathcal{K}; \Gamma_N \vdash_{\mathcal{A}} t : \tau \Rightarrow \Gamma_V$ with an over-approximation $\mathit{Inhabited}'(\mathcal{K}, \Gamma_N)$, such that*

$$\mathit{Inhabited}(\mathcal{K}, \Gamma_N) \subseteq \mathit{Inhabited}'(\mathcal{K}, \Gamma_N) \subseteq \{\Gamma \mid \Gamma :: \mathcal{K}\}.$$

By the proof of Theorem 15, our co-trivial model checking algorithm remains sound and complete for any such over-approximation.

▶ **Remark 18.** *Like Kobayashi's GTRecS algorithm [14], the co-trivial ATA model checking algorithm above runs in time polynomial in the size of HORS under the assumption that the other parameters (the size of the co-trivial automaton, the order of HORS, and the largest arity of non-terminals in HORS) are fixed. First, the number of iterations is also linear in the size of HORS since the largest size of type environments is linear in the size of HORS (under the fixed-parameter assumption above). The cost for computing $\mathcal{F}(\Gamma)$ is also linear in the size of HORS, as the inhabitance check can be performed in a constant time (again, under the fixed-parameter assumption above): see Appendix C.*

4 Trivial ATA Model Checking

This section presents a saturation-based algorithm for *trivial* ATA model checking. *Co-trivial* model checking is more natural for the saturation-based method, but many typical program verification problems are reduced to trivial ATA model checking problems for HORS [13, 11, 19], so that a program is safe if and only if the tree generated by a corresponding HORS is accepted by a trivial ATA. Although trivial ATA model checking can be reduced to co-trivial model checking by negating the trivial ATA (so that the co-trivial ATA accepts "error configurations" of a program), that approach is not good at *certifying* that a co-trivial property is *not* satisfied. When a co-trivial property is satisfied, then based on Theorem 8, one can generate a derivation tree for $\emptyset \vdash_{\mathcal{A},\mathcal{G}} S : q_I$ or the types of non-terminals required in the derivation tree as a certificate. When a co-trivial property is not satisfied, however, the possible witness is a fixedpoint of $\mathcal{F}$, which can be very large. It is also difficult to validate the witness independently of the model checking algorithm, especially when the saturation algorithm is combined with other static analyses as in [5] and our implementation reported in Section 5. This is in contrast with the existing trivial automata model checking algorithms for HORS [11, 14, 21], which can output the types of non-terminals as a certificate when a property is satisfied. Checking the certificates amounts to type checking for an intersection type system, which can be performed independently of the model checking algorithms. This section modifies the saturation-based algorithm so that it can deal with trivial model checking directly (rather than negating the property) and generate certificates.

We first clarify the notion of "certificates". Define the function $\mathit{Shrink}_{\mathcal{G}}$ on type environments by:

$$\mathit{Shrink}_{\mathcal{G}}(\Gamma) = \{F : \sigma_1 \to \cdots \to \sigma_k \to q \in \Gamma \mid \mathcal{R}(F) = \lambda x_1.\cdots\lambda x_\ell.t \text{ and } \Gamma, x_1 : \sigma_1, \ldots, x_\ell : \sigma_\ell \vdash_{\mathcal{A}} t : q\}.$$

Intuitively, $\mathit{Shrink}_{\mathcal{G}}(\Gamma)$ picks each type binding $F : \sigma_1 \to \cdots \to \sigma_k \to q$ in Γ, checks whether it is valid in the sense that the body of F has the same type, and filters out invalid ones. We omit the subscript $\mathcal{G}$ when it is clear from context. We write $\vdash \mathcal{G} : \Gamma$ if $\mathit{Shrink}_{\mathcal{G}}(\Gamma) = \Gamma$, We also write $\Gamma \vdash (\mathcal{G}, t) : \tau$ if $\vdash \mathcal{G} : \Gamma$ and $\Gamma \vdash t : \tau$ hold. The following theorem is due to Kobayashi [13]. It also follows immediately from Theorem 8: see Appendix B.

▶ **Theorem 19.** $\mathbf{Tree}(\mathcal{G})$ *is accepted by* $\mathcal{A}^\top$ *in trivial mode if and only if* $\Gamma \vdash (\mathcal{G}, S) : q_I$ *for some* Γ.

Based on Theorem 19, a type environment Γ such that $\Gamma \vdash (\mathcal{G}, S) : q_I$ serves as a certificate for $\mathbf{Tree}(\mathcal{G})$ being accepted by $\mathcal{A}$.

▶ **Example 20.** Recall $\mathcal{G}_1$ in Example 2. Let $\mathcal{A}_2 = (\{\mathtt{a} \mapsto 2, \mathtt{b} \mapsto 1, \mathtt{c} \mapsto 0\}, \{q_0, q_1\}, \Delta_2, q_0)$ where $\Delta_2 = \{(q_0, \mathtt{a}, \{(1, q_0), (2, q_0)\}), (q_0, \mathtt{b}, \{(1, q_1)\}), (q_0, \mathtt{c}, \emptyset), (q_1, \mathtt{b}, \{(1, q_1)\}), (q_1, \mathtt{c}, \emptyset)\}$. $\mathcal{A}_2$ describes the property that $\mathtt{a}$ cannot occur below $\mathtt{b}$. $\mathbf{Tree}(\mathcal{G}_1)$ is accepted by $\mathcal{A}_2$, and the type environment $\{S : q_0, F : q_0 \wedge q_1 \to q_0\}$ serves as a certificate. ◀

The existing algorithms for trivial automata model checking [11, 14] first compute an overapproximation Γ' of a possible certificate, compute the fixedpoint $\Gamma = \bigcap_{j\in\omega} \mathit{Shrink}^j(\Gamma')$, and check whether $S : q_I \in \Gamma$. We show below that such an overapproximation Γ' can be computed by using $\mathcal{F}$. For that purpose, we just need to replace the initial type environment $\emptyset$ with the type environment $\Gamma_0 = \{F : \underbrace{\top \to \cdots \to \top}_{\mathtt{ar}(\mathcal{N}(F))} \to q \mid F \in \mathit{dom}(\mathcal{N})\}$. $\mathbf{Terms}_{\Gamma_0}$ is the set of terms t such that $t^\perp$ is accepted by $\mathcal{A}^\top$. Thus, for $\Gamma' = \bigcup_{i\in\omega} \mathcal{F}^i(\Gamma_0)$, $\mathbf{Terms}_{\Gamma'}$ is

```
Γ′ := Γ0;
while not(F_{G,A}(Γ′)=Γ′) do
  (Γ′ := F_{G,A}(Γ′); Γ := Γ′; (while not(Shrink(Γ)=Γ) do Γ := Shrink(Γ));
   if S : q_I ∈ Γ then return Γ );
return FALSE
```

Figure 3 Trivial ATA model checking algorithm HorSatT.

the set $\{t \mid t \longrightarrow_{\mathcal{G}}^{*} t' \text{ and } t'^{\perp} \text{ is accepted by } \mathcal{A}^{\top}\}$. The type environment Γ' itself is not a certificate: in fact, *any* term t that has a non-terminal as its head is an element of $\mathbf{Terms}_{\Gamma'}$, since $t^{\perp} = \perp$ and $\perp$ is accepted by $\mathcal{A}^{\top}$. As the following theorem shows, however, Γ' serves as an overapproximation of a possible certificate.

▶ **Theorem 21.** *$S : q_I \in \bigcap_{j\in\omega} \mathit{Shrink}^j(\bigcup_{i\in\omega} \mathcal{F}^i(\Gamma_0))$ if and only if $\mathbf{Tree}(\mathcal{G})$ is accepted by $\mathcal{A}^{\top}$ in the trivial mode.*

Based on the theorem, we obtain the trivial ATA model checking algorithm shown in Figure 3. The outer loop repeatedly computes $\mathcal{F}(\Gamma_0), \mathcal{F}^2(\Gamma_0), \ldots$, until it converges to a fixedpoint, or finds a certificate. The inner loop computes $\Gamma = \bigcap_{j\in\omega} \mathit{Shrink}^j(\mathcal{F}^i(\Gamma_0))$. If $S : q_I \in \Gamma$ then the algorithm terminates and returns Γ as a certificate. Otherwise, the algorithm eventually returns `FALSE` when a fixedpoint of $\mathcal{F}$ is reached but $S : q_I \in \Gamma$ does not hold. Like the co-trivial ATA model checking algorithm in Figure 2, the algorithm terminates as soon as a certificate is found.

▶ **Example 22.** Recall $\mathcal{G}_1$ in Example 2 and $\mathcal{A}_2$ in Example 20. $\Gamma_0, \mathcal{F}(\Gamma_0), \mathcal{F}^2(\Gamma_0), \ldots$ are computed as follows.

$$\Gamma_0 = \{S : q_0, S : q_1, F : \top \to q_0, F : \top \to q_1\}$$
$$\mathcal{F}(\Gamma_0) = \Gamma_0 \cup \{F : q_0 \to q_0\} \qquad \mathcal{F}^2(\Gamma_0) = \mathcal{F}(\Gamma_0) \cup \{F : q_0 \wedge q_1 \to q_0\} = \mathcal{F}^3(\Gamma_0)$$

For $\Gamma' = \mathcal{F}^2(\Gamma_0)$, $\mathit{Shrink}^i(\Gamma')(i = 1, 2, \ldots)$ are:

$$\mathit{Shrink}(\Gamma') = \{S : q_0, S : q_1, F : q_0 \to q_0, F : q_0 \wedge q_1 \to q_0\}$$
$$\mathit{Shrink}^2(\Gamma') = \{S : q_0, F : q_0 \wedge q_1 \to q_0\} = \mathit{Shrink}^3(\Gamma').$$

Since $S : q_0 \in \mathit{Shrink}^2(\Gamma') = \bigcap_{j\in\omega} \mathit{Shrink}^j(\Gamma')$, we can conclude that $\mathbf{Tree}(\mathcal{G})$ is accepted by $\mathcal{A}_2$ in trivial mode.

▶ **Remark 23.** GTRecS *algorithm [14] is obtained by replacing $\mathcal{F}$ in Figure 3 with the function Expand in [14]. The functions Expand and $\mathcal{F}$ are quite different, however; the former uses a* game-semantic *idea [14, 24] to propagate the requirement that S should have type q_I in the* forward *direction, while the latter propagates information backwards using* purely type-based *techniques. Although both algorithms enjoy the fixed-parameter linear time complexity [14], according to experiments (see Section 5),* GTRecS *tends to be much slower. This is attributed to the game-semantics interpretation of intersection types. For example, a function of type $q_1 \to \cdots \to q_n \to q$ may have any type of the form $\sigma_1 \to \cdots \to \sigma_n \to q$ for $\sigma_i \in \{\top, q_i\}$ in the interpretation of [14]; thus the number of possible types blows up.*

5 Experiments

We have implemented a new higher-order model checker incorporating both the co-trivial HorSat and trivial HorSatT model checking algorithms described respectively in Sections 3

and 4. As in [5], we have optimized the algorithms by using forward flow analysis, to exclude irrelevant type bindings; more precisely, the definition of $\mathit{Inhabited}(\mathcal{K}, \Gamma_N)$ in Section 3 has been replaced by

$$\{\Gamma \mid \Gamma :: \mathcal{K} \text{ and } \forall x \in \mathit{dom}(\Gamma).\exists s \in \mathbf{ATerms}_{\mathcal{N},\Sigma,\mathcal{K}(x)} \cap \mathbf{Flow}(x).\forall x : \tau \in \Gamma.\Gamma_N \vdash^{-}_{\mathcal{A}} s : \tau\}.$$

where $\mathbf{Flow}(x)$ is an overapproximation of the terms that may flow to x in a reduction sequence from the start symbol S. We use 0CFA to compute $\mathbf{Flow}(x)$. The implementation was compared to other type-based model checkers for HORS: TRecS [11], GTRecS2 [15] (which is a successor of GTRecS [14]) and TravMC [21], and a saturation-based model checker `CSHORe` [5] for CPDS. Since GTRecS2 has different variants of the algorithm for proving safety properties and their complements (eventually giving up if it cannot prove the property) we only ran the appropriate version of GTRecS2 on each example.

The benchmark suite consists of five categories of inputs (separated by horizontal lines), which have been collected from different applications of trivial automata model checking of HORS [20, 23, 19, 26, 16]: the first one from the HMTT verification tool [20], the second from software model checker MoCHi [19], the third from the PMRS model checker [23], the fourth from exact flow analysis [26], and the fifth from applications to data compression [16]. The inputs have been automatically generated from program verification problems except for those in the fifth category. We have chosen relatively large programs from each category, so the benchmarks represent "hard instances". The benchmarks marked by "(neg)" expect the output of model checking to be "No"; "Yes" is expected for the others. Some tools such as TRecS and TravMC can take certain extensions of HORS as input; benchmarks with such extensions were reformulated as a pure HORS in every case, which might result in a longer run-time than on the original. The implementation and benchmarks are available at `http://www-kb.is.s.u-tokyo.ac.jp/~koba/horsat/`.

We ran the experiments on an Acer Aspire TimeLineX 4820T laptop computer with an Intel Core i5 M430 CPU and 4GB of RAM. The operating system was GNU/Linux (Fedora 17) with kernel version 3.6.2-4. TRecS, GTRecS2 and HorSat were compiled with OCaml version 3.12.1, `CSHORe` was run with OpenJDK IcedTea version 1.7.0 and TravMC using Mono version 3.0.2. A memout of 2GB and a timeout of 300s was given to each tool for each run. The order and size of the HORS are respectively listed in the O and Sz columns (where the size of a HORS is defined as the total number of occurrences of symbols in the righthand side of the rewriting rules). The T, G, TMC and C columns give the total run-times (in seconds) for the TRecS, GTRecS2, TravMC and `CSHORe` tools as each tool reports for itself. The HS and HST columns give the run-times as reported by HorSat and HorSatT.

Overall the benchmark results are favorable to our new algorithms HorSat and HorSatT. They are the best two tools in terms of the number of time-outs, although the state-of-the-art model checker TRecS is often better in terms of the run-times when it terminates. (The only time-out of HorSat is for `fibstring`, which is a pathological case where a huge string is concisely expressed by a HORS.) HorSat outperforms another saturation-based model checker `CSHORe` except for one instance (`fold_right`); this confirms the advantage of directly working on HORS without a detour to CPDS. According to further experiments (by Steven Ramsay) using the benchmark $\mathcal{G}_{m,n}$ [14], the running time of HorSat is not linear in the size of $\mathcal{G}_{m,n}$, even if we exclude out the time for 0CFA (whose worst case complexity is cubic time) to compute $\mathbf{Flow}(x)$. We believe that this is due the naiveness of the current implementation, and that an improved implementation would make the running time almost linear in the size of $\mathcal{G}_{m,n}$.

Table 2 Comparison of model-checking tools.

Benchmark file	Ord	Sz	T	G	TMC	C	HS	HST
`jwig-cal_main`	2	7627	0.090	0.902	**0.081**	—	13.137	5.306
`specialize_cps`	3	2731	—	—	—	5.145	1.702	**0.956**
`xhtmlf-div-2` **(neg)**	2	3003	**0.327**	51.8	—	—	12.392	2.697
`xhtmlf-m-ch`	2	3027	**0.331**	18.558	—	—	9.282	2.496
`fold_fun_list`	7	1346	0.724	—	—	4.152	**0.180**	0.729
`fold_right`	5	1310	—	—	—	**2.996**	34.796	7.958
`search-e-ch` **(neg)**	6	837	**0.011**	—	0.489	9.547	0.403	0.921
`zip`	4	2952	—	—	—	19.299	**3.501**	—
`filepath`	2	5956	—	—	—	1.059	**0.586**	6.860
`filter-nonzero` **(neg)**	5	482	**0.008**	0.486	0.206	3.337	0.067	0.147
`filter-nonzero-1`	5	888	0.272	—	—	11.116	**0.223**	1.203
`map-plusone-2`	5	704	1.227	—	20.080	5.518	**0.113**	0.609
`cfa-life2`	14	7648	—	—	—	—	**2.860**	—
`cfa-matrix-1`	8	2944	—	—	—	—	**0.450**	5.345
`cfa-psdes`	7	1819	—	—	—	6.761	**0.185**	1.410
`tak` **(neg)**	8	451	—	—	7.763	—	1.570	**0.429**
`dna`	2	411	**0.029**	0.072	0.467	—	0.126	0.335
`g45`	4	55	—	2.576	—	—	**0.019**	0.017
`fibstring`	4	29	—	**0.179**	102.583	—	—	—
`l`	3	35	—	0.010	23.322	0.439	**0.002**	0.006

6 Related Work

Early model-checking algorithms for HORS [10, 1, 22, 9] were mainly used for showing the decidability of model checking problems, and suffer from the k-EXPTIME worst-case complexity [22] for almost all inputs. To our knowledge, Kobayashi [11] proposed the first practical algorithm for trivial automata model checking, and implemented a model checker TRecS [12]. His algorithm reduces the start symbol S in a finite number of steps, and infers the types of non-terminals by observing how each non-terminal symbol is used in the partial reduction sequence. The inferred type environment is then used as an over-approximation of the fixedpoint of *Shrink* in Section 4. Some other practical algorithms [14, 21] have since been developed. Except GTRecS in a pathological case (the fifth category in the benchmark), however, they failed to show a clear practical advantage over TRecS. Those algorithms are all based on a type-based characterization of the problem, and propagate information *forwards*, starting with the goal to prove that S has type q_I. Broadbent et al. [4, 5] have recently proposed a quite different algorithm for CPDS, which uses *backward* propagation.

As already noted in Section 1, our new algorithms HorSat and HorSatT bridge the gap between the two families of model checking algorithms mentioned above. On the one hand, HorSat and HorSatT are strongly related to the type-based algorithms in that they use Kobayashi's type-based characterization of model checking [13], and the algorithms can be viewed as an (optimized) fixed-point computation for a function on intersection type environments. On the other hand, HorSat is also related to the saturation algorithm for CPDS, in that both propagate information backwards, starting from the set of error configurations. Our representation of a set of terms as a type environment is superficially

quite different from Broadbent et al.'s stack automata used to represent CPDS configurations (which are variations of alternating automata). Based on Rehof and Urzyczyn's result [25], however, a type environment can also be regarded as an alternating tree automaton that accepts the set $\mathbf{Terms}_\Gamma$ of terms well-typed under Γ (see Section C in Appendix). Thus, both approaches essentially represent the set of states reachable to accepting configurations by using (variants of) alternating automata. A more precise connection on this point will be discussed in a companion paper [3], by using a streamlined version of CPDS.

As mentioned in Section 1, the model checking of HORS has recently been applied to automated program analysis and verification [13, 11, 23, 26]. Those applications should benefit from the performance advantage of HorSat; in fact, we have recently replaced the underlying model checker TRecS with HorSat in the work on exact flow analysis [26] and observed a speed up by an order of magnitude in several cases.

7 Conclusion

We have presented the first algorithm for model-checking HORS using intersection types that employs a *backward* mode of inference, à la saturation algorithm for CPDS [4]; previous type-based algorithms use forward inference. We have also implemented a prototype model checker and confirmed that it often outperforms previous model checkers for HORS.

This paper lays the foundation for further work on backward mode saturation-like algorithms using types. It is worth mentioning that the set of *(forwards)-reachable* terms is irregular (when viewed as a language consisting of abstract syntax trees) and existing forward-mode algorithms must in some sense approximate the set of reachable terms. On the other hand, Rehof and Urzyczyn's construction [25] applied to the type environment computed by saturation shows that the set of terms *backward-reachable* from error-terms is *regular*. In this respect backward algorithms are arguably more natural.

Acknowledgment. We thank anonymous referees for useful comments. This work was partially supported by Kakenhi 23220001. The first author was supported by JSPS and AvH Postdoc. Fellowships.

References

1 Klaus Aehlig. A finite semantics of simply-typed lambda terms for infinite runs of automata. *Logical Methods in Computer Science*, 3(3), 2007.

2 A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *International Conference on Concurrency Theory*, pages 135–150, 1997.

3 Christopher Broadbent and Naoki Kobayashi. Streamlining collapsible pushdown systems and their model checking, 2013. Draft.

4 Christopher H. Broadbent, Arnaud Carayol, Matthew Hague, and Olivier Serre. A saturation method for collapsible pushdown systems. In *Proceedings of ICALP 2012*, volume 7392 of *LNCS*, pages 165–176. Springer-Verlag, 2012.

5 Christopher H. Broadbent, Arnaud Carayol, Matthew Hague, and Olivier Serre. C-SHORe: A collapsible approach to verifying higher-order programs. In *Proceedings of ICFP 2013*, 2013.

6 Arnaud Carayol and Olivier Serre. Collapsible pushdown automata and labeled recursion schemes: Equivalence, safety and effective selection. In *Proceedings of LICS 2012*, pages 165–174. IEEE, 2012.

7 A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems. In *Proc. 2nd Int. Workshop on Verification of Infinite State Systems (INFINITY'97), Bologna, Italy, July 11–12, 1997*, volume 9 of *ENTCS*. Elsevier, 1997.

8 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer-Verlag, 2002.

9 Matthew Hague, Andrzej Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. In *Proceedings of 23rd Annual IEEE Symposium on Logic in Computer Science*, pages 452–461. IEEE Computer Society, 2008.

10 Teodor Knapik, Damian Niwinski, and Pawel Urzyczyn. Higher-order pushdown trees are easy. In *FoSSaCS 2002*, volume 2303 of *LNCS*, pages 205–222. Springer-Verlag, 2002.

11 Naoki Kobayashi. Model-checking higher-order functions. In *Proceedings of PPDP 2009*, pages 25–36. ACM Press, 2009.

12 Naoki Kobayashi. TRecS: A type-based model checker for recursion schemes. `http://www-kb.is.s.u-tokyo.ac.jp/~koba/trecs/`, 2009.

13 Naoki Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. In *Proc. of POPL*, pages 416–428, 2009.

14 Naoki Kobayashi. A practical linear time algorithm for trivial automata model checking of higher-order recursion schemes. In *Proceedings of FoSSaCS 2011*, volume 6604 of *LNCS*, pages 260–274. Springer-Verlag, 2011.

15 Naoki Kobayashi. GTRecS2: A model checker for recursion schemes based on games and types. A tool available at `http://www-kb.is.s.u-tokyo.ac.jp/~koba/gtrecs2/`, 2012.

16 Naoki Kobayashi, Kazutaka Matsuda, Ayumi Shinohara, and Kazuya Yaguchi. Functional programs as compressed data. *Higher-Order and Symbolic Computation*, 2013. To appear. A preliminary version appeared in Proceedings of PEPM12.

17 Naoki Kobayashi and C.-H. Luke Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *Proceedings of LICS 2009*, pages 179–188. IEEE Computer Society Press, 2009.

18 Naoki Kobayashi and C.-H. Luke Ong. Complexity of model checking recursion schemes for fragments of the modal mu-calculus. *Logical Methods in Computer Science*, 7(4), 2011.

19 Naoki Kobayashi, Ryosuke Sato, and Hiroshi Unno. Predicate abstraction and CEGAR for higher-order model checking. In *Proc. of PLDI*, pages 222–233, 2011.

20 Naoki Kobayashi, Naoshi Tabuchi, and Hiroshi Unno. Higher-order multi-parameter tree transducers and recursion schemes for program verification. In *Proc. of POPL*, pages 495–508, 2010.

21 Robin P. Neatherway, Steven James Ramsay, and C.-H. Luke Ong. A traversal-based algorithm for higher-order model checking. In *ACM SIGPLAN International Conference on Functional Programming (ICFP '12)*, pages 353–364, 2012.

22 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS 2006*, pages 81–90. IEEE Computer Society Press, 2006.

23 C.-H. Luke Ong and Steven Ramsay. Verifying higher-order programs with pattern-matching algebraic data types. In *Proc. of POPL*, pages 587–598, 2011.

24 C.-H. Luke Ong and Takeshi Tsukada. Two-level game semantics, intersection types, and recursion schemes. In *Proceedings of ICALP 2012*, volume 7392 of *LNCS*, pages 325–336. Springer-Verlag, 2012.

25 Jakob Rehof and Pawel Urzyczyn. Finite combinatory logic with intersection types. In *Proceedings of TLCA 2011*, volume 6690 of *LNCS*, pages 169–183. Springer, 2011.

26 Yoshihiro Tobita, Takeshi Tsukada, and Naoki Kobayashi. Exact flow analysis by higher-order model checking. In *Proceedings of FLOPS 2012*, volume 7294 of *LNCS*, pages 275–289. Springer, 2012.

A On Theorem 8

Kobayashi and Ong's result [17] directly imply the following special cases of Theorem 8.

▶ **Theorem 24.** *Suppose* $\mathbf{Tree}(\mathcal{G})$ *does not contain* $\bot$*. Then,* $\mathbf{Tree}(\mathcal{G})$ *is accepted by* $\mathcal{A}$ *in the trivial mode if and only if there is a* possibly infinite *derivation tree for* $\emptyset \vdash_{\mathcal{A},\mathcal{G}} S : q_I$.

▶ **Theorem 25.** *Suppose* $\mathbf{Tree}(\mathcal{G})$ *does not contain* $\bot$*. Then,* $\mathbf{Tree}(\mathcal{G})$ *is accepted by* $\mathcal{A}$ *in the co-trivial mode if and only if there is a finite derivation tree for* $\emptyset \vdash_{\mathcal{A},\mathcal{G}} S : q_I$.

We sketch how to derive Theorem 8 from Theorems 24 and 25. Suppose $\mathbf{Tree}(\mathcal{G})$ may contain $\bot$. Let $\mathcal{G}'$ be a HORS that is obtained by adding a special terminal *loop* of arity 1, and replacing each rule $F\,x_1 \cdots x_k \to t$ of $\mathcal{G}$ with $F\,x_1 \cdots x_k \to \mathit{loop}(t)$. Then, $\mathbf{Tree}(\mathcal{G}')$ does not contain $\bot$, and $\mathbf{Tree}(\mathcal{G})$ is obtained from $\mathbf{Tree}(\mathcal{G}')$ by removing every infinite sequence *loop* with $\bot$ and removing other occurrences of *loop*. Let $\mathcal{A}'$ be the ATA obtained from $\mathcal{A}$ by adding the transition rule $(q, \mathit{loop}, \{(1, q)\})$ for every state q. Then, we have:

$$\begin{aligned}
& \mathbf{Tree}(\mathcal{G}) \text{ is accepted by } \mathcal{A} \text{ in the co-trivial mode} \\
\Leftrightarrow\ & \mathbf{Tree}(\mathcal{G}') \text{ is accepted by } \mathcal{A}' \text{ in the co-trivial mode} \\
\Leftrightarrow\ & \text{there is a finite derivation for } \emptyset \vdash_{\mathcal{A}',\mathcal{G}'} S : q_I. \\
\Leftrightarrow\ & \text{there is a finite derivation for } \emptyset \vdash_{\mathcal{A},\mathcal{G}} S : q_I.
\end{aligned}$$

Similarly, we have:

$$\begin{aligned}
& \mathbf{Tree}(\mathcal{G}) \text{ is accepted by } \mathcal{A} \text{ in the trivial mode} \\
\Leftrightarrow\ & \mathbf{Tree}(\mathcal{G}') \text{ is accepted by } \mathcal{A}' \text{ in the trivial mode} \\
\Leftrightarrow\ & \text{there is a possibly infinite derivation for } \emptyset \vdash_{\mathcal{A}',\mathcal{G}'} S : q_I. \\
\Leftrightarrow\ & \text{there is a possibly infinite derivation for } \emptyset \vdash_{\mathcal{A},\mathcal{G}} S : q_I.
\end{aligned}$$

B Proofs

B.1 Proofs for Section 3

Proof of Lemma 9

Suppose that $\mathbf{Tree}(\mathcal{G})$ is accepted by $\mathcal{A}$. Then there exists a finite run-tree R of $\mathcal{A}$ over $\mathbf{Tree}(\mathcal{G})$. Let D be the relevant part of the domain of $\mathbf{Tree}(\mathcal{G})$, i.e., $D = \{\pi \mid (\pi, a) \in \mathit{codom}(R)\}$. By the definition of $\mathbf{Tree}(\mathcal{G})(= \bigsqcup\{t^\bot \mid S \longrightarrow_{\mathcal{G}}^* t\})$, there exists t such that $S \longrightarrow_{\mathcal{G}}^* t$ and $U \subseteq \mathit{dom}(t^\bot)$ with $\mathbf{Tree}(\mathcal{G})(\pi) = t^\bot(\pi)$ for every $\pi \in U$. Then, R is also a run-tree of $\mathcal{A}^\bot$ over $t^\bot$. Thus, t satisfies the required property.

Conversely, suppose that $S \longrightarrow_{\mathcal{G}}^* t$ and $t^\bot$ is accepted by $\mathcal{A}^\bot$. Since $\mathcal{A}^\bot$ has no transition rule on $\bot$, a run-tree of $\mathcal{A}^\bot$ over $t^\bot$ must also be a run-tree of $\mathcal{A}$ over $\mathbf{Tree}(\mathcal{G})$. Thus, $\mathbf{Tree}(\mathcal{G})$ is accepted by $\mathcal{A}$. ◀

Proof of Lemma 10

Let $\mathcal{A} = (\Sigma, Q, \Delta, q_I)$. We show that $\emptyset \vdash_{\mathcal{A}}^- t : q$ if and only if $t^\bot$ is accepted by (Σ, Q, Δ, q), by induction on the structure of t. If $\emptyset \vdash_{\mathcal{A}}^- t : q$, then t must be of the form $a\,t_1 \cdots t_k$ and:

$$(q, a, \{(i, q_j) \mid i \in \{1, \ldots, k\}, j \in I_i\}) \in \Delta \qquad \emptyset \vdash_{\mathcal{A}}^- t_i : q_j \text{ for each } i \in \{1, \ldots, k\}, j \in I_i$$

By the induction hypothesis, ${t_i}^\bot$ is accepted by (Σ, Q, Δ, q_j) for each $i \in \{1, \ldots, k\}, j \in I_i$. Thus, $t^\bot = a\,{t_1}^\bot \cdots {t_k}^\bot$ is accepted by (Σ, Q, Δ, q).

Conversely, suppose that $t^{\perp}$ is accepted by (Σ, Q, Δ, q). Then t must be of the form $a\,t_1 \cdots t_k$. So, we have: $(q, a, \{(i, q_j) \mid i \in \{1, \ldots, k\}, j \in I_i\}) \in \Delta$ for some $I_1, \ldots, I_k$ and ${t_i}^{\perp}$ is accepted by (Σ, Q, Δ, q_j) for each $i \in \{1, \ldots, k\}, j \in I_i$. By the induction hypothesis, $\emptyset \vdash^{-}_{\mathcal{A}} t_i : q_j$ for each $i \in \{1, \ldots, k\}, j \in I_i$. Thus, we have $\emptyset \vdash^{-}_{\mathcal{A}} t : q$ as required. ◀

▶ **Lemma 26.** *Let s and t be applicative terms. If $\Gamma \vdash^{-}_{\mathcal{A}} [t/x]s : \tau$, then there exist a (possibly empty) set I and $\{\tau_i \mid i \in I\}$ (where ℓ may be 0) such that $\Gamma \cup \{x : \tau_i \mid i \in I\} \vdash^{-}_{\mathcal{A}} s : \tau$ and $\Gamma \vdash^{-}_{\mathcal{A}} t : \tau_i$ for each $i \in I$.*

Proof. The proof proceeds by induction on the structure of s.

- Case $s = a$ or $s = y \neq x$: The required result holds for $I = \emptyset$.
- Case $s = x$: The result holds for $I = \{1\}$ and $\tau_1 = \tau$.
- Case $s = s_1 s_2$: In this case, we have

$$\Gamma \vdash^{-}_{\mathcal{A}} [t/x]s_1 : \textstyle\bigwedge_{j \in J} \tau'_j \to \tau \qquad \Gamma \vdash^{-}_{\mathcal{A}} [t/x]s_2 : \tau'_j \text{ for each } j \in J$$

By the induction hypothesis, we have:

$$\Gamma \cup \{x : \tau_i \mid i \in I_0\} \vdash^{-}_{\mathcal{A}} s_1 : \textstyle\bigwedge_{j \in J} \tau'_j \to \tau \qquad \Gamma \cup \{x : \tau_i \mid i \in I_j\} \vdash^{-}_{\mathcal{A}} s_2 : \tau'_j \text{ for each } j \in J$$
$$\Gamma \vdash^{-}_{\mathcal{A}} t : \tau_i \text{ for each } i \in I \cup \textstyle\bigcup_{j \in J} I_j.$$

Let $I = I_0 \cup \bigcup_{j \in J} I_j$. Then, we have $\Gamma \cup \{x : \tau_i \mid i \in I\} \vdash^{-}_{\mathcal{A}} s : \tau$ as required.

◀

Proof of Lemma 12

This follows by repeated applications of Lemma 26. ◀

▶ **Lemma 27.** *Let $\mathcal{G}^{(m)}$ and $t^{(m)}$ as defined in the proof of Theorem 15. If $\mathcal{F}^m(\emptyset) \cup \Gamma \vdash^{-}_{\mathcal{A}} t : \tau$ then $\Gamma \vdash_{\mathcal{A}^{\perp}, \mathcal{G}^{(m)}} t^{(m)} : \tau$.*

Proof. This follows by double induction on m and the derivation of $\mathcal{F}^m(\emptyset) \vdash^{-}_{\mathcal{A}} t : \tau$, with case analysis on the last rule used for deriving $\mathcal{F}^m(\emptyset) \cup \Gamma \vdash^{-}_{\mathcal{A}} t : \tau$. Since the other cases are trivial, we show only the case where $t = F$, with $F : \tau \notin \Gamma$. In this case, $F : \tau \in \mathcal{F}^m(\emptyset)$ holds for some $m \geq 1$. By the definition of $\mathcal{F}$ and Lemmas 13, we have

$$\tau = \Gamma_V(x_1) \to \cdots \to \Gamma_V(x_k) \to q \qquad \mathcal{R}(F) = \lambda x_1. \cdots \lambda x_\ell. s \qquad \mathcal{F}^{m-1}(\emptyset) \cup \Gamma_V \vdash^{-}_{\mathcal{A}} t : q$$

By the induction hypothesis, we have $\Gamma_V \vdash_{\mathcal{A}^{\perp}, \mathcal{G}} t^{(m-1)} : q$. By using T-NT, we obtain $\Gamma \vdash_{\mathcal{A}^{\perp}, \mathcal{G}} F^{(m)} : \tau$ as required. ◀

▶ **Lemma 28** (substitution). *If $\Gamma \cup \{x : \tau_i \mid i \in I\} \vdash_{\mathcal{A}, \mathcal{G}} s : q$ and $\Gamma \vdash_{\mathcal{A}, \mathcal{G}} t : \tau_i$ for every $i \in I$, with $x \notin dom(\Gamma)$, then $\Gamma \vdash_{\mathcal{A}, \mathcal{G}} [t/x]s : q$.*

Proof. This follows by straightforward induction on the structure of s. ◀

▶ **Lemma 29** (type preservation). *If $\emptyset \vdash_{\mathcal{A}, \mathcal{G}} t : q$ and $t \longrightarrow_{\mathcal{G}} t'$, then $\emptyset \vdash_{\mathcal{A}, \mathcal{G}} t' : q$.*

Proof. This follows by induction on the derivation of $t \longrightarrow_{\mathcal{G}} t'$. Since the induction step is trivial, we show only the case where $t = F\,t_1 \cdots t_k$ and $t' = [t_1/x_1, \ldots, t_k/x_k]s$ with $\mathcal{R}(F) = \lambda x_1. \cdots \lambda x_k. s$. In this case, we have:

$$\{x_i : \tau_j \mid i \in \{1, \ldots, k\}, j \in I_i\} \vdash^{-}_{\mathcal{A}} \mathcal{G}s : q \qquad \emptyset \vdash^{-}_{\mathcal{A}} \mathcal{G}t_i : \tau_j \text{ for each } i \in \{1, \ldots, k\}, j \in I_i$$

By the substitution lemma (Lemma 28), we have $\emptyset \vdash^{-}_{\mathcal{A}} \mathcal{G}[t_1/x_1, \ldots, t_k/x_k]s : q$ as required. ◀

B.2 Proofs for Section 4

Proof of Theorem 19

This follows immediately from Theorem 8. Note that there is a fixedpoint Γ of $\mathit{Shrink}_{\mathcal{G}}$ such that $S : q_I \in \Gamma$ if and only if there is a possibly infinite derivation tree for $\emptyset \vdash_{\mathcal{A}} S : q_I$. To see this, note that if there is a possibly infinite derivation tree for $\emptyset \vdash_{\mathcal{A}} S : q_I$, then the set $\{F : \tau \mid \emptyset \vdash_{\mathcal{A}} F : \tau \text{ occurs in the derivation tree}\}$ is a fixed-point of Shrink, and conversely, one can construct an infinite derivation tree for $\emptyset \vdash_{\mathcal{A}} S : q_I$ from a fixedpoint of Shrink. ◀

The rest of this subsection is devoted to the proof of Theorem 21.

▶ **Lemma 30.** *Suppose $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}', S)$ and $\mathcal{R} \subseteq \mathcal{R}'$. If $\Gamma \vdash (\mathcal{G}, t) : q$ and $s \longrightarrow_{\mathcal{R}} t$, then $\mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}(\Gamma) \vdash (\mathcal{G}, s) : q$.*

Proof. By Lemma 14, we have $\mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}(\Gamma) \vdash s : q$. So, it remains to show $\mathit{Shrink}(\mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}(\Gamma)) = \mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}(\Gamma)$. Suppose $F : \sigma_1 \to \cdots \to \sigma_k \to q \in \mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}(\Gamma) \setminus \Gamma$ and $\mathcal{R}(F) = \lambda x_1. \cdots \lambda x_\ell.t$. By the definition of $\mathcal{F}$ and Lemma 13, we have $\Gamma, x_1 : \sigma_1, \ldots, x_\ell : \sigma_\ell \vdash_{\mathcal{A}} t : q$. Thus, we have $\mathcal{F}_{\mathcal{G},\mathcal{A},\mathcal{R}}(\Gamma), x_1 : \sigma_1, \ldots, x_\ell : \sigma_\ell \vdash_{\mathcal{A}} t : q$ as required. ◀

▶ **Lemma 31.** *If $\mathbf{Tree}(\mathcal{G})$ is accepted by $\mathcal{A}^{\top}$ in the trivial mode, then $S : q_I$ is an element of $\bigcap_{j\in\omega} \mathit{Shrink}^j(\bigcup_{i\in\omega} \mathcal{F}^i(\Gamma_0))$.*

Proof. Let $\mathcal{G}$ be $(\Sigma, \mathcal{N}, \mathcal{R}, S)$, where $\mathit{dom}(\mathcal{N}) = \{F_1, \ldots, F_n\}$ and $S = F_1$. Let $\Gamma_{\mathbf{max}}$ be the largest type environment that conforms to $\mathcal{N}$, i.e., $\{F : \tau \mid \tau :: \mathcal{N}(F)\}$. Let m be the number of type bindings $|\Gamma_{\mathbf{max}}|$. Let $\mathcal{G}^{(m)}$ be the HORS $(\Sigma \cup \{\bot \mapsto 0\}, \mathcal{N}^{(m)}, \mathcal{R}^{(m)}, F_1^{(m)})$ constructed as an approximation of $\mathcal{G}$ in the proof of Theorem 15. Let $\mathcal{R}'^{(m)}$ be the following subset of $\mathcal{R}^{(m)}$:

$$\mathcal{R}'^{(m)} = \{F_i^{(j)} \mapsto [F_1^{(j-1)}/F_1, \ldots, F_n^{(j-1)}/F_n]\mathcal{R}(F_i)\}$$

By the strong normalization of the simply-typed λ-calculus, $F_1^{(m)} \longrightarrow^*_{\mathcal{R}'^{(m)}} t \not\longrightarrow_{\mathcal{R}'^{(m)}}$ for some t, and there is a corresponding reduction sequence $F_1 \longrightarrow^*_{\mathcal{G}} t'$ of $\mathcal{G}$ such that $t'^{\bot} = t^{\bot}$. Thus, $t^{\bot}$ is accepted by $\mathcal{A}^{\top}$. By Lemma 10, we have $\emptyset \vdash^-_{\mathcal{A}} t^{\bot} : q_I$. Thus, we have $\Gamma_0^{(0)} \vdash^-_{\mathcal{A}} t : q_I$, where $\Gamma_0^{(0)} = \{F^{(0)} : \tau \mid F : \tau \in \Gamma_0\}$. By Lemma 30, we have $\Gamma' \vdash (\mathcal{G}^{(m)}, F_1^{(m)}) : q_I$ for $\Gamma' = \bigcup_{i\in\omega} \mathcal{F}^i_{\mathcal{G}^{(m)},\mathcal{A}^{\top},\mathcal{R}'^{(m)}}(\Gamma_0^{(0)})$. Let Γ'_j be: $\{F_i : \tau \mid F_i^{(j')} : \tau \in \Gamma' \wedge j' \geq j\}$. By the condition $\Gamma' = \bigcup_{i\in\omega} \mathcal{F}^i_{\mathcal{G}^{(m)},APT^{\top},\mathcal{R}'^{(m)}}(\Gamma_0^{(0)})$, we have $\Gamma'_0 \subseteq \bigcup_{i\in\omega} \mathcal{F}^i_{\mathcal{G},\mathcal{A},\mathcal{R}}(\Gamma_0)$. Furthermore, Γ'_j forms a monotonically decreasing sequence: $\Gamma'_0 \supseteq \Gamma'_1 \supseteq \cdots \supseteq \Gamma'_m$. Since $|\Gamma'_0| \leq m$ and $|\Gamma'_m| \geq |\{F_1^{(m)} : q_I\}| = 1$, there exists $k\,(< m)$ such that $\Gamma'_k = \Gamma'_{k+1}$. By the condition $\Gamma' \vdash (\mathcal{G}^{(m)}, F_1^{(m)}) : q_I$, we have $\vdash \mathcal{G}^{(m)} : \Gamma'$, which implies:

$$\Gamma{\downarrow}_{\{F_1^{(k)},\ldots,F_n^{(k)}\}}, x_1 : \sigma_1, \ldots, x_\ell : \sigma_\ell \vdash^-_{\mathcal{A}} [F_1^{(k)}/F_1, \ldots, F_n^{(k)}/F_n]t : \tau$$

for every $F^{(k+1)} : \tau \in \Gamma'$ and $\mathcal{R}(F) = \lambda x_1. \cdots \lambda x_\ell.t$. Here, $\Gamma{\downarrow}_S$ denotes $\{F : \tau \in \Gamma \mid F \in S\}$. Since $\Gamma'_k = \Gamma'_{k+1}$, the above condition implies: $\Gamma'_k, x_1 : \sigma_1, \ldots, x_\ell : \sigma_\ell \vdash^-_{\mathcal{A}} t : \tau$ for every $F : \tau \in \Gamma'_k$ and $\mathcal{R}(F) = \lambda x_1. \cdots \lambda x_\ell.t$, i.e., $\mathit{Shrink}_{\mathcal{G}}(\Gamma'_k) = \Gamma'_k$. Therefore, we have:

$$S : q_I \in \Gamma'_m \subseteq \Gamma'_k = \bigcap_{j\in\omega} \mathit{Shrink}^j(\Gamma'_k) \subseteq \bigcap_{j\in\omega} \mathit{Shrink}^j(\Gamma'_0) \subseteq \bigcap_{j\in\omega} \mathit{Shrink}^j(\bigcup_{i\in\omega} \mathcal{F}^i(\Gamma_0))).$$

◀

Proof of Theorem 21

The "only if" direction follows immediately from Theorem 19. The "if" direction follows from Lemma 31. ◀

C An algorithm to check the inhabitance condition

C.1 The construction

Computing $\mathcal{F}(\Gamma)$ requires a procedure to check $\Gamma \overset{?}{\in} \mathit{Inhabited}(\mathcal{K}, \Gamma_N)$. Following Rehof and Urzyczyn [25], we can reduce it the emptiness problem for an alternating tree automaton. This also serves to demonstrate how our algorithm can be interpreted as manipulating alternating tree automata (the stack automata manipulated by the CPDS algorithm can also be seen as a kind of alternating tree automaton). Given Γ_N, define an ATA $\mathcal{A}' = (\Sigma', Q', \Delta', q_I')$ by:

$$
\begin{aligned}
\Sigma' &= \{a \mapsto 0 \mid a \in \mathit{dom}(\Sigma)\} \cup \{F \mapsto 0 \mid F \in \mathit{dom}(\mathcal{N})\} \\
&\quad \cup \{@ \mapsto 2\} \\
Q' &= \{(\tau, \kappa) \mid \tau :: \kappa, \text{ and } \tau \text{ occurs (as a sub-expression)} \\
&\qquad \text{in } \Gamma_N \text{ or a type of a constant}\} \\
\Delta' &= \{((\tau,\kappa), a, \emptyset) \mid \emptyset \vdash^-_{\mathcal{A}} a : \tau \text{ and } \kappa = \underbrace{\mathsf{o} \to \cdots \to \mathsf{o}}_{\Sigma(a)} \to \mathsf{o}\} \\
&\quad \cup \{((\tau, \mathcal{N}(F)), F, \emptyset) \mid F : \tau \in \Gamma_N\} \\
&\quad \cup \{((\tau,\kappa), @, \\
&\qquad \{(1, (\textstyle\bigwedge_{i\in I} \tau_i \to \tau, \kappa' \to \kappa))\} \cup \{(2, (\tau_i, \kappa')) \mid i \in I\}) \\
&\qquad\quad \mid (\textstyle\bigwedge_{i\in I} \tau_i \to \tau, \kappa' \to \kappa) \in Q\} \\
q_I' &= (q_I, \mathsf{o})
\end{aligned}
$$

By the construction, $t^\sharp$ is accepted by $(\Sigma', Q', \Delta', (\tau, \kappa))$ if and only if $t \in \mathbf{ATerms}_{\mathcal{N},\Sigma,\kappa}$ and $\Gamma_N \vdash^-_{\mathcal{A}} t : \tau$, where $t^\sharp$ is the tree representation of term t, with an application $t_1 t_2$ is expressed by a tree:

Thus, to check $\Gamma \in \mathit{Inhabited}(\mathcal{K}, \Gamma_N)$, it suffices to check that for every $x \in \mathit{dom}(\Gamma)$, the intersection:

$$\bigcap_{x:\tau\in\Gamma} \mathcal{L}(\Sigma', Q', \Delta', (\tau, \mathcal{K}(x)))$$

is non-empty, where $\mathcal{L}(\mathcal{A})$ denotes the set of trees accepted by $\mathcal{A}$ (in the co-trivial mode).

During the computation of $\mathcal{F}^m(\emptyset)$ (where $m = 1, 2, \ldots$), the automaton $\mathcal{A}'$ for $\Gamma_N = \mathcal{F}^m(\emptyset)$ can be constructed incrementally.

C.2 A remark on the complexity of the emptiness check

For the purposes of HORS model-checking, we are interested in the complexity of our algorithms when the arity of sorts (and hence types) and the sizes of the property automaton (and in particular Q) are bounded. This is because these tend to be small compared to the size of the HORS itself.

In this subsection we explain why the inhabitance check used in the HorSat algorithm can be regarded (under the assumptions above) as a constant-time operation when considering the

theoretical worst-case complexity. However, this is not intended as a practical algorithm (the constant is still generally very large) but just to demonstrate that theoretically inhabitance checking does not negate the fixed parameter tractability of the algorithm.

First observe that, for the purposes of inhabitance checking, it is only necessary to have a single non-terminal bound to any given type. More precisely, let Γ_N be a set of type bindings for non-terminals, and $\mathcal{K}$ a kind-environment such that $\Gamma_N :: \mathcal{K}$. Suppose for each sort κ and each set $T \subseteq \mathbf{ITypes}_\kappa$ we have a fresh non-terminal $F_{\kappa,T}$. Define:

$$\begin{aligned} U_{\mathcal{K},\Gamma_N} &:= \{F_{\kappa,T} : \kappa \mid \exists G.\forall \tau \in T.G : \tau \in \Gamma_N \text{ with } \tau :: \kappa\} \\ &\quad \cup \{F_{\kappa,T} : \kappa \mid \exists a \in \mathit{dom}(\Sigma).\forall \tau \in T.\emptyset \vdash_{\mathcal{A},\mathcal{G}} a : \tau \text{ with } \tau :: \kappa\} \\ \Delta_{\mathcal{K},\Gamma_N} &:= \{F_{\kappa,T} : \tau \mid F_{\kappa,T} \in \mathit{dom}(U_{\mathcal{K},\Gamma_N}) \text{ and } \tau \in T\} \end{aligned}$$

It should be clear that

$$\mathit{Inhabited}(\mathcal{K},\Gamma_N) = \mathit{Inhabited}(U_{\mathcal{K},\Gamma_N}, \Delta_{\mathcal{K},\Gamma_N})$$

After all, given a term t witnessing $\{x : \tau \mid \tau \in T\} \in \mathit{Inhabited}(\mathcal{K},\Gamma_N)$ we can construct a term t' witnessing $\{x : \tau \mid \tau \in T\} \in \mathit{Inhabited}(U_{\mathcal{K},\Gamma_N}, \Delta_{\mathcal{K},\Gamma_N})$ by replacing each non-terminal G occurring in t with $F_{\kappa,R}$ where R is the maximal set satisfying $\{G : \tau \mid \tau \in R\} \subseteq \Gamma$ and $R :: \kappa$, and replacing every terminal a with $F_{\kappa,R}$ where R is the maximal set satisfying $\emptyset \vdash_{\mathcal{A},\mathcal{G}} a : \tau$ for every $\tau \in R$, with $R :: \kappa$. Conversely, given a term t' witnessing $\{x : \tau \mid \tau \in T\} \in \mathit{Inhabited}(U_{\mathcal{K},\Gamma_N}, \Delta_{\mathcal{K},\Gamma_N})$, we can construct a term t witnessing $\{x : \tau \mid \tau \in T\} \in \mathit{Inhabited}(\mathcal{K},\Gamma_N)$ by replacing every occurrence $F_{\kappa,R}$ in t' by either a G such that $G : \tau \in \Gamma$ for every $\tau \in R$ or $a \in \mathit{dom}(\Sigma)$ such that $\emptyset \vdash_{\mathcal{A},\mathcal{G}} a : \tau$ for every $\tau \in R$ (one of which must exist by definition).

Note that the size of $\Delta_{\mathcal{K},\Gamma_N}$ is bounded by a constant, namely the number of different well-sorted sets of intersection types. Thus in particular the emptiness of $\mathit{Inhabited}(U_{\mathcal{K},\Gamma_N}, \Delta_{\mathcal{K},\Gamma_N})$ can be determined in constant time (for example by the algorithm sketched in the previous section whose run-time depends only on $|\Delta_{\mathcal{K},\Gamma_N}|$ and the number of intersection types). So assuming that $\Delta_{\mathcal{K},\Gamma_N}$ has already been computed, $\mathit{Inhabited}(\mathcal{K},\Gamma_N)$ can be computed in constant time.

We now observe how $\Delta_{\mathcal{K},\Gamma_N}$ can be grown incrementally together with Γ_N as the algorithm progresses (rather than being recomputed on each inhabitance check). Since $\Gamma_N := \emptyset$ at the start of the algorithm, $\Delta_{\mathcal{K},\Gamma_N}$ is correspondingly initialized to $\Delta_{\emptyset,\emptyset}$.

As an aid the HorSat algorithm could maintain a table H with $\mathit{dom}(\mathcal{N})$ as keys where:

$$H(G) = \{P \subseteq \mathbf{ITypes}_\kappa \mid G : \kappa \in \mathcal{K} \text{ and } \forall \tau \in P.G : \tau \in \Gamma\}$$

This table grows as Γ_N is grown with the progression of the HorSat algorithm. Whenever a new set P' is added to $H(G)$ for some $G \in \mathit{dom}(\mathcal{N}) \cup \mathit{dom}(\Sigma)$, P' is also added to $\Delta_{\mathcal{K},\Gamma_N}$.

Whenever a new type binding for a non-terminal $G : \tau$ is added to Γ_N, this must be processed against every member P of $H(G)$ with $P \cup \{\tau\}$ then being added to $H(G)$.

Since the number of intersection types is fixed (and so the number of sets of intersection types is fixed) this must involve only constantly many comparisons. Thus assuming that $H(G)$ can be looked up in constant time, the overhead to HorSat for maintaining $H(G)$ would also only be constant for each addition to Γ_N (and it is only upon each addition to Γ_N that the inhabitance check is performed).

Descriptive complexity of approximate counting CSPs

Andrei Bulatov*[1], Victor Dalmau†[2], and Marc Thurley[3]

1 **School of Computing Science, Simon Fraser University, Burnaby, Canada**
abulatov@sfu.ca
2 **Department of Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain**
victor.dalmau@upf.edu
3 **Oracle, Buenos Aires, Argentina**
marc.thurley@googlemail.com

Abstract

Motivated by Fagin's characterization of NP, Saluja et al. have introduced a logic based framework for expressing counting problems. In this setting, a counting problem (seen as a mapping $\mathcal{C}$ from structures to non-negative integers) is 'defined' by a first-order sentence φ if for every instance $\mathbf{A}$ of the problem, the number of possible satisfying assignments of the variables of φ in $\mathbf{A}$ is equal to $\mathcal{C}(A)$. The logic RHΠ$_1$ has been introduced by Dyer et al. in their study of the counting complexity class #BIS. The interest in the class #BIS stems from the fact that, it is quite plausible that the problems in #BIS are not #P-hard, nor they admit a fully polynomial randomized approximation scheme. In the present paper we investigate which counting constraint satisfaction problems #CSP($\mathbf{H}$) are definable in the monotone fragment of RHΠ$_1$. We prove that #CSP($\mathbf{H}$) is definable in monotone RHΠ$_1$ whenever $\mathbf{H}$ is invariant under meet and join operations of a distributive lattice. We prove that the converse also holds if $\mathbf{H}$ contains the equality relation. We also prove similar results for counting CSPs expressible by linear Datalog. The results in this case are very similar to those for monotone RHΠ$_1$, with the addition that $\mathbf{H}$ has, additionally, $\top$ (the greatest element of the lattice) as a polymorphism.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic

Keywords and phrases Constraint Satisfaction Problems, Approximate Counting, Descriptive Complexity.

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.149

1 Introduction

Constraint Satisfaction Problems (CSPs) form a rich class of algorithmic problems with applications in many areas of computer science. In a CSP the goal is to find an assignment to variables subject to specified constraints. It has been observed by Feder and Vardi [19] that CSPs can be viewed as homomorphisms problems: given two relational structures $\mathbf{A}$ and $\mathbf{H}$, decide if there is a homomorphism from $\mathbf{A}$ to $\mathbf{H}$.

In this paper we consider *counting* constraint satisfaction problems (#CSPs), in which the problem of computing the number of solutions of a given CSP instance. Substantial amount

* supported by NSERC Discovery grant
† supported by MICINN grant TIN2010-20967-C04-02

of attention has been paid in the last decade to the complexity and algorithms for problems of the form CSP($\mathbf{H}$) (see, for example, [1, 3, 5, 8, 23]) and #CSP($\mathbf{H}$) [4, 6, 7, 10, 16, 17, 18, 21], in which the target structure $\mathbf{H}$ is fixed. In the case of exact counting, there is a complete complexity classification of problems #CSP($\mathbf{H}$) [4, 17, 18], which states that every problem of this form is either solvable in polynomial time, or complete in #P. This classification was recently extended to computing partition functions of weighted homomorphisms [10, 21].

Very few non-trivial counting problems can be solved using a polynomial-time deterministic algorithm. When efficient exact counting is not possible one might try to find a good approximation. Dyer et al. [14] argued that the most natural model of efficient approximation is the one by means of fully polynomial randomized approximation schemes (FPRAS), where the desired approximation error is a part of input, randomization is allowed, and the algorithm must stop within time polynomial in the size of the input and the bound on the approximation error. The approximation complexity of counting problems is then measured through approximation preserving, or AP-reductions, designed so that the class of problems solvable by FPRAS is closed under AP-reductions (more details can be found in §2.3).

The approximation complexity of #CSP($\mathbf{H}$) for 2-element structures $\mathbf{H}$ is determined in [15]. It turns out that, along with problems admitting a FPRAS (indeed, even solvable exactly in polynomial-time) and #P-hard problems, there are also problems that apparently do not fall into any of these two categories. Furthermore, all the problems that seemingly lie strictly between the class of problems admitting an FPRAS and the class of #P-hard problems are interreducible with each other and with other natural and well-studied problems (see also [14]) such as the problem of counting independent sets in a bipartite graph, denoted #BIS. It is argued in [14] that the set of problems interreducible with #BIS form a separate complexity class different from both FPRAS and #P. This class includes the problem of finding the number of downsets and the problem of finding the number of antichains in a partially ordered set, SAT-based problems such as finding the number of satisfying assignments of a CNF in which every clause is an implication or a unit clause, certain graph homomorphism problems, e.g., BeachConfig [14], and many others.

In this paper we shed light on the complexity of approximate counting CSPs by studying its descriptive complexity. We follow Saluja et al.'s framework [27] for studying the logical definability of counting problems. Let φ be a first-order formula that can have first and second-order free variables. In the setting of [27], a counting problem $\mathcal{C}$ (seen as a mapping from structures over a finite signature τ to non-negative integers) is *defined* by formula φ if, for every structure $\mathbf{A}$ with signature τ, $\mathcal{C}(\mathbf{A})$ is equal to the number of different interpretations of the free variables that make φ true on $\mathbf{A}$.

For example, the problem #IS of counting the number of independent sets of a graph $G = (V, E)$ is defined by the sentence

$$\forall x, y\ (\neg E(x, y) \vee \neg I(x) \vee \neg I(y)),$$

where I is a monadic second order free variable.

It is shown in [27] that, on ordered structures, the class #P coincides with the class #FO of counting problems definable by a first-order formula. Furthermore it is also shown that the expressiveness of subclasses of #FO obtained by restricting the quantifier alternation depth form the strict hierarchy

$$\#\Sigma_0 = \#\Pi_0 \subset \#\Sigma_1 \subset \#\Pi_1 \subset \#\Sigma_2 \subset \#\Pi_2 = \#\mathrm{FO},$$

where $\#L$ denotes the set of counting problems definable by a formula in L.

A different approach to expressing counting problems over graphs in logical terms has been developed in a series of papers by Makowsky et al. (see [24] and the references therein). The framework there is more liberal allowing to define a wide range of graph invariants and polynomials, such as the the chromatic polynomial, various generalizations of the Tutte polynomial, matching polynomials, interlace polynomials, and many others.

Dyer et al. have introduced the logic $\mathrm{RH\Pi}_1 \subseteq \Pi_1$ (to be defined below) in their study of the class of problems AP-interreducible with #BIS. It is shown in [14] that all problems in $\#\mathrm{RH\Pi}_1$ are AP-reducible to #BIS. Also, many problems AP-interreducible with #BIS (for example all problems listed in §2.3.1) are known to be in $\#\mathrm{RH\Pi}_1$. The logic $\mathrm{RH\Pi}_1$ contains all first-order formulas of the form $\forall \mathbf{y}\, \psi$, where ψ is a quantifier-free CNF, in which every clause has at most one occurrence of a unnegated second-order variable and at most one occurrence of a negated second-order variable.

Our main result concerns the monotone fragment of $\mathrm{RH\Pi}_1$, namely, the subset of $\mathrm{RH\Pi}_1$ containing all formulas in which every relation from τ (the signature of the input structure) appears only negatively. It is natural to consider monotone logics in the context of CSP as for every input structure $\mathbf{A}$ of $\#\mathrm{CSP}(\mathbf{H})$, every atomic formula with a predicate from τ holding in $\mathbf{A}$ makes the existence of a homomorphism from $\mathbf{A}$ to $\mathbf{H}$ less likely. Furthermore, it follows from the results of [20] that in the decision variant (that is, if our goal is merely to decide if the number of homomorphisms is greater than zero) monotone $\mathrm{RH\Pi}_1$ is as expressive as full $\mathrm{RH\Pi}_1$.

To tackle our question we consider the algebraic invariance properties of the relations in $\mathbf{H}$, namely, so-called polymorphisms of $\mathbf{H}$ (see §2.1). This approach has lead to impressive progress in the study of the complexity of $\mathrm{CSP}(\mathbf{H})$ and of $\#\mathrm{CSP}(\mathbf{H})$.

We prove that $\#\mathrm{CSP}(\mathbf{H})$ is definable by a monotone $\mathrm{RH\Pi}_1$ formula whenever $\mathbf{H}$ has, as polymorphisms, the meet and join operations of a distributive lattice. We also show that this is the best-possible considering only the algebraic invariants of $\mathbf{H}$. In particular, we prove that if $\#\mathrm{CSP}(\mathbf{H})$ is definable in monotone $\mathrm{RH\Pi}_1$ and $\mathbf{H}$ contains one relation interpreted as the equality then $\mathbf{H}$ must be invariant under the meet and join operations of a distributive lattice. Since the set of polymorphisms of a structure does not change if one adds the equality relation to it, our results imply a complete characterization for the definability of #CSPs in monotone $\mathrm{RH\Pi}_1$ under the assumption that every pair of structures $\mathbf{H}$ and $\mathbf{H}'$ with the same polymorphisms give rise to counting CSPs, $\#\mathrm{CSP}(\mathbf{H})$ and $\#\mathrm{CSP}(\mathbf{H}')$ that are both definable or both undefinable in monotone $\mathrm{RH\Pi}_1$. Although the majority of the properties of CSPs investigated so far are completely determined by the algebraic invariants (see [26]), there are some which are not [25].

As a byproduct of our main result we obtain a similar characterization of definability on the fragment of monotone $\mathrm{RH\Pi}_1$ known as linear Datalog. Linear Datalog has been investigated in the decision CSPs as a tool to show the membership in NL [2, 11, 12, 13]. For our purposes, linear Datalog is precisely the class of monotone $\mathrm{RH\Pi}_1$-formulas that do not contain free first-order variables and where every clause of its quantifier-free part contains an unnegated second-order variable. We show that $\#\mathrm{CSP}(\mathbf{H})$ is definable in linear Datalog if $\mathbf{H}$ has, as polymorphisms, the join, meet, and top operations of a distributive lattice and that the converse holds whenever $\mathbf{H}$ contains the equality relation.

2 Preliminaries

2.1 Basic definitions

Let A be a finite set. A k-ary tuple $(a_1, \ldots, a_k)$ over A is any element of A^k. We shall use boldface letters to denote tuples of any length. A k-ary relation on A is a collection of k-ary tuples over A or, alternatively, a subset of A^k. A *relational signature* (also relational vocabulary) τ is a collection of relational symbols (also called predicates), in which every symbol has an associated arity. A *(relational) structure* $\mathbf{A}$ with signature τ (also called *τ-structure*) consists of a set A called the *universe* of $\mathbf{A}$, and for each symbol $R \in \tau$, say of arity k, a k-ary relation $R^{\mathbf{A}}$ on A, called the *intepretation* of R in $\mathbf{A}$. We shall use the same boldfaced and slanted capital letters to denote a structure and its universe, respectively. In this paper all signatures and structures are finite. A *fact* of a relational structure is any atomic formula holding in it. Sometimes we will regard relational structures as a universe and a collection of facts on it.

Let R be a relation on a set A and $f\colon A^n \to A$ an n-ary operation on the same set. Operation f is said to be a *polymorphism* of R if for any choice $\mathbf{a}_1, \ldots, \mathbf{a}_n$ of tuples from R the tuple $f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ obtained by applying f component-wise also belongs to R. Then, it is also said that R is *invariant* under f. Operation f is a polymorphism of a relational structure $\mathbf{A}$ if it is a polymorphism of every relation in $\mathbf{A}$.

Let A, B be finite sets and let $f\colon A \to B$. For every tuple $\mathbf{a}$ on A we use $f(\mathbf{a})$ to denote the tuple on B obtained by applying f to $\mathbf{a}$ component-wise. Similarly, for every relation R on A we use $f(R)$ to denote $\{f(\mathbf{a}) \mid \mathbf{a} \in R\}$. Let $\mathbf{A}, \mathbf{B}$ be relational structures of the same signature with universes A and B, respectively. Mapping f is said to be a *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ if for any symbol R from τ, $f(R^{\mathbf{A}}) \subseteq R^{\mathbf{B}}$. If, furthermore, $B \subseteq A$ and f acts as the identity on B then f is said to be a *retraction*. A homomorphism f from $\mathbf{A}$ to $\mathbf{B}$ is said to be an *isomorphism* if it is bijective and f^{-1} is a homomorphism from $\mathbf{B}$ to $\mathbf{A}$.

A *lattice* $\mathbf{H}$ is a structure with a universe H equipped with two binary operations $\sqcap, \sqcup\colon H \times H \to H$ (see, e.g., [22]) satisfying the following conditions for any $x, y, z \in H$: (1) $x \sqcap x = x \sqcup x = x$, (2) $x \sqcap y = y \sqcap x$, $x \sqcup y = y \sqcup x$, (3) $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$, $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$, (4) $x \sqcap (x \sqcup y) = x \sqcup (x \sqcap y) = x$. Lattice $\mathbf{H}$ is said to be *distributive* if it satisfies an additional equation $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$. Every lattice has an associated partial order $\leq$ on its universe given by $x \leq y$ if and only if $x \sqcap y = x$.

2.2 Constraint satisfaction problem

For a relational structure $\mathbf{H}$ an instance of the *constraint satisfaction problem* $\mathrm{CSP}(\mathbf{H})$ is a structure $\mathbf{A}$ of the same signature. The goal in $\mathrm{CSP}(\mathbf{H})$ is to decide whether or not there is a homomorphism from $\mathbf{A}$ to $\mathbf{H}$. In the *counting constraint satisfaction problem* $\#\mathrm{CSP}(\mathbf{H})$ the objective is to find the number of such homomorphisms.

▶ **Example 1.** In a 3-SAT problem we are given a propositional formula φ in conjunctive normal form whose clauses contain 3 literals (3-CNF). The task is to decide if φ is satisfiable. As is easily seen, the 3-SAT problem is equivalent to $\mathrm{CSP}(\mathbf{H}_{3\text{-}\mathsf{SAT}})$, where $\mathbf{H}_{3\text{-}\mathsf{SAT}}$ is the relational structure with universe $\{0, 1\}$ that contains, for every $a, b, c \in \{0, 1\}$, the relation $R_{a,b,c} = \{0, 1\}^3 \setminus \{(a, b, c)\}$. In the counting version of 3-SAT, denoted #3-SAT, the goal is to find the number of satisfying assignments of a 3-CNF formula. Clearly, this problem can be represented as $\#\mathrm{CSP}(\mathbf{H}_{3\text{-}\mathsf{SAT}})$.

▶ **Example 2.** Let F be a finite field. The LINEAR SYSTEM(F) problem over F is the problem of checking the consistency of a given system of linear equations over F. This

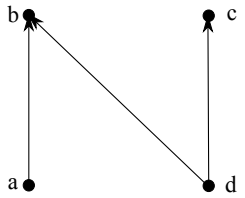


Figure 1 The $\mathbf{H}_{\mathsf{BIS}}$ graph.

problem cannot be represented as a CSP because the set of possible linear equations that can appear in an instance is infinite, while we only allow finite structures. However, for any system of linear equations one can easily obtain an equivalent system in which every equation contains at most 3 variables (although it may be necessary to introduce new variables). Hence, LINEAR SYSTEM(F) reduces to the restricted problem 3-LINEAR SYSTEM(F), in which only equations with 3 variables are allowed. For every $\alpha, \beta, \gamma, \delta \in F$, denote by $R_{\alpha\beta\gamma\delta}$, the ternary relation that contains all tuples $(x, y, z) \in F^3$ satisfying the equation $\alpha x + \beta y + \gamma z = \delta$. Then the 3-LINEAR SYSTEM(F) problem can be represented as CSP($\mathbf{H}_{\mathsf{LIN}}$), where $\mathbf{H}_{\mathsf{LIN}}$ is the relational structure with universe F equipped with all relations $R_{\alpha\beta\gamma\delta}$, $\alpha, \beta, \gamma, \delta \in F$. The counting version of this problem, #CSP($\mathbf{H}_{\mathsf{LIN}}$), concerns finding the number of solutions of a system of linear equations.

2.3 Counting and approximation

Counting CSPs is a particular case of *counting problems*. For every problem $\mathcal{L}$ in NP, one can associate a corresponding counting problem; namely, the problem of counting the accepting paths of a nondeterministic Turing machine deciding $\mathcal{L}$ in polynomial time. The set of problems defined this way is denoted by #P.

▶ **Example 3.** In the counting Bipartite Independent Set problem (#BIS) we are given a bipartite graph $\mathbf{G}$ and asked to find the number of independent sets in $\mathbf{G}$. Let $\mathbf{H}_{\mathsf{BIS}}$ be the digraph shown in Fig. 1. Given a bipartite graph $\mathbf{G}$ with bipartition (V_1, V_2) let $\mathbf{G}'$ be the digraph obtained by orienting all edges from V_1 to V_2. As is easily seen, the number of homomorphisms from $\mathbf{G}'$ to $\mathbf{H}_{\mathsf{BIS}}$ equals the number of independent sets in $\mathbf{G}$, as the preimage of $\{a, c\}$ is an independent set of $\mathbf{G}$. Thus, #BIS can be easily 'reduced' to #CSP($\mathbf{H}_{\mathsf{BIS}}$), but it is not clear if it can be represented as a counting CSP.

Algorithms and the complexity of counting problems, including counting CSPs, have attracted considerable amount of attention starting from the seminal paper by Valiant [28]. The complexity of exact counting CSPs is largely known, see, [4, 17, 18]. Every problem of the form #CSP($\mathbf{H}$) is either solvable in polynomial time or is complete in #P under polynomial time reductions[1].

The approximation complexity of #CSP($\mathbf{H}$) is much more diverse. Let $\mathcal{C}$ be a counting problem and, for an instance I of $\mathcal{C}$, let us denote the solution of I by $\#I$. For $\varepsilon > 0$, a randomized algorithm Alg is said to be an *ε-approximating algorithm* for the problem $\mathcal{C}$ if for any instance I of $\mathcal{C}$ it returns a number $\mathsf{Alg}(I)$ such that

$$\mathsf{Pr}\left[e^{-\varepsilon} < \frac{\mathsf{Alg}(I)}{\#I} < e^{\varepsilon}\right] \geq \frac{2}{3}.$$

[1] In fact, the class #P is not closed under polynomial time reductions; therefore it is technically more correct to say that these problems are complete in $\mathrm{FP}^{\#\mathrm{P}}$

Arguably, the most general, but still practical type of approximation algorithm for counting problems is *fully polynomial randomized approximation schemes* (*FPRAS*): An algorithm Alg is said to be an FPRAS for a counting problem $\mathcal{C}$ if it takes as input an instance I of $\mathcal{C}$ and a number $\varepsilon > 0$, outputs a number $\mathsf{Alg}(I, \varepsilon)$ satisfying the inequality above, and works in time polynomial in $|I|$ and $\log \frac{1}{\varepsilon}$. To compare the relative complexity of approximating counting problems one uses *approximation preserving reduction* (or *AP-reduction* for short). If $\mathcal{A}$ and $\mathcal{B}$ are counting problems, an AP-reduction from $\mathcal{A}$ to $\mathcal{B}$ is a probabilistic algorithm Alg, using $\mathcal{B}$ as an oracle, that takes as input a pair (I, ε) where I is an instance of $\mathcal{A}$ and $0 < \varepsilon < 1$, and satisfies the following three conditions: (i) every oracle call made by Alg is of the form (I', δ), where I' is an instance of $\mathcal{B}$, and $0 < \delta < 1$ is an error bound such that $\log \frac{1}{\delta}$ is bounded by a polynomial in the size of I and $\log \frac{1}{\varepsilon}$; (ii) the algorithm Alg meets the specifications for being approximation scheme for $\mathcal{A}$ whenever the oracle meets the specification for being approximation scheme for $\mathcal{B}$; and (iii) the running time of Alg is polynomial in the size of I and $\log \frac{1}{\varepsilon}$. If an AP-reduction from $\mathcal{A}$ to $\mathcal{B}$ exists we write $\mathcal{A} \leq_{\mathrm{AP}} \mathcal{B}$, and say that $\mathcal{A}$ is *AP-reducible to* $\mathcal{B}$.

2.3.1 The class of problems AP-interreducible with #BIS

The two most natural approximation complexity classes are FPRAS, the class of problems solvable by an FPRAS, and the class $\mathrm{FP}^{\#\mathrm{P}}$, the class of problems AP-interreducible with #SAT (note that #P is not closed under AP-reductions). In [14] Dyer et al. argued that #BIS (see Example 3) defines a class of its own: No FPRAS is known for this problem, and it is not believed to be interreducible with #SAT. There are many natural and well studied problems that are AP-interreducible with #BIS. The following list contains some examples:

- #DOWNSET. A downset in a partial order $(P, \leq)$ is a set $A \subseteq P$ such that whenever $b \in A$ and $a \leq b$, the element a belongs to A. The #DOWNSET problem asks, given a partial order $(P, \leq)$ to find the number of downsets in P.
- #ANTICHAIN. An antichain in a partial order $(P, \leq)$ is a set $C \subseteq P$ such that $a \leq b$ for no $a, b \in C$. In the #ANTICHAIN problem we are required, given a partial order $(P, \leq)$, to find the number of antichains. #ANTICHAIN and #DOWNSET are essentially the same problem. Clearly, every downset $A \subseteq P$ is determined by the set of its maximal elements that form an antichain. Conversely, if $C \subseteq P$ is an antichain then the set $\{a \in P \mid a \leq b \text{ for some } b \in C\}$ is a downset.
- #IMPLICATION. Let φ be a 2-CNF, in which every clause is of the form $\neg x \vee y$, or, equivalently, $x \to y$. In the #IMPLICATION problem, given such a 2-CNF, the goal is to compute the number of its satisfying assignments. There are easy AP-reductions between #DOWNSET and #IMPLICATION. In one direction, the downsets of a partial order $(P, \leq)$ are exactly the satisfying assignments of the formula that includes clause $b \to a$ for every pair $a, b \in P$ with $a \leq b$. For the opposite direction, every instance φ of #IMPLICATION can be represented as a digraph $G(\varphi)$, in which the nodes are the variables of φ and edges (x, y) correspond to clauses $x \to y$. The set of strongly connected components $P(\varphi)$ of $G(\varphi)$ can be equipped with the natural partial order: $U_1 \leq U_2$ for $U_1, U_2 \in P(\varphi)$ if and only if there is a directed path from a node from U_2 to a node from U_1. It is straightforward to see that the number of satisfying assignments of φ equals the number of downsets in $P(\varphi)$.

  Also, #IMPLICATION is precisely $\#\mathrm{CSP}(\mathbf{H}_{\mathsf{IMP}})$, where $\mathbf{H}_{\mathsf{IMP}}$ is the digraph shown in Fig. 2.

Figure 2 The $\mathbf{H}_{\mathsf{IMP}}$ digraph.

A classification of problems of the form #CSP($\mathbf{H}$) for 2-element structures $\mathbf{H}$ according to their approximation complexity given in [15] provides another evidence of the significance of #BIS. Indeed, every such problem turns out to be either solvable exactly in polynomial time (and so belongs to FPRAS), or is AP-interreducible with #SAT, or else is AP-interreducible with #BIS.

2.4 Descriptive complexity of (approximate) counting problems

Motivated by Fagin's characterization of NP, Saluja et al. [27] have introduced a logic based framework for expressing counting problems. In what follows we describe the setting of [27].

Let τ and σ be finite signatures, let $\mathcal{C}$ be a counting problem (seen as a mapping from τ-structures to non-negative integers), let $\varphi(\mathbf{z})$ be a first-order formula with signature $\tau \cup \sigma$ with free (first-order) variables $\mathbf{z}$, and let $\mathbf{A}$ be a τ-structure. Formula φ is *monadic* if all predicates in σ have arity at most one. An $\mathbf{A}$-*assignment* for φ (or just an assignment, if $\mathbf{A}$ and φ are clear) is a pair $(\mathbf{T}, \mathbf{a})$ where $\mathbf{T}$ and $\mathbf{a}$ are interpretations of σ and $\mathbf{z}$, respectively, over the universe, A, of $\mathbf{A}$. We write $(\mathbf{A}, \mathbf{T})$ to denote the $(\tau \cup \sigma)$-structure with universe A where every $R \in \tau$ is interpreted as in $\mathbf{A}$ and every $I \in \sigma$ is interpreted as in $\mathbf{T}$. We say that $(\mathbf{T}, \mathbf{a})$ *satisfies* φ if $(\mathbf{A}, \mathbf{T}) \models \varphi(\mathbf{a})$ that is, if $\varphi(\mathbf{a})$ is true on the structure $(\mathbf{A}, \mathbf{T})$. We say that φ *defines* $\mathcal{C}$ if for every τ-structure $\mathbf{A}$

$$\mathcal{C}(\mathbf{A}) = |\{(\mathbf{T}, \mathbf{a}) \mid (\mathbf{A}, \mathbf{T}) \models \varphi(\mathbf{a})\}|$$

We note here that we deviate—although only formally—from the framework in Sajula et al. in the following sense: we use predicate symbols in σ to represent second-order variables. Hence, our first-order formulas only have, formally, first-order free variables.

We shall denote by #FO the set of all counting problems definable by a first-order formula. For every fragment L of FO we define $\#L$ as the set of all counting problems definable by a formula in L. An structure $\mathbf{A}$ is ordered if it has a binary relation that is interpreted as a total order on the universe.

▶ **Theorem 4** ([27]). *On ordered structures, the class #P coincides with the class #FO. In fact, #P is the class of all counting problems definable with a* Π_2 *formula.*

Saluja et al. [27] study the expressiveness of subclasses of #FO obtained by restricting the quantifier alternation depth obtaining the strict hierarchy

$$\#\Sigma_0 = \#\Pi_0 \subset \#\Sigma_1 \subset \#\Pi_1 \subset \#\Sigma_2 \subset \#\Pi_2 = \#\mathrm{FO}$$

Dyer et al. [14] introduced the fragment $\mathrm{RH}\Pi_1 \subseteq \Pi_1$ in their study of the complexity class of problems AP-interreducible with #BIS. A first-order formula $\varphi(\mathbf{z})$ with signature $\tau \cup \sigma$ is in $\mathrm{RH}\Pi_1$ if it is of the form $\forall \mathbf{y} \psi(\mathbf{y}, \mathbf{z})$ where ψ is a quantifier-free CNF in which every clause has at most one occurrence of an unnegated relation symbol from σ and at most one occurrence of a negated symbol from σ. The part Π_1 in notation $\#\mathrm{RH}\Pi_1$ indicates that the formula involves only universal quantification, and RH indicates that ψ is in 'restricted Horn' form.

It is shown in [14] that all problems in #RHΠ$_1$ are AP-reducible to #BIS. Also, many problems AP-interreducible with #BIS (for example all problems listed in §2.3.1) are known to be in #RHΠ$_1$.

▶ **Example 5.** Consider the problem #DOWNSET. By encoding a partial order $(P, \leq)$ as a structure with a binary relation we can define #DOWNSET with the RHΠ$_1$-sentence

$$\forall x, y \ (I(x) \vee \neg(x \leq y) \vee \neg I(y))$$

Our ultimate goal is to characterize under which circumstances #CSP($\mathbf{H}$) belongs to #RHΠ$_1$. Towards this end, in this paper we consider the fragment of RHΠ$_1$ obtained by requiring that, in addition, the predicates from τ occur only negatively. The resulting logic is called *monotone* RHΠ$_1$.

It makes sense to restrict to monotone formulas in the context of problems of the form #CSP($\mathbf{H}$) as the addition of more facts to an input structure cannot increase the number of homomorphisms. Indeed, all problems listed in §2.3.1 are also definable by a formula in monotone RHΠ$_1$. Furthermore, it follows from the results of [20] that, if our goal is merely to decide if the number of homomorphisms is greater than zero, monotone RHΠ$_1$ is as expressive as RHΠ$_1$. Additionally we will deal exclusively with unordered structures as the analysis for ordered structures becomes much more complicated.

We say that a τ-structure $\mathbf{H}$ *contains equality* if τ contains a binary relational symbol eq that is interpreted as the equality on the set H (that is, $\mathbf{H}^{\text{eq}} = \{(b, b) \mid b \in H\}$). Note that we do not require that eq is interpreted as the equality in the instances of #CSP($\mathbf{H}$).

We are now in a position to state the main result of the paper.

▶ **Theorem 6.** *For every structure* $\mathbf{H}$ *the following holds:*

1. *If* $\mathbf{H}$ *has polymorphisms* $x \sqcap y$ *and* $x \sqcup y$ *for some distributive lattice* $(H; \sqcap, \sqcup)$ *then there exists a monotone* RHΠ$_1$*-formula defining* #CSP($\mathbf{H}$).
2. *Furthermore, if* $\mathbf{H}$ *contains equality then the converse also holds.*

Observe that since the set of polymorphisms of a structure $\mathbf{H}$ does not change if one adds the equality relation to it, it follows that the sufficient condition of Theorem 6 is the best it can be achieved by considering only the algebraic invariants of $\mathbf{H}$. Also, note that it follows from Theorem 6 that the problem of deciding whether for a relational structure $\mathbf{H}$ containing equality the problem #CSP($\mathbf{H}$) is definable in monotone RHΠ$_1$ belongs to NP. Indeed, after guessing lattice operation $\sqcap, \sqcup$ on the universe of $\mathbf{H}$, it is polynomial time to verify that these binary operations are polymorphisms of the structure.

3 Reduction to the monadic case

In this section, as a first step toward proving the necessary condition of Theorem 6, we prove the following proposition.

▶ **Proposition 7.** *For every structure* $\mathbf{H}$*, if* #CSP($\mathbf{H}$) *is definable in monotone* RHΠ$_1$ *then it is also definable by a monadic monotone formula from* RHΠ$_1$ *without free variables.*

In what follows, τ and σ are finite vocabularies, $\mathbf{H}$ is a τ-structure, and $\varphi(\mathbf{z})$ is a monotone RHΠ$_1$-formula with signature $\tau \cup \sigma$ defining #CSP($\mathbf{H}$).

For every $n \geq 1$, define $\mathbf{Isol}_n$ (from *isolated nodes*) to be the τ-structure with universe $\{1, \ldots, n\}$, where all relations are interpreted as the empty set.

▶ **Lemma 8.** *φ is a sentence (i.e, has no free variables).*

Proof. Consider structure $\mathbf{Isol}_p$ where p is a prime number that does not divide $|H|$. Let k be the number of free variables of φ and for every $\mathbf{a} \in A^k$ let $n(\mathbf{a})$ be

$$|\{(\mathbf{T}, \mathbf{a}) \mid (\mathbf{Isol}_p, \mathbf{T}) \models \varphi(\mathbf{a})\}|$$

Clearly $\sum_{\mathbf{a}\in A^k} n(\mathbf{a}) = |H|^p$. Consider the following equivalence relation θ on A^k: two tuples $\mathbf{a}, \mathbf{a}' \in A^k$ are θ-related if $\mathbf{a}' = h(\mathbf{a})$ for some bijection $h : A \to A$. Clearly, if $\mathbf{a}$ and $\mathbf{a}'$ are θ-related then $n(\mathbf{a}) = n(\mathbf{a}')$.

For every $\mathbf{a} \in A^k$ we shall denote by $\mathbf{a}_\theta$ the θ-class containing $\mathbf{a}$. Hence $|H|^p = \sum_{\mathbf{a}\in A^k} n(\mathbf{a}) = \sum_{\mathbf{a}_\theta \in (A^k)_\theta} |\mathbf{a}_\theta| \cdot n(\mathbf{a})$, where $(A^k)_\theta$ denotes the set of all θ-classes. Note that $|\mathbf{a}_\theta| = p(p-1)\cdots(p-m+1)$, where m is the number of different elements in $\mathbf{a}$. We are in a position to show that $k = 0$. If $k > 0$ then p divides $|\mathbf{a}_\theta|$ for any $\mathbf{a}$ and hence p divides $\sum_{\mathbf{a}\in A^k} n(\mathbf{a}) = |H|^p$, but p does not divide $|H|$, a contradiction. ◀

Consequently, from now on we can assume that φ is a sentence. Let $\mathbf{A}$ be a τ-structure and let $\mathbf{T}$ be any $\mathbf{A}$-assignment. It will be convenient to regard, alternatively, $\mathbf{T}$ as the collection of all atomic formulas $I(\mathbf{a})$ that hold in $\mathbf{T}$.

The following lemma is a direct consequence of the fact that every clause of an $\mathrm{RH}\Pi_1$-formula has at most one occurrence of an unnegated relation symbol from σ and at most one occurrence of a negated symbol from σ.

▶ **Lemma 9.** *Let $\mathbf{A}$ be a τ-structure. The set of all $\mathbf{A}$-assignments (seen as a collection of facts) satisfying φ is closed under union and intersection.*

▶ **Lemma 10.** *Let I be any predicate in σ with arity $k \geq 2$. Then*

$$\varphi \models \forall x_1, \ldots, x_k, y_1, \ldots, y_k \quad \neg(x_i = y_i) \vee \neg I(x_1, \ldots, x_k) \vee I(y_1, \ldots, y_k)$$

for some $1 \leq i \leq k$

Proof. For every $L \subseteq \{1, \ldots, k\}$, we define μ_L to be the sentence

$$\forall x_1, \ldots, x_k, y_1, \ldots, y_k \quad (\bigvee_{i\in L} \neg(x_i = y_i)) \vee \neg I(x_1, \ldots, x_k) \vee I(y_1, \ldots, y_k).$$

Observe that μ_L expresses the fact that $I(z_1, \ldots, z_k)$ only depends on the variables $z_i, i \in L$.

It follows easily that $\mu_J \wedge \mu_K \models \mu_L$ for every $J, K, L \subseteq \{1, \ldots, k\}$ with $J \cap K \subseteq L$. Note that the sentences appearing in the statement of the lemma are precisely the class of all sentences of the form μ_L where L is a singleton. Hence, the lemma follows by a direct application of the above property provided we are able to show the following:

For every different $i, j \in \{1, \ldots, k\}$ there exists L with $\{i, j\} \not\subseteq L$ and such that $\varphi \models \mu_L$.

To simplify the notation we shall prove the claim only for $i = k-1$ and $j = k$. Consider the structure $\mathbf{Isol}_{2n+k-2}$ with n large enough. Let X be the set containing all atomic formulas of the form $I(1, \ldots, k-2, a, b)$ where $a \in \{k-1, \ldots, n+k-2\}$ and $b \in \{n+k-1, \ldots, 2n+k-2\}$.

We claim that there exists some atomic formula $I(1, \ldots, k-2, a, b) \in X$ such that every satisfying assignment containing $I(1, \ldots, k-2, a, b)$ contains also some other atomic formula in X. Indeed, otherwise, since the set of satisfying assignments is closed under union, we could construct for every non-empty subset $Y \subseteq X$ a satisfying assignment containing all atomic predicates from Y and none of the atomic predicates from $Y \setminus X$. This would lead to a contradiction, as the set of satisfying assignments would be at least 2^{n^2}, which grows

asymptotically faster than $|H|^{2n+k-2}$ (the number of homomorphisms from $\mathbf{Isol}_{2n+k-2}$ to $\mathbf{H}$).

Thus there exists an atomic formula $I(1,\dots,k-2,a,b) \in X$ such that every satisfying assignment containing $I(1,\dots,k-2,a,b)$ contains also some other atomic formula in X. Consider first the case in which there exists at at least one satisfying assignment containing $I(1,\dots,k-2,a,b)$. Then, the *smallest*, with respect to inclusion, satisfying assignment containing $I(1,\dots,k-2,a,b)$ (by Lemma 9 such an assignment exists) also contains some other atomic predicate $I(1,\dots,k-2,a',b')$ in X. By the monotonicity of φ it follows that φ implies

$$\forall v_1,\dots,v_{2n+k-2} \;\; \neg I(v_1,\dots,v_{k-2},v_a,v_b) \vee I(v_1,\dots,v_{k-2},v_{a'},v_{b'}),$$

which after renaming variables is equivalent to μ_L for $L = \{1,\dots,k-2,k-1\}$ or $L = \{1,\dots,k-2,k\}$. Secondly, assume that there is no satisfying assignment containing $I(1,\dots,k-2,a,b)$. Then

$$\varphi \models \forall v_1,\dots,v_{2n+k-2} \;\; \neg I(v_1,\dots,v_{k-2},v_a,v_b).$$

It follows easily that, in this case, φ implies any formula of the form μ_L. ◀

Proof of Proposition 7. Let $\mathbf{H}$ be a τ-structure and let φ be a monotone RHΠ$_1$-formula with signature $\tau \cup \sigma$ defining #CSP($\mathbf{H}$). By Lemma 8, φ has no free variables.

Pick any predicate I in σ with arity $k \geq 2$. By Lemma 10

$$\varphi \models \forall x_1,\dots,x_k,y_1,\dots,y_k \;\; \neg(x_i = y_i) \vee \neg I(x_1,\dots,x_k) \vee I(y_1,\dots,y_k)$$

for some $1 \leq i \leq k$. Let σ' be obtained from σ by replacing I by a new unary predicate I', and let φ' be the sentence with signature $\tau \cup \sigma'$ obtained from φ by replacing every atomic formula of the form $I(z_1,\dots,z_k)$ by $I'(z_i)$. It follows easily that φ' has the same number of satisfying assignments as φ and one non-monadic predicate less. Iterating we obtain a sentence that contains only monadic predicates. ◀

4 Necessary condition

We start with proving item (2) of Theorem 6. In what follows $\mathbf{H}$ is a τ-structure and φ is a monadic monotone RHΠ$_1$-sentence with signature $\tau \cup \sigma$ defining #CSP($\mathbf{H}$). It will be convenient to assume that σ does not contain 0-ary predicate symbols. This can be achieved by replacing every 0-ary relation symbol $I \in \sigma$ with a new unary relation symbol I', adding to φ the clause $\neg I'(x) \vee I'(y)$, and replacing every atomic formula of the form $R()$ in φ with $R'(x)$ (x and y are bound variables in φ).

Let $\mathbf{A}$ be a τ-structure. Since all the predicate symbols in σ are unary one can establish a bijection from the set of all $\mathbf{A}$-assignments to the set of mappings from A to 2^σ. In particular, we associate with every $\mathbf{A}$-assignment $\mathbf{T}$ a mapping $h : A \to 2^\sigma$ where for every $a \in A$

$$h(a) = \{I \in \sigma \mid I(a) \text{ holds in } \mathbf{T}\}.$$

We shall use $\mathbf{T}_h$ to denote the $\mathbf{A}$-assignment associated with a mapping h. Let also $\mathrm{Sol}(\mathbf{A},\varphi)$ is given by

$$\mathrm{Sol}(\mathbf{A},\varphi) = \{h : A \to 2^\sigma \mid (\mathbf{A},\mathbf{T}_h) \models \varphi\}.$$

▶ **Lemma 11.** *Let* $\mathbf{A}$, $\mathbf{B}$ *be* τ*-structures and let* g *be a homomorphism from* $\mathbf{A}$ *to* $\mathbf{B}$*. For every* $f : B \to 2^\sigma$ *the following holds:*

1. *If* $f \in \mathrm{Sol}(\mathbf{B}, \varphi)$ *then* $f \circ g \in \mathrm{Sol}(\mathbf{A}, \varphi)$*.*
2. *If* $g(A) = B$ *and* $|\mathrm{Sol}(\mathbf{B}, \varphi)| = |\mathrm{Sol}(\mathbf{A}, \varphi)|$*, then for any* $h \in \mathrm{Sol}(\mathbf{A}, \varphi)$ *there is* $f \in \mathrm{Sol}(\mathbf{B}, \varphi)$ *such that* $h = f \circ g$*.*

Proof. (1) Follows directly from the monotonicity of φ. (2) Since $g(A) = B$ the set $\{f \circ g \mid f \in \mathrm{Sol}(\mathbf{B}, \varphi)\}$ contains $|\mathrm{Sol}(\mathbf{B}, \varphi)|$ different mappings and, consequently, $\mathrm{Sol}(\mathbf{A}, \varphi)$ cannot contain any other one. ◀

▶ **Lemma 12.** *Let* $\mathbf{A}$ *be a* τ*-structure, let* $a, a' \in A$*, and let* $\mathbf{B}$ *be the* τ*-structure obtained by adding the fact* $\mathrm{eq}(a, a')$ *to* $\mathbf{A}$*. For every* $f : A \to 2^\sigma$ *the following holds:*

$$f \in \mathrm{Sol}(\mathbf{B}, \varphi) \text{ if and only if } f \in \mathrm{Sol}(\mathbf{A}, \varphi) \text{ and } f(a) = f(a')$$

Proof. We can assume wlog. that $\mathbf{A}$ (and hence $\mathbf{B}$) contains equalities $\mathrm{eq}(a, a)$ and $\mathrm{eq}(a', a')$ as the addition of $\mathrm{eq}(a, a)$ and $\mathrm{eq}(a', a')$ does not alter $\hom(\mathbf{A}, \mathbf{H})$ or $\hom(\mathbf{B}, \mathbf{H})$, and, consequently, it cannot alter $\mathrm{Sol}(\mathbf{A}, \varphi)$ or $\mathrm{Sol}(\mathbf{B}, \varphi)$ either.

($\Rightarrow$) Assume $f \in \mathrm{Sol}(\mathbf{B}, \varphi)$. It follows directly from Lemma 11(1) that $f \in \mathrm{Sol}(\mathbf{A}, \varphi)$ so it only remains to show that $f(a) = f(a')$. Again by Lemma 11(1) it is only necessary to prove the statement in the case when $\mathbf{A}$ does not contain any other fact besides the equalities $\mathrm{eq}(a, a)$ and $\mathrm{eq}(a', a')$. Indeed, if the claim is true for such structure $\mathbf{A}'$ then the identity homomorphisms from $\mathbf{A}'$ to $\mathbf{A}$ witnesses, with help of Lemma 11(1), that it is also true for $\mathbf{A}$. Let $g : A \to A \setminus \{a'\}$ be the mapping that sends a' to a and acts as the identify otherwise. Lemma 11(2) implies that $f = h \circ g$ for some $h \in \mathrm{Sol}(g(\mathbf{B}), \varphi)$ and hence $f(a) = f(a')$.

($\Leftarrow$) Let $\varphi = \forall \mathbf{y} \psi(\mathbf{y})$, let n be the number of variables in $\mathbf{y}$, and let $\mathbf{C}$ be the τ-structure obtained by adding to $\mathbf{A}$ the chain of equalities

$$\mathrm{eq}(a, a_1), \mathrm{eq}(a_1, a_2), \ldots, \mathrm{eq}(a_n, a_{n+1}), \mathrm{eq}(a_{n+1}, a'),$$

where $a_1, \ldots, a_{n+1}$ are new elements not occurring in $\mathbf{A}$. Assume that $f \in \mathrm{Sol}(\mathbf{A}, \varphi)$ and $f(a) = f(a')$, and let $h : C \to 2^\sigma$ be the extension of f that sets $h(a_i) = f(a)$ for every $i = 1, \ldots, n + 1$.

We claim that $h \in \mathrm{Sol}(\mathbf{C}, \varphi)$. Let $\mathbf{c}$ be any instantiation of $\mathbf{y}$ over C. There exists some element a_i that does not appear in $\mathbf{c}$. Let $\mathbf{C}'$ be obtained by removing from $\mathbf{C}$ the equalities involving a_i. There is a retraction g from $\mathbf{C}'$ to $\mathbf{A}$ that maps a_j to a if $j \leq i$ and to a' otherwise. By Lemma 11(1) $h = f \circ g$ belongs to $\mathrm{Sol}(\mathbf{C}', \varphi)$ and, hence, $(\mathbf{T}_h, \mathbf{C}') \models \psi(\mathbf{c})$. Since a_i does not appear in $\mathbf{c}$ we have $(\mathbf{T}_h, \mathbf{C}) \models \psi(\mathbf{c})$ as well. Since $(\mathbf{T}_h, \mathbf{C}) \models \psi(\mathbf{c})$ holds for every instantiation $\mathbf{c}$ of $\mathbf{y}$, the claim follows.

Let g be any retraction from $\mathbf{C}$ to $\mathbf{B}$ with $g(a_i) \in \{a, a'\}$ for every $i = 1, \ldots, n+1$. Since $h = f \circ g$ it follows from Lemma 11(2) that $f \in \mathrm{Sol}(\mathbf{B}, \varphi)$. ◀

For every $R \in \tau$ of arity, say, k, let $\mathbf{J}_R$ be the τ-structure with universe $\{1, \ldots, k\}$ containing only fact $R(1, \ldots, k)$. Recall the definition of $\mathbf{Isol}_n$ in the beginning of §3. We define $\mathbf{J}_\varphi$ to be the τ-structure with universe $\mathrm{J}_\varphi = \{h(1) \mid h \in \mathrm{Sol}(\mathbf{Isol}_1, \varphi)\}$ such that for every $R \in \tau$

$$R^{\mathbf{J}_\varphi} = \{(h(1), \ldots, h(k)) \mid h \in \mathrm{Sol}(\mathbf{J}_R, \varphi)\}$$

The next two lemmas follow directly from the definition of $\mathbf{J}_\varphi$.

▶ **Lemma 13.** *Let* $\mathbf{A}$ *be any* τ*-structure and let* $f \in \mathrm{Sol}(\mathbf{A}, \varphi)$*. Then* f *is a homomorphism from* $\mathbf{A}$ *to* $\mathbf{J}_\varphi$*.*

Proof. First, let $a \in A$, and $g : \mathbf{Isol}_1 \to \mathbf{A}$ taking 1 to a. By Lemma 11, $f \circ g \in \mathrm{Sol}(\mathbf{Isol}_1, \varphi)$, implying $f(a)$ belongs to the universe of $\mathbf{J}_\varphi$. Similarly, let $R \in \tau$ and let $(a_1, \ldots, a_k) \in R^{\mathbf{A}}$. The mapping $i \overset{g}{\mapsto} a_i$ defines a homomorphism from $\mathbf{J}_R$ to $\mathbf{A}$. By Lemma 11, $f \circ g \in \mathrm{Sol}(\mathbf{J}_R, \varphi)$, which is equivalent to say that $R(f(a_1), \ldots, f(a_k))$ holds in $\mathbf{J}_\varphi$. ◀

▶ **Lemma 14.** *If* $\mathbf{H}$ *contains equality then* $\mathbf{H}$ *and* $\mathbf{J}_\varphi$ *are isomorphic.*

Proof. Let X and Y be sets, let F be a collection of mappings from X to Y, and let $\mathrm{equiv}(X)$ be the set of all equivalence relations in X. For every $\theta \in \mathrm{equiv}(X)$ we denote by F_θ the collection of all $f \in F$ such that $f(i) = f(j)$ whenever i and j are θ-related.

Let $\mathbf{A}$ be any τ-structure. For every $\theta \in \mathrm{equiv}(A)$ we define $\mathbf{A}_\theta$ to be the structure that is obtained by adding to $\mathbf{A}$ all facts of the form $\mathrm{eq}(a, a')$ where a and a' are θ-related. For every $\theta \in \mathrm{equiv}(A)$ we have

$$|\mathrm{Sol}(\mathbf{A}, \varphi)_\theta| = |\mathrm{Sol}(\mathbf{A}_\theta, \varphi)| = |\hom(\mathbf{A}_\theta, \mathbf{H})| = |\hom(\mathbf{A}, \mathbf{H})_\theta|,$$

where the first equality follows from Lemma 12 and the other equalities follow directly from the definitions. Consequently, $\mathrm{Sol}(\mathbf{A}, \varphi)$ and $\hom(\mathbf{A}, \mathbf{H})$ contain the same number of injective mappings. This follows from the fact that the number of injective mappings in $\mathrm{Sol}(\mathbf{A}, \varphi)$ and the number of injective mappings in $\hom(\mathbf{A}, \mathbf{H})$ are completely determined by the values $|\mathrm{Sol}(\mathbf{A}, \varphi)_\theta|, \theta \in \mathrm{equiv}(A)$, and $|\hom(\mathbf{A}, \mathbf{H})_\theta|, \theta \in \mathrm{equiv}(A)$, respectively, according to the Möbius inversion formula. By setting $\mathbf{A} = \mathbf{H}$ we infer that $\mathrm{Sol}(\mathbf{H}, \varphi)$ contains an injective mapping h that, by Lemma 13, is an homomorphism from $\mathbf{H}$ to $\mathbf{J}_\varphi$. Since $|H| = |\mathrm{J}_\varphi|$, homomorphism h must be, in fact, a bijective homomorphism. For every relation symbol $R \in \tau$, we have $h(R^{\mathbf{H}}) \subseteq R^{\mathbf{J}_\varphi}$, because h is a homomorphism. We also have $|R^{\mathbf{J}_\varphi}| = |R^{\mathbf{H}}| = |h(R^{\mathbf{H}})|$ where the first equality follows from the definition of $\mathbf{J}_\varphi$ and the second one follows from the fact that h is a bijection. It follows that $h(R^{\mathbf{H}}) = R^{\mathbf{J}_\varphi}$. Consequently, h is an isomorphism. ◀

Proof of Theorem 6(2). Let $\mathbf{H}$ be a τ-structure that contains equality such that $\#\mathrm{CSP}(\mathbf{H})$ is definable in monotone $\mathrm{RH}\Pi_1$. By Lemma 14 $\mathbf{H}$ is isomorphic to $\mathbf{J}_\varphi$. Since by Lemma 9 $\mathbf{J}_\varphi$ has polymorphisms $x \cap y$ and $x \cup y$, the theorem follows. ◀

Lemma 14 fails if $\mathbf{H}$ does not contain equality as the following example shows. Let $\mathbf{H}$ be the digraph with universe $\{0, 1\}$ containing only edge $(0, 1)$. Consider the monotone $\mathrm{RH}\Pi_1$-sentence φ with $\sigma = \{I\}$

$$\forall x, y, z\ (\neg E(x, y) \vee I(x)) \wedge (\neg E(x, y) \vee I(y)) \wedge (\neg E(x, y) \vee \neg E(y, z))$$

It is not difficult to see that φ defines $\#\mathrm{CSP}(\mathbf{H})$ and that $\mathbf{H}$ is not isomorphic to $\mathbf{J}_\varphi$. Still, $\mathbf{H}$ is invariant under the meet and join of a distributive lattice, namely, $(\{0, 1\}, \vee, \wedge)$.

5 Sufficient condition

In this section we shall prove item (1) of Theorem 6. Throughout this section τ is a finite signature and $\mathbf{H}$ is a τ-structure with polymorphisms $x \sqcap y$ and $x \sqcup y$ for some distributive lattice $(H; \sqcap, \sqcup)$. Our goal is to show that there exists a monotone (monadic) $\mathrm{RH}\Pi_1$-sentence φ defining $\#\mathrm{CSP}(\mathbf{H})$.

It is well known (see, e.g., [22, Theorem 9, Corollary 11, Corollary 14, Ch. II.1]) that, since $(H; \sqcap, \sqcup)$ is distributive, there is an isomorphism g from $(H; \sqcap, \sqcup)$ to a sublattice of the

lattice of subsets of some finite set S. It will be convenient to assume wlog. that $g(\top) = S$ and $g(\bot) \neq \emptyset$ where $\top$ and $\bot$ are the top and bottom elements, respectively, of $(H; \sqcap, \sqcup)$.

Let k be the maximum arity of a relation in τ, and let us define σ to have one monadic predicate for each symbol in S. To simplify notation we shall use the same symbol to represent a member of S and its associate predicate in σ. Sentence φ is defined to be the monotone monadic RHΠ$_1$-sentence $\forall \mathbf{x}\ \psi(\mathbf{x})$ with signature $\tau \cup \sigma$, where $\mathbf{x}$ has size k and $\psi(\mathbf{x})$ contains all clauses $\chi(\mathbf{x})$ with at most one occurrence of an unnegated symbol from σ and at most one occurrence of a negated symbol from σ such that $(\mathbf{H}, \mathbf{T}_g) \models \forall \mathbf{x}\ \chi(\mathbf{x})$ (recall the definition of $\mathbf{T}_g$ given in the beginning of §4). For every $I \in \sigma$ we shall denote by $\mathbf{T}_g^I$ the interpretation of I in $\mathbf{T}_g$, that is, the relation $\{a \in H \mid I \in g(a)\}$.

▶ **Lemma 15.** *Let $b, b' \in H$ and let $X = \mathbf{T}_g^I$ for some $I \in \sigma$. Then:*
1. *$b \sqcup b' \in X \Leftrightarrow b \in X$ or $b' \in X$*
2. *$b \sqcap b' \in X \Leftrightarrow b \in X$ and $b' \in X$*

Proof. Follows directly from the definitions. ◀

▶ **Lemma 16.** *Let $\mathbf{A}$ be a τ-structure and let $h \in \mathrm{Sol}(\mathbf{A}, \varphi)$. Then $g^{-1} \circ h$ is well defined and belongs to $\hom(\mathbf{A}, \mathbf{H})$.*

Proof. Let $a \in A$ and let $\mathcal{Y}$ be a nonempty collection of subsets of H. $\mathcal{Y}$ is said to be *consistent* with $h(a)$ if for every $I \in \sigma$ the following holds:

$$I \in h(a) \Leftrightarrow Y \subseteq \mathbf{T}_g^I \text{ for some } Y \in \mathcal{Y}.$$

We claim that if there is a set $\mathcal{Y}$ consistent with $h(a)$ then $g^{-1}(h(a))$ is well defined and is equal to $\sqcup_{Y \in \mathcal{Y}} \sqcap Y$, where $\sqcap Y$ denotes the meet of all elements from Y. To see this, let $b = \sqcup_{Y \in \mathcal{Y}} \sqcap Y$. It follows from the definition of consistency and Lemma 15 that for every $I \in \sigma$

$$I \in h(a) \Leftrightarrow b \in \mathbf{T}_g^I,$$

which is equivalent to saying that $g(b) = h(a)$.

For every $a \in A$, let $\mathcal{Y}_a$ be the set $\{\mathbf{T}_g^I \mid I \in h(a)\}$. We have $\emptyset \neq g(\bot) \subseteq h(a)$, and hence $\mathcal{Y}_a$ is non-empty. We claim that $\mathcal{Y}_a$ is consistent with $h(a)$. Let $I \in \sigma$ and consider the two cases:

- $I \in h(a)$. In this case $\mathcal{Y}_a$ contains $\mathbf{T}_g^I$ and we are done.
- $I \notin h(a)$. Assume, towards a contradiction that $\mathbf{T}_g^J \subseteq \mathbf{T}_g^I$ for some $J \in h(a)$. This implies, by the definition of φ, that φ contains the clause $\neg J(x) \vee I(x)$, in contradiction with the fact that $I \notin h(a)$ and $J \in h(a)$.

Since for every $a \in A$, $\mathcal{Y}_a$ is a nonempty collection of sets consistent with $h(a)$, it follows that $g^{-1} \circ h$ is well defined. Now, let us prove that $g^{-1} \circ h \in \hom(\mathbf{A}, \mathbf{H})$.

Let $R \in \tau$ and let $(a_1, \ldots, a_k) \in \mathbf{A}^R$. For every $i = 1, \ldots, k$ and every $I \in h(a_i)$, let $Y_{i,I}$ be the set of all tuples $(b_1, \ldots, b_k) \in R^{\mathbf{H}}$ where $b_i \in \mathbf{T}_g^I$. The set $Y_{i,I}$ satisfies the following two claims:

Claim 1: $Y_{i,I} \neq \emptyset$. Otherwise, $\forall \mathbf{x}\ (\neg R(x_1, \ldots, x_k) \vee \neg I(x_i))$ holds in $(\mathbf{H}, \mathbf{T}_g)$, which implies that φ contains the clause $\neg R(x_1, \ldots, x_k) \vee \neg I(x_i)$, in contradiction with the fact that $(a_1, \ldots, a_k) \in \mathbf{A}^R$ and $I \in h(a_i)$.

Claim 2: For every $j = 1, \ldots, k$ and every $J \notin h(a_j)$, set $Y_{i,I}$ contains a tuple with $b_j \notin \mathbf{T}_g^J$. Otherwise, $\forall \mathbf{x}\ (\neg R(x_1, \ldots, x_k) \vee \neg I(x_i) \vee I(x_j))$ holds in $(\mathbf{H}, \mathbf{T}_g)$, which implies that φ contains the clause $\neg R(x_1, \ldots, x_k) \vee \neg I(x_i) \vee J(x_j)$, in contradiction with the fact that $(a_1, \ldots, a_k) \in \mathbf{A}^R$, $I \in h(a_i)$, and $J \notin h(a_j)$.

Let

$$\mathbf{c} = (c_1, \dots, c_k) = \bigsqcup_{1 \le i \le k, I \in h(a_i)} \sqcap Y_{i,I}$$

Since $g(\bot) \subseteq h(a_i)$, Claim 1 above guarantees that the right term is not void. It follows from Claims 1 and 2 above that for every $j = 1, \dots, k$ the set $\{\mathrm{proj}_j\, Y_{i,I} \mid 1 \le i \le k, I \in h(a_i)\}$ (where $\mathrm{proj}_j\, Y_{i,I}$ denotes the *projection* of $Y_{i,I}$ to its jth coordinate) is consistent with $h(a_j)$. This implies that $c_j = (g^{-1} \circ h)(a_j)$ for every $j = 1, \dots, k$. Since $\mathbf{c}$ is obtained by iterative application of $\sqcup$ and $\sqcap$ to tuples in $R^{\mathbf{H}}$ we conclude that $\mathbf{c} \in R^{\mathbf{H}}$ and we are done. ◀

Proof of Theorem 6(1). Let $\mathbf{A}$ be any τ-structure. It follows from the definition of φ that $(\mathbf{H}, \mathbf{T}_g) \models \varphi$ or, equivalently, that $g \in \mathrm{Sol}(\mathbf{H}, \varphi)$. Then, by Lemma 11(1) $f \mapsto g \circ f$ defines a mapping from $\hom(\mathbf{A}, \mathbf{H})$ and $\mathrm{Sol}(\mathbf{A}, \varphi)$. This mapping is injective (because so is g) and exhaustive (by Lemma 16). ◀

6 Counting problems and linear Datalog

Datalog has a long and successful history as a tool in database theory and the study of the decision CSP (see [9] and the references therein). In this section we show how Datalog is also related to counting CSPs, and, more generally, to counting problems. As a byproduct of the proof of Theorem 6 we obtain a characterization of counting CSPs that can be represented through certain Datalog programs.

Datalog is a language of logic programs primarily developed in database theory. Let τ and σ be finite signatures. Symbols from τ are called *EDBs* (for Extensional DataBase symbols), and symbols from σ are called *IDBs* (for Intensional DataBase symbols). Every Datalog program is a collection of rules of the form

$$\psi_1 :- \psi_2, \dots, \psi_m,$$

where $\psi_1, \dots, \psi_m$ are atomic formulas using predicates from $\tau \cup \sigma$. The left side of the rule is called the *head* of the rule, and the predicate symbol occurring in it must be an IDB. The right hand side is called the *body* of the rule and might contain both IDBs and EDBs. A rule is called *linear* if its body contains at most one occurrence of an IDB. A Datalog program is said to be linear if all its rules are linear. The linear fragment of Datalog has been used for decision CSPs (see [2, 11, 12, 13]). In particular, the decision CSPs expressible through linear Datalog belong to the class NL. Moreover, it is conjectured that the converse is also true.

A Datalog program P is applied to a τ-structure $\mathbf{A}$ using the semantics of fixed points. Let $\mathbf{T}$ be an interpretation of the IDBs on the universe A of $\mathbf{A}$. Then $\mathbf{T}$ is said to be a *fixed point* of P on $\mathbf{A}$ if for every rule the following holds: for every interpretation of the variables of the rule that makes the body of the rule true given the intepretation of the IDBs (in $\mathbf{T}$) and of the EDBs (in $\mathbf{A}$), the head of the rule must hold in $\mathbf{T}$. Since the intersection of two fixed points is again a fixed point, there is always a least fixed point of a Datalog program. To express a decision CSP in terms of Datalog one should consider programs that contain a distinguished 'goal' IDB (usually null-ary). The corresponding CSP has no solution on input $\mathbf{A}$ if and only if the least fixed point of the program on $\mathbf{A}$ contains the goal IDB. The link to counting CSPs works in a different way. Here we consider not only the least fixed point, but all fixed points of a Datalog program. For a τ-structure $\mathbf{H}$ we say that a Datalog program P with EDBs from τ defines the problem $\#\mathrm{CSP}(\mathbf{H})$ if for any τ-structure $\mathbf{A}$ the number of homomorphisms from $\mathbf{A}$ to $\mathbf{H}$ equals the number of fixed points of P on $\mathbf{A}$.

For the purpose of this paper, however, we do not have to define the semantics of Datalog as above. Instead, we view linear Datalog as a fragment of monotone $\mathrm{RH\Pi}_1$ and fixed points as satisfying assignments of the corresponding formulas.

▶ **Lemma 17.** *Every linear Datalog program is equivalent to a monotone* $\mathrm{RH\Pi}_1$ *sentence in which each clause contains an unnegated predicate symbol from* σ *(equivalently, an IDB). Conversely, every monotone* $\mathrm{RH\Pi}_1$ *formula of this kind is equivalent to a Datalog program.*

Proof. The lemma easily follows from the observation that every rule $\psi_1 : -\psi_2, \dots, \psi_m$ of Datalog is equivalent to the clause $\psi_1 \vee \neg\psi_2 \vee \dots \vee \neg\psi_m$ of a monotone $\mathrm{RH\Pi}_1$ sentence. Note that the clause contains a positive occurrence (namely the symbol in ψ_1) of an IDB. Conversely, every clause like that with exactly one unnegated predicate from σ can be translated into a Datalog rule. ◀

Since Datalog is a proper subset of monotone $\mathrm{RH\Pi}_1$, it is likely to be less expressive.

▶ **Theorem 18.** *For every structure* $\mathbf{B}$ *the following holds:*

1. *If* $\mathbf{H}$ *has polymorphisms* $x \sqcap y$ *and* $x \sqcup y$ *for some distributive lattice* $(H; \sqcap, \sqcup)$ *and the nullary operation* $\top$ *(returning the greatest element of the lattice) then there is a linear Datalog program that defines* $\#\mathrm{CSP}(\mathbf{H})$
2. *Furthermore, if* $\mathbf{H}$ *contains equality then the converse also holds.*

Proof. (1) It has been show in §5 that, under the hypothesis of item (1), $\#\mathrm{CSP}(\mathbf{H})$ is definable by a monadic monotone $\mathrm{RH\Pi}_1$-sentence φ. Recall the definitions of φ, g, S, σ, and $\mathbf{T}_g$ from §5.

Assume now, additionally, that $\mathbf{H}$ is invariant under the nullary operation $\top$ returning the top element of the lattice. Let $\psi(\mathbf{x})$ be any clause in φ. We just need to show that some relation symbol from σ occurs unnegated in $\psi(\mathbf{x})$. We have $g(\top) = S$ by assumption (here and during the rest of the proof we shall slightly abuse the notation by using $\top$ to denote also the element in H that $\top$ returns). It follows that $\top \in \mathbf{T}_g^I$ for every $I \in \sigma$. Also, since $\mathbf{H}$ is invariant under $\top$ it follows that $(\top, \dots, \top) \in \mathbf{H}^R$ for every $R \in \tau$. By definition $(\mathbf{H}, \mathbf{T}_g) \models \forall \mathbf{x}\ \chi(\mathbf{x})$. Hence, if one instantiates all variables in $\mathbf{x}$ to $\top$ then one obtains an assigment that falsifies all negated atomic formulas in χ. Consequently, $\chi(\mathbf{x})$ must contain one unnegated atomic formula. Since $\chi(\mathbf{x})$ is monotone the predicate symbol of this unnegated atomic formula must be from σ.

(2) Recall the definition of J_φ from §4. It has been show in §4 that $\mathbf{J}_\varphi$ is invariant under set-theoretic union and intersection and that, under the hypothesis of item (2), $\mathbf{H}$ is isomorphic to $\mathbf{J}_\varphi$. Assume now that φ is a linear Datalog program. It is only necessary to show that, additionally, $\mathbf{J}_\varphi$ is invariant under the nullary operation returning σ (the top element in the lattice $(\mathrm{J}_\varphi, \cap, \cup)$). Let R be any predicate symbol in τ. Since every clause of φ has an occurrence of an unnegated predicate symbol from σ it follows that the mapping $\{1, \dots, k\} \mapsto \sigma$ belongs to $\mathrm{Sol}(\mathbf{J}_R, \varphi)$. Hence, by definition $(\sigma, \dots, \sigma) \in R^{\mathbf{J}_\varphi}$. ◀

References

1 Libor Barto and Marcin Kozik. Constraint satisfaction problems of bounded width. In *FOCS*, pages 595–603, 2009.

2 Libor Barto, Marcin Kozik, and Ross Willard. Near unanimity constraints have bounded pathwidth duality. In *LICS*, pages 125–134, 2012.

3 Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006.

4 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. In *ICALP (1)*, pages 646–661, 2008.

5 Andrei A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Log.*, 12(4):24, 2011.

6 Andrei A. Bulatov and Víctor Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. *Information and Computation*, 205(5):651–678, 2007.

7 Andrei A. Bulatov and Martin Grohe. The complexity of partition functions. *Theor. Comput. Sci.*, 348(2-3):148–186, 2005.

8 Andrei A. Bulatov, Peter Jeavons, and Andrei A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005.

9 Andrei A. Bulatov, Andrei A. Krokhin, and Benoit Larose. Dualities for constraint satisfaction problems. In *Complexity of Constraints*, pages 93–124, 2008.

10 Jin-Yi Cai and Xi Chen. Complexity of counting CSP with complex weights. In *STOC*, pages 909–920, 2012.

11 Catarina Carvalho, Víctor Dalmau, and Andrei A. Krokhin. CSP duality and trees of bounded pathwidth. *Theor. Comput. Sci.*, 411(34-36):3188–3208, 2010.

12 Víctor Dalmau. Linear datalog and bounded path duality of relational structures. *Logical Methods in Computer Science*, 1(1), 2005.

13 Víctor Dalmau and Andrei A. Krokhin. Majority constraints have bounded pathwidth duality. *Eur. J. Comb.*, 29(4):821–837, 2008.

14 Martin E. Dyer, Leslie Ann Goldberg, Catherine S. Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2003.

15 Martin E. Dyer, Leslie Ann Goldberg, and Mark Jerrum. An approximation trichotomy for Boolean #CSP. *J. Comput. Syst. Sci.*, 76(3-4):267–277, 2010.

16 Martin E. Dyer, Leslie Ann Goldberg, and Mike Paterson. On counting homomorphisms to directed acyclic graphs. *J. ACM*, 54(6), 2007.

17 Martin E. Dyer and David Richerby. On the complexity of #CSP. In *STOC*, pages 725–734, 2010.

18 Martin E. Dyer and David Richerby. The #CSP dichotomy is decidable. In *STACS*, pages 261–272, 2011.

19 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

20 Tomás Feder and Moshe Y. Vardi. Homomorphism closed vs existential positive. In *LICS*, pages 311–320, 2003.

21 Leslie Ann Goldberg, Martin Grohe, Mark Jerrum, and Marc Thurley. A complexity dichotomy for partition functions with mixed signs. In *STACS*, pages 493–504, 2009.

22 G. Grätzer. *General Lattice Theory*. Birkhäuser Verlag, Basel, 2003.

23 Pawel M. Idziak, Petar Markovic, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM J. Comput.*, 39(7):3023–3037, 2010.

24 Tomer Kotek and Johann Makowsky. Connection matrices and the definability of graph parameters. In *CSL*, pages 411–425, 2009.

25 Benoit Larose, Cynthia Loten, and Claude Tardif. A characterisation of first-order constraint satisfaction problems. *Logical Methods in Computer Science*, 3(4), 2007.

26 Benoit Larose and Pascal Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.*, 410(18):1629–1647, 2009.

27 Sanjeev Saluja, K. V. Subrahmanyam, and Madhukar N. Thakur. Descriptive complexity of #P functions. *J. Comput. Syst. Sci.*, 50(3):493–505, 1995.

28 L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

What is Decidable about Partially Observable Markov Decision Processes with omega-Regular Objectives*

Krishnendu Chatterjee, Martin Chmelik, and Mathieu Tracol

IST Austria
Klosterneuburg, Austria

Abstract

We consider partially observable Markov decision processes (POMDPs) with ω-regular conditions specified as parity objectives. The qualitative analysis problem given a POMDP and a parity objective asks whether there is a strategy to ensure that the objective is satisfied with probability 1 (resp. positive probability). While the qualitative analysis problems are known to be undecidable even for very special cases of parity objectives, we establish decidability (with optimal EXPTIME-complete complexity) of the qualitative analysis problems for POMDPs with all parity objectives under finite-memory strategies. We also establish optimal (exponential) memory bounds.

1998 ACM Subject Classification D.2.4 Formal methods

Keywords and phrases POMDPs, Omega-regular objectives, Parity objectives, Qualitative analysis

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.165

1 Introduction

Partially observable Markov decision processes (POMDPs). *Markov decision processes (MDPs)* are standard models for probabilistic systems that exhibit both probabilistic and nondeterministic behavior [16]. MDPs have been used to model and solve control problems for stochastic systems [13]: nondeterminism represents the freedom of the controller to choose a control action, while the probabilistic component of the behavior describes the system response to control actions. In *perfect-observation (or perfect-information) MDPs (PIMDPs)* the controller can observe the current state of the system to choose the next control actions, whereas in *partially observable MDPs (POMDPs)* the state space is partitioned according to observations that the controller can observe i.e., given the current state, the controller can only view the observation of the state (the partition the state belongs to), but not the precise state [22]. POMDPs provide the appropriate model to study a wide variety of applications such as in computational biology [12], speech processing [21], software verification [6], robot planning [17], to name a few. In verification of probabilistic systems, MDPs have been adopted as models for concurrent probabilistic systems [10], under-specified probabilistic systems [4], and applied in diverse domains [3, 18]. POMDPs also subsume many other powerful computational models such as probabilistic automata [25, 23] (since probabilistic automata are a special case of POMDPs with a single observation).

* The research was supported by Austrian Science Fund (FWF) Grant No P 23499- N23, FWF NFN Grant No S11407-N23 (RiSE), ERC Start grant (279307: Graph Games), and Microsoft faculty fellows award.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 165–180

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The class of ω-regular objectives. An objective specifies the desired set of behaviors (or paths) for the controller. In verification and control of stochastic systems an objective is typically an ω-regular set of paths. The class of ω-regular languages extends classical regular languages to infinite strings, and provides a robust specification language to express all commonly used specifications [28]. In a parity objective, every state of the MDP is mapped to a non-negative integer priority (or color) and the goal is to ensure that the minimum priority (or color) visited infinitely often is even. Parity objectives are a canonical way to define such ω-regular specifications. Thus POMDPs with parity objectives provide the theoretical framework to study problems such as the verification and control of stochastic systems.

Qualitative and quantitative analysis. The analysis of POMDPs with parity objectives can be classified into qualitative and quantitative analysis. Given a POMDP with a parity objective and a start state, the *qualitative analysis* asks whether the objective can be ensured with probability 1 (*almost-sure winning*) or positive probability (*positive winning*); whereas the *quantitative analysis* asks whether the objective can be satisfied with probability at least λ for a given threshold $\lambda \in (0, 1)$.

Importance of qualitative analysis. The qualitative analysis of MDPs is an important problem in verification that is of interest independent of the quantitative analysis problem. There are many applications where we need to know whether the correct behavior arises with probability 1. For instance, when analyzing a randomized embedded scheduler, we are interested in whether every thread progresses with probability 1 [11]. Even in settings where it suffices to satisfy certain specifications with probability $\lambda < 1$, the correct choice of λ is a challenging problem, due to the simplifications introduced during modeling. For example, in the analysis of randomized distributed algorithms it is quite common to require correctness with probability 1 (see, e.g., [24, 27]). Furthermore, in contrast to quantitative analysis, qualitative analysis is robust to numerical perturbations and modeling errors in the transition probabilities. Thus qualitative analysis of POMDPs with parity objectives is a fundamental theoretical problem in verification and analysis of probabilistic systems.

Previous results. On one hand POMDPs with parity objectives provide a rich framework to model a wide variety of practical problems, on the other hand, most theoretical results established for POMDPs are *negative* (undecidability) results. There are several deep undecidability results established for the special case of probabilistic automata (that immediately imply undecidability for the more general case of POMDPs). The basic undecidability results are for probabilistic automata over finite words (that can be considered as a special case of parity objectives). The quantitative analysis problem is undecidable for probabilistic automata over finite words [25, 23]; and it was shown in [19] that even the following approximation version is undecidable: for any fixed $0 < \epsilon < \frac{1}{2}$, given a probabilistic automaton and the guarantee that either (a) there is a word accepted with probability at least $1 - \epsilon$; or (ii) all words are accepted with probability at most ϵ; decide whether it is case (i) or case (ii). The almost-sure (resp. positive) problem for probabilistic automata over finite words reduces to the non-emptiness question of universal (resp. non-deterministic) automata over finite words and is PSPACE-complete (resp. solvable in polynomial time). However, another related decision question whether for every $\epsilon > 0$ there is a word that is accepted with probability at least $1 - \epsilon$ (the value 1 problem) is undecidable for probabilistic automata over finite words [14]. Also observe that all undecidability results for probabilistic automata over finite words carry over to POMDPs where the controller is restricted to finite-memory strategies. In [20], the authors consider POMDPs with finite-memory strategies under expected rewards, but the general problem remains undecidable. For qualitative analysis of

POMDPs with parity objectives, deep undecidability results were shown for very special cases of parity objectives (even in the special case of probabilistic automata). It was shown in [2] that the almost-sure (resp. positive) problem is undecidable for probabilistic automata with coBüchi (resp. Büchi) objectives which are special cases of parity objectives that use only two priorities. In summary the most important theoretical results are negative (they establish undecidability results).

Our contributions. The undecidability proofs for the qualitative analysis of POMDPs with parity objectives crucially require the use of *infinite-memory* strategies for the controller. In all practical applications, the controller must be a *finite-state* controller to be implementable. Thus for all practical purposes the relevant question is the existence of finite-memory controllers. The quantitative analysis problem remains undecidable even under finite-memory controllers as the undecidability results are established for probabilistic automata over finite words. In this work we study the most prominent remaining theoretical open question (that is also of practical relevance) for POMDPs with parity objectives that whether the qualitative analysis of POMDPs with parity objectives is decidable or undecidable for finite-memory strategies (i.e., finite-memory controllers). Our main result is the *positive* result that the qualitative analysis of POMDPs with parity objectives is *decidable* under finite-memory strategies. Moreover, for qualitative analysis of POMDPs with parity objectives under finite-memory strategies, we establish optimal complexity bounds both for strategy complexity as well as computational complexity. Our contributions are as follows (summarized in Table 1):

1. *(Strategy complexity).* Our first result shows that *belief-based stationary* strategies are not sufficient (where a belief-based stationary strategy is based on the subset construction that remembers the possible set of current states): we show that there exist POMDPs with coBüchi objectives where finite-memory almost-sure winning strategy exists but there exists no randomized belief-based stationary almost-sure winning strategy. All previous results about decidability for almost-sure winning in sub-classes of POMDPs crucially relied on the sufficiency of randomized belief-based stationary strategies that allowed standard techniques like subset construction to establish decidability. However, our counter-example shows that previous techniques based on simple subset construction (to construct an exponential size PIMDP) are not adequate to solve the problem. Before the result for parity objectives, we consider a slightly more general form of objectives, called Muller objectives. For a Muller objective a set $\mathcal{F}$ of subsets of colors is given and the set of colors visited infinitely often must belong to $\mathcal{F}$. We show our main result that given a POMDP with $|S|$ states and a Muller objective with d colors (priorities), if there is a finite-memory almost-sure (resp. positive) winning strategy, then there is an almost-sure (resp. positive) winning strategy that uses at most $\mathsf{Mem}^* = 2^{2\cdot|S|} \cdot (2^{2^d})^{|S|}$ memory. Developing on our result for Muller objectives, for POMDPs with parity objectives we show that if there is a finite-memory almost-sure (resp. positive) winning strategy, then there is an almost-sure (resp. positive) winning strategy that uses at most $2^{3\cdot d\cdot|S|}$ memory. Our exponential memory upper bound for parity objectives is optimal as it is shown in [8] that almost-sure winning strategies require at least exponential memory even for the very special case of reachability objectives in POMDPs.

2. *(Computational complexity).* We present an exponential time algorithm for the qualitative analysis of POMDPs with parity objectives under finite-memory strategies, and thus obtain an EXPTIME upper bound. The EXPTIME-hardness follows from [8] for

Table 1 Strategy and computational complexity for POMDPs. UB:Upper bound; LB: Lower bound. New results in bold fonts.

Objectives		Almost-sure		Positive	
		Inf. Mem.	Fin. Mem.	Inf. Mem.	Fin. Mem.
Büchi	Strategy	Exp . (belief)	Exp . (belief)	Inf. mem.	UB: **Exp. $2^{6\cdot\|S\|}$** LB: Exp. **(belief not suf.)**
	Complexity	EXP-c.	EXP-c.	Undec.	**EXP-c.**
coBüchi	Strategy	Inf. mem	UB: **Exp. $2^{6\cdot\|S\|}$** LB: Exp. **(belief not suf.)**	UB: Exp. LB: Exp. **(belief not suf.)**	UB: Exp. LB: Exp. **(belief not suf.)**
	Complexity	Undec.	**EXP-c.**	EXP-c.	EXP-c.
Parity	Strategy	Inf. mem	UB: **Exp. $2^{3\cdot d\cdot\|S\|}$** LB: Exp. **(belief not suf.)**	Inf. mem	UB: **Exp. $2^{3\cdot d\cdot\|S\|}$** LB: Exp. **(belief not suf.)**
	Complexity	Undec.	**EXP-c.**	Undec.	**EXP-c.**

the special case of reachability and safety objectives, and thus we obtain the optimal EXPTIME-complete computational complexity result.[1]

Technical contributions. The key technical contribution for the decidability result is as follows. Since belief-based stationary strategies are not sufficient, standard subset construction techniques do not work. For an arbitrary finite-memory strategy we construct a projected strategy that collapses memory states based on a projection graph construction given the strategy. The projected strategy at a collapsed memory state plays uniformly over actions that were played at all the corresponding memory states of the original strategy. The projected strategy thus plays more actions with positive probability. The key challenge is to show the bound on the size of the projection graph, and to show that the projected strategy, even though plays more actions, does not destroy the structure of the recurrent classes of the original strategy. For parity objectives, we show a reduction from general parity objectives to parity objectives with two priorities on a polynomially larger POMDP and from our general result for Muller objectives obtain the optimal memory complexity bounds for parity objectives. For the computational complexity result, we show how to construct an exponential size special class of POMDPs (which we call belief-observation POMDPs where the belief is always the current observation) and present polynomial time algorithms for the qualitative analysis of the special belief-observation POMDPs of our construction. Full proofs are available as technical report, Feb 20, 2013, `https://repository.ist.ac.at/109/`.

2 Definitions

In this section we present the basic definitions of POMDPs, strategies (policies), ω-regular objectives, and the winning modes.

Notations. For a finite set X, we denote by $\mathcal{P}(X)$ the set of subsets of X (the power set of X). A probability distribution f on X is a function $f : X \to [0, 1]$ such that $\sum_{x\in X} f(x) = 1$, and we denote by $\mathcal{D}(X)$ the set of all probability distributions on X. For $f \in \mathcal{D}(X)$ we denote by $\text{Supp}(f) = \{x \in X \mid f(x) > 0\}$ the support of f.

▶ **Definition 1** (POMDPs). A *Partially Observable Markov Decision Process (POMDP)* is a tuple $G = (S, A, \delta, \mathcal{O}, \gamma, s_0)$ where: (i) S is a finite set of states; (ii) A is a finite alphabet of

[1] Recently, Nain and Vardi (personal communication, to appear LICS 2013) considered the finite-memory strategies problem for one-sided partial-observation games and established 2EXPTIME upper bound. Our work is independent and establishes optimal (EXPTIME-complete) complexity bounds for POMDPs.

actions; (iii) $\delta : S \times A \to \mathcal{D}(S)$ is a *probabilistic transition function* that given a state s and an action $a \in A$ gives the probability distribution over the successor states, i.e., $\delta(s,a)(s')$ denotes the transition probability from state s to state s' given action a; (iv) $\mathcal{O}$ is a finite set of *observations*; (v) $\gamma : S \to \mathcal{O}$ is an *observation function* that maps every state to an observation; and (vi) s_0 is the initial state.

Given $s, s' \in S$ and $a \in A$, we also write $\delta(s'|s,a)$ for $\delta(s,a)(s')$. For an observation o, we denote by $\gamma^{-1}(o) = \{s \in S \mid \gamma(s) = o\}$ the set of states with observation o. For a set $U \subseteq S$ of states and $O \subseteq \mathcal{O}$ of observations we denote $\gamma(U) = \{o \in \mathcal{O} \mid \exists s \in U.\ \gamma(s) = o\}$ and $\gamma^{-1}(O) = \bigcup_{o \in O} \gamma^{-1}(o)$. For technical convenience we consider that the initial state s_0 has a unique observation.

Plays, cones and belief-updates. A *play* (or a path) in a POMDP is an infinite sequence $(s_0, a_0, s_1, a_1, s_2, a_2, \ldots)$ of states and actions such that for all $i \geq 0$ we have $\delta(s_i, a_i)(s_{i+1}) > 0$. We write Ω for the set of all plays. For a finite prefix $w \in (S \cdot A)^* \cdot S$ of a play, we denote by $\mathsf{Cone}(w)$ the set of plays with w as the prefix (i.e., the cone or cylinder of the prefix w), and denote by $\mathsf{Last}(w)$ the last state of w. For a finite prefix $w = (s_0, a_0, s_1, a_1, \ldots, s_n)$ we denote by $\gamma(w) = (\gamma(s_0), a_0, \gamma(s_1), a_1, \ldots, \gamma(s_n))$ the observation and action sequence associated with w. For a finite sequence $\rho = (o_0, a_0, o_1, a_1, \ldots, o_n)$ of observations and actions, the *belief* $\mathcal{B}(\rho)$ after the prefix ρ is the set of states in which a finite prefix of a play can be after the sequence ρ of observations and actions, i.e., $\mathcal{B}(\rho) = \{s_n = \mathsf{Last}(w) \mid w = (s_0, a_0, s_1, a_1, \ldots, s_n), w \text{ is a prefix of a play, and for all } 0 \leq i \leq n.\ \gamma(s_i) = o_i\}$. The belief-updates associated with finite-prefixes are as follows: for prefixes w and $w' = w \cdot a \cdot s$ the belief update is defined inductively as $\mathcal{B}(\gamma(w')) = \left(\bigcup_{s_1 \in \mathcal{B}(\gamma(w))} \mathrm{Supp}(\delta(s_1, a))\right) \cap \gamma^{-1}(s)$.

Strategies. A *strategy (or a policy)* is a recipe to extend prefixes of plays and is a function $\sigma : (S \cdot A)^* \cdot S \to \mathcal{D}(A)$ that given a finite history (i.e., a finite prefix of a play) selects a probability distribution over the actions. Since we consider POMDPs, strategies are *observation-based*, i.e., for all histories $w = (s_0, a_0, s_1, a_1, \ldots, a_{n-1}, s_n)$ and $w' = (s'_0, a_0, s'_1, a_1, \ldots, a_{n-1}, s'_n)$ such that for all $0 \leq i \leq n$ we have $\gamma(s_i) = \gamma(s'_i)$ (i.e., $\gamma(w) = \gamma(w')$), we must have $\sigma(w) = \sigma(w')$. In other words, if the observation sequence is the same, then the strategy cannot distinguish between the prefixes and must play the same. We now present an equivalent definition of strategies such that the memory is explicit.

▶ **Definition 2** (Strategies with memory and memoryless strategies)**.** A *strategy* with memory is a tuple $\sigma = (\sigma_u, \sigma_n, M, m_0)$ where: (i) *(Memory set).* M is a denumerable set (finite or infinite) of memory elements (or memory states). (ii) *(Action selection function).* The function $\sigma_n : M \to \mathcal{D}(A)$ is the *action selection function* that given the current memory state gives the probability distribution over actions. (iii) *(Memory update function).* The function $\sigma_u : M \times \mathcal{O} \times A \to \mathcal{D}(M)$ is the *memory update function* that given the current memory state, the current observation and action, updates the memory state probabilistically. (iv) *(Initial memory).* The memory state $m_0 \in M$ is the initial memory state. A strategy is a *finite-memory* strategy if the set M of memory elements is finite. A strategy is *pure (or deterministic)* if the memory update function and the action selection function are deterministic. A strategy is *memoryless (or stationary)* if it is independent of the history but depends only on the current observation, and can be represented as a function $\sigma : \mathcal{O} \to \mathcal{D}(A)$.

▶ Remark. It was shown in [7] that in POMDPs pure strategies are as powerful as randomized strategies, hence in sequel we omit discussions about pure strategies.

Probability measure. Given a strategy σ, the unique probability measure obtained given σ is denoted as $\mathbb{P}^\sigma(\cdot)$. We first define the measure $\mu^\sigma(\cdot)$ on cones. For $w = s_0$ we have

$\mu^\sigma(\mathsf{Cone}(w)) = 1$, and for $w = s$ where $s \neq s_0$ we have $\mu^\sigma(\mathsf{Cone}(w)) = 0$; and for $w' = w \cdot a \cdot s$ we have $\mu^\sigma(\mathsf{Cone}(w')) = \mu^\sigma(\mathsf{Cone}(w)) \cdot \sigma(w)(a) \cdot \delta(\mathsf{Last}(w), a)(s)$. By Caratheódary's extension theorem, the function $\mu^\sigma(\cdot)$ can be uniquely extended to a probability measure $\mathbb{P}^\sigma(\cdot)$ over Borel sets of infinite plays [5].

Objectives. An *objective* in a POMDP G is a measureable set $\varphi \subseteq \Omega$ of plays. For a play $\rho = (s_0, a_0, s_1, a_1, s_2 \ldots)$, we denote by $\mathrm{Inf}(\rho) = \{s \in S \mid \forall i \geq 0 \cdot \exists j \geq i : s_j = s\}$ the set of states that occur infinitely often in ρ. We consider the following objectives.

- *Reachability and safety objectives.* Given a set $\mathcal{T} \subseteq S$ of target states, the *reachability* objective $\mathsf{Reach}(\mathcal{T}) = \{(s_0, a_0, s_1, a_1, s_2 \ldots) \in \Omega \mid \exists k \geq 0 : s_k \in \mathcal{T}\}$ requires that a target state in $\mathcal{T}$ is visited at least once. Dually, the *safety* objective $\mathsf{Safe}(\mathcal{T}) = \{(s_0, a_0, s_1, a_1, s_2 \ldots) \in \Omega \mid \forall k \geq 0 : s_k \in \mathcal{T}\}$ requires that only states in $\mathcal{T}$ are visited.
- *Büchi and coBüchi objectives.* Given a set $\mathcal{T} \subseteq S$ of target states, the *Büchi* objective $\mathsf{Buchi}(\mathcal{T}) = \{\rho \in \Omega \mid \mathrm{Inf}(\rho) \cap \mathcal{T} \neq \emptyset\}$ requires that a state in $\mathcal{T}$ is visited infinitely often. Dually, the *coBüchi* objective $\mathsf{coBuchi}(\mathcal{T}) = \{\rho \in \Omega \mid \mathrm{Inf}(\rho) \subseteq \mathcal{T}\}$ requires that only states in $\mathcal{T}$ are visited infinitely often.
- *Parity objectives.* For $d \in \mathbb{N}$, let $p : S \to \{0, 1, \ldots, d\}$ be a *priority function* that maps each state to a non-negative integer priority. The *parity* objective $\mathsf{Parity}(p) = \{\rho \in \Omega \mid \min\{p(s) \mid s \in \mathrm{Inf}(\rho)\} \text{ is even}\}$ requires that the smallest priority that appears infinitely often is even.
- *Muller objectives.* Let D be a set of colors, and $\mathsf{col} : S \to D$ be a color mapping function that maps every state to a color. A Muller objective $\mathcal{F}$ consists of a set of subsets of colors and requires that the set of colors visited infinitely often belongs to $\mathcal{F}$, i.e., $\mathcal{F} \in \mathcal{P}(\mathcal{P}(D))$ and $\mathsf{Muller}(\mathcal{F}) = \{\rho \in \Omega \mid \{\mathsf{col}(s) \mid s \in \mathrm{Inf}(\rho)\} \in \mathcal{F}\}$.

Given a set $U \subseteq S$ we will denote by $p(U)$ the set of priorities of the set U given by the priority function p, i.e., $p(U) = \{p(s) \mid s \in U\}$, and similarly $\mathsf{col}(U) = \{\mathsf{col}(s) \mid s \in U\}$. Büchi and coBüchi objectives are parity objectives with two priorities; and parity objectives are a special case of Muller objectives. However, given a POMDP with a Muller objective with color set D, an equivalent POMDP with $|S| \cdot |D|!$ states and a parity objective with $|D|^2$ priorities can be constructed with the latest appearance record (LAR) construction of [15].

Winning modes. Given a POMDP, an objective φ, and a class $\mathcal{C}$ of strategies, we say that: a strategy $\sigma \in \mathcal{C}$ is *almost-sure winning* (resp. *positive winning*) if $\mathbb{P}^\sigma(\varphi) = 1$ (resp. $\mathbb{P}^\sigma(\varphi) > 0$); and a strategy $\sigma \in \mathcal{C}$ is *quantitative winning*, for a threshold $\lambda \in (0, 1)$, if $\mathbb{P}^\sigma(\varphi) \geq \lambda$. We first precisely summarize related works in the following Theorem.

▶ **Theorem 3** (Previous results [25, 23, 2, 26, 8]). *The following assertions hold for POMDPs with the class $\mathcal{C}$ of all infinite-memory (randomized or pure) strategies: (1) The quantitative winning problem is undecidable for safety, reachability, Büchi, coBüchi, parity, and Muller objectives. (2) The almost-sure winning problem is EXPTIME-complete for safety, reachability, and Büchi objectives; and undecidable for coBüchi, parity, and Muller objectives. (3) The positive winning problem is PTIME-complete for reachability objectives, EXPTIME-complete for safety and coBüchi objectives; and undecidable for Büchi, parity, and Muller objectives.*

Explanation of the previous results and implications under finite-memory strategies. All the undecidability results follow from the special case of probabilistic automata: the undecidability of the quantitative problem for probabilistic automata follows from [25, 23, 9]. The undecidability for positive winning for Büchi and almost-sure winning for coBüchi objectives was established in [1, 2]. For the decidable results, the optimal complexity results for safety

objectives can be obtained from the results of [26] and all the other results follow from [8, 2]. If the classes of strategies are restricted to finite-memory strategies, then the undecidability results for quantitative winning still hold, as they are established for reachability objectives and for reachability objectives finite-memory suffices. The most prominent and important open question is whether the almost-sure and positive winning problems are decidable for parity and Muller objectives in POMDPs under finite-memory strategies.

3 Strategy Complexity

In this section we will first show that belief-based stationary strategies are not sufficient for finite-memory almost-sure winning strategies in POMDPs with coBüchi objectives; and then present the upper bound on memory size required for finite-memory almost-sure and positive winning strategies in POMDPs with Muller objectives, and finally for parity objectives. We start with some basic results about Markov chains.

Markov chains, recurrent classes, and reachability. A Markov chain $\overline{G} = (\overline{S}, \overline{\delta})$ consists of a *finite* set $\overline{S}$ of states and a probabilistic transition function $\overline{\delta} : \overline{S} \to \mathcal{D}(\overline{S})$. Given the Markov chain, we consider the graph $(\overline{S}, \overline{E})$ where $\overline{E} = \{(\overline{s}, \overline{s}') \mid \delta(\overline{s}' \mid \overline{s}) > 0\}$. A *recurrent class* $\overline{C} \subseteq \overline{S}$ of the Markov chain is a bottom strongly connected component (scc) in the graph $(\overline{S}, \overline{E})$ (a bottom scc is an scc with no edges out of the scc). We denote by $\mathsf{Rec}(\overline{G})$ the set of recurrent classes of the Markov chain, i.e., $\mathsf{Rec}(\overline{G}) = \{\overline{C} \mid \overline{C} \text{ is a recurrent class}\}$. Given a state $\overline{s}$ and a set $\overline{U}$ of states, we say that $\overline{U}$ is reachable from $\overline{s}$ if there is a path from $\overline{s}$ to some state in $\overline{U}$ in the graph $(\overline{S}, \overline{E})$. Given a state $\overline{s}$ of the Markov chain we denote by $\mathsf{Rec}(\overline{G})(\overline{s}) \subseteq \mathsf{Rec}(\overline{G})$ the subset of the recurrent classes reachable from $\overline{s}$ in $\overline{G}$. A state is recurrent if it belongs to a recurrent class.

▶ **Lemma 4.** *For a Markov chain $\overline{G} = (\overline{S}, \overline{\delta})$ with Muller objective $\mathsf{Muller}(\mathcal{F})$ (or parity objective $\mathsf{Parity}(p)$), a state $\overline{s}$ is almost-sure winning (resp. positive winning) if for all recurrent classes $\overline{C} \in \mathsf{Rec}(\overline{G})(\overline{s})$ (resp. for some recurrent class $\overline{C} \in \mathsf{Rec}(\overline{G})(\overline{s})$) reachable from $\overline{s}$ we have $\mathsf{col}(\overline{C}) \in \mathcal{F}$ ($\min(p(\overline{C}))$ is even for the parity objective).*

Markov chains $G \restriction \sigma$ under finite-memory strategies σ. We now define Markov chains obtained by fixing finite-memory strategies in a POMDP G. A finite-memory strategy $\sigma = (\sigma_u, \sigma_n, M, m_0)$ induces a finite-state Markov chain $(S \times M, \delta_\sigma)$, denoted $G \restriction \sigma$, with the probabilistic transition function $\delta_\sigma : S \times M \to \mathcal{D}(S \times M)$: given $s, s' \in S$ and $m, m' \in M$, the transition $\delta_\sigma\big((s', m') \mid (s, m)\big)$ is the probability to go from state (s, m) to state (s', m') in one step under the strategy σ. The probability of transition can be decomposed as follows: (i) First an action $a \in A$ is sampled according to the distribution $\sigma_n(m)$; (ii) then the next state s' is sampled according to the distribution $\delta(s, a)$; and (iii) finally the new memory m' is sampled according to the distribution $\sigma_u(m, \gamma(s'), a)$ (i.e., the new memory is sampled according to σ_u given the old memory, new observation and the action). More formally, we have: $\delta_\sigma\big((s', m') \mid (s, m)\big) = \sum_{a \in A} \sigma_n(m)(a) \cdot \delta(s, a)(s') \cdot \sigma_u(m, \gamma(s'), a)(m')$.

Belief-based stationary strategies not sufficient. For all previous decidability results for almost-sure winning in POMDPs, the key was to show that *belief-based stationary* strategies are sufficient. In POMDPs with Büchi objectives, belief-based stationary strategies are sufficient for almost-sure winning, and we now show that in POMDPs with coBüchi objectives finite-memory almost-sure winning strategies may exist whereas no belief-based stationary ones.

▶ **Example 5.** We consider a POMDP G with state space $\{s_0, X, X', Y, Y', Z, Z'\}$ and action set $\{a, b\}$, and let $U = \{X, X', Y, Y', Z, Z'\}$. From the initial state s_0 all the other states

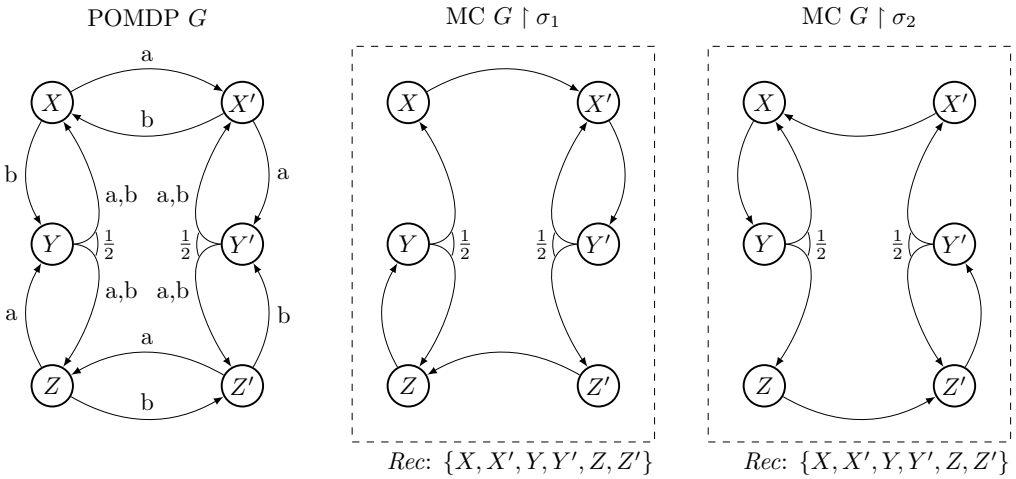


Figure 1 Belief is not sufficient.

are reached with uniform probability in one-step, i.e., for all $s' \in U = \{X, X', Y, Y', Z, Z'\}$ we have $\delta(s_0, a)(s') = \delta(s_0, b)(s') = \frac{1}{6}$. The transitions from the other states (shown in Figure 1) are as follows: (i) $\delta(X, a)(X') = 1$ and $\delta(X, b)(Y) = 1$; (ii) $\delta(X', a)(Y') = 1$ and $\delta(X', b)(X) = 1$; (iii) $\delta(Z, a)(Y) = 1$ and $\delta(Z, b)(Z') = 1$; (iv) $\delta(Z', a)(Z) = 1$ and $\delta(Z', b)(Y') = 1$; (v) $\delta(Y, a)(X) = \delta(Y, b)(X) = \delta(Y, a)(Z) = \delta(Y, b)(Z) = \frac{1}{2}$; and (vi) $\delta(Y', a)(X') = \delta(Y', b)(X') = \delta(Y', a)(Z') = \delta(Y', b)(Z') = \frac{1}{2}$. All states in U have the same observation. The coBüchi objective is given by the target set $\{X, X', Z, Z'\}$, i.e., Y and Y' must be visited only finitely often. The belief initially after one-step is the set U since from s_0 all of them are reached with positive probability. The belief is always the set U since every state has an input edge for every action, i.e., if the current belief is U (i.e., the set of states that the POMDP is currently in with positive probability is U), then irrespective of whether a or b is chosen all states of U are reached with positive probability and hence the belief set is again U. There are three belief-based stationary strategies: (i) σ_1 that plays always a; (ii) σ_2 that plays always b; or (iii) σ_3 that plays both a and b with positive probability. For all the three strategies, the Markov chains obtained have the whole set U as the recurrent class (see Figure 1 for the Markov chains $G \upharpoonright \sigma_1$ and $G \upharpoonright \sigma_2$), and hence both Y and Y' are visited infinitely often with probability 1 violating the coBüchi objective. The strategy σ_4 that plays action a and b alternately gives rise to the Markov chain $G \upharpoonright \sigma_4$ shown in Figure 2 (i.e., σ_4 has two memory states a and b, in memory state a it plays action a and switches to memory state b, and in memory state b it plays action b and switches to memory state a). The recurrent classes do not intersect with (Y, m) or (Y', m), for memory state $m \in \{a, b\}$, and hence σ_4 is a finite-memory almost-sure winning strategy. ◀

Upper bound on memory. For the following of the section, we fix a POMDP $G = (S, A, \delta, \mathcal{O}, \gamma, s_0)$, with a Muller objective $\mathsf{Muller}(\mathcal{F})$ with the set D of colors and a color mapping function col. We will denote by $\mathfrak{D}$ the powerset of the powerset of the set D of colors, i.e., $\mathfrak{D} = \mathcal{P}(\mathcal{P}(D))$; and note that $|\mathfrak{D}| = 2^{2^d}$, where $d = |D|$. Our goal is to prove the following fact: given a finite-memory almost-sure (resp. positive) winning strategy σ on G there exists a finite-memory almost-sure (resp. positive) winning strategy σ' on G, of memory size at most $\mathsf{Mem}^* = 2^{|S|} \cdot 2^{|S|} \cdot |\mathfrak{D}|^{|S|}$.

Overview of the proof. We first present an overview of our proof. (i) Given an arbitrary finite-memory strategy σ we will consider the Markov chain $G \upharpoonright \sigma$ arising by fixing the

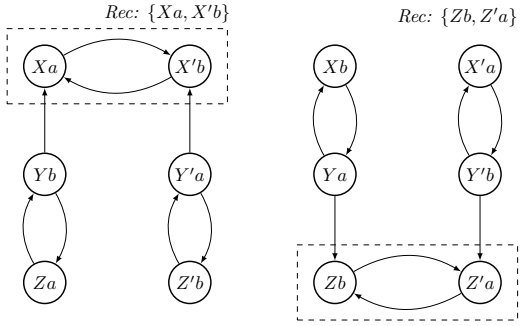


Figure 2 The Markov chain $G \upharpoonright \sigma_4$.

strategy. (ii) Given the Markov chain we will define a projection graph that depends on the recurrent classes of the Markov chain. The projection graph is of size at most Mem^*. (iii) Given the projection graph we will construct a projected strategy with memory size at most Mem^* that preserves the recurrent classes of the Markov chain $G \upharpoonright \sigma$.

Notations. Given $Z \in \mathfrak{D}^{|S|}$ and given $s \in S$, we write $Z(s)$ (which is in $\mathfrak{D} = \mathcal{P}(\mathcal{P}(D))$) for the s-component of Z. For two sets U_1 and U_2 and $U \subseteq U_1 \times U_2$, we denote by $\mathsf{Proj}_i(U)$ for $i \in \{1, 2\}$ the projection of U on the i-th component.

Basic definitions for the projection graph. We now introduce notions associated with the finite Markov chain $G \upharpoonright \sigma$ that will be essential in defining the projection graph.

▶ **Definition 6** (Recurrence set functions)**.** Let σ be a finite-memory strategy with memory M on G for the Muller objective with the set D of colors, and let $m \in M$.

- *(Function set recurrence).* The function $\mathsf{SetRec}_\sigma(m) : S \to \mathfrak{D}$ maps every state $s \in S$ to the projections of colors of recurrent classes reachable from (s, m) in $G \upharpoonright \sigma$. Formally, $\mathsf{SetRec}_\sigma(m)(s) = \{\mathsf{col}(\mathsf{Proj}_1(U)) \mid U \in \mathsf{Rec}(G \upharpoonright \sigma)((s, m))\}$, i.e., we consider the set $\mathsf{Rec}(G \upharpoonright \sigma)((s, m))$ of recurrent classes reachable from the state (s, m) in $G \upharpoonright \sigma$, obtain the projections on the state space S and consider the colors of states in the projected set. We will in sequel consider $\mathsf{SetRec}_\sigma(m) \in \mathfrak{D}^{|S|}$.
- *(Function boolean recurrence).* The function $\mathsf{BoolRec}_\sigma(m) : S \to \{0, 1\}$ is such that for all $s \in S$, we have $\mathsf{BoolRec}_\sigma(m)(s) = 1$ if there exists $U \in \mathsf{Rec}(G \upharpoonright \sigma)((s, m))$ such that $(s, m) \in U$, and 0 if not. Intuitively, $\mathsf{BoolRec}_\sigma(m)(s) = 1$ if (s, m) belongs to a recurrent class in $G \upharpoonright \sigma$ and 0 otherwise. In sequel we will consider $\mathsf{BoolRec}_\sigma(m) \in \{0, 1\}^{|S|}$.

▶ **Lemma 7.** *Let $s, s' \in S$ and $m, m' \in M$ be such that (s', m') is reachable from (s, m) in $G \upharpoonright \sigma$. Then $\mathsf{SetRec}_\sigma(m')(s') \subseteq \mathsf{SetRec}_\sigma(m)(s)$.*

▶ **Definition 8** (Projection graph)**.** Let σ be a finite-memory strategy. We define the *projection graph* $\mathsf{PrGr}(\sigma) = (V, E)$ associated to σ as follows:

- *(Vertex set).* The set of vertices is $V = \{(U, \mathsf{BoolRec}_\sigma(m), \mathsf{SetRec}_\sigma(m)) \mid U \subseteq S \text{ and } m \in M\}$.
- *(Edge labels).* The edges are labeled by actions in A.
- *(Edge set).* Let $U \subseteq S$, $m \in M$ and $a \in \mathrm{Supp}(\sigma_n(m))$. Let $\overline{U} = \bigcup_{s \in U} \mathrm{Supp}(\delta(s, a))$ denote the set of possible successors of states in U given action a. We add the following set of edges in E: Given (U', m') such that there exists $o \in \mathcal{O}$ with $\gamma^{-1}(o) \cap \overline{U} = U'$ and $m' \in \mathrm{Supp}(\sigma_u(m, o, a))$, we add the edge $(U, \mathsf{BoolRec}_\sigma(m), \mathsf{SetRec}_\sigma(m)) \xrightarrow{a} (U', \mathsf{BoolRec}_\sigma(m'), \mathsf{SetRec}_\sigma(m'))$ to E. Intuitively, the update from U to U' is the update of the belief, i.e., if the previous belief is the set U of states, and the current observation

is o, then the new belief is U'; the update of m to m' is according to the support of the memory update function; and the $\mathsf{BoolRec}$ and SetRec functions for the memories are given by σ.

- *(Initial vertex).* The *initial vertex* of $\mathsf{PrGr}(\sigma)$ is the vertex $(\{s_0\}, \mathsf{BoolRec}_\sigma(m_0), \mathsf{SetRec}_\sigma(m_0))$.

Note that $V \subseteq \mathcal{P}(S) \times \{0,1\}^{|S|} \times \mathfrak{D}^{|S|}$, and hence $|V| \leq \mathsf{Mem}^*$. For the rest of the section we fix a finite-memory strategy σ that uses memory M. We now define projected strategies: intuitively the projected strategy collapses memory with same $\mathsf{BoolRec}$ and SetRec functions, and at a collapsed memory state plays uniformly the union of the actions played at the corresponding memory states.

▶ **Definition 9** (Projected strategy $proj(\sigma)$)**.** Let $\mathsf{PrGr}(\sigma) = (V, E)$ be the projection graph of σ. We define the following projected strategy $\sigma' = proj(\sigma) = (\sigma'_u, \sigma'_n, M', m'_0)$:

- *(Memory set).* The memory set of $proj(\sigma)$ is $M' = V = \{(U, \mathsf{BoolRec}_\sigma(m), \mathsf{SetRec}_\sigma(m)) \mid U \subseteq S \text{ and } m \in M\}$.
- *(Initial memory).* The initial memory state of $proj(\sigma)$ is $m'_0 = (\{s_0\}, \mathsf{BoolRec}_\sigma(m_0), \mathsf{SetRec}_\sigma(m_0))$.
- *(Memory update).* Let $m = (U, B, L) \in M'$, $o \in \mathcal{O}$ and $a \in A$. Then $\sigma'_u(m, o, a)$ is the uniform distribution over the set $\{m' = (U', B', L') \in M' \mid m \xrightarrow{a} m' \in E \text{ and } U' \subseteq \gamma^{-1}(o)\}$.
- *(Action selection).* Given $m \in M'$, the action selection function $\sigma'_n(m)$ is the uniform distribution over $\{a \in A \mid \exists m' \in M' \text{ s.t. } m \xrightarrow{a} m' \in E\}$.

Let $(V, E) = \mathsf{PrGr}(\sigma)$ be the projection graph, and let $\sigma' = proj(\sigma)$ be the projected strategy. The chain $G \upharpoonright \sigma'$ is a finite-state Markov chain, with state space $S \times M'$, which is a subset of $S \times \mathcal{P}(S) \times \{0,1\}^{|S|} \times \mathfrak{D}^{|S|}$.

Random variable notations. For all $n \geq 0$ we write X_n, Y_n, C_n, Z_n, W_n for the random variables which correspond respectively to the projection of the n-th state of the Markov chain $G \upharpoonright \sigma'$ on the S component, the $\mathcal{P}(S)$ component, the $\{0,1\}^{|S|}$ component, the $\mathfrak{D}^{|S|}$ component, and the n-th action, respectively.

Run of the Markov chain $G \upharpoonright \sigma'$. A *run* on $G \upharpoonright \sigma'$ is a sequence $r = (X_0, Y_0, C_0, Z_0) \xrightarrow{W_0} (X_1, Y_1, C_1, Z_1) \xrightarrow{W_1} \ldots$ such that each finite prefix of r is generated with positive probability on the chain, i.e., for all $i \geq 0$, we have (i) $W_i \in \mathrm{Supp}(\sigma'_n(Y_i, C_i, Z_i))$; (ii) $X_{i+1} \in \mathrm{Supp}(\delta(X_i, W_i))$; and (iii) $(Y_{i+1}, C_{i+1}, Z_{i+1}) \in \mathrm{Supp}(\sigma'_u((Y_i, C_i, Z_i), \gamma(X_{i+1}), W_i))$.

In the following lemma we show that reachability in the Markov chain $G \upharpoonright \sigma$ implies reachability in the Markov chain $G \upharpoonright \sigma'$. Intuitively, the result follows from the fact that the projected strategy σ' plays in the collapsed memory state uniformly all actions that were played at all the corresponding memory states of the original strategy σ.

▶ **Lemma 10.** *Let $\sigma' = proj(\sigma)$ be the projected strategy of σ. Given $s, s' \in S$ and $m, m' \in M$, if (s', m') is reachable from (s, m) in $G \upharpoonright \sigma$, then for all $Y \subseteq S$ such that $(s, Y, \mathsf{BoolRec}_\sigma(m), \mathsf{SetRec}_\sigma(m))$ is a state of $G \upharpoonright \sigma'$, there exists $Y' \subseteq S$ such that $(s', Y', \mathsf{BoolRec}_\sigma(m'), \mathsf{SetRec}_\sigma(m'))$ is reachable from $(s, Y, \mathsf{BoolRec}_\sigma(m), \mathsf{SetRec}_\sigma(m))$ in $G \upharpoonright \sigma'$.*

Proof. Suppose first that (s', m') is reachable from (s, m) in $G \upharpoonright \sigma$ in one step. Let $Y \subseteq S$ be such that $(s, Y, \mathsf{BoolRec}_\sigma(m), \mathsf{SetRec}_\sigma(m))$ is a state of $G \upharpoonright \sigma'$. Then there exists an edge in the projection graph of σ from $(Y, \mathsf{BoolRec}_\sigma(m), \mathsf{SetRec}_\sigma(m))$ to another vertex $(Y', \mathsf{BoolRec}_\sigma(m'), \mathsf{SetRec}_\sigma(m'))$. As a consequence, there exists $Y' \subseteq S$ such

that $(s', Y', \mathsf{BoolRec}_\sigma(m'), \mathsf{SetRec}_\sigma(m'))$ is reachable from $(s, Y, \mathsf{BoolRec}_\sigma(m), \mathsf{SetRec}_\sigma(m))$ in $G \restriction \sigma'$.

We conclude the proof by induction: if (s', m') is reachable from (s, m) in $G \restriction \sigma$, then there exists a sequence of couples $(s_1, m_1), (s_2, m_2), ..., (s_i, m_i)$ such that $(s_1, m_1) = (s, m)$, $(s_i, m_i) = (s', m')$, and for all $j \in \{1, ..., i-1\}$ we have that (s_{j+1}, m_{j+1}) is reachable from (s_j, m_j) in one step. Using the proof for an elementary step (or one step) inductively on such a sequence, we get the result. ◀

In the following lemma we establish the crucial properties of the Markov chain obtained from the projected strategy.

▶ **Lemma 11.** *Let $X_0 \in S$, $Y_0 \in \mathcal{P}(S)$, $C_0 \in \{0,1\}^{|S|}$ and $Z_0 \in \mathfrak{D}^{|S|}$, and let $r = (X_0, Y_0, C_0, Z_0) \xrightarrow{W_0} (X_1, Y_1, C_1, Z_1) \xrightarrow{W_1} ...$ be a run on $G \restriction \sigma'$ with a starting state (X_0, Y_0, C_0, Z_0). Then for all $n \geq 0$ the following assertions hold:*

- $X_{n+1} \in \mathrm{Supp}(\delta(X_n, W_n))$.
- *$Z_n(X_n)$ is not empty.*
- $Z_{n+1}(X_{n+1}) \subseteq Z_n(X_n)$.
- $(Y_n, C_n, Z_n) \xrightarrow{W_n} (Y_{n+1}, C_{n+1}, Z_{n+1})$ *is an edge in E, where $(V, E) = \mathsf{PrGr}(\sigma)$.*
- *If $C_n(X_n) = 1$, then $C_{n+1}(X_{n+1}) = 1$.*
- *If $C_n(X_n) = 1$, then $|Z_n(X_n)| = 1$; and if $\{Z\} = Z_n(X_n)$, then for all $j \geq 0$ we have $\mathsf{col}(X_{n+j}) \in Z$.*

Proof. We prove the last point. Suppose (X_n, Y_n, C_n, Z_n) is such that $C_n(X_n) = 1$. Let $m \in M$ be an arbitrary memory state such that $C_n = \mathsf{BoolRec}_\sigma(m)$ and $Z_n = \mathsf{SetRec}_\sigma(m)$. By hypothesis, since $C_n(X_n) = 1$, it follows that (X_n, m) is a recurrent state in the Markov chain $G \restriction \sigma$. As a consequence, only one recurrent class $R \subseteq S \times M$ of $G \restriction \sigma$ is reachable from (X_n, m), and (X_n, m) belongs to this class. Hence $Z_n(X_n) = \{\mathsf{col}(\mathsf{Proj}_1(R))\}$, and thus $|Z_n(X_n)| = 1$. It also follows that all states (X', m') reachable in one step from (X_n, m) also belong to the recurrent class R. It follows that $X_{n+1} \in \mathsf{Proj}_1(R)$ and hence $\mathsf{col}(X_{n+1}) \in \mathsf{col}(\mathsf{Proj}_1(R))$. By induction for all $j \geq 0$ we have $\mathsf{col}(X_{n+j}) \in \mathsf{col}(\mathsf{Proj}_1(R))$. The desired result follows. ◀

We now introduce the final notion that is required to complete the proof. The notion is that of a pseudo-recurrent state. Intuitively a state (X, Y, C, Z) is pseudo-recurrent if Z contains exactly one recurrent subset, X belongs to the subset and it will follow that for some memory $m \in M$ (of certain desired property) (X, m) is a recurrent state in the Markov chain $G \restriction \sigma$. The important property that is useful is that once a pseudo-recurrent state is reached, then C and Z remain invariant (follows from Lemma 11).

▶ **Definition 12** (Pseudo-recurrent states)**.** Let $X \in S$, $Y \subseteq S$, $C \in \{0,1\}^{|S|}$, and $Z \in \mathfrak{D}^{|S|}$. Then the state (X, Y, C, Z) is called *pseudo-recurrent* if there exists $Z_\infty \subseteq D$ such that: (i) $Z(X) = \{Z_\infty\}$, (ii) $\mathsf{col}(X) \in Z_\infty$, and (iii) $C(X) = 1$.

▶ **Lemma 13.** *Let (X, Y, C, Z) be a pseudo-recurrent state. If (X', Y', C', Z') is reachable from (X, Y, C, Z) in $G \restriction \sigma'$, then (X', Y', C', Z') is also a pseudo-recurrent state and $Z'(X') = Z(X)$.*

We establish the following key properties of pseudo-recurrent states with the aid of the properties of Lemma 11. Firstly, with probability 1 a run of a Markov chain $G \restriction \sigma'$ reaches a pseudo-recurrent state.

▶ **Lemma 14.** *Let $X \in S$, $Y \in \mathcal{P}(S)$, $C \in \{0,1\}^{|S|}$, and $Z \in \mathfrak{D}^{|S|}$. Then almost-surely (with probability 1) a run on $G \restriction \sigma'$ from any starting state (X, Y, C, Z) reaches a pseudo-recurrent state.*

Proof. We show that given (X, Y, C, Z) there exists a pseudo-recurrent state (X', Y', C', Z') which is reachable from (X, Y, C, Z) in $G \restriction \sigma'$. First let us consider the Markov chain $G \restriction \sigma$ obtained from the original finite-memory strategy σ with memory M. Let $m \in M$ be such that $C = \mathsf{BoolRec}_\sigma(m)$ and $Z = \mathsf{SetRec}_\sigma(m)$. We will now show that the result is a consequence of Lemma 10. First we know that there exists $t \in S$ and $m' \in M$ such that (t, m') is recurrent and reachable from (X, m) with positive probability in $G \restriction \sigma$. Let $R \subseteq S \times M$ be the unique recurrent class such that $(t, m') \in R$, and $Z_\infty = \{\mathsf{col}(\mathsf{Proj}_1(R))\}$. By Lemma 10, this implies that from (X, Y, C, Z) we can reach a state (X', Y', C', Z') such that (i) $X' = t$; (ii) $Z'(X') = \{Z_\infty\}$; (iii) $\mathsf{col}(X') \in Z_\infty$; and (iv) $C'(X') = 1$. Hence (X', Y', C', Z') is a pseudo-recurrent state. This shows that from all states with positive probability a pseudo-recurrent state is reached, and since it holds for all states with positive probability, it follows that it holds for all states with probability 1. ◀

Moreover, for every projection Z_B of a reachable recurrent class in the Markov chain $G \restriction \sigma$, there exists a pseudo-recurrent state (X', Y', C', Z') reachable in $G \restriction \sigma'$ such that $Z'(X') = \{Z_B\}$.

▶ **Lemma 15.** *Let (X, Y, C, Z) be a state of $G \restriction \sigma'$, and let $Z_B \in Z(X)$. Then there exists a pseudo-recurrent state (X', Y', C', Z') which is reachable from (X, Y, C, Z) and such that $Z'(X') = \{Z_B\}$.*

Finally, if we consider a pseudo-recurrent state, and consider the projection on the state space of the POMDP G of the recurrent classes reachable and consider the colors, then they coincide with $Z(X)$.

▶ **Lemma 16.** *Let (X, Y, C, Z) be a pseudo-recurrent state, then we have $Z(X) = \mathsf{SetRec}_{\sigma'}(m')(X)$, where $m' = (Y, C, Z)$.*

Proof. Let (X, Y, C, Z) be a pseudo-recurrent state, and let Z_∞ be such that $Z(X) = \{Z_\infty\}$. First, by Lemma 13, we know that if (X', Y', C', Z') is reachable from (X, Y, C, Z) in $G \restriction \sigma'$, then $\mathsf{col}(X') \in Z_\infty$. This implies that for all $Z_B \in \mathsf{SetRec}_{\sigma'}(m')(X)$, where $m' = (Y, C, Z)$, we have $Z_B \subseteq Z_\infty$. Second, by Lemma 10, if (X', Y', C', Z') is reachable from (X, Y, C, Z) in $G \restriction \sigma'$ and $\ell \in Z_\infty$, then there exists (X'', Y'', C'', Z'') reachable from (X', Y', C', Z') such that $\mathsf{col}(X'') = \ell$. This implies that for all $Z_B \in \mathsf{SetRec}_{\sigma'}(m')(X)$, where $m' = (Y, C, Z)$, we have $Z_\infty \subseteq Z_B$. Thus, $\mathsf{SetRec}_{\sigma'}(m')(X) = \{Z_\infty\} = Z(X)$. ◀

With the key properties we prove the main lemma (Lemma 17) which shows that the color sets of the projections of the recurrent classes on the state space of the POMDP coincide for σ and $\sigma' = \mathit{proj}(\sigma)$. Lemma 17 and Lemma 4 yield Theorem 18.

▶ **Lemma 17.** *Consider a finite-memory strategy $\sigma = (\sigma_u, \sigma_n, M, m_0)$ and the projected strategy $\sigma' = \mathit{proj}(\sigma) = (\sigma'_u, \sigma'_n, M', m'_0)$. Then we have $\mathsf{SetRec}_{\sigma'}(m'_0)(s_0) = \mathsf{SetRec}_\sigma(m_0)(s_0)$; i.e., the colors of the projections of the recurrent classes of the two strategies on the state space of the POMDP G coincide.*

Proof. For the proof, let $X = s_0$, $Y = \{s_0\}$, $C = \mathsf{BoolRec}_\sigma(m_0)$, $Z = \mathsf{SetRec}_\sigma(m_0)$. We need to show that $\mathsf{SetRec}_{\sigma'}(m'_0)(X) = Z(X)$, where $m'_0 = (Y, C, Z)$. We show inclusion in both directions.

First inclusion:$(Z(X) \subseteq \mathsf{SetRec}_{\sigma'}(m'_0)(X))$. Let $Z_B \in Z(X)$. By Lemma 15, there exists (X', Y', C', Z') which is reachable in $G \restriction \sigma'$ from (X, Y, C, Z), which is pseudo-recurrent, and such that $Z'(X') = \{Z_B\}$. By Lemma 16, we have $Z'(X') = \mathsf{SetRec}_{\sigma'}(m')(X')$ where $m' = (Y', C', Z')$. By Lemma 7, we have $\mathsf{SetRec}_{\sigma'}(m')(X') \subseteq \mathsf{SetRec}_{\sigma'}(m'_0)(X)$. This proves that $Z_B \in \mathsf{SetRec}_{\sigma'}(m'_0)(X)$.
Second inclusion: $(\mathsf{SetRec}_{\sigma'}(m'_0)(X) \subseteq Z(X))$. Conversely, let $Z_B \in \mathsf{SetRec}_{\sigma'}(m'_0)(X)$. Since $G \restriction \sigma'$ is a finite Markov chain, there exists (X', Y', C', Z') which is reachable from (X, Y, C, Z) in $G \restriction \sigma'$ and such that:

- $\{Z_B\} = \mathsf{SetRec}_{\sigma'}(m')(X')$, where $m' = (Y', C', Z')$.
- For all (X'', Y'', C'', Z'') reachable from (X', Y', C', Z') in $G \restriction \sigma'$ we have $\{Z_B\} = \mathsf{SetRec}_{\sigma'}(m'')(X'')$ where $m'' = (Y'', C'', Z'')$.

The above follows from the following property of a finite Markov chain: given a state s of a finite Markov chain and a recurrent class R reachable from s, from all states t of R the recurrent class reachable from t is R only. The condition is preserved by a projection on colors of states in R. By Lemma 14, there exists a pseudo-recurrent state (X'', Y'', C'', Z'') which is reachable from (X', Y', C', Z', W') in $G \restriction \sigma'$. By Lemma 16, we know that $Z''(X'') = \mathsf{SetRec}_{\sigma'}(m'')(X'')$ where $m'' = (Y'', C'', Z'')$. Since $\mathsf{SetRec}_{\sigma'}(m'')(X'') = \{Z_B\}$, and since by Lemma 11 (third point) we have $Z''(X'') \subseteq Z'(X') \subseteq Z(X)$, we get that $Z_B \in Z(X)$. ◀

▶ **Theorem 18.** *Given a POMDP G and a Muller objective $\mathsf{Muller}(\mathcal{F})$ with the set D of colors, if there is a finite-memory almost-sure (resp. positive) winning strategy σ, then the projected strategy $\mathrm{proj}(\sigma)$, with memory of size at most $\mathsf{Mem}^* = 2^{2\cdot|S|} \cdot |\mathfrak{D}|^{|S|}$ (where $\mathfrak{D} = \mathcal{P}(\mathcal{P}(D))$), is also an almost-sure (resp. positive) winning strategy.*

Büchi and coBüchi objectives are parity (thus Muller) objectives with 2 priorities (or colors) (i.e., $d = 2$), and from Theorem 18 we obtain an upper bound of $2^{6\cdot|S|}$ on memory size for them. However, applying the result of Theorem 18 for Muller objectives to parity objectives we obtain a double exponential bound. We establish Theorem 19: for item (1), we present a reduction (details in appendix) that for almost-sure (resp. positive) winning given a POMDP with $|S|$ states and a parity objective with $2 \cdot d$ priorities constructs an equivalent POMDP with $d \cdot |S|$ states with coBüchi (resp. Büchi) objectives (and thus applying Theorem 18 we obtain the $2^{3\cdot d\cdot|S|}$ upper bound); and item (2) follows from Example 5 (and [8] for lower bounds for reachability and safety objectives).

▶ **Theorem 19.** *Given a POMDP G and a parity objective $\mathsf{Parity}(p)$ with the set D of d priorities, the following assertions hold: (1) If there is a finite-memory almost-sure (resp. positive) winning strategy, then there is an almost-sure (resp. positive) winning strategy with memory of size at most $2^{3\cdot d\cdot|S|}$. (2) Finite-memory almost-sure (resp. positive) winning strategies require exponential memory in general, and belief-based stationary strategies are not sufficient in general for finite-memory almost-sure (resp. positive) winning strategies.*

4 Computational Complexity

We will present an exponential time algorithm to solve almost-sure winning in POMDPs with coBüchi objectives under finite-memory strategies (and our polynomial time reduction for parity objectives to coBüchi objectives for POMDPs allows our results to carry over to parity objectives). The results for positive Büchi is similar. The naive algorithm would be to enumerate over all finite-memory strategies with memory bounded by $2^{6\cdot|S|}$, this leads to an algorithm that runs in double-exponential time. Instead our algorithm consists of two steps: (1) given a POMDP G we first construct a special kind of a POMDP $\widehat{G}$ such

that there is a finite-memory winning strategy in G iff there is a randomized memoryless winning strategy in $\widehat{G}$; and (2) then show how to solve the special kind of POMDPs under randomized memoryless strategies in time polynomial in the size of $\widehat{G}$. We introduce the special kind of POMDPs which we call belief-observation POMDPs which satisfy that the current belief is always the set of states with current observation.

▶ **Definition 20.** A POMDP $G = (S, A, \delta, \mathcal{O}, \gamma, s_0)$ is a *belief-observation POMDP* iff for every finite prefix $w = (s_0, a_0, s_1, a_1, \ldots, s_n)$ with the observation sequence $\rho = \gamma(w)$, the belief $\mathcal{B}(\rho)$ is equal to the set of states with the observation $\gamma(s_n)$, i.e., $\mathcal{B}(\rho) = \{s \in S \mid \gamma(s) = \gamma(s_n)\}$.

POMDPs to belief-observation POMDPs. We will construct a belief-observation POMDP $\widehat{G}$ from a POMDP G for almost-sure winning with coBüchi objectives. Since we are interested in coBüchi objectives, for the sequel of this section we will denote by $M = 2^S \times \{0,1\}^{|S|} \times \mathfrak{D}^{|S|}$, i.e., all the possible beliefs $\mathcal{B}$, $\mathsf{BoolRec}$ and SetRec functions (recall that $\mathfrak{D}$ is $\mathcal{P}(\mathcal{P}(\{1,2\}))$ for coBüchi objectives). If there exists a finite-memory almost-sure winning strategy σ, then the projected strategy $\sigma' = \mathit{proj}(\sigma)$ is also a finite-memory almost-sure winning strategy (by Theorem 18) and will use memory $M' \subseteq M$. The size of the constructed POMDP $\widehat{G}$ will be exponential in the size of the original POMDP G and polynomial in the size of the memory set M (and $|M| = 2^{6 \cdot |S|}$ is exponential in the size of the POMDP G). We define the set $M_{\mathsf{coBuchi}} \subseteq M$ as the memory elements, where for all states s in the belief component of the memory, the set $\mathsf{SetRec}(s)$ contains only a set with priority two, i.e., there is no state with priority 1 in the reachable recurrent classes according to SetRec. Formally, $M_{\mathsf{coBuchi}} = \{(Y, B, L) \in M \mid \forall s \in Y, L(s) = \{\{2\}\}\}$. The POMDP $\widehat{G}$ is constructed such that it allows all possible ways that a projected strategy of a finite-memory almost-sure winning strategy could play in G. Informally, since beliefs are part of states of $\widehat{G}$ it is belief-observation; and since possible memory states of projected strategies are part of the state space, we only need to consider memoryless strategies. We will now present a polynomial time algorithm for the computation of the almost-sure winning set for the belief-observation POMDP $\widehat{G}$ with state space $\widehat{S}$ for coBüchi objectives under randomized memoryless strategies.

Almost-sure winning observations. For an objective φ, we denote by $\mathsf{Almost}(\varphi) = \{o \in \mathcal{O} \mid$ there exists a randomized memoryless strategy σ such that for all $s \in \gamma^{-1}(o)$. $\mathbb{P}_s^\sigma(\varphi) = 1\}$ the set of observations such that there is a randomized memoryless strategy to ensure winning with probability 1 from all states of the observation. Also note that since we consider belief-observation POMDPs we can only consider beliefs that correspond to all states of an observation.

Almost-sure winning for coBüchi objectives. We show that the computation can be achieved by computing almost-sure winning regions for safety and reachability objectives. The steps of the computation are as follows: *(Step 1).* Let $F \subseteq \widehat{S}$ be the set of states of $\widehat{G}$ where some actions can be played consistent with a projected strategy of a finite-memory strategy, and we first compute $\mathsf{Almost}(\mathsf{Safe}(F))$. *(Step 2).* Let $\widehat{S}_{wpr} \subseteq \widehat{S}$ denote the subset of states that intuitively correspond to *winning pseudo-recurrent (wpr)* states, i.e., formally it is defined as follows: $\widehat{S}_{wpr} = \{(s, (Y, B, L)) \mid B(s) = 1, L(s) = \{\{2\}\}$ and $\widehat{p}(s) = 2\}$. In the POMDP restricted to $\mathsf{Almost}(\mathsf{Safe}(F))$ we compute the set of observations $W_2 = \mathsf{Almost}(\mathsf{Reach}(\widehat{S}_{wpr}))$. We show that $W_2 = \mathsf{Almost}(\mathsf{coBuchi}(\widehat{p}^{-1}(2)))$, and then show that in belief-observation POMDPs almost-sure safety and reachability sets can be computed in polynomial time (and thus obtain Theorem 23).

▶ **Lemma 21.** $\mathsf{Almost}(\mathsf{coBuchi}(\widehat{p}^{-1}(2))) = W_2$.

Proof. We prove the inclusion $W_2 \subseteq \mathsf{Almost}(\mathsf{coBuchi}(\widehat{p}^{-1}(2)))$. Let $o \in W_2$ be an observation in W_2, and we show how to construct a randomized memoryless almost-sure winning strategy ensuring that $o \in \mathsf{Almost}(\mathsf{coBuchi}(\widehat{p}^{-1}(2)))$. Let σ be the strategy produced by the computation of $\mathsf{Almost}(\mathsf{Reach}(\widehat{S}_{wpr}))$. We will show that the same strategy ensures also $\mathsf{Almost}(\mathsf{coBuchi}(\widehat{p}^{-1}(2)))$. As in every observation o the strategy σ plays only a subset of actions that are available in the POMDP restricted to $\mathsf{Almost}(\mathsf{Safe}(F)$ (to ensure safety in F), where $F = \widehat{S} \setminus \widehat{s}_b$, the loosing absorbing state $\widehat{s}_b$ is not reachable. Intuitively, the state $\widehat{s}_b$ is reached whenever a strategy violates the structure of a projected strategy. Also with probability 1 the set $\widehat{S}_{wpr}$ is reached. We show that for all states $(s, (Y, B, L)) \in \widehat{S}_{wpr}$ that all the states reachable from $(s, (Y, B, L))$ have priority 2 according to $\widehat{p}$. Therefore ensuring that all recurrent classes reachable from $\widehat{S}_{wpr}$ have minimal priority 2. In the construction of the POMDP $\widehat{G}$, the only actions allowed in a state $(s, (Y, B, L))$ satisfy that for all states $\widehat{s} \in Y$ if $B(\widehat{s}) = 1$, $L(\widehat{s}) = \{Z_\infty\}$ and $\widehat{p}(s) \in Z_\infty$ for some $Z_\infty \subseteq \{1, 2\}$, then for all states $\widehat{s}' \in \mathrm{Supp}(\delta(s, a))$ we have that $p(\widehat{s}') \in Z_\infty$. As all states in $(s, (Y, B, L)) \in \widehat{S}_{wpr}$ have $L(s) = \{\{2\}\}$, it follows that any state reachable in the next step has priority 2. Let $(s', Y', (Y, B, L), a)$ be an arbitrary state reachable from $(s, (Y, B, L))$ in one step. By the previous argument we have that the priority $\widehat{p}((s', Y', (Y, B, L), a)) = 2$. Similarly the only allowed memory-update actions (Y', B', L') from state $(s', Y', (Y, B, L), a)$ satisfy that whenever $\widehat{s} \in Y$ and $B(\widehat{s}) = 1$, then for all $\widehat{s}' \in \mathrm{Supp}(\delta(\widehat{s}, a))$, we have that $B'(\widehat{s}') = 1$ and similarly we have that $L'(s')$ is a non-empty subset of $L(s)$, i.e., $L'(s') = \{\{2\}\}$. Therefore the next reachable state $(s', (Y', B', L'))$ is again in $\widehat{S}_{wpr}$. In other words, from states $(s, (Y, B, L))$ in $\widehat{S}_{wpr}$ in all future steps only states with priority 2 are visited, i.e., $\mathsf{Safe}(\widehat{p}^{-1}(2))$ is ensured which ensures the coBüchi objective. As the states in $\widehat{S}_{wpr}$ are reached with probability 1 and from them all recurrent classes reachable have only states that have priority 2, the desired result follows. ◀

▶ **Lemma 22.** *For $T \subseteq S$ and $F \subseteq S$, the set $Y^* = \mathsf{Almost}(\mathsf{Safe}(F))$ can be computed in linear time; and the set $Z^* = \mathsf{Almost}(\mathsf{Buchi}(T))$ and $\mathsf{Almost}(\mathsf{Reach}(T))$ can be computed in quadratic time for belief-observation POMDPs.*

▶ **Theorem 23.** *(1) Given a POMDP G with $|S|$ states and a parity objective with d priorities, the decision problem of the existence (and the construction if one exists) of a finite-memory almost-sure (resp. positive) winning strategy can be solved in $2^{O(|S|\cdot d)}$ time. (2) The decision problem of given a POMDP and a parity objective whether there exists a finite-memory almost-sure (resp. positive) winning strategy is EXPTIME-complete.*

Concluding remarks. Our EXPTIME-algorithm for parity objectives, and the LAR reduction of Muller objectives to parity objectives [15] give an $2^{O(d! \cdot d^2 \cdot |S|)}$ time algorithm for Muller objectives with d colors for POMDPs with $|S|$ states, i.e., exponential in $|S|$ and double exponential in d. Note that the Muller objective specified by the set $\mathcal{F}$ maybe in general itself double exponential in d.

Acknowledgement. We thank Sumit Nain and Moshe Vardi for sharing their work with us.

References

1 C. Baier, N. Bertrand, and M. Größer. On decision problems for probabilistic Büchi automata. In *FoSSaCS*, LNCS 4962, pages 287–301. Springer, 2008.

2 C. Baier, M. Größer, and N. Bertrand. Probabilistic omega-automata. *J. ACM*, 59(1), 2012.

3 C. Baier and J-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
4 A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *FSTTCS 95*, volume 1026 of *LNCS*, pages 499–513. Springer-Verlag, 1995.
5 P. Billingsley, editor. *Probability and Measure*. Wiley-Interscience, 1995.
6 P. Cerný, K. Chatterjee, T. A. Henzinger, A. Radhakrishna, and R. Singh. Quantitative synthesis for concurrent programs. In *Proc. of CAV*, LNCS 6806, pages 243–259. Springer, 2011.
7 K. Chatterjee, L. Doyen, H. Gimbert, and T. A. Henzinger. Randomness for free. In *MFCS*, 2010.
8 K. Chatterjee, L. Doyen, and T. A. Henzinger. Qualitative analysis of partially-observable Markov decision processes. In *MFCS*, pages 258–269, 2010.
9 A. Condon and R. J. Lipton. On the complexity of space bounded interactive proofs. In *FOCS*, pages 462–467, 1989.
10 C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
11 L. de Alfaro, M. Faella, R. Majumdar, and V. Raman. Code-aware resource management. In *EMSOFT 05*. ACM, 2005.
12 R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge Univ. Press, 1998.
13 J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, 1997.
14 H. Gimbert and Y. Oualhadj. Probabilistic automata on finite words: Decidable and undecidable problems. In *Proc. of ICALP*, LNCS 6199, pages 527–538. Springer, 2010.
15 Y. Gurevich and L. Harrington. Trees, automata, and games. In *STOC'82*, pages 60–65, 1982.
16 H. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.
17 H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.
18 M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In *TOOLS' 02*, pages 200–204. LNCS 2324, Springer, 2002.
19 O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif. Intell.*, 147(1-2):5–34, 2003.
20 N. Meuleau, K-E. Kim, L. P. Kaelbling, and A.R. Cassandra. Solving pomdps by searching the space of finite policies. In *UAI*, pages 417–426, 1999.
21 M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
22 C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12:441–450, 1987.
23 A. Paz. *Introduction to probabilistic automata*. Academic Press, 1971.
24 A. Pogosyants, R. Segala, and N. Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distributed Computing*, 13(3):155–186, 2000.
25 M.O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.
26 J. H. Reif. The complexity of two-player games of incomplete information. *JCSS*, 29, 1984.
27 M.I.A. Stoelinga. Fun with FireWire: Experiments with verifying the IEEE1394 root contention protocol. In *Formal Aspects of Computing*, 2002.
28 W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer, 1997.

Infinite-state games with finitary conditions*

Krishnendu Chatterjee[1] and Nathanaël Fijalkow[2]

1 **IST Austria, Klosterneuburg, Austria**
`krishnendu.chatterjee@ist.ac.at`
2 **LIAFA, Université Denis Diderot-Paris 7, France**
Institute of Informatics, University of Warsaw, Poland
`nath@mimuw.edu.pl`

Abstract

We study two-player zero-sum games over infinite-state graphs equipped with ωB and finitary conditions.

Our first contribution is about the strategy complexity, *i.e* the memory required for winning strategies: we prove that over general infinite-state graphs, memoryless strategies are sufficient for finitary Büchi, and finite-memory suffices for finitary parity games.

We then study pushdown games with boundedness conditions, with two contributions. First we prove a collapse result for pushdown games with ωB-conditions, implying the decidability of solving these games. Second we consider pushdown games with finitary parity along with stack boundedness conditions, and show that solving these games is EXPTIME-complete.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Two-player games, Infinite-state systems, Pushdown games, Bounds in omega-regularity, Synthesis

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.181

1 Introduction

Games on graphs. Two-player games played on graphs is a powerful mathematical framework to analyze several problems in computer science as well as mathematics. In particular, when the vertices of the graph represent the states of a reactive system and the edges represent the transitions, then the synthesis problem (Church's problem) asks for the construction of a winning strategy in a game played on the graph [12, 30]. Game-theoretic formulations have also proved useful for the verification, refinement, and compatibility checking of reactive systems [4]; and has deep connection with automata theory and logic, *e.g* the celebrated decidability result of monadic second-order logic over infinite trees due to Rabin [33].

Omega-regular conditions: strengths and weaknesses. In the literature, two-player games on finite-state graphs with ω-regular conditions have been extensively studied [21, 22, 25, 26]. The class of ω-regular languages provides a robust specification language for solving control and verification problems (see, *e.g*, [32]). Every ω-regular condition can be decomposed into a safety part and a liveness part [2]. The safety part ensures that the component will not do anything "bad" (such as violate an invariant) within any finite number of transitions. The liveness part ensures that the component will do something "good" (such as proceed, or

* The first author was supported by Austrian Science Fund (FWF) Grant No P23499 – N23, FWF NFN Grant No S11407-N23 (RiSE), ERC Start grant (279307: Graph Games), and Microsoft faculty fellows award. The second author has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 259454 (GALE) and n° 239850 (SOSNA).

respond, or terminate) in the long-run. Liveness can be violated only in the limit, by infinite sequences of transitions, as no bound is stipulated on when the "good" thing must happen. This infinitary, classical formulation of liveness has both strengths and weaknesses. A main strength is robustness, in particular, independence from the chosen granularity of transitions. Another important strength is simplicity, allowing liveness to serve as an abstraction for complicated safety conditions. For example, a component may always respond in a number of transitions that depends, in some complicated manner, on the exact size of the stimulus. Yet for correctness, we may be interested only that the component will respond "eventually". However, these strengths also point to a weakness of the classical definition of liveness: it can be satisfied by components that in practice are quite unsatisfactory because no bound can be put on their response time.

Stronger notion of liveness: finitary conditions. For the weakness of the infinitary formulation of liveness, alternative and stronger formulations of liveness have been proposed. One of these is *finitary* liveness [3]: it is satisfied if *there exists* a bound N such that every stimulus is followed by a response within N transitions. Note that it does not insist on a response within a known bound N (*i.e*, every stimulus is followed by a response within N transitions), but on response within some unknown bound, which can be arbitrarily large; in other words, the response time must not grow forever from one stimulus to the next. In this way, finitary liveness still maintains the robustness (independence of step granularity) and simplicity (abstraction of complicated safety conditions) of traditional liveness, while removing unsatisfactory implementations.

All ω-regular languages can be defined by a deterministic parity automaton; the parity condition assigns to each state an integer representing a priority, and requires that in the limit, every odd priority is followed by a lower even priority. Its finitary counterpart, the finitary parity condition, strenghtens this by requiring the existence of a bound N such that in the limit every odd priority is followed by a lower even priority within N transitions.

Games with finitary conditions. Games over finite graphs with finitary conditions have been studied in [15], leading to very efficient algorithms: finitary parity games can be solved in polynomial time. In this paper, we study games over infinite graphs with finitary conditions, and then focus on the widely studied class of pushdown games, which model sequential programs with recursion. This line of work belongs to the tradition of infinite-state systems and games, (see *e.g* [1, 11]). Pushdown games with the classical reachability and parity conditions have been studied in [5, 37]. It has been established in [37] that the problem of deciding the winner in pushdown parity games is EXPTIME-complete. However, little is known about pushdown games with boundedness conditions; one notable exception is parity and stack boundedness conditions [10, 24]. The stack boundedness condition naturally arises with the synthesis problem in mind, since bounding the stack amounts to control the depth of recursion calls of the sequential program.

Bounds in ω-regularity. The finitary conditions are closely related to the line of work initiated by Bojańczyk in [7], where the MSO $+ \mathbb{B}$ logic was defined, generalizing MSO over infinite words by adding a bounding quantifier $\mathbb{B}$. The satisfiability problem for this logic has been deeply investigated (see for instance [7, 8, 9]), but the decidability for the general case is still open. A fragment of MSO $+ \mathbb{B}$ over infinite words was shown to be decidable in [8], by introducing the model of ωB-automata, which manipulate counters. They perform three kind of actions on counters: increment (i), reset (r) or nothing (ε). The relation with finitary conditions has been investigated in [13], where it is shown that automata with finitary conditions exactly correspond to star-free ωB-expressions.

Regular cost-functions. A different perspective for bounds in ω-regularity was developed by Colcombet in [16] with functions instead of languages, giving rise to the theory of regular cost-functions and cost-MSO. The decidability of cost-MSO over finite trees was established in [20], but its extension over infinite trees is still open, and would imply the decidability of the index of the non-deterministic Mostowski hierarchy [19], a problem open for decades. A subclass of cost-MSO called temporal cost logic was introduced in [18] and is the counterpart of finitary conditions for regular cost-functions [13].

Quantification order. The essential difference between the approaches underlying the logics $\mathrm{MSO} + \mathbb{B}$ and cost-MSO is a quantifier switch. We illustrate this in the context of games: a typical property expressed in $\mathrm{MSO} + \mathbb{B}$ is "there exists a strategy, such that for all plays, there exists a bound on the counter values", while cost-MSO allows to express properties like "there exists a strategy, there exists a bound N, such that for all plays, the counter values are bounded by N". In other words, $\mathrm{MSO} + \mathbb{B}$ expresses non-uniform bounds while bounds in cost-MSO are uniform.

Memoryless determinacy for infinite-state games. Colcombet pointed out in [17] that the remaining difficulty to establish the decidability of cost-MSO over infinite trees is a good understanding of cost-games, and more specifically the cornerstone is to extend the memoryless determinacy of parity games over infinite arenas, following [21, 22, 26].

This paper investigates the subcase of finitary conditions.

Our contributions. We study two questions about infinite-state games with finitary conditions: the *memory requirements* of winning strategies and the *decidability* of solving a pushdown game.

Strategy complexity. We give (non-effective) characterizations of the winning regions for finitary games over countably infinite graphs, implying a complete picture of the strategy complexity. Most importantly, we show that for finitary Büchi conditions memoryless strategies suffice, and that for finitary parity conditions, memory of size $d/2 + 1$ suffices, where d is the number of priorities of the parity condition.

Pushdown games. We present two contributions.

First we consider pushdown games with ωB-conditions and prove the equivalence between the following: "there exists a strategy, such that for all plays, there exists a bound on the counter values" and "there exists a strategy, there exists a bound N, such that for all plays, *eventually* the counter values are bounded by N". We refer to this as a collapse result, as it reduces a quantification with non-uniform bounds (in the fashion of $\mathrm{MSO} + \mathbb{B}$) to one with uniform bounds (à la cost-MSO). Using this, we obtain the decidability of determining the winner in such games relying on previous results [6, 7].

Second we consider pushdown games with finitary parity along with stack boundedness conditions, and establish that solving these games is EXPTIME-complete.

All proofs are omitted, but can be found in the technical report [14].

2 Definitions

Arenas and games. The games we consider are played on an *arena* $\mathcal{A} = (V, (V_E, V_A), E)$, which consists of a (potentially infinite but countable) graph (V, E) and a partition (V_E, V_A) of the vertex set V. A vertex is controlled by Eve and depicted by a circle if it belongs to V_E and controlled by Adam and depicted by a square if it belongs to V_A. Playing consists in moving a pebble along the edges: initially placed on a vertex v_0, the pebble is sent along an edge chosen by the player who controls the vertex. From this infinite interaction results a *play* π, which is an infinite sequence of vertices $v_0, v_1, \ldots$ where for all i, we have $(v_i, v_{i+1}) \in E$, *i.e*

π is an infinite path in the graph. We denote by Π the set of all plays, and define *conditions* for a player by sets of winning plays $\Omega \subseteq \Pi$. The games are zero-sum, which means that if Eve's condition is Ω, then Adam's condition is $\Pi \setminus \Omega$, usually denoted by "CoΩ" (the conditions are opposite). Formally, a *game* is given by $\mathcal{G} = (\mathcal{A}, \Omega)$ where $\mathcal{A}$ is an arena and Ω a condition. A condition Ω is prefix-independent if it is closed under adding and removing prefixes.

Strategies. A *strategy* for a player is a function that prescribes, given a finite history of the play, the next move. Formally, a *strategy* for Eve is a function $\sigma : V^* \cdot V_E \to V$ such that for a finite history $w \in V^*$ and a current vertex $v \in V_E$, the prescribed move is legal, *i.e* along an edge: $(v, \sigma(w \cdot v)) \in E$. Strategies for Adam are defined similarly, and usually denoted by τ. Once a game $\mathcal{G} = (\mathcal{A}, \Omega)$, a starting vertex v_0 and strategies σ for Eve and τ for Adam are fixed, there is a unique play denoted by $\pi(v_0, \sigma, \tau)$, which is said to be winning for Eve if it belongs to Ω. The sentence "Eve wins from v_0" means that she has a winning strategy from v_0, that is a strategy σ such that for all strategies τ for Adam, the play $\pi(v_0, \sigma, \tau)$ is winning. By "solving the game", we mean (algorithmically) determine the winner. We denote by $\mathcal{W}_E(\mathcal{G})$ the set of vertices from which Eve wins, also referred to as winning set, or winning region, and analogously $\mathcal{W}_A(\mathcal{G})$ for Adam. A very important theorem in game theory, due to Martin [29], states that Borel games (that is, where the condition is Borel) are determined, *i.e* we have $\mathcal{W}_E(\mathcal{G}) \cup \mathcal{W}_A(\mathcal{G}) = V$, *i.e* from any vertex, exactly one of the two players has a winning strategy. Throughout this paper, we only consider Borel conditions, hence our games are determined.

Memory structures. We define memory structures and strategies relying on memory structures. A *memory structure* $\mathcal{M} = (M, m_0, \mu)$ for an arena $\mathcal{A}$ consists of a set M of memory states, an initial memory state $m_0 \in M$, and an update function $\mu : M \times E \to M$. A memory structure is similar to an automaton synchronized with the arena: it starts from m_0 and reads the sequence of edges produced by the arena. Whenever an edge is taken, the current memory state is updated using the update function μ. A strategy relying on a memory structure $\mathcal{M}$, whenever it picks the next move, considers only the current vertex and the current memory state: it is thus given by a next-move function $\nu : V_E \times M \to V$. Formally, given a memory structure $\mathcal{M}$ and a next-move function ν, we can define a strategy σ for Eve by $\sigma(w \cdot v) = \nu(v, \mu^*(w \cdot v))$, where μ is extended to $\mu^* : V^+ \to M$. A strategy with memory structure $\mathcal{M}$ has finite memory if M is a finite set. It is *memoryless*, or *positional* if M is a singleton: in this case, the choice for the next move only depends on the current vertex, and can be described as a function $\sigma : V_E \to V$.

Attractors. Given $F \subseteq V$, define $\mathrm{Pre}(F)$ as the union of $\{u \in V_E \mid \exists (u, v) \in E, v \in F\}$ and $\{u \in V_A \mid \forall (u, v) \in E, v \in F\}$. The attractor sequence is the step-by-step computation of the least fixpoint of the monotone function $X \mapsto F \cup \mathrm{Pre}(X)$:

$$\begin{cases} \mathrm{Attr}^{\mathrm{E}}_0(F) = F \\ \mathrm{Attr}^{\mathrm{E}}_{k+1}(F) = \mathrm{Attr}^{\mathrm{E}}_k(F) \cup \mathrm{Pre}(\mathrm{Attr}^{\mathrm{E}}_k(F)) \end{cases}$$

The sequence $(\mathrm{Attr}^{\mathrm{E}}_k(F))_{k \geq 0}$ is increasing with respect to set inclusion, so it has a limit, denoted $\mathrm{Attr}^{\mathrm{E}}(F)$, the attractor to F. An attractor strategy to $F \subseteq V$ for Eve is a memoryless strategy that ensures from $\mathrm{Attr}^{\mathrm{E}}(F)$ to reach F within a finite number of steps. Specifically, an attractor strategy to F from $\mathrm{Attr}^{\mathrm{E}}_N(F)$ ensures to reach F within the next N steps.

ω-regular conditions. We define the Büchi and parity conditions. We equip the arena with a coloring function $c : V \to [d]$ where $[d] = \{0, \ldots, d\}$ is the set of *colors* or *priorities*. For a play π, let $\mathrm{Inf}(\pi) \subseteq [d]$ be the set of colors that appear infinitely often in π. The parity

condition is defined by $\text{Parity}(c) = \{\pi \mid \min(\text{Inf}(\pi)) \text{ is even}\}$, *i.e* it is satisfied if the lowest color visited infinitely often is even. Here, the color set $[d]$ is interpreted as a set of priorities, even priorities being "good" and odd priorities "bad", and lower priorities preferable to higher ones. As a special case, the class of Büchi conditions are defined using the color set $[1] = \{0, 1\}$ (*i.e* $d = 1$). Setting F as $c^{-1}(0) \subseteq V$, we define $\text{Büchi}(F) = \{\pi \mid 0 \in \text{Inf}(\pi)\}$, *i.e* the Büchi condition $\text{Büchi}(F)$ requires that infinitely many times vertices in F are reached. We usually call F the Büchi set and say that a vertex is Büchi if it belongs to F. The dual is $\text{CoBüchi}(F)$ condition, which requires that finitely many times vertices in F are reached.

ωB-conditions. We equip the arena with k counters and an update function $C : E \to \{\varepsilon, i, r\}^k$, associating to each edge an action for each counter. The value of a counter along a play is incremented by the action i, reset by r and left unchanged by ε. We say that a counter is bounded along a play if the set of values assumed is finite, and denote by Bounded the set of plays where all counters are bounded, and $\text{Bounded}(N)$ if bounded by N. Conditions of the form $\text{Bounded} \cap \text{Parity}(c)$ are called ωB-conditions.

Note that the bound requirement is not uniform: a strategy is winning if for all plays, there exists a bound N such that the counters are bounded by N and the parity condition is satisfied. In other words, the bound N depends on the path. The sentence "Eve wins for the bound N" means that Eve has a strategy which ensures the bound N uniformly: for all plays, the counters are bounded by the same N. Similarly, the sentence "the strategy (for Adam) breaks the bound N" means that it ensures that for all plays, either some counter reaches the value N or the parity condition is not satisfied.

Finitary conditions. Finitary conditions add bounds requirements over ω-regular conditions [3]. Given a coloring function $c : V \to [d]$, and a position k we define:

$$\text{dist}_k(\pi, c) = \inf_{k' \geq k} \left\{ k' - k \mid \begin{array}{c} c(\pi_{k'}) \text{ is even, and} \\ c(\pi_{k'}) \leq c(\pi_k) \end{array} \right\};$$

i.e $\text{dist}_k(\pi, c)$ is the "waiting time" by means of number of steps from the k^{th} vertex to a preferable priority (that is, even and lower). The finitary parity winning condition $\text{FinParity}(c)$ was defined as follows in [15]: $\text{FinParity}(c) = \{\pi \mid \limsup_k \text{dist}_k(\pi, c) < \infty\}$, *i.e* the finitary parity condition requires that the supremum limit of the distance sequence is bounded. In the special case where $d = 1$, this defines the finitary Büchi condition: setting $F = c^{-1}(0)$, we denote $\text{dist}_k(\pi, F) = \inf\{k' - k \mid k' \geq k, \pi_{k'} \in F\}$, *i.e* $\text{dist}_k(\pi, F)$ is the number of transitions followed from the k^{th} vertex before reaching the next Büchi vertex. Then $\text{FinBüchi}(F) = \{\pi \mid \limsup_k \text{dist}_k(\pi, F) < \infty\}$. In the context of finitary conditions, the sentence "the strategy (for Adam) breaks the bound N" means that the strategy ensures that for all plays, there exists a position k such that $\text{dist}_k(\pi, c) > N$.

▶ Remark. As defined, finitary conditions do not form a subclass of ωB-conditions; however, one can easily show that there exists a deterministic ωB-automaton which recognize $\text{FinParity}(c)$ (see *e.g* [13]), so finitary parity games easily reduce to ωB games by composing with this deterministic automaton.

▶ **Example 1.** We conclude this section by an example witnessing the difference between playing a Büchi condition and a finitary Büchi condition over an infinite arena. This is in contrast to the case of finite arenas, where winning for Büchi and finitary Büchi conditions are equivalent. Figure 1 presents an infinite arena where only Adam has moves; he loses the Büchi game but wins for the finitary Büchi game. We give two representations: on the left as a pushdown game (defined in Section 4), and on the right as an infinite-state game.

A play consists in rounds, each starting whenever the pebble hits the leftmost vertex. In a round, Adam follows the top path, remaining in Büchi vertices; he may decide at any

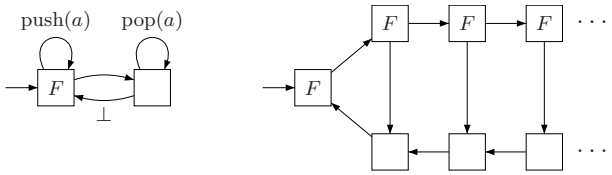


Figure 1 Adam loses the Büchi game but wins the finitary Büchi game.

point to follow an edge down, following the bottom Büchi-free path before geting back to the leftmost vertex. Whatever Adam does, infinitely many Büchi vertices will be visited, so Adam loses the Büchi game. However, by going further and further to the right (*e.g* for i steps in the i^{th} round), Adam ensures longer and longer paths without Büchi vertices, hence wins the finitary Büchi game.

3 Strategy complexity for finitary conditions over infinite-state games

In this section we give characterizations of the winning regions for finitary conditions over infinite arenas, and use them to establish the strategy complexity.

Our motivation to prove the existence of finite-memory strategies comes from automata theory, where several constructions rely on the existence of memoryless winning strategies (for parity games): for instance to complement tree automata [22], or to simulate alternating two-way tree automata by non-deterministic ones [36]. For this, one needs to prove the existence of finite-memory strategies whose size only depends on the condition. We present such a result in the following theorem.

▶ **Theorem 2** (Strategy complexity for finitary games). *The following assertions hold:*

1. *For all finitary Büchi games, there exists a memoryless winning strategy for Eve from her winning set.*
2. *For all finitary parity games, there exists a winning strategy for Eve from her winning set that uses at most $d/2+1$ memory states, where d is the number of colors.*

3.1 Bounded and uniform conditions

To obtain Theorem 2, we take five steps, summarized in Figure 2, which involve two variants of finitary conditions: uniform and bounded.

Figure 2 Results implications.

Uniform conditions. The bound $N \in \mathbb{N}$ is made explicit; for instance the uniform Büchi condition is $\text{Büchi}(F, N) = \{\pi \mid \limsup_k \text{dist}_k(\pi, F) \leq N\}$.

Bounded conditions. The requirement is not in the limit, but from the start of the play, *i.e* the distance function is bounded rather than eventually bounded; for instance the bounded parity condition is $\text{BndParity}(c) = \{\pi \mid \sup_k \text{dist}_k(\pi, c) < \infty\}$.

The two variants can be combined, for instance the bounded uniform Büchi condition is $\text{BndBüchi}(F, N) = \{\pi \mid \sup_k \text{dist}_k(\pi, F) \leq N\}$. Let us point out that in the special case

of Büchi conditions, we have $\mathrm{BndBüchi}(F) = \mathrm{FinBüchi}(F)$, hence we can refer to these conditions either as bounded Büchi or as finitary Büchi.

3.2 Strategy complexity for bounded uniform Büchi games

Our first step is the study of bounded uniform Büchi games.

▶ **Theorem 3** (Strategy complexity for bounded uniform Büchi games)**.** *For all bounded uniform Büchi games with bound N, there exists a memoryless winning strategy for Eve from her winning set.*

We show that Eve's winning set can be described using a greatest fixpoint, which allows to define a winning memoryless strategy. We define a sequence $(Z_k)_{k\geq 0}$ of subsets of V:

$$\begin{cases} Z_0 = V \\ Z_{k+1} = \mathrm{Attr}_N^{\mathrm{E}}(F \cap \mathrm{Pre}(Z_k)) \end{cases}$$

As this sequence is decreasing with respect to set inclusion, it has a limit[1], equivalently defined as the greatest fixpoint of the monotone function $X \mapsto \mathrm{Attr}_N^{\mathrm{E}}(F \cap \mathrm{Pre}(X))$: we denote it by Z.

▶ **Lemma 4.** $Z = \mathcal{W}_E(\mathrm{BndBüchi}(F, N))$

The crucial point is the left-to-right inclusion, which consists in constructing a winning strategy for Eve from Z. Roughly speaking, the strategy is obtained by nesting attractor strategies on the slices defined by the sets $(Z_k)_{k\geq 0}$, and this strategy is memoryless.

3.3 From bounded uniform Büchi games to finitary Büchi games

We sketch the second step. The bounded uniform Büchi conditions are the prefix-dependent counterpart of the uniform Büchi conditions: $\mathrm{Büchi}(F, N) = V^* \cdot \mathrm{BndBüchi}(F, N)$. However, this equality does not imply the equality between $\mathcal{W}_E(\mathrm{Büchi}(F, N))$ and the attractor of $\mathcal{W}_E(\mathrm{BndBüchi}(F, N))$. The following properties hold:

1. $\mathcal{W}_E(\mathrm{BndBüchi}(F, N)) \subseteq \mathcal{W}_E(\mathrm{Büchi}(F, N))$,
2. if $\mathcal{W}_E(\mathrm{BndBüchi}(F, N))$ is empty then $\mathcal{W}_E(\mathrm{Büchi}(F, N))$ is empty.

We sketch the proof of the second item. Assume that Eve wins nowhere for $\mathrm{BndBüchi}(F, N)$, then $\mathcal{W}_A(\mathrm{BndBüchi}(F, N))$ is the set of all vertices: from everywhere Adam can break the bound N once. Repeating such a strategy, he can break the bound N infinitely often, so Adam wins everywhere for the condition $\mathrm{Büchi}(F, N)$, which implies $\mathcal{W}_E(\mathrm{Büchi}(F, N)) = \emptyset$.

The two properties above imply that the uniform Büchi winning set is obtained by a least fixpoint iteration using the bounded uniform Büchi winning set. In particular, the memoryless determinacy transfers from bounded uniform Büchi to uniform Büchi.

This technique will be used several times in the paper (see *e.g* [27] for similar fixpoint iterations). It consists in decomposing the uniform Büchi winning set into a sequence of disjoint subarenas called "slices", and define a positional strategy for each slice. Aggregating all those strategies yields a positional winning strategy for the uniform Büchi condition. The first slice is the attractor of the bounded uniform Büchi winning set, for which Eve has a

[1] This follows from our assumption that the arenas have a countable set of vertices. Here we could drop this assumption and define the sequence indexed by ordinals, which we avoided for the sake of readability.

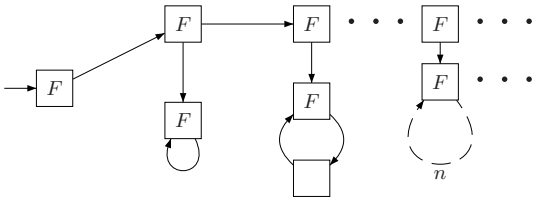


Figure 3 An infinite arena where Eve cannot predict the bound.

positional winning strategy for the uniform Büchi condition. We remove the first slice and proceed inductively with the remaining arena. The key observation is that a play consistent with the obtained strategy can only go down the slices, so eventually remains in one slice.
We sketch the third step. Denote by U the operator that associates to an arena the set of vertices $\mathrm{Attr}^{\mathrm{E}}\left(\bigcup_N \mathcal{W}_E(\text{Büchi}(F,N))\right)$. For the sake of readability, we also see U as a set of vertices once an arena is fixed. We first show how to obtain a memoryless strategy that wins for the condition FinBüchi(F) from U. This requires us to compose several memoryless strategies into one:

▶ **Lemma 5** (Union and memoryless strategies [24]). *Let $(\Omega_n)_{n\in\mathbb{N}}$ be a family of Borel conditions, and assume $\cup_n\Omega_n$ is prefix-independent. If for all n, Eve wins positionally for the condition Ω_n from V_n, then she wins positionally for the condition $\cup_n\Omega_n$ from $\cup_n V_n$.*

Intuitively, U is the set of vertices where Eve has a strategy to attract in a region won for some uniform Büchi condition. That is, from some point onwards, Eve can announce a bound N and claim "I will win for the condition Büchi(F,N)". However, it may be that even if Eve wins, she is never able to announce a bound: such a situation happens in Example 6.

▶ **Example 6.** Figure 3 presents an infinite one-player arena, where Eve wins yet is not able to announce a bound. A loop labeled n denotes a loop of length n, where a Büchi vertex is visited every n steps. In this game, as long as Adam decides to remain in the top path, Eve cannot claim that she will win for some uniform Büchi condition.

The following properties hold:
1. $U \subseteq \mathcal{W}_E(\text{FinBüchi}(F))$,
2. if U is empty then $\mathcal{W}_E(\text{FinBüchi}(F))$ is empty.

We sketch the second item. Assume that for all N, the winning set $\mathcal{W}_E(\text{Büchi}(F,N))$ is empty, so Adam wins for the condition CoBüchi(F,N) from everywhere: let τ_N be a winning strategy for Adam. From any vertex, the strategy τ_N ensures that at some point, there will be a sequence of N consecutive non-Büchi vertices. Playing in turns τ_1 until such a sequence occurs, then τ_2, and so on, ensures that the condition FinBüchi(F) is spoiled. Hence Adam wins everywhere for the condition CoFinBüchi(F), which implies $\mathcal{W}_E(\text{FinBüchi}(F)) = \emptyset$.

As for the second step, the winning region for finitary Büchi is obtained as the least fixpoint of the operator U, implying the memoryless determinacy for finitary Büchi.

We summarize in the following theorem the winning sets characterizations obtained for the three variants of Büchi conditions, using mu-calculus formulae with infinite disjunction.

▶ **Theorem 7** (Characterizations of the winning set).

$$\mathcal{W}_E(\text{BndBüchi}(F,N)) = \nu Z \cdot \mathrm{Attr}^{\mathrm{E}}_N(F \cap \mathrm{Pre}(Z))$$

$$\mathcal{W}_E(\text{Büchi}(F,N)) = \mu Y \cdot \nu Z \cdot \mathrm{Attr}^{\mathrm{E}}_N((F \cup Y) \cap \mathrm{Pre}(Z))$$

$$\mathcal{W}_E(\text{FinBüchi}(F)) = \mu X \cdot \left(\bigcup_{N\in\mathbb{N}} \mu Y \cdot \nu Z \cdot \mathrm{Attr}^{\mathrm{E}}_N((F \cup Y \cup X) \cap \mathrm{Pre}(Z)) \right)$$

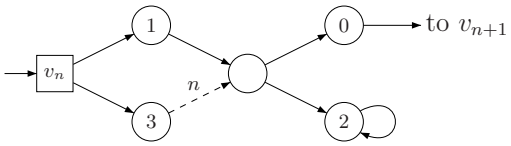


Figure 4 An infinite arena where Eve needs memory to win BndParity(c).

3.4 From finitary Büchi to finitary parity games

We sketch the fourth step, which is a reduction from bounded parity games to bounded Büchi games. We consider a coloring function $c : V \to [d]$, and assume d is even. Define the memory structure $\mathcal{M} = (\{1, 3, \ldots, d-1\} \cup \{d\}, m_0, \mu)$, where:

$$\mu(m, (v, v')) = \begin{cases} m & \text{if } c(v') \geq m \\ c(v') & \text{if } c(v') < m \text{ and } c(v') \text{ is odd} \\ d & \text{if } c(v') < m \text{ and } c(v') \text{ is even} \end{cases}$$

The initial memory state m_0 is $c(v_0)$ if $c(v_0)$ is odd, and d otherwise. Intuitively, this memory structure keeps track of the most urgent pending request.

Let $F = \{(v, d) \mid c(v) \text{ is even}\}$. We argue that $\mathcal{G} = (\mathcal{A}, \text{BndParity}(c))$ is equivalent to $\mathcal{G} \times \mathcal{M} = (\mathcal{A} \times \mathcal{M}, \text{BndBüchi}(F))$. This follows from the equivalence:

$$\pi \in \text{BndParity}(c) \quad \text{if and only if} \quad \widetilde{\pi} \in \text{BndBüchi}(F),$$

where $\widetilde{\pi}$ is the play in $\mathcal{G} \times \mathcal{M}$ corresponding to π. Since Eve has a memoryless winning strategy in any bounded Büchi game, which implies that she has a winning strategy using $\mathcal{M}$ as memory structure in the original bounded parity game $\mathcal{G}$.

▶ **Example 8.** Figure 4 presents an infinite arena, where for condition BndParity(c), Eve needs two memory states to win. This is in contrast with finite arenas, where she has memoryless winning strategies [15]. The label n on an edge indicates that the length of the path is n. A play is divided in rounds, and a round is as follows: first Adam makes a request, either 1 or 3, and then Eve either answers both requests and proceeds to the next round, or stops the play visiting color 2. Assume Eve uses a memoryless strategy, and consider two cases: either she chooses always 0, then Adam wins by choosing always 3, ensuring that the response time grows unbounded, or at some round she chooses 2, then Adam wins by choosing 1 at this particular round, ensuring that this last request will never be responded. However, if Eve answers correctly – that is choosing color 0 for the request 1, and color 2 for the request 3 – the bounded parity condition is satisfied; this requires two memory states.

The fifth step is very similar to the second, and is omitted here.

4 Pushdown ωB games

In this section we consider pushdown ωB games and prove a collapse result. Along with previous results [6, 7], this implies that determining the winner in such games is decidable.

Pushdown arenas. A pushdown process is a finite-state machine which features a stack: it is described as (Q, Γ, Δ) where Q is a finite set of control states, Γ is the stack alphabet and Δ is the transition relation. There is a special stack symbol denoted $\perp$ which does not

belong to Γ; we denote by $\Gamma_\perp$ the alphabet $\Gamma \cup \{\perp\}$. A configuration is a pair $(q, u\perp)$ (the top stack symbol is the leftmost symbol of u). There are three kinds of transitions in Δ:

- $(p, a, \text{push}(b), q)$: allowed if the top stack element is $a \in \Gamma_\perp$, the symbol $b \in \Gamma$ is pushed onto the stack.
- $(p, \text{pop}(a), q)$: allowed if the top stack element is $a \in \Gamma$, the top stack symbol a is popped from the stack.
- (p, a, skip, q): allowed if the top stack element is $a \in \Gamma_\perp$, the stack remains unchanged.

The symbol $\perp$ is never pushed onto, nor popped from the stack. The pushdown arena of a pushdown process is defined as $(Q \times \Gamma^*\perp, (Q_E \times \Gamma^*\perp, Q_A \times \Gamma^*\perp), E)$, where (Q_E, Q_A) is a partition of Q and E is given by the transition relation Δ. For instance if $(p, a, \text{push}(b), q) \in \Delta$, then $((p, aw\perp), (q, baw\perp)) \in E$, for all words w in Γ^*.

Conditions. The parity conditions for pushdown arenas are specified over the control states, *i.e* do not depend on the stack content. Formally, a coloring function is given by $c : Q \to [d]$, and extended to $c : Q \times \Gamma^*\perp \to [d]$ by $c(q, u\perp) = c(q)$.

We begin this section by giving an example witnessing an interesting phenomenon of pushdown games with ωB-conditions.

▶ **Example 9.** Figure 5 presents a pushdown ωB game where Eve wins but with arbitrarily large value of the counter, depending on Adam. Let us first look at the two bottom states: in the left-hand state at the bottom, Adam can push as many b's as he wishes, and moves the token to the state to its right, where all those b's are popped one at a time, incrementing the counter each time. In other words, each visit of the two bottom states allows Adam to announce a number N and to increment the counter by N. We now look at the states on the top line: the initial state is the leftmost one, where Adam can push an arbitrary number of a's. We see those a's as credits: from the central state, Adam can use one credit (*i.e* pop an a) to pay a visit to the two bottom states. When he runs out of credit, which will eventually happen, he moves the token to the rightmost state, where nothing happens anymore.

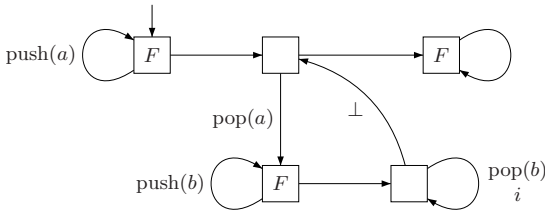


Figure 5 A pushdown ωB game where Eve wins but with arbitrarily large counter value.

$\mathcal{P}$-automata. We will use alternating $\mathcal{P}$-automata to recognize sets of configurations: an alternating $\mathcal{P}$-automaton $\mathcal{B} = (S, \delta, F)$ for the pushdown process (Q, Γ, Δ) is a classical alternating automaton over finite words: S is a finite set of control states, $\delta : S \times \Gamma \to \mathcal{B}^+(S)$ is the transition function (the notation $\mathcal{B}^+(S)$ denotes the positive boolean formulae over S) and F is the subset of S of final states. We assume that the set of states S contains Q. A configuration $(q, u\perp)$ is accepted by $\mathcal{B}$ if it is accepted with $q \in Q \subseteq S$ as initial state and the classical alternating semantics. A set of configurations is called *regular* if it is accepted by an alternating $\mathcal{P}$-automaton.

The following theorem shows that the winning region is regular for a wide class of conditions [34, 35].

▶ **Theorem 10** ([35]). *For all pushdown games, for all winning conditions $\Omega \subseteq Q^\omega$ that are Borel and prefix-independent, the set $\mathcal{W}_E(\Omega)$ is a regular set of configurations recognized by an alternating $\mathcal{P}$-automaton of size $|Q|$.*

4.1 The collapse result

We denote that LimitBounded(N) the set of plays which contain a suffix for which the counters are bounded by N.

▶ **Theorem 11** (The forgetful property). *For all pushdown ωB games, for all initial configurations, the following are equivalent:*

- $\exists \sigma$ *strategy for Eve,* $\forall \pi$ *plays,* $\exists N \in \mathbb{N}$, $\pi \in \text{Bounded}(N) \cap \text{Parity}(c)$,
- $\exists \sigma$ *strategy for Eve,* $\exists N \in \mathbb{N}$, $\forall \pi$ *plays,* $\pi \in \text{LimitBounded}(N) \cap \text{Parity}(c)$.

The intuition behind the name forgetful property is the following: even if a configuration carries an unbounded amount of information (since the stack may be arbitrarily large), this information cannot be forever transmitted along a play. Indeed, to increase the counter values significantly, Adam has to use the stack, consuming or forgetting its original information. Example 9 shows that the content of the stack can be used as "credit" for Adam, but also that if Eve wins then from some point onwards this credit vanishes.

We sketch the proof of the forgetful property. We abbreviate $\mathcal{W}_E(\text{LimitBounded}(N) \cap \text{Parity}(c))$ by $\mathcal{W}_E(N)$ and $\mathcal{W}_E(\text{Bounded} \cap \text{Parity}(c))$ by $\mathcal{W}_E$. The following properties hold:

1. $\mathcal{W}_E(0) \subseteq \mathcal{W}_E(1) \subseteq \mathcal{W}_E(2) \subseteq \cdots \subseteq \mathcal{W}_E$.
2. There exists N such that $\mathcal{W}_E(N) = \mathcal{W}_E(N+1) = \cdots$.
3. For such N, we have $V \setminus \mathcal{W}_E(N) \subseteq \mathcal{W}_A$, hence $\mathcal{W}_E = \mathcal{W}_E(N)$.

The first item is clear. For the second we rely on Theorem 10. For every N there exists $\mathcal{B}_N$ an alternating $\mathcal{P}$-automaton of size $|Q|$ recognizing $\mathcal{W}_E(N)$. Since there are finitely many alternating $\mathcal{P}$-automata of size $|Q|$, the increasing sequence of the set of configurations they recognize is ultimately constant, *i.e* there exists N such that $\mathcal{B}_N = \mathcal{B}_{N+1} = \ldots$. We now argue that the third item holds. From the complement of $\mathcal{W}_E(N)$, Adam can ensure to break the bound N, but also $N+1$, and so on, yet remaining there. Iterating such strategies ensures that the ωB-condition is spoiled, which concludes the proof.

▶ Remark. The above proof does not give a bound on N; indeed, the sequence $(\mathcal{B}_N)_{N \in \mathbb{N}}$ is ultimately constant, but the fact that two consecutive automata recognize the same set of configurations does not imply that from there on the sequence is constant. It follows that N can be *a priori* arbitrarily large.

We refer to [14] for examples showing that the bound N is at least doubly-exponential in the number of vertices, and exponential in the stack alphabet.

4.2 Decidability of pushdown ωB games

We give two proofs of decidability of solving pushdown games:

- First, we prove the decidability of solving pushdown finitary games, relying on the finite-memory results of Section 3 (Theorem 2), the collapse result (Theorem 11), and [7].
- Second, we prove the decidability of solving pushdown ωB games, generalizing the first item. This relies on the collapse result (Theorem 11) and [6].

We begin by proving the decidability of pushdown finitary games. Note that the second property in Theorem 11, namely:

$$\exists \sigma \text{ strategy for Eve}, \exists N \in \mathbb{N}, \forall \pi \text{ plays}, \ \pi \in \text{LimitBounded}(N) \cap \text{Parity}(c) \ ,$$

can be written as an existential bounding formula over infinite trees, whose satisfiability was proved decidable in [7]. This relies on an MSO interpretation of pushdown graphs into infinite trees, following [31]. Indeed, thanks to Theorem 2, Eve has a memoryless winning strategy in $\mathcal{G} \times \mathcal{M} = (\mathcal{A} \times \mathcal{M}, \text{FinParity}(c))$ from her winning set, and such a strategy can be described as a set of edges, hence as a monadic second-order variable.

The second proof relies on [6], where it is shown that the membership problem for two-way alternating parity cost-automata over regular trees is decidable; we reduce our problem to this. The first step is to reduce the problem of solving a pushdown ωB game to the membership problem for two-way alternating ωB automata over regular trees, following [28]. Let $\mathcal{A}$ be the two-way alternating ωB automaton obtained and t the regular tree. Now Theorem 11 implies that Eve wins the pushdown ωB game if and only if:

$$\exists N \in \mathbb{N}, \exists \sigma \text{ strategy for Eve}, \forall \pi \text{ plays}, \ \pi \in \text{LimitBounded}(N) \cap \text{Parity}(c) \ .$$

We construct a two-way alternating cost-automaton $\mathcal{A}'$ from $\mathcal{A}$ such that $\mathcal{A}'$ accepts t (as a cost-automaton) if and only if $\mathcal{A}$ accepts t (as an ωB automaton). The automaton $\mathcal{A}'$ is obtained by adding at each step the ability to reset all counters at the price of visiting a very bad color for the parity condition, which takes care of the difference between LimitBounded(N) and Bounded(N).

The main result of this section follows:

▶ **Theorem 12.** *Solving a pushdown ωB-game is decidable.*

5 Pushdown games with finitary and stack boundedness conditions

In this section, we consider pushdown games with finitary parity along with stack boundedness conditions, following [10, 24]. We prove that solving such games is EXPTIME-complete. This is achieved by a reduction which relies on two ideas, that we present separately; the first is a reduction from finitary parity to bounded parity, and the second a collapse result for finitary Büchi along with stack boundedness conditions. We then show how to combine them to obtain a complete reduction, with an optimal complexity.

We denote by BndSt the stack boundedness condition:

$$\text{BndSt} = \{\pi \mid \exists N, \begin{array}{c}\text{all configurations in } \pi \text{ have} \\ \text{stack height less than } N\end{array}\} \ .$$

5.1 A reduction from finitary parity to bounded parity

The reduction relies on a *restart* gadget. We consider a pushdown finitary parity game, given by the coloring function $c : Q \to [d]$, where we assume d to be odd. Between every transition we add a restart gadget, where Eve can choose either to follow the transition, or to restart: this entails that first a state with priority 0 is visited, where Adam can push on the stack a new symbol $\sharp$ an arbitrary number of times, and then go to a state with priority d, where he pops all the $\sharp$ symbols from the stack, before following the original transition. The intuition is the following: whenever Eve chooses to restart, visiting the vertex with priority 0 answers all previous requests, but this comes with the cost that Adam will be able to stay for an arbitrary long time on a state of odd priority. Therefore, Eve can restart only finitely many times. The gadget is represented in Figure 6.

▶ **Lemma 13.** *Eve wins the finitary parity game if and only if she wins the reduced bounded parity game.*

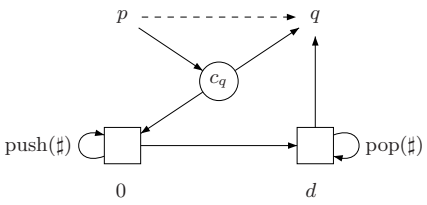


Figure 6 The restart gadget.

5.2 The special case of Büchi conditions

In the study of finitary games over finite graphs [15], the following observation is made: finitary Büchi coincide with Büchi, while finitary parity differs from parity as soon as three colors are involved. Over pushdown arenas, even finitary Büchi differs from Büchi, as noted in Example 1. Yet when intersected with the stack boundedness condition, we show that the case of finitary Büchi specializes again and collapses to Büchi.

▶ **Lemma 14.** *For all pushdown games,*

$$\mathcal{W}_E(\text{FinBüchi}(F) \cap \text{BndSt}) = \mathcal{W}_E(\text{Büchi}(F) \cap \text{BndSt}) \ .$$

The left-to-right inclusion is clear, since $\text{FinBüchi}(F) \subset \text{Büchi}(F)$. We prove the converse inclusion relying on memoryless determinacy for the condition $\text{Büchi}(F) \cap \text{BndSt}$ [10]: assume that σ is a memoryless strategy ensuring $\text{Büchi}(F) \cap \text{BndSt}$, and let π be a play consistent with σ. First note that between two visits of the same configuration, there must be a Büchi configuration, otherwise iterating this loop would be a play consistent with σ yet losing. The second observation is that since the stack height remains smaller than a bound N, the number of different configurations visited in π is finite and bounded by a function of N. The combination of these two arguments imply that π satisfies $\text{FinBüchi}(F)$.

5.3 The complete reduction

We show how to use both ideas to handle pushdown games with finitary parity and stack boundedness conditions. We present a three-step reduction, illustrated in Figure 7.

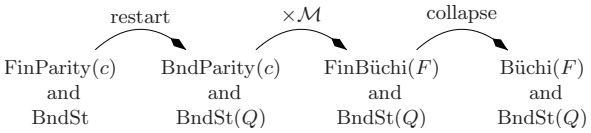


Figure 7 Sequence of reductions.

The first step is to adapt the reduction from finitary parity to bounded parity, now intersected with the stack boundedness condition. To this end, we need to modify the stack boundedness condition so that it ignores the configurations in the restart gadget; we define its restriction to Q:

$$\text{BndSt}(Q) = \{\pi \mid \exists N, \begin{array}{c}\text{all configurations in } \pi \\ \text{with control state in } Q \\ \text{have stack height less than } N\end{array} \} \ .$$

Now the reduction is from finitary parity and stack boundedness to bounded parity and restricted stack boundedness.

The second step is to compose with the memory structure from the fourth step of Section 3, giving an equivalent pushdown game with the condition finitary Büchi and restricted stack boundedness.

The third step is the collapse of finitary Büchi to Büchi. Note that the collapse stated in Lemma 14 deals with stack boundedness, not restricted to a subset of states. Indeed, the result does not hold in general for this modified stack boundedness condition, but it does hold here due to the special form of the restart gadget, that is used only finitely many times.

This three-step reduction produces in linear time an equivalent pushdown game with the condition Büchi and stack boundedness restricted to Q. It has been shown in [10, 24] that deciding the winner in a pushdown game with condition Büchi and stack boundedness is EXPTIME-complete; a slight modification of their techniques extends this to the restricted definition of stack boundedness.

▶ **Theorem 15.** *Determining the winner in a pushdown game with finitary parity and stack boundedness conditions is* EXPTIME*-complete.*

Conclusion. We studied boundedness games over infinite arenas, and investigated two questions. First, the strategy complexity over general infinite arenas; we proved that finite-memory winning strategies exist for finitary parity games. It remains open to extend this to cost-parity games [23]. Second, the decidability of pushdown games; we proved that pushdown ωB-games are decidable, and pushdown games with finitary parity along with stack boundedness conditions are EXPTIME-complete.

Acknowledgments. We thank Denis Kuperberg and Thomas Colcombet for sharing and explaining [6], Damian Niwinski for raising the question of pushdown finitary games, Olivier Serre for many inspiring discussions and Florian Horn for interesting suggestions. We are grateful to the anonymous reviewers for their valuable comments.

References

1 Parosh Aziz Abdulla, Ahmed Bouajjani, and Julien d'Orso. Monotonic and downward closed games. *J. Log. Comput.*, 18(1):153–169, 2008.
2 Bowen Alpern and Fred B. Schneider. Defining liveness. *Inf. Process. Lett.*, 21(4):181–185, 1985.
3 Rajeev Alur and Thomas A. Henzinger. Finitary fairness. *ACM Trans. Program. Lang. Syst.*, 20(6):1171–1194, 1998.
4 Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.
5 Rajeev Alur, Salvatore La Torre, and P. Madhusudan. Modular strategies for recursive game graphs. *Theor. Comput. Sci.*, 354(2):230–249, 2006.
6 Achim Blumensath, Thomas Colcombet, Denis Kuperberg, and Michael Vanden Boom, 2013. Personal communication.
7 Mikołaj Bojańczyk. A bounding quantifier. In *CSL*, pages 41–55, 2004.
8 Mikołaj Bojańczyk and Thomas Colcombet. Bounds in ω-regularity. In *LICS*, pages 285–296, 2006.
9 Mikołaj Bojańczyk and Szymon Toruńczyk. Weak MSO+U over infinite trees. In *STACS*, pages 648–660, 2012.

10 Alexis-Julien Bouquet, Olivier Serre, and Igor Walukiewicz. Pushdown games with unboundedness and regular conditions. In *FSTTCS*, pages 88–99, 2003.

11 Tomás Brázdil, Petr Jancar, and Antonín Kucera. Reachability games on extended vector addition systems with states. In *ICALP (2)*, pages 478–489, 2010.

12 J. Richard Büchi and Lawrence H. Landweber. Definability in the monadic second-order theory of successor. *J. Symb. Log.*, 34(2):166–170, 1969.

13 Krishnendu Chatterjee and Nathanaël Fijalkow. Finitary languages. In *LATA*, pages 216–226, 2011.

14 Krishnendu Chatterjee and Nathanaël Fijalkow. Infinite-state games with finitary conditions. *CoRR*, abs/1301.2661, 2013.

15 Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. Finitary winning in omega-regular games. *ACM Trans. Comput. Log.*, 11(1), 2009.

16 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *ICALP (2)*, pages 139–150, 2009.

17 Thomas Colcombet. Fonctions régulières de coût. Habilitation Thesis (in French), 2013.

18 Thomas Colcombet, Denis Kuperberg, and Sylvain Lombardy. Regular temporal cost functions. In *ICALP (2)*, pages 563–574, 2010.

19 Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *ICALP (2)*, pages 398–409, 2008.

20 Thomas Colcombet and Christof Löding. Regular cost functions over finite trees. In *LICS*, pages 70–79, 2010.

21 E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *FOCS*, pages 328–337, 1988.

22 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS*, pages 368–377, 1991.

23 Nathanaël Fijalkow and Martin Zimmermann. Cost-parity and cost-streett games. In *FSTTCS*, pages 124–135, 2012.

24 Hugo Gimbert. Parity and exploration games on infinite graphs. In *CSL*, pages 56–70, 2004.

25 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

26 Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *STOC*, pages 60–65, 1982.

27 Eryk Kopczyński. Half-positional determinacy of infinite games. In *ICALP (2)*, pages 336–347, 2006.

28 Orna Kupferman and Moshe Y. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *CAV*, pages 36–52, 2000.

29 Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

30 Robert McNaughton. Infinite games played on finite graphs. *Ann. Pure Appl. Logic*, 65(2):149–184, 1993.

31 David E. Muller and Paul E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theor. Comput. Sci.*, 37:51–75, 1985.

32 Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989.

33 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the AMS*, 141:1–23, 1969.

34 Olivier Serre. Note on winning positions on pushdown games with ω-regular conditions. *Inf. Process. Lett.*, 85(6):285–291, 2003.

35 Olivier Serre. *Contribution à l'étude des jeux sur des graphes de processus à pile*. PhD thesis, Université Paris 7 - Denis Diderot, 2006.

36 Moshe Y. Vardi. Reasoning about the past with two-way automata. In *ICALP*, pages 628–641, 1998.

37 Igor Walukiewicz. Pushdown processes: Games and model-checking. *Inf. Comput.*, 164(2):234–263, 2001.

Annotation-Free Sequent Calculi for Full Intuitionistic Linear Logic*

Ranald Clouston, Jeremy Dawson, Rajeev Goré, and Alwen Tiu

Logic and Computation Group, Research School of Computer Science,
The Australian National University, Canberra ACT 0200, Australia

Abstract

Full Intuitionistic Linear Logic (FILL) is multiplicative intuitionistic linear logic extended with par. Its proof theory has been notoriously difficult to get right, and existing sequent calculi all involve inference rules with complex annotations to guarantee soundness and cut-elimination. We give a simple and annotation-free display calculus for FILL which satisfies Belnap's generic cut-elimination theorem. To do so, our display calculus actually handles an extension of FILL, called Bi-Intuitionistic Linear Logic (BiILL), with an 'exclusion' connective defined via an adjunction with par. We refine our display calculus for BiILL into a cut-free nested sequent calculus with deep inference in which the explicit structural rules of the display calculus become admissible. A separation property guarantees that proofs of FILL formulae in the deep inference calculus contain no trace of exclusion. Each such rule is sound for the semantics of FILL, thus our deep inference calculus and display calculus are conservative over FILL. The deep inference calculus also enjoys the subformula property and terminating backward proof search, which gives the NP-completeness of BiILL and FILL.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Linear logic, display calculus, nested sequent calculus, deep inference

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.197

1 Introduction

Multiplicative Intuitionistic Linear Logic (MILL) contains as connectives only tensor $\otimes$, its unit I, and its residual $\multimap$, where we use I rather than the usual 1 to avoid a clash with the categorical notation for terminal object. The connective par $\parr$ and its unit $\bot$ are traditionally only introduced when we move to classical Multiplicative Linear Logic (MLL), but Hyland and de Paiva's Full Intuitionistic Linear Logic (FILL) [20] shows that a sensible notion of par can be added to MILL without collapse to classicality. FILL's semantics are categorical, with the interaction between the $(\otimes, I, \multimap)$ and $(\parr, \bot)$ fragments entirely described by the equivalent formulae shown below:

$$(p \otimes (q \parr r)) \multimap ((p \otimes q) \parr r) \qquad ((p \multimap q) \parr r) \multimap (p \multimap (q \parr r)) \tag{1}$$

The first formula is variously called *weak distributivity* [20, 11], *linear distributivity* [12], and *dissociativity* [14]. The second we call Grishin (b) [16]. Its converse, called Grishin (a), is not FILL-valid, and indeed adding it to FILL recovers MLL.

From a traditional sequent calculus perspective, FILL is the logic specified by taking a two-sided sequent calculus for MLL, which enjoys cut-elimination, and restricting its $(\multimap R_2)$

* We gratefully acknowledge the comments of the anonymous reviewers. This work is partly supported by the ARC Discovery Projects DP110103173 and DP120101244.

rule to apply only to "singletons on the right", giving $(\multimap R_1)$, as shown below:

$$(\multimap R_1)\ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \qquad\qquad (\multimap R_2)\ \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}$$

Since exactly this restriction converts Gentzen's LK for ordinary classical logic to Gentzen's LJ for intuitionistic logic, FILL arises very naturally. Unfortunately the resulting calculus fails cut-elimination [26]. (Note that there is also work on natural deduction and proof nets for FILL [12, 1, 24, 13]. In this setting the problems of cut-elimination are side-stepped; see the discussion of "essential cuts" in [12] in particular.)

Hyland and de Paiva [20] therefore sought a middle ground between the too weak $(\multimap R_1)$ and the unsound $(\multimap R_2)$ by annotating formulae with *term assignments*, and using them to restrict the application of $(\multimap R_2)$ - the restriction requires that the variable typed by A not appear free in the terms typed by Δ. Reasoning with freeness in the presence of variable binders is notoriously tricky, and a bug was subsequently found by Bierman [4] which meant that the proof of the sequent below requires a cut that is not eliminable:

$$(a \parr b) \parr c \vdash a, (b \parr c \multimap d) \parr e \multimap d \parr e \tag{2}$$

Bierman [4] presented two possible corrections to the term assignment system, one due to Bellin. These were subsequently refined by Bräuner and de Paiva [6] to replace the term assignments by rules annotated with a binary relation between formulae on the left and on the right of the turnstile, which effectively trace variable occurrence. The only existing annotation-free sequent calculi for FILL [15, 16] are incorrect. The first [15] uses $(\multimap R_2)$ without the required annotations, making it unsound, and also contains other transcription errors. The second [16] identifies FILL with 'Bi-Linear Logic', which fails weak distributivity and has an extra connective called 'exclusion', of which more shortly.

The existing correct annotated sequent calculi [4, 6] have some weaknesses. First, the introduction rules for a connective do not define that connective in isolation, as was Gentzen's ideal. Instead, they introduce $\multimap$ on the right only when the context in which the rule sits obeys the rule's side-condition. A consequence is that they cannot be used for naive backward proof search since we must apply the rule upwards blindly, and then check the side-conditions once we have a putative derivation. Second, the term-calculus that results from the annotations has not been shown to have any computational content since its sole purpose is to block unsound inferences by tracking variable occurrence [6]. Thus, FILL's close relationship with other logics is obscured by these complex annotational devices, leading to it being described as proof-theoretically "curious" [12], and leading others to conclude that FILL "does not have a satisfactory proof theory" [9].

We believe these difficulties arise because efforts have focused on an 'unbalanced' logic. We show that adding an 'exclusion' connective $\prec$, dual to $\multimap$, gives a fully 'balanced' logic, which we call Bi-Intuitionistic Linear Logic (BiILL). The beauty of BiILL is that it has a simple *display calculus* [3, 16] BiILL$_{dc}$ that inherits Belnap's general cut-elimination theorem "for free". A similar situation has already been observed in classical modal logic, where it has proved impossible to extend traditional Gentzen sequents to a uniform and general proof-theory encompassing the numerous extensions of normal modal logic K. Display calculi capture a large class of such modal extensions uniformly and modularly [27, 22] by viewing them as fragments of (the display calculi for) tense logics, which conservatively extend modal logic with two modalities $\blacklozenge$ and $\blacksquare$, respectively adjoint to the original $\Box$ and $\Diamond$.

In tense logics, the conservativity result is trivial since both modal and tense logics are defined with respect to the same Kripke semantics. With BiILL and FILL, however, there is

no such existing conservativity result via semantics. The conservativity of BiILL over FILL would follow if we could show that a derivation of a FILL formula in BiILL$_{dc}$ preserved FILL-validity downwards: unfortunately, this does not hold, as explained next.

Belnap's generic cut-elimination procedure applies to BiILL$_{dc}$ because of the "display property", whereby any substructure of a sequent can be displayed as the whole of either the antecedent or succedent. The display property for BiILL$_{dc}$ is obtained via certain reversible structural rules, called *display rules*, which encode the various adjunctions between the connectives, such as the one between par and exclusion. Any BiILL$_{dc}$-derivation of a FILL formula that uses this adjunction to display a substructure contains occurrences of a structural connective which is an exact proxy for exclusion. That is, a BiILL$_{dc}$-derivation of a FILL formula may require inference steps that have no meaning in FILL, thus we cannot use our display calculus BiILL$_{dc}$ directly to show conservativity of BiILL over FILL. We circumvent this problem by showing that the structural rules to maintain the display property become *admissible*, provided one uses *deep inference*.

Following a methodology established for bi-intuitionistic and tense logics [17, 18], we show that the display calculus for BiILL can be refined to a *nested sequent calculus* [21, 7], called BiILL$_{dn}$, which contains no explicit structural rules, and hence no cut rule, as long as its introduction rules can act "deeply" on any substructure in a given structure. To prove that BiILL$_{dn}$ is sound and complete for BiILL, we use an intermediate nested sequent calculus called BiILL$_{sn}$ which, similar to our display calculus, has explicit structural rules, including cut, and uses *shallow* inference rules that apply only to the topmost sequent in a nested sequent. Our shallow inference calculus BiILL$_{sn}$ can simulate cut-free proofs of our display calculus BiILL$_{dc}$, and vice versa. It enjoys cut-elimination, the display property and coincides with the deep-inference calculus BiILL$_{dn}$ with respect to (cut-free) derivability. Together these imply that BiILL$_{dn}$ is sound and (cut-free) complete for BiILL. Our deep nested sequent calculus BiILL$_{dn}$ also enjoys a *separation property*: a BiILL$_{dn}$-derivation of a formula A uses only introduction rules for the connectives appearing in A. By selecting from BiILL$_{dn}$ only the introduction rules for the connectives in FILL, we obtain a nested (cut-free and deep inference) calculus FILL$_{dn}$ which is complete for FILL. We then show that the rules of FILL$_{dn}$ are also sound for the semantics of FILL. The conservativity of BiILL over FILL follows since a FILL formula A which is valid in BiILL will be cut-free derivable in BiILL$_{dc}$, and hence in BiILL$_{dn}$, and hence in FILL$_{dn}$, and hence valid in FILL.

Viewed upwards, introduction rules for display calculi use shallow inference and can require disassembling structures into an appropriate form using the display rules, meaning that display calculi do not enjoy a "substructure property". The modularity of display calculi also demands explicit structural rules for associativity, commutativity and weak-distributivity. These necessary aspects of display calculi make them unsuitable for proof search since the various structural rules and reversible rules can be applied indiscriminately. As structural rules are admissible in the nested deep inference calculus BiILL$_{dn}$, proof search in it is easier to manage than in the display calculus. Using BiILL$_{dn}$, we show that the tautology problem for BiILL and FILL are in fact NP-complete.

For full proof details we refer readers to the extended version of this paper [10].

2 Display Calculi

2.1 Syntax

▶ **Definition 1.** BiILL-*formulae* are defined using the grammar below where p is from some fixed set of propositional variables

$$A ::= p \mid I \mid \bot \mid A \otimes A \mid A \parr A \mid A \multimap A \mid A \prec A$$

Antecedent and *succedent* BiILL-*structures* (also known as antecedent and succedent parts) are defined by mutual induction, where Φ is a structural constant and A is a BiILL-formula:

$$X_a ::= A \mid \Phi \mid X_a, X_a \mid X_a < X_s \qquad\qquad X_s ::= A \mid \Phi \mid X_s, X_s \mid X_a > X_s$$

FILL-formulae are BiILL-formulae with no occurrence of the exclusion connective $\prec$. FILL-structures are BiILL-structures with no occurrence of $<$, and containing only FILL-formulae. We stipulate that $\otimes$ and $\parr$ bind tighter than $\multimap$ and $\prec$, that comma binds tighter than $>$ and $<$, and resolve $A \multimap B \multimap C$ as $A \multimap (B \multimap C)$. A BiILL- (resp. FILL-) *sequent* is a pair comprising an antecedent and a succedent BiILL- (resp. FILL-) structure, written $X_a \vdash X_s$.

▶ **Definition 2.** We can translate sequents $X \vdash Y$ into formulae as $\tau^a(X) \multimap \tau^s(Y)$, given the mutually inductively defined antecedent and succedent τ-*translations*:

	A	Φ	X, Y	$X > Y$	$X < Y$
τ^a	A	I	$\tau^a(X) \otimes \tau^a(Y)$		$\tau^a(X) \prec \tau^s(Y)$
τ^s	A	$\bot$	$\tau^s(X) \parr \tau^s(Y)$	$\tau^a(X) \multimap \tau^s(Y)$	

Hence Φ and comma are *overloaded* to be translated into different connectives depending on their position. By uniformly replacing our structural connective $<$ with $>$, we could have also overloaded $>$ to stand for $\multimap$ and $\prec$, which would have avoided the blank spaces in the above table, but we have opted to use different connectives to help visually emphasise whether a given structure lives in BiILL or its fragment FILL.

The display calculi for FILL and BiILL are given in Fig. 1.

▶ Remark. For conciseness, we treat comma-separated structures as multisets and usually omit explicit use of (Ass $\vdash$), ($\vdash$ Ass), (Com $\vdash$) and ($\vdash$ Com). The *residuated pair* and *dual residuated pair* rules (rp) and (drp) are the *display postulates* which give Thm. 3 below. Our display postulates build in commutativity of comma, so the two (Com) rules are derivable. If we wanted to drop commutativity [12], we would have to use the more general display postulates from [16]. Note that (drp) may create the structure $<$ which has no meaning in FILL, so we will return to this issue. For now, observe that proofs of even apparently trivial FILL-sequents such as $(p \parr q) \parr r \vdash p, (q \parr r)$ require (drp) to 'move p out the way' so ($\vdash \parr$) can be applied. Another (drp) then eliminates the $<$ to restore p to the right. The rule ($\vdash$ Grnb) is the structural version of Grishin (b), the right hand formula of (1); the rule (Grnb $\vdash$) is equivalent. Fig. 2 gives a cut-free proof of the example from Bierman (2).

▶ **Theorem 3** (Display Property). *For every structure Z which is an antecedent (resp. succedent) part of the sequent $X \vdash Y$, there is a sequent $Z \vdash Y'$ (resp. $X' \vdash Z$) obtainable from $X \vdash Y$ using only (rp) and (drp), thereby displaying the Z as the whole of one side.*

▶ **Theorem 4** (Cut-Admissibility). *From cut-free* BiILL$_{dc}$*-derivations of $X \vdash A$ and $A \vdash Y$ there is an effective procedure to obtain a cut-free* BiILL$_{dc}$*-derivation of $X \vdash Y$.*

Proof. BiILL$_{dc}$ obeys Belnap's conditions for cut-admissibility [3]: see App. A. ◀

Cut and identity:

$$(\text{id})\quad p \vdash p \qquad (\text{cut})\ \frac{X \vdash A \qquad A \vdash Y}{X \vdash Y}$$

Logical rules:

$$(I \vdash)\ \frac{\Phi \vdash X}{I \vdash X} \qquad (\vdash I)\quad \Phi \vdash I$$

$$(\bot \vdash)\quad \bot \vdash \Phi \qquad (\vdash \bot)\ \frac{X \vdash \Phi}{X \vdash \bot}$$

$$(\otimes \vdash)\ \frac{A, B \vdash X}{A \otimes B \vdash X} \qquad (\vdash \otimes)\ \frac{X \vdash A \qquad Y \vdash B}{X, Y \vdash A \otimes B}$$

$$(\parr \vdash)\ \frac{A \vdash X \qquad B \vdash Y}{A \parr B \vdash X, Y} \qquad (\vdash \parr)\ \frac{X \vdash A, B}{X \vdash A \parr B}$$

$$(\multimap \vdash)\ \frac{X \vdash A \qquad B \vdash Y}{A \multimap B \vdash X > Y} \qquad (\vdash \multimap)\ \frac{X \vdash A > B}{X \vdash A \multimap B}$$

Structural rules:

$$(\text{rp})\ \frac{X \vdash Y > Z}{X, Y \vdash Z} \qquad (\text{rp})\ \frac{X, Y \vdash Z}{Y \vdash X > Z} \qquad (\text{drp})\ \frac{X < Y \vdash Z}{X \vdash Y, Z} \qquad (\text{drp})\ \frac{X \vdash Y, Z}{X < Z \vdash Y}$$

$$(\Phi \vdash)\ \frac{X, \Phi \vdash Y}{X \vdash Y} \qquad (\vdash \Phi)\ \frac{X \vdash \Phi, Y}{X \vdash Y}$$

$$(\text{Ass} \vdash)\ \frac{W, (X, Y) \vdash Z}{(W, X), Y \vdash Z} \qquad (\vdash \text{Ass})\ \frac{W \vdash (X, Y), Z}{W \vdash X, (Y, Z)}$$

$$(\text{Com} \vdash)\ \frac{X, Y \vdash Z}{Y, X \vdash Z} \qquad (\vdash \text{Com})\ \frac{X \vdash Y, Z}{X \vdash Z, Y}$$

$$(\text{Grnb} \vdash)\ \frac{W, (X < Y) \vdash Z}{(W, X) < Y \vdash Z} \qquad (\vdash \text{Grnb})\ \frac{W \vdash (X > Y), Z}{W \vdash X > (Y, Z)}$$

Further logical rules for BiILL$_{dc}$:

$$(\prec \vdash)\ \frac{A < B \vdash X}{A \prec B \vdash X} \qquad (\vdash \prec)\ \frac{X \vdash A \qquad B \vdash Y}{X < Y \vdash A \prec B}$$

Figure 1 FILL$_{dc}$ and BiILL$_{dc}$: display calculi for FILL and BiILL.

2.2 Semantics

▶ **Definition 5.** A *FILL-category* is a category equipped with

- a symmetric monoidal closed structure $(\otimes, I, \multimap)$
- a symmetric monoidal structure $(\parr, \bot)$
- a natural family of *weak distributivity* arrows $A \otimes (B \parr C) \to (A \otimes B) \parr C$.

A *BiILL-category* is a FILL-category where the $\parr$ bifunctor has a *co-closure* $\prec$, so there is a natural isomorphism between arrows $A \to B \parr C$ and $A \prec B \to C$.

▶ **Definition 6.** The *free* FILL- (resp. BiILL-) category has FILL- (resp. BiILL-) formulae as objects and the following arrows (quotiented by certain equations) where we are given objects A, A', A'', B, B' and arrows $f : A \to A', f' : A' \to A'', g : B \to B'$, $(\heartsuit, K) \in \{(\otimes, I), (\parr, \bot)\}$, and where the co-closure arrows exist in the free BiILL-category only:

Category: $A \xrightarrow{id} A \qquad A \xrightarrow{f' \circ f} A''$

Symmetric Monoidal: $A \heartsuit B \xrightarrow{f \heartsuit g} A' \heartsuit B' \qquad (A \heartsuit B) \heartsuit C \underset{\alpha^{-1}}{\overset{\alpha}{\rightleftarrows}} A \heartsuit (B \heartsuit C)$

$$K \heartsuit A \underset{\lambda^{-1}}{\overset{\lambda}{\rightleftarrows}} A \qquad A \heartsuit K \underset{\rho^{-1}}{\overset{\rho}{\rightleftarrows}} A \qquad A \heartsuit B \xrightarrow{\gamma} B \heartsuit A$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a \vdash a \qquad b \vdash b}{a \parr b \vdash a, b}\,(\parr\vdash) \qquad c \vdash c}{(a \parr b) \parr c \vdash a, b, c}\,(\parr\vdash)}{(a \parr b) \parr c < a \vdash b, c}\,\mathbf{(drp)}}{(a \parr b) \parr c < a \vdash b \parr c}\,(\vdash \parr) \qquad d \vdash d}{b \parr c \multimap d \vdash ((a \parr b) \parr c < a) > d}\,(\multimap\vdash) \qquad e \vdash e}{(b \parr c \multimap d) \parr e \vdash (((a \parr b) \parr c < a) > d), e}\,(\parr\vdash)}{(b \parr c \multimap d) \parr e \vdash ((a \parr b) \parr c < a) > d, e}\,(\vdash \text{Grnb})}{(b \parr c \multimap d) \parr e, ((a \parr b) \parr c < a) \vdash d, e}\,(\text{rp})}{(b \parr c \multimap d) \parr e, ((a \parr b) \parr c < a) \vdash d \parr e}\,(\vdash \parr)}{(a \parr b) \parr c < a \vdash (b \parr c \multimap d) \parr e > d \parr e}\,(\text{rp})}{(a \parr b) \parr c < a \vdash (b \parr c \multimap d) \parr e \multimap d \parr e}\,(\vdash\multimap)}{(a \parr b) \parr c \vdash a, (b \parr c \multimap d) \parr e \multimap d \parr e}\,\mathbf{(drp)}
$$

Figure 2 The cut-free FILL$_{dc}$-derivation of the example from Bierman.

Closed: $A \multimap B \xrightarrow{A\multimap g} A \multimap B' \qquad (A \multimap B) \otimes A \xrightarrow{\varepsilon} B \qquad A \xrightarrow{\eta} B \multimap A \otimes B$

Weak Distributivity: $A \otimes (A' \parr A'') \xrightarrow{\omega} (A \otimes A') \parr A''$

Co-Closed: $A \prec B \xrightarrow{f\prec B} A' \prec B \qquad A \parr B \prec A \xrightarrow{\varepsilon} B \qquad A \xrightarrow{\eta} B \parr (A \prec B)$

We will suppress explicit reference to the associativity and symmetry arrows.

▶ **Definition 7.** A FILL- (resp. BiILL-) sequent $X \vdash Y$ is *satisfied* by a FILL- (resp. BiILL-) category if, given any valuation of its propositional variables as objects, there exists an arrow $I \to \tau^a(X) \multimap \tau^s(Y)$. It is FILL- (resp. BiILL-) *valid* if it is satisfied by all such categories. In fact, we only need to check the free categories under their generic valuations.

▶ Remark. Those familiar with categorical logic will note that our use of category theory here is rather shallow, looking only at whether hom-sets are populated, and not at the rich structure of equivalences between proofs that categorical logic supports. This is an adequate basis for this work because the question of FILL-validity alone has proved so vexed.

▶ **Theorem 8.** BiILL$_{dc}$ *(Fig. 1) is* sound *and* cut-free complete *for BiILL-validity.*

Proof. BiILL$_{dc}$-proof rules and the arrows of the free BiILL-category are interdefinable. ◀

▶ **Corollary 9.** *The display calculus* FILL$_{dc}$ *is cut-free complete for FILL-validity.*

Proof. Because BiILL-categories are FILL-categories, and BiILL$_{dc}$ proofs of FILL-sequents are FILL$_{dc}$ proofs. ◀

We will return to the question of *soundness* for FILL$_{dc}$ in Sec. 4.

3 Deep Inference and Proof Search

We now present a refinement of the display calculus BiILL$_{dc}$, in the form of a nested sequent calculus, that is more suitable for proof search. A nested sequent is essentially just a structure in display calculus, but presented in a more sequent-like notation. This change of notation allows us to present the proof systems much more concisely. The proof system we are interested in is the deep inference system in Sec. 3.2, but we shall first present an intermediate system, BiILL$_{sn}$, which is closer to display calculus, and which eases the proof of correspondence between the deep inference calculus and the display calculus for BiILL.

3.1 The Shallow Inference Calculus

The syntax of nested sequents is given by the grammar below where A_i and B_j are formulae.

$$S \;\; T ::= S_1, \ldots, S_k, A_1, \ldots, A_m \Rightarrow B_1, \ldots, B_n, T_1, \ldots, T_l$$

We use Γ and Δ for multisets of formulae and use P, Q, S, T, X, Y, etc., for sequents, and $\mathcal{S}$, $\mathcal{X}$, etc., for multisets of sequents and formulae. The empty multiset is $\cdot$ ('dot').

A nested sequent can naturally be represented as a tree structure as follows. The nodes of the tree are traditional two-sided sequents (i.e., pairs of multisets). The edges between nodes are labelled with either a $-$, denoting nesting to the left of the sequent arrow, or a $+$, denoting nesting to the right of the sequent arrow. For example, the nested sequent below can be visualised as the tree in Fig. 3 (i):

$$(e, f \Rightarrow g), (p, (u, v \Rightarrow x, y) \Rightarrow q, r), a, b \Rightarrow c, d, (\cdot \Rightarrow s) \tag{3}$$

A display sequent can be seen as a nested sequent, where $\vdash$, $>$ and $<$ are all replaced by $\Rightarrow$ and the unit Φ is represented by the empty multiset. The definition of a nested sequent incorporates implicitly the associativity and commutativity of comma, and the effects of its unit, via the multiset structure.

▶ **Definition 10.** Following Def. 2, we can translate nested sequents into equivalence classes of BiILL-formulae (modulo associativity, commutativity, and unit laws) via *τ-translations*:

$$\begin{aligned}&\tau^a(S_1, \ldots, S_k, A_1, \ldots, A_m \Rightarrow B_1, \ldots, B_n, T_1, \ldots, T_l)\\ &= (\tau^a(S_1) \otimes \cdots \otimes \tau^a(S_k) \otimes A_1 \otimes \cdots \otimes A_m) \prec (B_1 \parr \cdots \parr B_n \parr \tau^s(T_1) \parr \cdots \parr \tau^s(T_l))\end{aligned}$$

$$\begin{aligned}&\tau^s(S_1, \ldots, S_k, A_1, \ldots, A_m \Rightarrow B_1, \ldots, B_n, T_1, \ldots, T_l)\\ &= (\tau^a(S_1) \otimes \cdots \otimes \tau^a(S_k) \otimes A_1 \otimes \cdots \otimes A_m) \multimap (B_1 \parr \cdots \parr B_n \parr \tau^s(T_1) \parr \cdots \parr \tau^s(T_l)).\end{aligned}$$

The translations τ^a and τ^s differ only in their translation of the sequent symbol $\Rightarrow$ to $\multimap$ and $\prec$ respectively. Where $m = 0$, $A_1 \otimes \cdots \otimes A_m$ translates to I, and similarly $B_1 \parr \cdots \parr B_n$ translates to $\bot$ when $n = 0$. These translations each extend to a map from multisets of nested sequents and formulae to formulae: τ^a (resp. τ^s) acts on each sequent as above, leaves formulae unchanged, and connects the resulting formulae with $\otimes$ (resp. $\parr$). Empty multisets are mapped to I (resp. $\bot$).

A *context* is either a 'hole' $[\ ]$, called the *empty context*, or a sequent where exactly one node has been replaced by a hole $[\ ]$. Contexts are denoted by $X[\ ]$. We write $X[S]$ to denote a sequent resulting from replacing the hole $[\ ]$ in $X[\ ]$ with the sequent S. A non-empty context $X[\ ]$ is *positive* if the hole $[\ ]$ occurs immediately to the right of a sequent arrow $\Rightarrow$, and *negative* otherwise. This simple definition of polarities of a context is made possible by the use of the same symbol $\Rightarrow$ to denote the structural counterparts of $\multimap$ and $\prec$. As we shall see in Sec. 3.2, this overloading of $\Rightarrow$ allows a presentation of deep inference rules that ignores context polarity.

The shallow inference system BiILL$_{sn}$ for BiILL is given in Fig. 4. The main difference from BiILL$_{dc}$ is that we allow multiple-conclusion logical rules. This implicitly builds the Grishin (b) rules into the logical rules (see [10]).

▶ **Theorem 11.** *A formula is cut-free* BiILL$_{sn}$*-provable iff it is cut-free* BiILL$_{dc}$*-provable.*

▶ **Corollary 12.** *The cut rule is admissible in* BiILL$_{sn}$*.*

Just as in display calculus (Thm. 3), the display property holds for BiILL$_{sn}$.

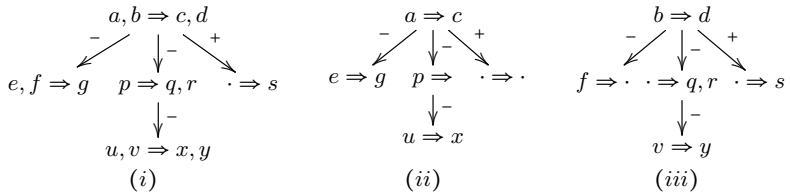


Figure 3 A tree representation of a nested sequent (i), and its partitions (ii and iii).

Cut and identity: $\dfrac{}{p \Rightarrow p}\ id \qquad \dfrac{\mathcal{S} \Rightarrow \mathcal{S}', A \quad A, \mathcal{T} \Rightarrow \mathcal{T}'}{\mathcal{S}, \mathcal{T} \Rightarrow \mathcal{S}', \mathcal{T}'}\ cut$

Structural rules:

$$\frac{\mathcal{S} \Rightarrow \mathcal{T}, \mathcal{T}'}{(\mathcal{S} \Rightarrow \mathcal{T}) \Rightarrow \mathcal{T}'}\ drp_1 \qquad \frac{\mathcal{S}, \mathcal{T} \Rightarrow \mathcal{T}'}{\mathcal{S} \Rightarrow (\mathcal{T} \Rightarrow \mathcal{T}')}\ rp_1 \qquad \frac{(\mathcal{S} \Rightarrow \mathcal{S}'), \mathcal{T} \Rightarrow \mathcal{T}'}{(\mathcal{S}, \mathcal{T} \Rightarrow \mathcal{S}') \Rightarrow \mathcal{T}'}\ gl$$

$$\frac{(\mathcal{S} \Rightarrow \mathcal{T}) \Rightarrow \mathcal{T}'}{\mathcal{S} \Rightarrow \mathcal{T}, \mathcal{T}'}\ drp_2 \qquad \frac{\mathcal{S} \Rightarrow (\mathcal{T} \Rightarrow \mathcal{T}')}{\mathcal{S}, \mathcal{T} \Rightarrow \mathcal{T}'}\ rp_2 \qquad \frac{\mathcal{S} \Rightarrow (\mathcal{S}' \Rightarrow \mathcal{T}'), \mathcal{T}}{\mathcal{S} \Rightarrow (\mathcal{S}' \Rightarrow \mathcal{T}', \mathcal{T})}\ gr$$

Logical rules:

$$\frac{}{\bot \Rightarrow \cdot}\ \bot_l \qquad \frac{\mathcal{S} \Rightarrow \mathcal{T}}{\mathcal{S} \Rightarrow \mathcal{T}, \bot}\ \bot_r \qquad \frac{\mathcal{S} \Rightarrow \mathcal{T}}{\mathcal{S}, \mathrm{I} \Rightarrow \mathcal{T}}\ \mathrm{I}_l \qquad \frac{}{\cdot \Rightarrow \mathrm{I}}\ \mathrm{I}_r$$

$$\frac{\mathcal{S}, A, B \Rightarrow \mathcal{T}}{\mathcal{S}, A \otimes B \Rightarrow \mathcal{T}}\ \otimes_l \qquad \frac{\mathcal{S} \Rightarrow A, \mathcal{T} \quad \mathcal{S}' \Rightarrow B, \mathcal{T}'}{\mathcal{S}, \mathcal{S}' \Rightarrow A \otimes B, \mathcal{T}, \mathcal{T}'}\ \otimes_r$$

$$\frac{\mathcal{S}, A \Rightarrow \mathcal{T} \quad \mathcal{S}', B \Rightarrow \mathcal{T}'}{\mathcal{S}, \mathcal{S}', A \parr B \Rightarrow \mathcal{T}, \mathcal{T}'}\ \parr_l \qquad \frac{\mathcal{S} \Rightarrow A, B, \mathcal{T}}{\mathcal{S} \Rightarrow A \parr B, \mathcal{T}}\ \parr_r$$

$$\frac{\mathcal{S} \Rightarrow A, \mathcal{T} \quad \mathcal{S}', B \Rightarrow \mathcal{T}'}{\mathcal{S}, \mathcal{S}', A \multimap B \Rightarrow \mathcal{T}, \mathcal{T}'}\ \multimap_l \qquad \frac{\mathcal{S} \Rightarrow \mathcal{T}, (A \Rightarrow B)}{\mathcal{S} \Rightarrow \mathcal{T}, A \multimap B}\ \multimap_r$$

$$\frac{\mathcal{S}, (A \Rightarrow B) \Rightarrow \mathcal{T}}{\mathcal{S}, A \prec B \Rightarrow \mathcal{T}}\ \prec_l \qquad \frac{\mathcal{S} \Rightarrow A, \mathcal{T} \quad \mathcal{S}', B \Rightarrow \mathcal{T}'}{\mathcal{S}, \mathcal{S}' \Rightarrow A \prec B, \mathcal{T}, \mathcal{T}'}\ \prec_r$$

Figure 4 The shallow inference system BiILL$_{sn}$, where gl and gr capture Grishin (b).

▶ **Proposition 13** (Display property). *Let $X[\ ]$ be a positive (negative) context. For every $\mathcal{S}$, there exists $\mathcal{T}$ such that $\mathcal{T} \Rightarrow \mathcal{S}$ (respectively $\mathcal{S} \Rightarrow \mathcal{T}$) is derivable from $X[\mathcal{S}]$ using only the structural rules from $\{drp_1, drp_2, rp_1, rp_2\}$. Thus $\mathcal{S}$ is "displayed" in $\mathcal{T} \Rightarrow \mathcal{S}$ ($\mathcal{S} \Rightarrow \mathcal{T}$).*

3.2 The Deep Inference Calculus

A deep inference rule can be applied to any sequent within a nested sequent. This poses a problem in formalising context splitting rules, e.g., $\otimes$ on the right. To be sound, we need to consider a context splitting that splits an entire tree of sequents, as formalised next.

Given two sequents X_1 and X_2, their *merge set* $X_1 \bullet X_2$ is defined inductively as:

$$\begin{aligned} X_1 \bullet X_2 = \{\ (\Gamma_1, \Gamma_2, Y_1, \ldots, Y_m \Rightarrow \Delta_1, \Delta_2, Z_1, \ldots, Z_n)\ \mid\ & \\ X_1 = (\Gamma_1, P_1, \ldots, P_m \Rightarrow \Delta_1, Q_1, \ldots, Q_n)\ \text{ and } & \\ X_2 = (\Gamma_2, S_1, \ldots, S_m \Rightarrow \Delta_2, T_1, \ldots, T_n)\ \text{ and } & \\ Y_i \in P_i \bullet S_i \text{ for } 1 \le i \le m \text{ and } Z_j \in Q_j \bullet T_j \text{ for } 1 \le j \le n\ \ \} & \end{aligned}$$

Note that the merge set of two sequents may not always be defined since mergeable sequents need to have the same structure. Note also that, because there can be more than

Propagation rules:

$$\frac{X[\mathcal{S} \Rightarrow (A, \mathcal{S}' \Rightarrow \mathcal{T}'), \mathcal{T}]}{X[\mathcal{S}, A \Rightarrow (\mathcal{S}' \Rightarrow \mathcal{T}'), \mathcal{T}]}\ pl_1 \qquad \frac{X[(\mathcal{S} \Rightarrow \mathcal{T}, A), \mathcal{S}' \Rightarrow \mathcal{T}']}{X[(\mathcal{S} \Rightarrow \mathcal{T}), \mathcal{S}' \Rightarrow A, \mathcal{T}']}\ pr_1$$

$$\frac{X[\mathcal{S}, A, (\mathcal{S}' \Rightarrow \mathcal{T}') \Rightarrow \mathcal{T}]}{X[\mathcal{S}, (\mathcal{S}', A \Rightarrow \mathcal{T}') \Rightarrow \mathcal{T}]}\ pl_2 \qquad \frac{X[\mathcal{S} \Rightarrow \mathcal{T}, A, (\mathcal{S}' \Rightarrow \mathcal{T}')]}{X[\mathcal{S} \Rightarrow \mathcal{T}, (\mathcal{S}' \Rightarrow \mathcal{T}', A)]}\ pr_2$$

Identity and logical rules: In branching rules, $X[\ ] \in X_1[\ ] \bullet X_2[\ ]$, $\mathcal{S} \in \mathcal{S}_1 \bullet \mathcal{S}_2$ and $\mathcal{T} \in \mathcal{T}_1 \bullet \mathcal{T}_2$.

$$\frac{X[\ ], \mathcal{U} \text{ and } \mathcal{V} \text{ are hollow.}}{X[\mathcal{U}, p \Rightarrow p, \mathcal{V}]}\ id^d \qquad \frac{X[\ ], \mathcal{U} \text{ and } \mathcal{V} \text{ are hollow.}}{X[\bot, \mathcal{U} \Rightarrow \mathcal{V}]}\ \bot_l^d \qquad \frac{X[\mathcal{S} \Rightarrow \mathcal{T}]}{X[\mathcal{S} \Rightarrow \mathcal{T}, \bot]}\ \bot_r^d$$

$$\frac{X[\mathcal{S} \Rightarrow \mathcal{T}]}{X[\mathcal{S}, \mathrm{I} \Rightarrow \mathcal{T}]}\ \mathrm{I}_l^d \qquad \frac{X[\ ], \mathcal{U} \text{ and } \mathcal{V} \text{ are hollow.}}{X[\mathcal{U} \Rightarrow \mathrm{I}, \mathcal{V}]}\ \mathrm{I}_r^d$$

$$\frac{X[\mathcal{S}, A, B \Rightarrow \mathcal{T}]}{X[\mathcal{S}, A \otimes B \Rightarrow \mathcal{T}]}\ \otimes_l^d \qquad \frac{X_1[\mathcal{S}_1 \Rightarrow A, \mathcal{T}_1] \quad X_2[\mathcal{S}_2 \Rightarrow B, \mathcal{T}_2]}{X[\mathcal{S} \Rightarrow A \otimes B, \mathcal{T}]}\ \otimes_r^d$$

$$\frac{X_1[\mathcal{S}_1 \Rightarrow A, \mathcal{T}_1] \quad X_2[\mathcal{S}_2, B \Rightarrow \mathcal{T}_2]}{X[\mathcal{S}, A \multimap B \Rightarrow \mathcal{T}]}\ \multimap_l^d \qquad \frac{X[\mathcal{S} \Rightarrow \mathcal{T}, (A \Rightarrow B)]}{X[\mathcal{S} \Rightarrow \mathcal{T}, A \multimap B]}\ \multimap_r^d$$

$$\frac{X_1[\mathcal{S}_1, A \Rightarrow \mathcal{T}_1] \quad X_2[\mathcal{S}_2, B \Rightarrow \mathcal{T}_2]}{X[\mathcal{S}, A \parr B \Rightarrow \mathcal{T}]}\ \parr_l^d \qquad \frac{X[\mathcal{S} \Rightarrow A, B, \mathcal{T}]}{X[\mathcal{S} \Rightarrow A \parr B, \mathcal{T}]}\ \parr_r^d$$

$$\frac{X[\mathcal{S}, (A \Rightarrow B) \Rightarrow \mathcal{T}]}{X[\mathcal{S}, A \prec B \Rightarrow \mathcal{T}]}\ \prec_l^d \qquad \frac{X_1[\mathcal{S}_1 \Rightarrow A, \mathcal{T}_1] \quad X_2[\mathcal{S}_2, B \Rightarrow \mathcal{T}_2]}{X[\mathcal{S} \Rightarrow A \prec B, \mathcal{T}]}\ \prec_r^d$$

Figure 5 The deep inference system BiILL$_{dn}$.

one way to enumerate elements of a multiset in the left/right hand side of a sequent, the result of the merging of two nested sequents is a set, rather than a single nested sequent. When $X \in X_1 \bullet X_2$, we say that X_1 and X_2 are a *partition* of X. Fig. 3 (ii) and (iii) show a partitioning of the nested sequent (3) in the tree representation. Note that the partitions (ii) and (iii) must have the same tree structure as the original sequent (i).

Given two contexts $X_1[\ ]$ and $X_2[\ ]$ their merge set $X_1[\ ] \bullet X_2[\ ]$ is defined as follows:

If $X_1[\ ] = [\ ]$ and $X_2[\ ] = [\ ]$ then $X_1[\ ] \bullet X_2[\ ] = \{[\ ]\}$

If $X_1[\ ] = (\Gamma_1, Y_1[\ ], P_1, \ldots, P_m \Rightarrow \Delta_1, Q_1, \ldots, Q_n)$ and
$X_2[\ ] = (\Gamma_2, Y_2[\ ], S_1, \ldots, S_m \Rightarrow \Delta_2, T_1, \ldots, T_n)$ then
$X_1[\ ] \bullet X_2[\ ] = \{\ (\Gamma_1, \Gamma_2, Y[\ ], U_1, \ldots, U_m \Rightarrow \Delta_1, \Delta_2, V_1, \ldots, V_n) \mid$
$Y[\ ] \in Y_1[\ ] \bullet Y_2[\ ]$ and $U_i \in P_i \bullet S_i$ for $1 \le i \le m$ and
$V_j \in Q_j \bullet T_j$ for $1 \le j \le n\ \}$

If $X_1[\ ] = (\Gamma_1, P_1, \ldots, P_m \Rightarrow \Delta_1, Y_1[\ ], Q_1, \ldots, Q_n)$ and
$X_2[\ ] = (\Gamma_2, S_1, \ldots, S_m \Rightarrow \Delta_2, Y_2[\ ], T_1, \ldots, T_n)$ then
$X_1[\ ] \bullet X_2[\ ] = \{\ (\Gamma_1, \Gamma_2, U_1, \ldots, U_m \Rightarrow \Delta_1, \Delta_2, Y[\ ], V_1, \ldots, V_n) \mid$
$Y[\ ] \in Y_1[\ ] \bullet Y_2[\ ]$ and $U_i \in P_i \bullet S_i$ for $1 \le i \le m$ and
$V_j \in Q_j \bullet T_j$ for $1 \le j \le n\ \}$

If $X[\ ] = X_1[\ ] \bullet X_2[\ ]$ we say $X_1[\ ]$ and $X_2[\ ]$ are a *partition* of $X[\ ]$.

We extend the notion of a merge set between multisets of formulae and sequents as follows. Given $\mathcal{X} = \Gamma \cup \{X_1, \ldots, X_n\}$ and $\mathcal{Y} = \Delta \cup \{Y_1, \ldots, Y_n\}$ their merge set contains all multisets of the form: $\Gamma \cup \Delta \cup \{Z_1, \ldots, Z_n\}$ where $Z_i \in X_i \bullet Y_i$.

A nested sequent X (resp. a context $X[\ ]$) is said to be *hollow* iff it contains no occurrences of formulae. For example, $(\cdot \Rightarrow \cdot) \Rightarrow (\cdot \Rightarrow [\ ]), (\cdot \Rightarrow \cdot)$ is a hollow context.

The deep inference system for BiILL, called BiILL$_{dn}$, is given in Fig. 5. Fig. 6 shows a cut-free derivation of Bierman's example in BiILL$_{dn}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a \Rightarrow a, (\cdot \Rightarrow \cdot)}\, id^d \quad \dfrac{\dfrac{}{\cdot \Rightarrow (b \Rightarrow b)}\, id^d}{b \Rightarrow (\cdot \Rightarrow b)}\, pl_1}{\mathbf{a} \parr \mathbf{b} \Rightarrow a, (\cdot \Rightarrow b)}\, \parr_l^d \quad \dfrac{\dfrac{}{\cdot \Rightarrow (c \Rightarrow c)}\, id^d}{c \Rightarrow (\cdot \Rightarrow c)}\, pl_1}{(\mathbf{a} \parr \mathbf{b}) \parr \mathbf{c} \Rightarrow a, (\cdot \Rightarrow b, c)}\, \parr_l^d}{(a \parr b) \parr c \Rightarrow a, (\cdot \Rightarrow \mathbf{b} \parr \mathbf{c})}\, \parr_r^d \quad \dfrac{}{\cdot \Rightarrow (d \Rightarrow d)}\, id^d}{(a \parr b) \parr c \Rightarrow a, (\mathbf{b} \parr \mathbf{c} \multimap \mathbf{d} \Rightarrow d)}\, \multimap_l^d \quad \dfrac{}{\cdot \Rightarrow (e \Rightarrow e)}\, id^d}{(a \parr b) \parr c \Rightarrow a, ((\mathbf{b} \parr \mathbf{c} \multimap \mathbf{d}) \parr \mathbf{e} \Rightarrow d, e)}\, \parr_l^d}{(a \parr b) \parr c \Rightarrow a, ((b \parr c \multimap d) \parr e \Rightarrow \mathbf{d} \parr \mathbf{e})}\, \parr_r^d}{(a \parr b) \parr c \Rightarrow a, (\mathbf{b} \parr \mathbf{c} \multimap \mathbf{d}) \parr \mathbf{e} \multimap \mathbf{d} \parr \mathbf{e}}\, \multimap_r^d$$

Figure 6 A cut-free derivation of Bierman's example in BiILL$_{dn}$.

3.3 The Equivalence of the Deep and Shallow Nested Sequent Calculi

From BiILL$_{dn}$ to BiILL$_{sn}$, it is enough to show that every deep inference rule is *cut-free derivable* in BiILL$_{sn}$. For the identity and the constant rules, this follows from the fact that hollow structures can be weakened away, as they add nothing to provability (see [10]). For the other logical rules, a key idea to their soundness is that the context splitting operation is derivable in BiILL$_{sn}$. This is a consequence of the following lemma (see [10]).

▶ **Lemma 14.** *The following rules are derivable in* BiILL$_{sn}$ *without cut:*

$$\dfrac{(\mathcal{X}_1 \Rightarrow \mathcal{Y}_1), (\mathcal{X}_2 \Rightarrow \mathcal{Y}_2), \mathcal{U} \Rightarrow \mathcal{V}}{(\mathcal{X}_1, \mathcal{X}_2 \Rightarrow \mathcal{Y}_1, \mathcal{Y}_2), \mathcal{U} \Rightarrow \mathcal{V}}\, dist_l \qquad \dfrac{\mathcal{U} \Rightarrow \mathcal{V}, (\mathcal{X}_1 \Rightarrow \mathcal{Y}_1), (\mathcal{X}_2 \Rightarrow \mathcal{Y}_2)}{\mathcal{U} \Rightarrow \mathcal{V}, (\mathcal{X}_1, \mathcal{X}_2 \Rightarrow \mathcal{Y}_1, \mathcal{Y}_2)}\, dist_r$$

Intuitively, these rules embody the weak distributivity formalised by the Grishin (b) rule.

▶ **Lemma 15.** *If* $\mathcal{X} \in \mathcal{X}_1 \bullet \mathcal{X}_2$ *then the rules below are cut-free derivable in* BiILL$_{sn}$*:*

$$\dfrac{\mathcal{X}_1, \mathcal{X}_2, \mathcal{U} \Rightarrow \mathcal{V}}{\mathcal{X}, \mathcal{U} \Rightarrow \mathcal{V}}\, m_l \qquad \dfrac{\mathcal{U} \Rightarrow \mathcal{V}, \mathcal{X}_1, \mathcal{X}_2}{\mathcal{U} \Rightarrow \mathcal{V}, \mathcal{X}}\, m_r$$

Proof. This follows straightforwardly from Lem. 14. ◀

▶ **Lemma 16.** *Suppose* $X[\ ] \in X_1[\ ] \bullet X_2[\ ]$ *and suppose there exists* $Y[\ ]$ *such that for any* $\mathcal{U}$ *and any* $\rho \in \{drp_1, drp_2, rp_1, rp_2\}$*, the figure below left is a valid inference rule in* BiILL$_{sn}$*:*

$$\dfrac{Y[\mathcal{U}]}{X[\mathcal{U}]}\, \rho \qquad \dfrac{Y_1[\mathcal{U}]}{X_1[\mathcal{U}]}\, \rho \qquad \dfrac{Y_2[\mathcal{U}]}{X_2[\mathcal{U}]}\, \rho$$

Then there exists $Y_1[\ ]$ *and* $Y_2[\ ]$ *such that* $Y[\ ] \in Y_1[\ ] \bullet Y_2[\ ]$ *and the second and the third figures above are also valid instances of* ρ *in* BiILL$_{sn}$*.*

Proof. This follows from the fact that $X[\ ]$, $X_1[\ ]$ and $X_2[\ ]$ have exactly the same nested structure, so whatever display rule applies to one also applies to the others. ◀

▶ **Theorem 17.** *If a sequent* X *is provable in* BiILL$_{dn}$ *then it is cut-free provable in* BiILL$_{sn}$*.*

Proof. We show that every rule of BiILL$_{dn}$ is cut-free derivable in BiILL$_{sn}$. We show here a derivation of the rule $\multimap_l^d$; the rest can be proved similarly. So suppose the conclusion of the rule is $X[\mathcal{S}, A \multimap B \Rightarrow \mathcal{T}]$, and the premises are $X_1[\mathcal{S}_1 \Rightarrow A, \mathcal{T}_1]$ and $X_2[\mathcal{S}_2, B \Rightarrow \mathcal{T}_2]$, where $X[\ ] \in X_1[\ ] \bullet X_2[\ ]$, $\mathcal{S} \in \mathcal{S}_1 \bullet \mathcal{S}_2$ and $\mathcal{T} \in \mathcal{T}_1 \bullet \mathcal{T}_2$. There are two cases to consider,

depending on whether $X[\ ]$ is positive or negative. We show here the former case, as the latter case is similar. Prop. 13 entails that $X[\mathcal{S}, A \multimap B \Rightarrow \mathcal{T}]$ is display equivalent to $\mathcal{U} \Rightarrow (\mathcal{S}, A \multimap B \Rightarrow \mathcal{T})$ for some $\mathcal{U}$. By Lem. 16, we have $\mathcal{U}_1$ and $\mathcal{U}_2$ such that $\mathcal{U} \in \mathcal{U}_1 \bullet \mathcal{U}_2$, and $(\mathcal{U}_1 \Rightarrow \mathcal{V})$ and $(\mathcal{U}_2 \Rightarrow \mathcal{V})$ are display equivalent to, respectively, $X_1[\mathcal{V}]$ and $X_2[\mathcal{V}]$, for any $\mathcal{V}$. The derivation of $\multimap_l^d$ in BiILL$_{sn}$ is thus constructed as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{X_1[\mathcal{S}_1 \Rightarrow A, \mathcal{T}_1]}{\mathcal{U}_1 \Rightarrow (\mathcal{S}_1 \Rightarrow A, \mathcal{T}_1)}\ \text{Lem. 16}}{\mathcal{U}_1, \mathcal{S}_1 \Rightarrow A, \mathcal{T}_1}\ rp_2 \qquad \dfrac{\dfrac{X_2[\mathcal{S}_2, B \Rightarrow \mathcal{T}_2]}{\mathcal{U}_2 \Rightarrow (\mathcal{S}_2, B \Rightarrow \mathcal{T}_2)}\ \text{Lem. 16}}{\mathcal{U}_2, \mathcal{S}_2, B \Rightarrow \mathcal{T}_2}\ rp_2}{\mathcal{U}_1, \mathcal{U}_2, \mathcal{S}_1, \mathcal{S}_2, A \multimap B \Rightarrow \mathcal{T}_1, \mathcal{T}_2}\ \multimap_l}{\mathcal{U}, \mathcal{S}, A \multimap B \Rightarrow \mathcal{T}}\ ml; ml; mr}{\mathcal{U} \Rightarrow (\mathcal{S}, A \multimap B \Rightarrow \mathcal{T})}\ rp_1}{X[\mathcal{S}, A \multimap B \Rightarrow \mathcal{T}]}\ \text{Prop. 13}$$

◀

The other direction of the equivalence is proved by a permutation argument: we first add the structural rules to BiILL$_{dn}$, then we show that these structural rules permute up over all (non-constant) logical rules of BiILL$_{dn}$. Then when the structural rules appear just below the id^d or the constant rules, they become redundant. There are quite a number of cases to consider, but they are not difficult once one observes the following property of BiILL$_{dn}$: in every rule, every context in the premise(s) has the same tree structure as the context in the conclusion of the rule. This observation takes care of permuting up structural rules that affect only the context. The non-trivial cases are those where the application of the structural rules changes the sequent where the logical rule is applied. We illustrate a case in the following lemma. The detailed proof can be found in [10].

▶ **Lemma 18.** *The rules drp_1, rp_1, drp_2, rp_2, gl, and gr permute up over all logical rules of* BiILL$_{dn}$.

Proof. (*Outline*) We illustrate here a non-trivial interaction between a structural rule and $\multimap_l$, where the conclusion sequent of $\multimap_l$ is changed by that structural rule. The other non-trivial cases follow the same pattern, i.e., propagation rules are used to move the principal formula to the required structural context.

$$\dfrac{\dfrac{\mathcal{S}_1, \mathcal{T}_1 \Rightarrow C, \mathcal{U}_1 \quad \mathcal{S}_2, \mathcal{T}_2, B \Rightarrow \mathcal{U}_2}{\mathcal{S}, C \multimap B, \mathcal{T} \Rightarrow \mathcal{U}}\ \multimap_l}{\mathcal{S}, C \multimap B \Rightarrow (\mathcal{T} \Rightarrow \mathcal{U})}\ rp_1 \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\mathcal{S}_1, \mathcal{T}_1 \Rightarrow C, \mathcal{U}_1}{\mathcal{S}_1 \Rightarrow (\mathcal{T}_1 \Rightarrow C, \mathcal{U}_1)}\ rp_1 \qquad \dfrac{\mathcal{S}_2, \mathcal{T}_2, B \Rightarrow \mathcal{U}_2}{\mathcal{S}_2 \Rightarrow (\mathcal{T}_2, B \Rightarrow \mathcal{U}_2)}\ rp_1}{\mathcal{S} \Rightarrow (C \multimap B, \mathcal{T} \Rightarrow \mathcal{U})}\ \multimap_l}{\mathcal{S}, C \multimap B \Rightarrow (\mathcal{T} \Rightarrow \mathcal{U})}\ pl_1$$

◀

▶ **Theorem 19.** *If a sequent X is cut-free* BiILL$_{sn}$*-derivable then it is also* BiILL$_{dn}$*-derivable.*

▶ **Corollary 20.** *A formula is cut-free* BiILL$_{dc}$*-derivable iff it is* BiILL$_{dn}$*-derivable.*

4 Separation, Conservativity, and Decidability

In this section we return our attention to the relationship between our calculi and the categorical semantics (Defs. 5 and 6). Def. 10 gave a translation of nested sequents to formulae; we can hence define validity for nested sequents.

▶ **Definition 21.** A nested sequent S is BiILL-*valid* if there is an arrow $I \to \tau^s(S)$ in the free BiILL-category.

A nested sequent is a (nested) *FILL-sequent* if it has no nesting of sequents on the left of $\Rightarrow$, and no occurrences of $\prec$ at all. The formula translation of Def. 10 hence maps FILL-sequents to FILL-formulae. Such a sequent S is FILL-valid if there is an arrow $I \to \tau^s(S)$ in the free FILL-category.

The calculus BiILL$_{dn}$ enjoys a 'separation' property between the FILL fragment using only $\bot$, I, $\otimes$, $\parr$, and $\multimap$ and the dual fragment using only $\bot$, I, $\otimes$, $\parr$, $\prec$. Let us define FILL$_{dn}$ as the proof system obtained from BiILL$_{dn}$ by restricting to FILL-sequents and removing the rules pr_1, pl_2, $\prec^d_l$ and $\prec^d_r$.

▶ **Theorem 22** (Separation)**.** *Nested FILL-sequents are* FILL$_{dn}$*-provable iff they are* BiILL$_{dn}$*-provable.*

Proof. One direction, from FILL$_{dn}$ to BiILL$_{dn}$, is easy. The other holds because every sequent in a BiILL$_{dn}$ derivation of a FILL-sequent is also a FILL-sequent. ◀

Thm. 22 tells us that every deep inference proof of a FILL-sequent is entirely constructed from FILL-sequents, each with a τ-translation to FILL-formulae. This contrasts with display calculus proofs, which must introduce the FILL-untranslatable $<$ even for simple theorems. By separation, and the equivalence of BiILL$_{dc}$ and BiILL$_{dn}$ (Cor. 20), the conservativity of BiILL over FILL reduces to checking the soundness of each rule of FILL$_{dn}$.

▶ **Lemma 23.** *An arrow $A \otimes B \to C$ exists in the free FILL-category iff an arrow $A \to B \multimap C$ exists. Further, arrows of the following types exist for all formulae A, B, C:*

(i) $A \multimap B \multimap C \;\to\; A \otimes B \multimap C$ *and* $A \otimes B \multimap C \;\to\; A \multimap B \multimap C$

(ii) $(A \multimap B) \parr C \;\to\; A \multimap B \parr C$.

In the proofs below we will abuse notation by omitting explicit reference to τ^a and τ^s, writing $\Gamma_1 \multimap \Delta_1$ for $\tau^a(\Gamma_1) \multimap \tau^s(\Delta_1)$ for example.

▶ **Lemma 24.** *Let $X[\ ]$ be a positive FILL-context. If there exists an arrow $f : \tau^s(S) \to \tau^s(T)$ in the free FILL-category then there also exists an arrow $\tau^s(X[S]) \to \tau^s(X[T])$. Hence if $X[S]$ is FILL-valid then so is $X[T]$.*

▶ **Lemma 25.** *Given a multiset $\mathcal{V}$ of* hollow *FILL-sequents, there exists an arrow $\bot \to \tau^s(\mathcal{V})$ in the free FILL-category.*

Proof. We will prove this for a single sequent first, by induction on its size. The base case is the sequent $\cdot \Rightarrow \cdot$, whose τ^s-translation is $I \multimap \bot$. The existence of an arrow $\bot \to I \multimap \bot$ is, by Lem. 23, equivalent to the existence of $\bot \otimes I \to \bot$; this is the unit arrow ρ. The induction case involves the sequent $\cdot \to T_1, \ldots, T_l$, with each T_i hollow; the required arrow exists by composing the arrows given by the induction hypothesis with $\bot \to \bot \parr \cdots \parr \bot$. The multiset case then follows easily by considering the cases where $\mathcal{V}$ is empty and non-empty. ◀

▶ **Lemma 26.** *Given a multiset $\mathcal{T} \in \mathcal{T}_1 \bullet \mathcal{T}_2$ of sequents and formulae, there is an arrow $\tau^s(\mathcal{T}_1) \parr \tau^s(\mathcal{T}_2) \to \tau^s(\mathcal{T})$ in the free FILL-category.*

Proof. We prove this for a single sequent first, by induction on its size. The base case requires an arrow $(\Gamma_1 \multimap \Delta_1) \parr (\Gamma_2 \multimap \Delta_2) \;\to\; \Gamma_1 \otimes \Gamma_2 \multimap \Delta_1 \parr \Delta_2$ (ref. Lem. 14), which exists by Lem. 23(ii) and (i). The induction case follows similarly. The multiset case then follows easily by considering the cases where $\mathcal{T}$ is empty and non-empty. ◀

▶ **Lemma 27.** *Take* $X[\ ] \in X_1[\ ] \bullet X_2[\ ]$ *and* $\mathcal{T} \in \mathcal{T}_1 \bullet \mathcal{T}_2$*. Then the following arrows exist in the free FILL-category for all* A, B, Γ_1 *and* Γ_2*:*

(i) $\tau^s(X_1[\Gamma_1 \Rightarrow A, \mathcal{T}_1]) \otimes \tau^s(X_2[\Gamma_2 \Rightarrow B, \mathcal{T}_2]) \ \to \ \tau^s(X[\Gamma_1, \Gamma_2 \Rightarrow A \otimes B, \mathcal{T}])$;

(ii) $\tau^s(X_1[\Gamma_1 \Rightarrow A, \mathcal{T}_1]) \otimes \tau^s(X_2[\Gamma_2, B \Rightarrow \mathcal{T}_2]) \ \to \ \tau^s(X[\Gamma_1, \Gamma_2, A \multimap B \Rightarrow \mathcal{T}])$;

(iii) $\tau^s(X_1[\Gamma_1, A \Rightarrow \mathcal{T}_1]) \otimes \tau^s(X_2[\Gamma_2, B \Rightarrow \mathcal{T}_2]) \ \to \ \tau^s(X[\Gamma_1, \Gamma_2, A \parr B \Rightarrow \mathcal{T}])$;

Proof. All three cases follow by induction on the size of $X[\ ]$. In all three cases the induction step is easy, and so we focus on the base cases. By Lem. 23 the base case for (i) requires an arrow:

$$(\Gamma_1 \multimap A \parr \mathcal{T}_1) \otimes (\Gamma_2 \multimap B \parr \mathcal{T}_2) \otimes \Gamma_1 \otimes \Gamma_2 \quad \to \quad (A \otimes B) \parr \mathcal{T}. \tag{4}$$

By the 'evaluation' arrows ε there is an arrow from the left hand side of (4) to $(A \parr \mathcal{T}_1) \otimes (B \parr \mathcal{T}_2)$. Composing this with weak distributivity takes us to $((A \parr \mathcal{T}_1) \otimes B) \parr \mathcal{T}_2$, and then to $(A \otimes B) \parr \mathcal{T}_1 \parr \mathcal{T}_2$. Lem. 26 completes the result. The base cases for (ii) and (iii) follow by similar arguments (App. B). ◀

▶ **Theorem 28.** *For every rule of* FILL$_{dn}$*, if the premises are FILL-valid then so is the conclusion.*

Proof. As FILL-sequents nest no sequents to the left of $\Rightarrow$, we can modify the rules of Fig. 5 to replace the multisets $\mathcal{S}, \mathcal{S}'$ of sequents and formulae with multisets Γ, Γ' of formulae only, and remove the hollow multisets of sequents $\mathcal{U}$ entirely (see App. B).

Therefore by Lem. 24 the soundness of pl_1 amounts to the existence in the free FILL-category of an arrow

$$\Gamma \multimap (A \otimes \Gamma' \multimap \mathcal{T}') \parr \mathcal{T} \quad \to \quad \Gamma \otimes A \multimap (\Gamma' \multimap \mathcal{T}') \parr \mathcal{T}.$$

This follows by two uses of Lem. 23(i). Similarly pr_2 requires an arrow

$$\Gamma \multimap \mathcal{T} \parr A \parr (\Gamma' \multimap \mathcal{T}') \quad \to \quad \Gamma \multimap \mathcal{T} \parr (\Gamma' \multimap \mathcal{T}' \parr A)$$

which exists by Lem. 23(ii).

id^d: by induction on the size of $X[\ ]$. The base case requires an arrow $I \to p \multimap p \parr \mathcal{V}$, which exists by Lems. 25 and 23. Induction involves a sequent $\cdot \Rightarrow X[p \Rightarrow p, \mathcal{V}], \mathcal{T}'$, with $\mathcal{T}'$ hollow, and hence requires an arrow $I \to I \multimap X[p \Rightarrow p, \mathcal{V}] \parr \mathcal{T}'$. By Lem. 23 and the arrow $I \otimes I \to I$ we need an arrow $I \to X[p \Rightarrow p, \mathcal{V}] \parr \mathcal{T}'$; by the induction hypothesis we have $I \to X[p \Rightarrow p, \mathcal{V}]$; this extends to $I \to X[p \Rightarrow p, \mathcal{V}] \parr \bot$; Lem. 25 completes the proof.

$\bot_l^d$: by another induction on $X[\ ]$. The base case $I \to \bot \multimap \mathcal{V}$ follows by Lems. 23 and 25; induction follows as with id^d.

$\bot_r^d$: By Lem. 24 and the unit property of $\bot$.

I_l^d: By Lem. 24 we need an arrow $(\Gamma \multimap \mathcal{T}) \otimes \Gamma \otimes I \to \mathcal{T}$; this exists by the unit property of I and the 'evaluation' arrow ε.

I_r^d: another induction on $X[\ ]$. The base case arrow $I \to I \multimap I \parr \mathcal{V}$ exists by Lems. 23 and 25; induction follows as with id^d.

$\otimes_l^d$, $\multimap_r^d$, and $\parr_r^d$ are trivial by the formula translation.

$\otimes_r^d$: compose the arrow $I \to I \otimes I$ with the arrows defined by the validity of the premises, then use Lem. 27(i). $\multimap_l^d$ and $\parr_r^d$ follow similarly via Lem. 27(ii) and (iii). ◀

▶ **Theorem 29.** *A FILL-formula is* FILL*-valid iff it is* FILL$_{dn}$*-provable, and BiILL is conservative over FILL.*

Proof. By Cors. 9 and 20 and Thms. 22 and 28. ◀

Note that it is also possible to prove soundness of FILL$_{dn}$ w.r.t. FILL syntactically, i.e., via a translation into Schellinx's sequent calculus for FILL [26]. See [10] for details.

Thm. 29 gives us a sound and complete calculus for FILL that enjoys a genuine subformula property. This in turn allows one to prove NP-completeness of the tautology problem for FILL (i.e., deciding whether a formula is provable or not), as we show next. The complexity does not in fact change even when one adds exclusion to FILL.

▶ **Theorem 30.** *The tautology problems for BiILL and FILL are NP-complete.*

Proof. (*Outline.*) Membership in NP is proved by showing that every cut-free proof of a formula A in BiILL$_{dn}$ can be checked in PTIME in the size of A. This is not difficult to prove given that each connective in A is introduced exactly once in the proof. NP-hardness is proved by encoding Constants-Only MLL (COMLL), which is NP-hard [23], in FILL$_{dn}$. ◀

5 Conclusion

We have given three cut-free sequent calculi for FILL without complex annotations, showing that, far from being a curiosity that demands new approaches to proof theory, FILL is in a broad family of linear and substructural logics captured by display calculi.

Various substructural logics can be defined by using a (possibly non-associative or non-commutative) multiplicative conjunction and its left and right residual(s) (implications). Many of these logics have cut-free sequent calculi with comma-separated structures in the antecedent and a single formula in the succedent. Each of these logics has a dual logic with disjunction and its residual(s) (exclusions); their proof theory requires sequents built out of comma-separated structures in the succedent and a single formula in the antecedent. These logics can then be combined using numerous "distribution principles" [19, 25], of which weak distributivity is but one example. However, obtaining an adequate sequent calculus for these combinations is often non-trivial. On the other hand, display calculi for these logics, their duals, and their combinations, are extremely easy to obtain using the known methodology for building display calculi [3, 16]. We followed this methodology to obtain BiILL in this paper, but needed a conservativity result to ensure the resulting calculus BiILL$_{dc}$ was sound for FILL. We finally note some specific variations on FILL deserving particular attention.

Grishin (a). Adding the converse of Grishin (b) to FILL recovers MLL. For example $(B \multimap \bot) \parr C \vdash B \multimap C$ is provable using Grn(b), but its converse requires Grn(a). Thus there is another 'full' non-classical extension of MILL with Grishin (a) as its interaction principle *instead* of (b). We do not know what significance this logic may have.

Mix rules. It is easy to give structural rules for the *mix* sequents $A, B \vdash A, B$ and $\Phi \vdash \Phi$ which have been studied in FILL [12, 1] and so it is natural to ask if the results of this paper can be extended to them. Intriguingly, our new structural connectives suggest a new mix rule with sequent form $A < B \vdash B > A$ which, given Grishin (b), is stronger than the mix rule for comma (given Grishin (a), it is weaker).

Exponentials. Adding exponentials [5] to our display calculus for FILL may be possible [2].

Additives. While it has been suggested that FILL could be extended with additives, the only attempt in the literature is erroneous [15]. It is not clear how easy this extension would be [8, Sec. 1]; it is certainly not straightforward with the display calculus. The problem is most easily seen through the categorical semantics: additive conjunction $\wedge$ and its unit $\top$ are limits, and $p \parr -$ is a right adjoint in BiILL but is not necessarily so in FILL. But right adjoints preserve limits. Then BiILL plus additives is not conservative over FILL plus additives, because the sequents $(p \parr q) \wedge (p \parr r) \vdash p, (q \wedge r)$ and $\top \vdash p, \top$ are valid in the former but not the latter, despite the absence of $\prec$ or $<$. We are currently investigating solutions.

References

1 Gianluigi Bellin. Subnets of proof-nets in multiplicative linear logic with MIX. *Mathematical Structures in Computer Science*, 7(6):663–669, 1997.

2 N D Belnap. Linear logic displayed. *Notre Dame Journal of Formal Logic*, 31:15–25, 1990.

3 Nuel D Belnap. Display logic. *Journal of Philosophical Logic*, 11:375–417, 1982.

4 Gavin M. Bierman. A note on full intuitionistic linear logic. *APAL*, 79(3):281–287, 1996.

5 Torben Bräuner and Valeria de Paiva. Cut-elimination for full intuitionistic linear logic. Technical Report RS-96-10, Basic Research in Computer Science, 1996.

6 Torben Bräuner and Valeria de Paiva. A formulation of linear logic based on dependency-relations. In *CSL '97*, volume 1414 of *LNCS*, pages 129–148, 1997.

7 Kai Brünnler. Deep sequent systems for modal logic. *Archive for Mathematical Logic*, 48(6):551–577, 2009.

8 Bor-Yuh Evan Chang, Kaustuv Chaudhuri, and Frank Pfenning. A judgmental analysis of linear logic. Technical Report CMU-CS-03-131R, Carnegie Mellon University, 2003.

9 Kaustuv Chaudhuri. The inverse method for intuitionistic linear logic. Technical Report CMU-CS-03-140, Carnegie Mellon University, 2004.

10 Ranald Clouston, Jeremy Dawson, Rajeev Goré, and Alwen Tiu. Annotation-free sequent calculi for full intuitionistic linear logic – extended version. *arXiv:1307.0289*, 2013.

11 J.R.B. Cockett and R.A.G. Seely. Weakly distributive categories. In *Applications of Categories in Computer Science*, volume 177 of *London Math. Soc. Lect. Note Series*, pages 45–65, 1992.

12 J.R.B. Cockett and R.A.G. Seely. Proof theory for full intuitionistic linear logic, bilinear logic, and MIX categories. *Theory and Applications of Categories*, 3(5):85–131, 1997.

13 Valeria de Paiva and Eike Ritter. A Parigot-style linear λ-calculus for full intuitionistic linear logic. *Theory and Applications of Categories*, 17(3):30–48, 2006.

14 Kosta Došen and Zoran Petrić. *Proof-Theoretical Coherence*, volume 1 of *Studies in Logic*. College Publications, 2004.

15 Didier Galmiche and Eric Boudinet. Proofs, concurrent objects, and computations in a FILL framework. In *OBPDC*, volume 1107 of *LNCS*, pages 148–167. Springer, 1995.

16 Rajeev Goré. Substructural logics on display. *Log. J. IGPL*, 6(3):451–504, 1998.

17 Rajeev Goré, Linda Postniece, and Alwen Tiu. Cut-elimination and proof search for bi-intuitionistic tense logic. In *Advances in Modal Logic*, pages 156–177. College Publications, 2010.

18 Rajeev Goré, Linda Postniece, and Alwen Tiu. On the correspondence between display postulates and deep inference in nested sequent calculi for tense logics. *LMCS*, 7(2), 2011.

19 V N Grishin. On a generalization of the Ajdukiewicz-Lambek system. In *Studies in Non-classical Logics and Formal Systems*, pages 315–343. Nauka, 1983.

20 Martin Hyland and Valeria de Paiva. Full intuitionistic linear logic (extended abstract). *Ann. Pure Appl. Logic*, 64(3):273–291, 1993.

21 Ryo Kashima. Cut-free sequent calculi for some tense logics. *Studia Log.*, 53:119–135, 1994.

22 Marcus Kracht. Power and weakness of the modal display calculus. In Heinrich Wansing, editor, *Proof Theory of Modal Logics*, pages 92–121. Kluwer, 1996.

23 Patrick Lincoln and Timothy C. Winkler. Constant-only multiplicative linear logic is NP-complete. *TCS*, 135(1):155–169, 1994.

24 Simone Martini and Andrea Masini. Experiments in linear natural deduction. *TCS*, 176(1-2):159–173, 1997.

25 Michael Moortgat. Symmetric categorial grammar. *J. Philosophical Logic*, 38(6):681–710, 2009.

26 Harold Schellinx. Some syntactical observations on linear logic. *JLC*, 1(4):537–559, 1991.

27 Heinrich Wansing. Sequent calculi for normal modal proposisional logics. *JLC*, 4(2):125–142, 1994.

Propagation rules:

$$\frac{X[\Gamma \Rightarrow (A, \Gamma' \Rightarrow \mathcal{T}'), \mathcal{T}]}{X[\Gamma, A \Rightarrow (\Gamma' \Rightarrow \mathcal{T}'), \mathcal{T}]}\ pl_1 \qquad \frac{X[\Gamma \Rightarrow \mathcal{T}, A, (\Gamma' \Rightarrow \mathcal{T}')]}{X[\Gamma \Rightarrow \mathcal{T}, (\Gamma' \Rightarrow \mathcal{T}', A)]}\ pr_2$$

Identity and logical rules: In branching rules, $X[\ ] \in X_1[\ ] \bullet X_2[\ ]$ and $\mathcal{T} \in \mathcal{T}_1 \bullet \mathcal{T}_2$.

$$\frac{X[\ ] \text{ and } \mathcal{V} \text{ are hollow.}}{X[p \Rightarrow p, \mathcal{V}]}\ id^d \qquad \frac{X[\ ] \text{ and } \mathcal{V} \text{ are hollow.}}{X[\bot \Rightarrow \mathcal{V}]}\ \bot_l^d \qquad \frac{X[\Gamma \Rightarrow \mathcal{T}]}{X[\Gamma \Rightarrow \mathcal{T}, \bot]}\ \bot_r^d$$

$$\frac{X[\Gamma \Rightarrow \mathcal{T}]}{X[\Gamma, \mathrm{I} \Rightarrow \mathcal{T}]}\ \mathrm{I}_l^d \qquad \frac{X[\ ] \text{ and } \mathcal{V} \text{ are hollow.}}{X[\cdot \Rightarrow \mathrm{I}, \mathcal{V}]}\ \mathrm{I}_r^d$$

$$\frac{X[\Gamma, A, B \Rightarrow \mathcal{T}]}{X[\Gamma, A \otimes B \Rightarrow \mathcal{T}]}\ \otimes_l^d \qquad \frac{X_1[\Gamma_1 \Rightarrow A, \mathcal{T}_1] \quad X_2[\Gamma_2 \Rightarrow B, \mathcal{T}_2]}{X[\Gamma_1, \Gamma_2 \Rightarrow A \otimes B, \mathcal{T}]}\ \otimes_r^d$$

$$\frac{X_1[\Gamma_1 \Rightarrow A, \mathcal{T}_1] \quad X_2[\Gamma_2, B \Rightarrow \mathcal{T}_2]}{X[\Gamma_1, \Gamma_2, A \multimap B \Rightarrow \mathcal{T}]}\ \multimap_l^d \qquad \frac{X[\Gamma \Rightarrow \mathcal{T}, (A \Rightarrow B)]}{X[\Gamma \Rightarrow \mathcal{T}, A \multimap B]}\ \multimap_r^d$$

$$\frac{X_1[\Gamma_1, A \Rightarrow \mathcal{T}_1] \quad X_2[\Gamma_2, B \Rightarrow \mathcal{T}_2]}{X[\Gamma_1, \Gamma_2, A \parr B \Rightarrow \mathcal{T}]}\ \parr_l^d \qquad \frac{X[\Gamma \Rightarrow A, B, \mathcal{T}]}{X[\Gamma \Rightarrow A \parr B, \mathcal{T}]}\ \parr_r^d$$

Figure 7 The deep inference system FILL$_{dn}$.

A Display Calculus

We outline the conditions that are easily checked to confirm that display calculi enjoy cut-admissibility (Thm. 4):

▶ **Definition 31** (Belnap's Conditions C1-C8)**.** The set of display conditions appears in various guises in the literature. Here we follow the presentation given in Kracht [22].

(C1) Each formula variable occurring in some premise of a rule ρ is a subformula of some formula in the conclusion of ρ.

(C2) *Congruent parameters* is a relation between parameters of the identical structure variable occurring in the premise and conclusion sequents.

(C3) Each parameter is congruent to at most one structure variable in the conclusion. Equivalently, no two structure variables in the conclusion are congruent to each other.

(C4) Congruent parameters are either all antecedent or all succedent parts of their respective sequent.

(C5) A formula in the conclusion of a rule ρ is either the entire antecedent or the entire succedent. Such a formula is called a **principal formula** of ρ.

(C6/7) Each rule is closed under simultaneous substitution of arbitrary structures for congruent parameters.

(C8) If there are rules ρ and σ with respective conclusions $X \vdash A$ and $A \vdash Y$ with formula A principal in both inferences (in the sense of C5) and if *cut* is applied to yield $X \vdash Y$, then either $X \vdash Y$ is identical to either $X \vdash A$ or $A \vdash Y$; or it is possible to pass from the premises of ρ and σ to $X \vdash Y$ by means of inferences falling under *cut* where the cut-formula always is a proper subformula of A.

B Conservativity of BiILL over FILL

Fig. 7 explicitly gives the proof rules for FILL$_{dn}$, the nested sequent calculus with deep inference for FILL. These are easily derived from BiILL$_{dn}$ (Fig. 5).

Proof of Lemma 23. This is basic category theory; we give one example to illustrate the techniques used. Given an arrow $f : A \otimes B \to C$, we get a new arrow $A \to B \multimap C$ by composing $B \multimap f$ with the 'co-evaluation' arrow $\eta : A \to B \multimap (A \otimes B)$. ◀

Proof of Lemma 24. By induction on the size of $X[\ ]$. The base case, where $X[\ ]$ is a hole, is trivial. The induction case involves a context $\Gamma \Rightarrow X[\ ], \mathcal{T}$ and hence requires an arrow

$$\Gamma \multimap X[S] \parr \mathcal{T} \quad \to \quad \Gamma \multimap X[T] \parr \mathcal{T}.$$

This exists by the induction hypothesis and the inductive definitions of Lem. 6. The validity of $X[S]$ then transfers to $X[T]$ via composition with the arrow $I \to X[S]$. ◀

Proof of Lemma 27(ii) and (iii). (ii): The base case requires an arrow

$$(\Gamma_1 \multimap A \parr \mathcal{T}_1) \otimes (\Gamma_2 \otimes B \multimap \mathcal{T}_2) \otimes \Gamma_1 \otimes \Gamma_2 \otimes (A \multimap B) \quad \to \quad \mathcal{T}. \tag{5}$$

Applying an evaluation to the left of (5) gives $(A \parr \mathcal{T}_1) \otimes (\Gamma_2 \otimes B \multimap \mathcal{T}_2) \otimes \Gamma_2 \otimes (A \multimap B)$; weak distributivity gives $\mathcal{T}_1 \parr (A \otimes (\Gamma_2 \otimes B \multimap \mathcal{T}_2) \otimes \Gamma_2 \otimes (A \multimap B))$; two more evaluations give $\mathcal{T}_1 \parr \mathcal{T}_2$ and Lem. 26 completes the result.

(iii): The base case requires an arrow

$$(\Gamma_1 \otimes A \multimap \mathcal{T}_1) \otimes (\Gamma_2 \otimes B \multimap \mathcal{T}_2) \otimes \Gamma_1 \otimes \Gamma_2 \otimes (A \parr B) \quad \to \quad \mathcal{T}. \tag{6}$$

Two applications of weak distributivity map the left of (6) to

$$((\Gamma_1 \otimes A \multimap \mathcal{T}_1) \otimes \Gamma_1 \otimes A) \parr ((\Gamma_2 \otimes B \multimap \mathcal{T}_2) \otimes \Gamma_2 \otimes B).$$

Two evaluations and Lem. 26 complete the result. ◀

C Annotated Sequent Calculi Proofs

On the next page we present cut-free proofs of the Bierman example (2) in the style of the three cut-free annotated sequent calculi in the literature: that due to Bierman [4]; that due to Bellin reported in [4], and that due to Bräuner and de Paiva [6]. Note that all three proofs contain the same sequence of proof rules; strip out the annotations and they are MLL proofs of the sequent. The difference between the calculi lies in the nature of their annotations, all of which come into play to verify that the final rule application, of $(\multimap R)$, is legal. The reader is invited to compare these proofs to those presented in the paper using display calculus (Fig. 2) and deep inference (Fig. 6).

Bierman-style proof; $(\multimap R)$ is legal because v and $(w \parr x \multimap y) \parr z$ share no free variables.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{v : a \vdash v : a \qquad w : b \vdash w : b}{v \parr w : a \parr b \vdash v : a, w : b} \qquad x : c \vdash x : c}{(v \parr w) \parr x : (a \parr b) \parr c \vdash v : a, w : b, x : c}}{(v \parr w) \parr x : (a \parr b) \parr c \vdash v : a, w \parr x : b \parr c} \qquad y : d \vdash y : d}{(v \parr w) \parr x : (a \parr b) \parr c, w \parr x \multimap y : b \parr c \multimap d \vdash v : a, y : d} \qquad z : e \vdash z : e}{(v \parr w) \parr x : (a \parr b) \parr c, (w \parr x \multimap y) \parr z : (b \parr c \multimap d) \parr e \vdash v : a, y : d, z : e}}{(v \parr w) \parr x : (a \parr b) \parr c, (w \parr x \multimap y) \parr z : (b \parr c \multimap d) \parr e \vdash v : a, y \parr z : d \parr e}}{(v \parr w) \parr x : (a \parr b) \parr c \vdash v : a, \lambda (w \parr x \multimap y) \parr z^{(b \parr c \multimap d) \parr e}.(y \parr z) : (b \parr c \multimap d) \parr e \multimap d \parr e}$$

Bellin-style proof; $(\multimap R)$ is legal because r is not free in let t be $u \parr$ - in let u be $v \parr$ - in v. We apologise for the extremely small font size necessary to fit this proof on the page.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{v : a \vdash v : a \qquad w : b \vdash w : b}{u : a \parr b \vdash \text{let } u \text{ be } v \parr \text{ - in } v : a, \text{let } u \text{ be - } \parr w \text{ in } w : b} \qquad x : c \vdash x : c}{t : (a \parr b) \parr c \vdash \text{let } t \text{ be } u \parr \text{ - in let } u \text{ be } v \parr \text{ - in } v : a, \text{let } t \text{ be } u \parr \text{ - in let } u \text{ be - } \parr w \text{ in } w : b, \text{let } t \text{ be - } \parr x \text{ in } x : c}}{t : (a \parr b) \parr c \vdash \text{let } t \text{ be } u \parr \text{ - in let } u \text{ be } v \parr \text{ - in } v : a, (\text{let } t \text{ be } u \parr \text{ - in let } u \text{ be - } \parr w \text{ in } w) \parr (\text{let } t \text{ be - } \parr x \text{ in } x) : b \parr c} \qquad y : d \vdash y : d}{t : (a \parr b) \parr c, s : b \parr c \multimap d \vdash \text{let } t \text{ be } u \parr \text{ - in let } u \text{ be } v \parr \text{ - in } v : a, (s(\text{let } t \text{ be } u \parr \text{ - in let } u \text{ be - } \parr w \text{ in } w) \parr (\text{let } t \text{ be - } \parr x \text{ in } x)) : d} \qquad z : e \vdash z : e}{t : (a \parr b) \parr c, r : (b \parr c \multimap d) \parr e \vdash \text{let } t \text{ be } u \parr \text{ - in let } u \text{ be } v \parr \text{ - in } v : a, \text{let } r \text{ be } s \parr \text{ - in } (s(\text{let } t \text{ be } u \parr \text{ - in let } u \text{ be - } \parr w \text{ in } w) \parr (\text{let } t \text{ be - } \parr x \text{ in } x)) : d, \text{let } s \text{ be - } \parr z \text{ in } z : e}}{t : (a \parr b) \parr c, r : (b \parr c \multimap d) \parr e \vdash \text{let } t \text{ be } u \parr \text{ - in let } u \text{ be } v \parr \text{ - in } v : a, (\text{let } r \text{ be } s \parr \text{ - in } (s(\text{let } t \text{ be } u \parr \text{ - in let } u \text{ be - } \parr w \text{ in } w) \parr (\text{let } t \text{ be - } \parr x \text{ in } x))) \parr (\text{let } s \text{ be - } \parr z \text{ in } z) : d \parr e}}{t : (a \parr b) \parr c \vdash \text{let } t \text{ be } u \parr \text{ - in let } u \text{ be } v \parr \text{ - in } v : a, \lambda r^{(b \parr c \multimap d) \parr e}.(\text{let } r \text{ be } s \parr \text{ - in } (s(\text{let } t \text{ be } u \parr \text{ - in let } u \text{ be - } \parr w \text{ in } w) \parr (\text{let } t \text{ be - } \parr x \text{ in } x))) \parr (\text{let } s \text{ be - } \parr z \text{ in } z) : (b \parr c \multimap d) \parr e \multimap d \parr e}$$

Bräuner and de Paiva-style proof; $(\multimap R)$ is legal because $(b \parr c \multimap d) \parr e$ is not related to a.

$$\begin{array}{rl}
(a, a) & \dfrac{}{a \vdash a} \quad \dfrac{}{b \vdash b} \; (b, b) \\
(a \parr b, a), (a \parr b, b) & \dfrac{a \parr b \vdash a, b}{} \quad \dfrac{}{c \vdash c} \; (c, c) \\
((a \parr b) \parr c, a), ((a \parr b) \parr c, b), ((a \parr b) \parr c, c) & \dfrac{(a \parr b) \parr c \vdash a, b, c}{} \\
((a \parr b) \parr c, a), ((a \parr b) \parr c, b \parr c) & \dfrac{(a \parr b) \parr c \vdash a, b \parr c}{} \quad \dfrac{}{d \vdash d} \; (d, d) \\
((a \parr b) \parr c, a), ((a \parr b) \parr c, d), (b \parr c \multimap d, d) & \dfrac{(a \parr b) \parr c, b \parr c \multimap d \vdash a, d}{} \quad \dfrac{}{e \vdash e} \; (e, e) \\
((a \parr b) \parr c, a), ((a \parr b) \parr c, d), ((b \parr c \multimap d) \parr e, d), ((b \parr c \multimap d) \parr e, e) & \dfrac{(a \parr b) \parr c, (b \parr c \multimap d) \parr e \vdash a, d, e}{} \\
((a \parr b) \parr c, a), ((a \parr b) \parr c, d \parr e), ((b \parr c \multimap d) \parr e, d \parr e) & \dfrac{(a \parr b) \parr c, (b \parr c \multimap d) \parr e \vdash a, d \parr e}{} \\
((a \parr b) \parr c, a), ((a \parr b) \parr c, (b \parr c \multimap d) \parr e \multimap d \parr e) & \dfrac{}{(a \parr b) \parr c \vdash a, (b \parr c \multimap d) \parr e \multimap d \parr e}
\end{array}$$

Deciding the weak definability of Büchi definable tree languages*

Thomas Colcombet[1], Denis Kuperberg[2], Christof Löding[3], and Michael Vanden Boom[4]

1,2 CNRS and LIAFA, Université Paris Diderot
Paris, France
{thomas.colcombet,denis.kuperberg}@liafa.univ-paris-diderot.fr

3 Informatik 7, RWTH Aachen University
Aachen, Germany
loeding@cs.rwth-aachen.de

4 Department of Computer Science, University of Oxford
Oxford, England
michael.vandenboom@cs.ox.ac.uk

Abstract

Weakly definable languages of infinite trees are an expressive subclass of regular tree languages definable in terms of weak monadic second-order logic, or equivalently weak alternating automata. Our main result is that given a Büchi automaton, it is decidable whether the language is weakly definable. We also show that given a parity automaton, it is decidable whether the language is recognizable by a nondeterministic co-Büchi automaton.

The decidability proofs build on recent results about cost automata over infinite trees. These automata use counters to define functions from infinite trees to the natural numbers extended with infinity. We reduce to testing whether the functions defined by certain "quasi-weak" cost automata are bounded by a finite value.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Tree automata, weak definability, decidability, cost automata, boundedness

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.215

1 Introduction

Infinite trees are often used as a model for representing the possible behaviours of a system. Various classes of automata and logic have been introduced in order to reason effectively about the properties of such systems. In particular, regular tree languages capture a rich class of properties which can be defined using logic (monadic second-order logic, or a fixpoint logic called the modal mu-calculus) and automata (including nondeterministic parity automata).

The *weakly definable languages* are a proper subclass of regular tree languages. These languages can be expressed in weak monadic second-order logic (in which second-order quantification is restricted only to finite sets), but they can be described in many other ways. For instance, Rabin [20] proved that they are precisely the languages for which the

* The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°259454 and the project Anr 2010 Blan 0202 02 Frec.

language and its complement are recognizable by nondeterministic Büchi automata. Arnold and Niwiński [2] showed that they can also be defined using the alternation-free mu-calculus. Muller et al. [15] showed that these languages can be equivalently defined by a form of alternating automata, called weak automata.

One reason these languages are so well studied is that they subsume temporal logics like CTL but still admit efficient (linear time) model-checking. Hence, they represent an expressive class of languages with good computational properties.

Given some regular language of infinite trees in the form of, say, an arbitrary parity automaton, it would be helpful to be able to decide whether it is weakly definable. We call this the *weak definability problem*.

Related work

The weak definability problem is related to the *nondeterministic parity index problem* or *Mostowski index problem*, which asks, given a regular language L of infinite trees and a finite set of priorities P, is there a nondeterministic parity automaton using only priorities P that recognizes L.[1] A nondeterministic parity automaton assigns to each state a priority. A run is a labelling of the tree with states, and a run is accepting if the highest priority occurring infinitely often on each branch is even. We say that a language L is P *definable* if there is a nondeterministic parity automaton using priorities P accepting exactly the trees in L. In the special case that $P = \{1, 2\}$ (respectively, $P = \{0, 1\}$), we say the language is *Büchi definable* (respectively, *co-Büchi definable*), since a nondeterministic parity automaton using priorities P can be viewed as a nondeterministic Büchi automaton (respectively, nondeterministic co-Büchi automaton).

Decidability of the nondeterministic parity index problem is known when P is restricted to a single priority (see [13]). More interesting, Niwiński and Walukiewicz [18] established decidability for any P as long as the input language is a deterministic tree language (a proper decidable subclass of regular tree languages).

This parity index problem is connected to the weak definability problem because of Rabin's characterization of weakly definable languages: a regular tree language is weakly definable if and only if the language and its complement are Büchi definable [20]. Since an automaton recognizing the complement of a regular language can be found effectively [19, 7], the weak definability problem reduces to determining whether the language and its complement are Büchi definable. Hence, one route to proving the decidability of weak definability would be to prove the decidability of Büchi definability. Although Karpiński reported such a solution already in the 1970s, there is no known written proof [9]. Hence, the weak definability problem is decidable when the input is a deterministic tree language, but is open in general.

Colcombet and Löding [5] proposed an alternative approach to solving this problem using *cost automata*. Cost automata are traditional finite state automata enriched with a finite set of counters. Instead of only accepting or rejecting some input structure, cost automata assign a value based on the evolution of counter values during the run. A cost automaton $\mathcal{A}$ can be viewed as defining a function $[\![\mathcal{A}]\!]$ from the set of structures under consideration to the natural numbers (extended with a special infinity symbol ∞). The idea is that cost automata can count some behaviour within the input.

[1] This is called the nondeterministic parity index problem since the desired automaton is nondeterministic. There are variants of this problem for different models of automata. See [13] for an overview.

The exact values of the function defined by a cost automaton do not matter. Instead, the functions are only considered up to an equivalence relation $\approx$ called the *boundedness relation* which ignores exact values but preserves all boundedness properties of the function.

Variants of these automata were famously used by Hashiguchi [8], and later Kirsten [10], to show the decidability of the star height problem over finite words. The idea is that a question about language theory like the star height problem is reduced to deciding whether the function defined by a cost automaton is limited, *i.e.* whether there is a natural number bounding the output over all accepted inputs. This is a special case of deciding whether two functions defined by cost automata are equivalent up to $\approx$. Because the boundedness relation is decidable for functions defined using cost automata over finite words ([3, 4]), this work on cost automata provided an alternative proof for a large part of the proof of decidability of the star height problem.

In this paper, we are dealing with cost automata working over infinite trees rather than finite words. Colcombet and Löding [5] provided a reduction of the decidability of the nondeterministic parity index problem to the decidability of $\approx$ for *cost-parity automata*: parity automata over infinite trees enriched with counters. Unfortunately, the decidability of $\approx$ for this richer class of cost-parity automata over infinite trees is open, so the general parity index problem remains open.

Independently, Afshari and Quickert also gave a reduction of the weak definability problem for Büchi definable languages to a question of boundedness, but decidability is also open using their approach [1].

Contributions

Our contribution is to show that recent results developed for the cost automata over infinite trees can be applied to solving a special case of the weak definability problem when the input language (or its complement) is Büchi definable. This subsumes the previously known decidability result for deterministic input. Along the way, we also point out an application to deciding whether a language is co-Büchi definable.

In other words, we show that the following problems are decidable:

- Given an alternating parity automaton over infinite trees, is there a nondeterministic co-Büchi automaton recognizing the same language? (Theorem 9)
- Given an alternating Büchi automaton, alternating co-Büchi automaton, or deterministic parity automaton over infinite trees, is there a weak automaton recognizing the same language? (Theorem 10)

The constructions make use of quasi-weak cost automata, which were introduced in [11]. This form of automaton is a new variant of weak automaton with counters, and we believe this application demonstrates the utility of this new automaton model.

Notation and Conventions

We write $\mathbb{N}$ for the set of non-negative integers and $\mathbb{N}_\infty$ for the set $\mathbb{N} \cup \{\infty\}$, ordered by $0 < 1 < \cdots < \infty$.

We work with an arbitrary finite alphabet $\mathbb{A}$. The set of finite (respectively, infinite) words over $\mathbb{A}$ is $\mathbb{A}^*$ (respectively, $\mathbb{A}^\omega$) and the empty word is ϵ. For notational simplicity we work only with infinite binary trees. Let $\mathcal{T} = \{0,1\}^*$ be the unlabelled infinite binary tree. The set $\mathcal{T}_\mathbb{A}$ of *complete* $\mathbb{A}$*-labelled binary trees* is composed of mappings $t : \mathcal{T} \to \mathbb{A}$. A *branch* π is a word $\{0,1\}^\omega$. A *frontier* E is a set of positions in $\mathcal{T}$ such that for all branches π in $\mathcal{T}$, $E \cap \pi$ is a singleton. For frontiers E and E', we write $E < E'$ if for every branch π, if

$\{x\} = E \cap \pi$ and $\{x'\} = E' \cap \pi$, then x is a strict ancestor of x'. If x, y are nodes in $\mathcal{T}$, with x a strict ancestor of y, we write $[x, y)$ for the set of nodes that are strict ancestors of y and descendants of x, including x itself.

2 Cost automata and cost functions

2.1 Cost automata

The automata that we consider are traditional automata over infinite trees equipped with a finite set of counters Γ which can be incremented $\mathtt{ic}$, reset $\mathtt{r}$, or left unchanged ε (but cannot be used to affect the flow of the automaton). Let $\mathbb{B} := \{\mathtt{ic}, \mathtt{r}, \varepsilon\}$ denote this alphabet of counter actions. Each counter starts with value zero, and the value of a sequence of actions is the supremum of the values achieved during this sequence. For instance the finite sequence $(\mathtt{ic})(\mathtt{ic})\mathtt{r}\varepsilon(\mathtt{ic})\varepsilon$ has value 2, the infinite sequence $((\mathtt{ic})\mathtt{r})^\omega$ has value 1, and the infinite sequence $(\mathtt{ic})\mathtt{r}(\mathtt{ic})^2\mathtt{r}(\mathtt{ic})^3\mathtt{r}\ldots$ has value ∞. If there are several counters, only the sequence with the maximal value is taken into account.

Formally, an *(alternating) B-Büchi automaton* $\langle Q, \mathbb{A}, q_0, \Gamma, F, \delta\rangle$ on infinite trees has a finite set of states Q, alphabet $\mathbb{A}$, initial state $q_0 \in Q$, a finite set Γ of counters, accepting states $F \subseteq Q$, and transition function $\delta : Q \times \mathbb{A} \to \mathcal{B}^+(\{0,1\} \times \mathbb{B}^\Gamma \times Q)$, where $\mathcal{B}^+(\{0,1\} \times \mathbb{B}^\Gamma \times Q)$ is the set of positive boolean combinations, written as a disjunction of conjunctions, of elements $(d, \nu, q) \in \{0,1\} \times \mathbb{B}^\Gamma \times Q$.

We view running a B-automaton $\mathcal{A}$ on an input tree t as a game $\mathcal{A} \times t$ between two players: Eve is in charge of the disjunctive choices and tries to minimize counter values while satisfying the acceptance condition, and Adam is in charge of the conjunctive choices and tries to maximize counter values or show the acceptance condition is not satisfied. Because the transition function is given as a disjunction of conjunctions, we can consider that at each position, Eve first chooses a disjunct, and then Adam chooses a single tuple (d, ν, q) in this disjunct.

A *play* of $\mathcal{A}$ on input t is a sequence $q_0, (d_1, \nu_1, q_1), (d_2, \nu_2, q_2), \ldots$ compatible with t and δ, *i.e.* q_0 is initial, and for all $i \in \mathbb{N}$, $(d_{i+1}, \nu_{i+1}, q_{i+1})$ appears in $\delta(q_i, t(d_1 \ldots d_i))$.

A *strategy* for Eve (respectively, Adam) in the game $\mathcal{A} \times t$ is a function that fixes the next choice of Eve (respectively, Adam), based on the history of the play (respectively, the history of the play and Eve's choice of disjunct). Notice that choosing a strategy for Eve and a strategy for Adam fixes a play in $\mathcal{A} \times t$. We say a play π is *compatible* with a strategy σ for Eve if there is some strategy σ' for Adam such that σ and σ' yield the play π.

A play π is *accepting* for the *Büchi condition* specified by F if there is $q \in F$ appearing infinitely often in π. Given a play π from a B-automaton $\mathcal{A}$, the value of π is ∞ if π is not accepting and is the supremum of the counter values achieved during the play otherwise.

We assign a value to a strategy σ for Eve by taking

$$\text{value}(\sigma) := \sup\{\text{value}(\pi) : \pi \text{ is compatible with } \sigma\}.$$

Likewise, the value assigned to the game $\mathcal{A} \times t$ is

$$\text{value}(\mathcal{A} \times t) := \inf\{\text{value}(\sigma) : \sigma \text{ is a strategy for Eve in the game } \mathcal{A} \times t\}.$$

We view $\mathcal{A}$ as defining a function $[\![\mathcal{A}]\!] : \mathcal{T}_\mathbb{A} \to \mathbb{N}_\infty$ such that $[\![\mathcal{A}]\!](t) := \text{value}(\mathcal{A} \times t)$. Hence, in a B-automaton like this, Eve's goal is to satisfy the acceptance condition while minimizing the counter values.

If for all $(q, a) \in Q \times \mathbb{A}$, $\delta(q, a)$ is of the form $\bigvee_i (0, \nu_i^0, q_i^0) \wedge (1, \nu_i^1, q_i^1)$, then we say the automaton is *nondeterministic*. We define a *run* to be the set of possible plays compatible

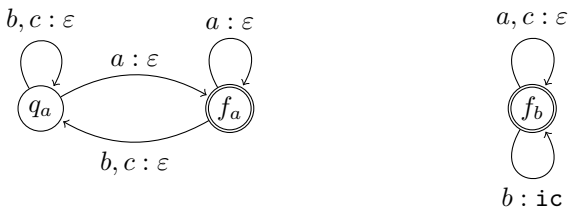
with some fixed strategy of Eve. Since the only choices of Adam are in the branching, a run labels the entire binary tree with states, and choosing a branch yields a unique play of the automaton. A run is accepting if it is accepting on all branches, and the value assigned to a run of a B-automaton is the supremum of the values across all branches. For nondeterministic automata, the choices of Eve and Adam in the game described above can be viewed as Eve picking some $(0, \nu_i^0, q_i^0) \wedge (1, \nu_i^1, q_i^1)$, and Adam choosing a direction (which uniquely determines which atom Adam picks from the conjunction). Note that unless otherwise indicated, we assume automata are alternating.

We can define other types of cost automata simply by varying the acceptance condition. Given a set $F \subseteq Q$, a play is accepting for a *co-Büchi condition* specified by F if states from F occur only finitely often on the play. Given a set of priorities P and a mapping $\Omega : Q \to P$ assigning a priority to every state, a play is accepting for the *parity condition* specified by Ω if the maximum priority occurring infinitely often on the play is even. B-automata with these acceptance conditions will be called B-parity or B-co-Büchi automata as expected. Cost automata is a more general term that includes all of these types of automata with counters.

▶ **Example 1.** Let $\mathbb{A} = \{a, b, c\}$. We describe an alternating B-Büchi automaton $\mathcal{A}$ which computes the function $g : \mathcal{T}_{\mathbb{A}} \to \mathbb{N}_\infty$ defined by $g(t) = 0$ if some branch of t has infinitely many a, and $g(t) = \sup\{|\pi|_b : \pi \text{ is a branch of } t\}$ otherwise, where $|\pi|_b$ denotes the number of b-labelled nodes on the branch π.

Informally, Eve has an initial choice between trying to find a branch with infinitely many a's, or counting the number of b's on a branch chosen by Adam. Formally, let $\mathcal{A} = \langle \{q_0, q_a, f_a, f_b\}, \mathbb{A}, q_0, \{\gamma\}, \{f_a, f_b\}, \delta \rangle$. For any letter $x \in \mathbb{A}$, we have $\delta(q_0, x) = [(0, \varepsilon, q_a) \vee (1, \varepsilon, q_a)] \vee [(0, \varepsilon, f_b) \wedge (1, \varepsilon, f_b)]$.

Then, one of the following automata is deterministically executed, the first one on a branch chosen by Eve, the second one on a branch chosen by Adam.



For instance, $\delta(q_a, a) = (0, \varepsilon, f_a) \vee (1, \varepsilon, f_a)$, and $\delta(f_b, b) = (0, \mathtt{ic}, f_b) \wedge (1, \mathtt{ic}, f_b)$.

Notice that if the letter labelling the root is b, the counter is not incremented. This can result in a difference of 1 between $g(t)$ and $[\![\mathcal{A}]\!](t)$. We will see in Section 2.3 that we can still consider that $\mathcal{A}$ defines g, up to some equivalence relation between functions (noted $[\![\mathcal{A}]\!] \approx g$).

2.2 Variants of weakness for cost automata

Traditional *weak automata* are alternating Büchi automata with a restriction that no cycle of the automaton visits both accepting and rejecting states (this is equivalent to the original definition in [15]).

In the cost setting, there are two natural variants of weakness. We say an alternating B-Büchi automaton is

- B-*weak* if in all cycles, either all states are accepting or all states are rejecting;
- B-*quasi-weak* if in all cycles that contain both an accepting state and a rejecting state, there is a counter that is incremented but not reset.

The weakness property is the traditional notion of weakness, and implies that every play in the game associated with this automaton has to eventually stabilise, either in a strongly connected component where all states are accepting (so the play is winning for Eve), or in a strongly connected component where no state is accepting (so the play is winning for Adam). In fact, this stabilisation occurs after at most $|Q|$-many *changes of mode* between accepting states and rejecting states (where Q is the set of states of the automaton). B-weak automata were studied in [22].

Similarly, the quasi-weakness property implies that any play that does not stabilise after kn-changes of mode (for some constant k depending on the automaton but not depending on the input structure) has a counter with value greater than n. Hence, a play with some bounded value $n \in \mathbb{N}$ must have stabilised in accepting states. B-quasi-weak automata were introduced in [11] where they were shown to be strictly more expressive than B-weak automata, but not as expressive as general B-parity automata.

Thus, the difference between these models is in the number of allowed mode changes: unrestricted for B-Büchi automata; bounded by some function of the value for B-quasi-weak automata; and bounded by some constant for B-weak automata.

▶ Remark. By definition, a B-weak or B-quasi-weak automaton is a special type of alternating B-Büchi automaton. However, a B-weak or B-quasi-weak automaton can always be converted to a B-co-Büchi automaton defining the same function. This can be accomplished by simply complementing the set F of accepting states in the original B-quasi-weak automaton, and then viewing the automaton as an alternating B-co-Büchi automaton; the transition function does not change at all.

We can switch so easily between the Büchi form and co-Büchi form because the only difference between the Büchi and co-Büchi semantics occurs when the play switches infinitely many times between accepting and rejecting states. This is impossible in B-weak automata, and occurs only in plays of infinite counter value in B-quasi-weak automata (when the value of the play is ∞, regardless of the acceptance condition).

2.3 Cost function equivalence

Let $\mathcal{D}$ be some domain of input structures, and $\mathcal{F}_{\mathcal{D}}$ the set of functions : $\mathcal{D} \to \mathbb{N}_\infty$. We say a function $f : \mathcal{D} \to \mathbb{N}_\infty$ is *bounded* on some set $U \subseteq \mathcal{D}$ if there is some $n \in \mathbb{N}$ such that $f(u) \leq n$ for all $u \in U$.

The *domination relation* $\preccurlyeq$ and *boundedness relation* $\approx$ are defined as follows. Given $f, g : \mathcal{D} \to \mathbb{N}_\infty$,

$f \preccurlyeq g$ if for all $U \subseteq \mathcal{D}$, if g is bounded on U then f is bounded on U.

Likewise, $f \approx g$ if $f \preccurlyeq g$ and $g \preccurlyeq f$. In other words,

$f \approx g$ if for all $U \subseteq \mathcal{D}$, f is bounded on U if and only if g is bounded on U.

This means that f and g satisfying $f \approx g$ may not agree on exact values but do agree on boundedness properties across all subsets of the domain of input structures.

▶ **Example 2.** Let $\mathcal{D} = \mathcal{T}_{\mathbb{A}}$ for $\mathbb{A} = \{a, b, c\}$.

- Consider the functions $|\cdot|_a$ and $|\cdot|_b$ mapping a tree $t \in \mathcal{T}_{\mathbb{A}}$ to the number of a-labelled nodes and b-labelled nodes, respectively, in t. Then $|\cdot|_a \approx 2|\cdot|_a$ since a set of trees has bounded output via $|\cdot|_a$ if and only if it has bounded output via the function $2|\cdot|_a$. However, $|\cdot|_a \not\approx |\cdot|_b$ since the family of trees $(t_i)_{i\in\mathbb{N}}$ where t_i has no occurrences of a

and i occurrences of b has a bounded output via $|\cdot|_a$ but unbounded output via $|\cdot|_b$. We can also take the singleton $\{t_b\}$ as a witness, where t_b is the full binary tree labelled only with b's: $|\cdot|_a$ is bounded on $\{t_b\}$ but $|\cdot|_b$ is not, since value ∞ is considered unbounded.

- Given $L \subseteq \mathcal{T}_{\mathbb{A}}$, let χ_L denote the *characteristic function* that maps everything in L to 0 and everything else to ∞. Then for $K, L \subseteq \mathcal{T}_{\mathbb{A}}$, we have $\chi_L \preccurlyeq \chi_K$ if and only if $K \subseteq L$. Likewise, for $L \subseteq \mathcal{T}_{\mathbb{A}}$ and $f : \mathcal{T}_{\mathbb{A}} \to \mathbb{N}_\infty$, $f \approx \chi_L$ if and only if f is bounded on L and $f(t) = \infty$ for all $t \notin L$.

A *cost function over* $\mathcal{D}$ is an equivalence class of $\mathcal{F}_{\mathcal{D}}/\approx$, so we also refer to $\approx$ as *cost function equivalence*. In practice, a cost function (denoted $f, g, \ldots$) will be represented by one of its elements in $\mathcal{F}_{\mathcal{D}}$. In this paper, $\mathcal{D}$ will usually be $\mathcal{T}_{\mathbb{A}}$. The function $[\![\mathcal{A}]\!]$ defined by a cost automaton $\mathcal{A}$ will always be considered as a cost function, *i.e.* only considered up to $\approx$.

3 Expressivity of traditional automata on infinite trees

For readers who are unfamiliar with regular tree languages, we briefly review in this section some results about traditional automata over infinite trees. We refer the reader to [21, 6] for more information.

By setting $\Gamma = \emptyset$, the definitions in the previous section correspond to the traditional definitions of automata over infinite trees. In this case, $L(\mathcal{A}) \subseteq \mathcal{T}_{\mathbb{A}}$ denotes the set of trees t for which there exists a strategy for Eve in $\mathcal{A} \times t$ such that every play π in this strategy satisfies the acceptance condition. We say $\mathcal{A}$ *recognizes* the *language* $L(\mathcal{A})$. The function $[\![\mathcal{A}]\!]$ defined by $\mathcal{A}$ is $\chi_{L(\mathcal{A})}$ (recall that χ_L is the characteristic function of L mapping every tree in L to 0 and all other trees to ∞). Given some language $L \subseteq \mathcal{T}_{\mathbb{A}}$, we write $\overline{L}$ for the complement $\mathcal{T}_{\mathbb{A}} \setminus L$ of L.

In terms of expressivity, parity automata (in either their alternating or nondeterministic form) capture all regular languages of infinite trees. Languages recognized by alternating and nondeterministic Büchi and co-Büchi automata are strict subclasses of the regular tree languages. Deterministic parity automata are strictly less expressive than alternating co-Büchi automata.

Alternating automata can be easily complemented through *dualization*. The dual $\widetilde{\mathcal{U}}$ of an alternating automaton $\mathcal{U}$ is obtained by switching conjunctions and disjunctions in the transition formulas, and dualizing the acceptance condition. For the parity condition, this amounts to incrementing each priority by 1. Likewise, the dual of a Büchi (respectively, co-Büchi) condition specified by F is a co-Büchi (respectively, Büchi) condition specified by F. In each case, the dual automaton $\widetilde{\mathcal{U}}$ recognizes the complement of $\mathcal{U}$ [16].

Thanks to the so-called "breakpoint construction" [14], alternating Büchi automata are expressively equivalent to nondeterministic Büchi automata. Therefore, a tree language is Büchi definable if it is recognizable using either a nondeterministic or alternating Büchi automaton. On the other hand, alternating co-Büchi automata are strictly more expressive than nondeterministic co-Büchi automata (what we call co-Büchi definable). This means that the complement of a co-Büchi definable language is Büchi definable. The complement of a Büchi definable language can be recognized by an alternating co-Büchi automaton but is not necessarily co-Büchi definable.

A language is weakly definable if it is recognizable by a weak automaton, *i.e.* an alternating Büchi automaton satisfying the weakness condition described earlier. Equivalently, a language L is weakly definable if and only if L and $\overline{L}$ are Büchi definable [20, 12]. Stated in terms of alternating automata, the weakly definable languages are precisely the languages recognizable

by both alternating Büchi and alternating co-Büchi automata. The dual of a weak automaton is weak, so weakly definable languages are closed under complement [15].

We pause to mention some well-known examples of weakly definable and Büchi definable languages.

▶ **Example 3.** Let $\mathbb{A} = \{a, b\}$. Consider $L_1, L_2 \subseteq \mathcal{T}_{\mathbb{A}}$ such that L_1 is the language of trees with infinitely many a's on every branch and L_2 is the language of trees with infinitely many a's on some branch.

- L_1 and $\overline{L_1}$ are weakly definable and Büchi definable.
- L_2 is Büchi definable but not weakly definable.
- $\overline{L_2}$ is neither Büchi definable nor weakly definable, but is co-Büchi definable.

4 Reducing to cost function equivalence

4.1 Description of the contribution

We seek to reduce questions about language definability to deciding cost function equivalence for functions defined by cost automata.

Colcombet and Löding [5] provided such a reduction for the parity index problem.

▶ **Theorem 4** ([5][2])**.** *Given a parity automaton $\mathcal{A}$ and a desired set of priorities P, there exists effectively a nondeterministic B-parity automaton $\mathcal{E}$ using priorities P such that the following are equivalent:*

- *$L(\mathcal{A})$ is recognizable by a nondeterministic parity automaton using priorities P,*
- *$[\![\mathcal{E}]\!] \approx \chi_{L(\mathcal{A})}$.*

The rough idea behind the construction is that the automaton $\mathcal{E}$ "guesses" a run of $\mathcal{A}$ (in fact, a run of a special normalized form for $\mathcal{A}$) and tries to map the priorities in $\mathcal{A}$ to the desired set of priorities P. The counters are used as a way to measure mistakes in this mapping, and it turns out that the function defined by $\mathcal{E}$ has a bounded value for all $t \in L(\mathcal{A})$ if and only if $L(\mathcal{A})$ is actually recognizable by a nondeterministic parity automaton using priorities P.

Our main contribution in this paper is to provide a reduction of the weak definability problem for Büchi definable languages to the decidability of $\approx$ for B-quasi-weak automata.

▶ **Theorem 5.** *Given a nondeterministic Büchi automaton $\mathcal{U}$ with $L = L(\mathcal{U})$, there exists effectively a B-quasi-weak automaton $\mathcal{B}$ such that the following are equivalent:*

- *L is weakly definable,*
- *$[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$.*

Recall that early work by Rabin [20] characterized weak definability in terms of Büchi definability of the language and its complement. Kupferman and Vardi [12] built on Rabin's work by providing an explicit construction of a weak automaton from two complementary nondeterministic Büchi automata.

Our construction is derived from this work. However, here we are only given a single Büchi automaton to start, so the constructed cost automaton "guesses" information about the

[2] The notation and terminology used in [5] is different than in this paper. The "distance-parity" automata in that work are B-parity automata here. The reduction is to the uniform universality problem of an automaton $\mathcal{U}_{ij}$, which asks whether $[\![\mathcal{U}_{ij}]\!]$ is equivalent to the constant function 0 (*i.e.* whether it is universal, and has a bounded value across all inputs). The construction of $\mathcal{U}_{ij}$ uses an automaton $\mathcal{A}_{ij}$, which corresponds to $\mathcal{E}$ here.

complementary automaton, and uses the counters to determine whether this complementary automaton is also Büchi (and therefore if the language is weak). Although this reduction was inspired by Theorem 4, it does not rely on it and the proof ideas are quite different.

We start by giving a brief description of the construction in [12], before proceeding to our reduction.

4.2 Proof from Kupferman and Vardi

The proofs in [20] and [12] begin with an analysis of composed runs of two nondeterministic Büchi automata $\mathcal{U} = \langle Q_\mathcal{U}, \mathbb{A}, q_0^\mathcal{U}, F_\mathcal{U}, \delta_\mathcal{U}\rangle$ and $\mathcal{U}' = \langle Q_{\mathcal{U}'}, \mathbb{A}, q_0^{\mathcal{U}'}, F_{\mathcal{U}'}, \delta_{\mathcal{U}'}\rangle$. Let $M := |Q_\mathcal{U}| \cdot |Q_{\mathcal{U}'}|$.

Recall that a frontier E is a set of nodes of t such that for any branch π of t, $E \cap \pi$ is a singleton. Frontiers can be compared: we write $E < E'$ if for any branch π, the only node in $\pi \cap E$ is a strict ancestor of the one in $\pi \cap E'$. Kupferman and Vardi [12] define a *trap* for $\mathcal{U}$ and $\mathcal{U}'$ to be a strictly increasing sequence of frontiers $\{\epsilon\} = E_0 < E_1 < \cdots < E_M$ such that there exists a tree t, a run R of $\mathcal{U}$ on t, and a run R' of $\mathcal{U}'$ on t satisfying the following properties: for all $0 \leq i < M$ and for all branches π in t, there exists $x, x' \in [e_i^\pi, e_{i+1}^\pi)$ such that $R(x) \in F_\mathcal{U}$ and $R'(x') \in F_{\mathcal{U}'}$ where $e_0^\pi < \cdots < e_M^\pi$ is the set of nodes from $E_0, \ldots, E_M$ induced by π. The set of positions $[e_i^\pi, e_{i+1}^\pi)$ can be viewed as an *accepting block* that witnesses an accepting state from both $\mathcal{U}$ and $\mathcal{U}'$.

A pumping argument from [20] shows that the existence of a trap implies $L(\mathcal{U}) \cap L(\mathcal{U}') \neq \emptyset$. We use this property in our proof below and therefore state it as a lemma.

▶ **Lemma 6** ([20],[12]). *If there is a trap for two nondeterministic Büchi automata $\mathcal{U}$ and $\mathcal{U}'$, then $L(\mathcal{U}) \cap L(\mathcal{U}') \neq \emptyset$.*

Now assume that $\mathcal{U}$ and $\mathcal{U}'$ are nondeterministic Büchi automata such that $L(\mathcal{U}')$ is the complement of $L(\mathcal{U})$. In this case, the existence of a trap implies a contradiction (which is why it is called a trap in [12]).

By taking advantage of this trap condition, Kupferman and Vardi [12] show how to use the complementary nondeterministic Büchi automata $\mathcal{U}$ and $\mathcal{U}'$ in order to construct a weak automaton $\mathcal{W}$ recognizing $L(\mathcal{U})$. The general idea is that Eve (respectively, Adam) selects a run of $\mathcal{U}$ (respectively, $\mathcal{U}'$). The acceptance condition in $\mathcal{W}$ requires that any time an accepting state from $\mathcal{U}'$ is seen, an accepting state from $\mathcal{U}$ is eventually seen. Because the existence of a trap is impossible, these accepting blocks only need to be counted up to M times before the automaton is allowed to enter an accepting sink state. Hence, the automaton $\mathcal{W}$ keeps track of the number of blocks (up to M) in the state. Since each block contains at most two changes of mode between accepting and rejecting states, this means there is no cycle visiting both accepting and rejecting states, so $\mathcal{W}$ is weak.

The idea for the proof of correctness is that if $t \in L(\mathcal{U})$, then Eve has a strategy to play in the game $\mathcal{W} \times t$ (*i.e.* play the accepting run of $\mathcal{U}$). On the other hand, if $t \in L(\mathcal{U}')$ and we assume for the sake of contradiction that Eve has a winning strategy in $\mathcal{W} \times t$, then this strategy and the accepting run of $\mathcal{U}'$ can be used to build a trap, which is impossible.

We now turn to our construction and the proof of Theorem 5.

4.3 Construction

Let us now describe the construction of the automaton $\mathcal{B}$ in Theorem 5, which bears a resemblance to the construction in [12]. However, the construction from [12] explicitly uses a Büchi automaton $\mathcal{U}'$ for the complement of $\mathcal{U}$, and furthermore uses an explicit counter in the state space to count up to value M (the value used in the definition of a trap, see above).

We only use the existence of such an automaton $\mathcal{U}'$ in one direction of the correctness proof, and replace the explicit counting by a single B-counter.

Let $\mathcal{U} = \langle Q_\mathcal{U}, \mathbb{A}, q_0^\mathcal{U}, F_\mathcal{U}, \delta_\mathcal{U}\rangle$ be a nondeterministic Büchi automaton without counters. Let $\widetilde{\mathcal{U}}$ be the dual of $\mathcal{U}$, obtained by exchanging conjunctions and disjunctions in the transition formulas of $\mathcal{U}$, and changing the acceptance condition from a Büchi condition specified by F to a co-Büchi condition specified by the same F. The automaton $\widetilde{\mathcal{U}}$ recognizes the complement of $\mathcal{U}$ [16]. Note that dualization of an automaton $\mathcal{U}$ also results in the dualization of the game $\mathcal{U} \times t$: the roles of Adam and Eve exchange (we use this fact in the proof below).

The construction of $\mathcal{B}$ can roughly be described as follows. It consists of two copies of $\widetilde{\mathcal{U}}$, the states of the first of these copies are non-accepting, the states of the second copy are accepting. In the first copy, the transition function of $\widetilde{\mathcal{U}}$ is extended to allow a nondeterministic choice between staying inside the first copy or jumping to the second copy (so each transition of $\widetilde{\mathcal{U}}$ is doubled in a nondeterministic way). In the second copy, the transitions from states in $F_\mathcal{U}$ go back to the first copy and perform an increment on the counter. The other transitions of the second copy, from states not in $F_\mathcal{U}$, stay in the second copy and do not increment the counter.

Formally, define the B-Büchi automaton $\mathcal{B} := \langle Q_\mathcal{B}, \mathbb{A}, q_0^\mathcal{B}, \{\gamma_{\mathrm{alt}}\}, F_\mathcal{B}, \delta_\mathcal{B}\rangle$ with the following components. $Q_\mathcal{B} = Q_\mathcal{U} \times \{R, A\}$ (R and A for rejecting and accepting), $q_0^\mathcal{B} = \langle q_0^\mathcal{U}, R\rangle$, and $F_\mathcal{B} = Q_\mathcal{U} \times \{A\}$.

For $q \in Q_\mathcal{U}$ and $a \in \mathbb{A}$ with $\delta_\mathcal{U}(q, a) = \bigvee_{i=0}^{n_q}(0, q_i^0) \wedge (1, q_i^1)$, we define $\delta_\mathcal{B}$ as follows:

$$\delta_\mathcal{B}(\langle q, R\rangle, a) = \bigwedge_{i=0}^{n_q}(0, \varepsilon, \langle q_i^0, R\rangle) \vee (0, \varepsilon, \langle q_i^0, A\rangle) \vee (1, \varepsilon, \langle q_i^1, R\rangle) \vee (1, \varepsilon, \langle q_i^1, A\rangle).$$

If $q \notin F_\mathcal{U}$ then

$$\delta_\mathcal{B}(\langle q, A\rangle, a) = \bigwedge_{i=0}^{n_q}(0, \varepsilon, \langle q_i^0, A\rangle) \vee (1, \varepsilon, \langle q_i^1, A\rangle)$$

and if $q \in F_\mathcal{U}$ then

$$\delta_\mathcal{B}(\langle q, A\rangle, a) = \bigwedge_{i=0}^{n_q}(0, \mathtt{ic}, \langle q_i^0, R\rangle) \vee (1, \mathtt{ic}, \langle q_i^1, R\rangle).$$

Note that the two copies of $\mathcal{U}$ that are used in $\mathcal{B}$ have the structure of $\widetilde{\mathcal{U}}$ with the additional branching between the two copies, as described above. We use this fact in the proof below to copy strategies of Eve or Adam in $\widetilde{\mathcal{U}} \times t$ to $\mathcal{B} \times t$.

4.4 Proof of correctness

It is easy to see that $\mathcal{B}$ is quasi-weak: any cycle going through both accepting and rejecting states has to take a transition from the second copy of $\widetilde{\mathcal{U}}$ to the first one, thereby incrementing the single counter γ_{alt} of $\mathcal{B}$ (which is never reset). It remains to prove that $\mathcal{B}$ has the property claimed in Theorem 5, namely that $[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$ if and only if L is weakly definable.

For one direction, assume that $[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$. Then there exists $N \in \mathbb{N}$ such that for all $t \in \overline{L}$, there is an accepting run ρ of $\mathcal{B}$ on t with $\mathrm{value}(\rho) \leq N$. Hence, there is a weak automaton $\mathcal{B}'$ based on $\mathcal{B}$ that simulates γ_{alt} in the state up to value N, and enters a special rejecting state as soon as it would exceed this bound. Then $L(\mathcal{B}') = \overline{L}$, and the dual of $\mathcal{B}'$ is a weak automaton defining L. Notice that this part does not depend on how $\mathcal{B}$ is built: the existence of any quasi-weak automaton $\mathcal{A}$ with $[\![\mathcal{A}]\!] \approx \chi_L$ or $[\![\mathcal{A}]\!] \approx \chi_{\overline{L}}$ would imply that L is weakly definable.

For the other direction, assume that L is weakly definable. We first show that $\mathcal{B}$ has value ∞ on all trees from L (this does not use the fact that L is weakly definable). So let $t \in L$ be a tree. Then Eve has a strategy in $\mathcal{U} \times t$ that ensures that $F_\mathcal{U}$ is visited infinitely

often. It follows that in the dual game $\widetilde{\mathcal{U}} \times t$ Adam has a strategy σ that ensures that $F_{\mathcal{U}}$ is visited infinitely often. Adam can use this strategy in $\mathcal{B} \times t$, regardless of which copy is currently used. Note that if the current copy is R, it is Eve who can choose to stay in R or to switch to A. However when the current copy is A, the plays goes back to R as soon as a state from $F_{\mathcal{U}}$ is seen, which depends on Adam's choices. A resulting play will either stay in R eventually, or it alternates infinitely often between A and R because σ ensures that $F_{\mathcal{U}}$ is visited infinitely often. In both cases, the value of the resulting play is ∞.

We now finish the correctness proof by showing that $\mathcal{B}$ is bounded on $\overline{L}$. Since L is weakly definable, there is a nondeterministic Büchi automaton $\mathcal{U}'$ for $\overline{L}$. Let $M := |Q_{\mathcal{U}}| \cdot |Q_{\mathcal{U}'}|$.

Fix some $t \in \overline{L}$. We consider a game that in some sense corresponds to the game $(\mathcal{U}' \times \widetilde{\mathcal{U}}) \times t$. Accordingly, the positions of the game are $Q_{\mathcal{U}'} \times Q_{\mathcal{U}} \times \{0,1\}^*$. One round of the game consists of the following moves, assuming the current position is (q', q, x):

1. Eve chooses a transition of $\mathcal{U}'$, that is, a conjunction $(0, p'_0) \wedge (1, p'_1)$ that appears in $\delta_{\mathcal{U}'}(q', t(x))$.
2. Adam chooses a transition of $\mathcal{U}$, that is, a conjunction $(0, p_0) \wedge (1, p_1)$ that appears in $\delta_{\mathcal{U}}(q, t(x))$.
3. Eve chooses a direction $d \in \{0, 1\}$.

As usual, the game continues from (p'_d, p_d, xd). Note that the moves 2 and 3 are the moves from $\mathcal{U} \times t$ with the roles of Eve and Adam exchanged, and thus correspond to the moves in $\widetilde{\mathcal{U}} \times t$.

Because $t \in \overline{L}$, Eve has a strategy σ to ensure that $F_{\mathcal{U}'}$ is visited infinitely often. Assume that Adam has a strategy τ against such a σ that enforces in each compatible play more than M many alternations between $F_{\mathcal{U}'}$ and $F_{\mathcal{U}}$. Since Eve is choosing the directions, τ works along all branches of t, and therefore would define a trap for $\mathcal{U}$, $\mathcal{U}'$, and t. Since $L(\mathcal{U}) \cap L(\mathcal{U}') = \emptyset$, this would contradict Lemma 6. We can conclude that Eve has a strategy σ that visits $F_{\mathcal{U}'}$ infinitely often and at the same time keeps the number of alternations between $F_{\mathcal{U}'}$ and $F_{\mathcal{U}}$ bounded by M (here, we use the determinacy of the game with the corresponding winning condition for Eve).

From σ we can derive a strategy in $\mathcal{B} \times t$ for Eve that accepts t with value bounded by M. Recall that the moves 2 and 3 correspond to the game $\widetilde{\mathcal{U}} \times t$ and thus Eve can use σ in $\mathcal{B}$ (keeping track of her moves in $\mathcal{U}'$ in her memory). We only have to define when Eve switches from $Q_{\mathcal{U}} \times \{R\}$ to $Q_{\mathcal{U}} \times \{A\}$. She does this whenever the state of $\mathcal{U}'$ in the original game is in $F_{\mathcal{U}'}$. If she plays according to this strategy, then each play switches from $Q_{\mathcal{U}} \times \{R\}$ to $Q_{\mathcal{U}} \times \{A\}$ eventually, because σ ensures infinitely many visits to $F_{\mathcal{U}'}$. Furthermore, Adam can switch back to $Q_{\mathcal{U}} \times \{R\}$ at most M times, because σ ensures at most M alternations between $F_{\mathcal{U}'}$ and $F_{\mathcal{U}}$. Hence, each play eventually remains in $Q_{\mathcal{U}} \times \{A\}$ and has value at most $M \in N$. Since this is true for all $t \in \overline{L}$, $\mathcal{B}$ is bounded on $\overline{L}$ as desired.

This finishes the correctness proof for the construction of $\mathcal{B}$ and thus the proof of Theorem 5.

5 Deciding special cases of cost function equivalence

5.1 Special cases of cost function equivalence over infinite trees

Although we do not know how to decide $\preccurlyeq$ for all cost functions over infinite trees, we can show it is decidable in some cases.

▶ **Theorem 7.** *Let f, g be cost functions over infinite trees. Then $f \preccurlyeq g$ is decidable if these two conditions are satisfied:*

- *f is given by a parity automaton (without counters) or a B-co-Büchi automaton;*
- *g is given by a parity automaton (without counters), B-Büchi automaton, or nondeterministic B-parity automaton.*

This is a slight extension of the decidability of cost function equivalence reported in [22] and [11], but the proof method is the same. The procedure tries to find a witness showing $f \not\preccurlyeq g$. Such a witness is a family of trees with bounded value via g but unbounded value via f. In order to do this, the automata are first converted into nondeterministic forms (so these witnesses can be "guessed"). B-automata are good for g because a single run of a nondeterministic B-automaton can witness a low value. There is a dual form called S-automata which are good for f since they can witness a high value with a single run. We will not describe this dual form here (we refer to the appendix for some details on this model). The types of automata appearing in Theorem 7 describe when we know when we can convert alternating cost automata into these required forms. From there, deciding $\preccurlyeq$ can be reduced to solving parity games on finite graphs. We refer the interested reader to [23] for more information.

Because B-quasi-weak automata can be viewed as either B-co-Büchi or B-Büchi automata (see the Remark at the end of Section 2.2), this means that $f \approx g$ is decidable when f and g are given by B-quasi-weak automata.

▶ **Corollary 8.** *Let f, g be cost functions over infinite trees. Then $f \approx g$ is decidable if each function is given by either a parity automaton (without counters) or a B-quasi-weak automaton.*

In the remainder of this section, we state the decidability results that follow from Theorem 7 and Corollary 8 and the reductions given in Theorem 4 and Theorem 5.

5.2 Deciding co-Büchi definability

In this section we show that it is decidable for a given parity automaton whether there exists an equivalent nondeterministic co-Büchi automaton.

The decidability of co-Büchi definability is a corollary of Theorem 7 and Theorem 4.

▶ **Theorem 9.** *Given a parity automaton $\mathcal{A}$ over infinite trees, it is decidable whether or not there is a nondeterministic co-Büchi automaton $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof. Given $\mathcal{A}$ we apply Theorem 4 with $P = \{0, 1\}$ to obtain a nondeterministic B-co-Büchi automaton $\mathcal{E}$ such that $[\![\mathcal{E}]\!] \approx \chi_{L(\mathcal{A})}$ if and only if $L(\mathcal{A})$ is recognizable by a nondeterministic co-Büchi automaton. According to Theorem 7 it is decidable whether $[\![\mathcal{E}]\!] \approx \chi_{L(\mathcal{A})}$ because the types of the automata $\mathcal{A}$ and $\mathcal{E}$ are such that both directions, $[\![\mathcal{E}]\!] \preccurlyeq \chi_{L(\mathcal{A})}$ and $\chi_{L(\mathcal{A})} \preccurlyeq [\![\mathcal{E}]\!]$ can be decided. ◀

One route to proving the decidability of the full parity index problem would be to prove a more general decidability result for $\approx$, for arbitrary alternating B-parity automata, but this problem remains open.

5.3 Deciding weak definability of Büchi definable languages

We now turn to the main result, deciding weakness for Büchi definable languages. This follows from Corollary 8 and the reduction in Theorem 5.

▶ **Theorem 10.** *Given an alternating Büchi automaton, alternating co-Büchi automaton, or deterministic parity automaton $\mathcal{A}$ with $L = L(\mathcal{A})$, it is decidable whether there is a weak automaton $\mathcal{W}$ such that $L(\mathcal{W}) = L$, or equivalently whether L is definable in weak monadic second-order logic.*

Proof. Note that Corollary 8 implies that $[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$ is decidable when $\mathcal{B}$ is B-quasi-weak. Hence, Theorem 5 implies the decidability of the weak definability problem when the input is a nondeterministic Büchi automaton. The other inputs can all be transformed into a nondeterministic Büchi automaton for the language or its complement.

If the input is an alternating Büchi automaton, then an equivalent nondeterministic Büchi automaton can be constructed ([14, 17]), and Theorem 5 can be applied. If the input is an alternating co-Büchi automaton, then the complement is an alternating Büchi automaton and we can use the previous case to decide whether $\overline{L}$ is weak. Since weakly definable languages are closed under complement, $\overline{L}$ is weak if and only if L is weak.

Finally, if the input is a deterministic parity automaton (not necessarily Büchi), then we reduce to other cases since a nondeterministic Büchi automaton can be constructed for the complement language: the nondeterministic Büchi automaton guesses a branch in the run of the deterministic parity automaton and checks that it is rejecting. As mentioned earlier, the decidability of this case was known already from [18] using different methods. ◀

6 Conclusion

We showed that given a Büchi automaton, it is decidable whether the language it recognizes is weak. We also showed that it is decidable whether a regular tree language is recognizable by a nondeterministic co-Büchi automaton, for arbitrary input. These are particular cases of the Mostowski index problem, for which very few intermediate results are known.

We used the recent formalism of quasi-weak automata to prove these results. We believe this shows that this model is flexible and well-suited for the weak definability problem. Moreover, it provides additional motivation to develop the theory of cost functions, since we demonstrated that it can give rise to new decidability procedures.

The current understanding of the theory of cost functions over infinite trees does not allow us to state a more general theorem, because decidability of the boundedness problem for such cost functions is still open in general. The inherent difficulty of this problem can be explained by the complex interplay between classic acceptance conditions and counter behaviour.

References

1 Bahareh Afshari and Sandra Quickert. Personal communication, 2012.

2 André Arnold and Damian Niwiński. Fixed point characterization of weak monadic logic definable sets of trees. In *Tree Automata and Languages*, pages 159–188. 1992.

3 Mikołaj Bojańczyk and Thomas Colcombet. Bounds in ω-regularity. In *LICS*, pages 285–296. IEEE Computer Society, 2006.

4 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *ICALP (2)*, volume 5556 of *LNCS*, pages 139–150. Springer, 2009.

5 Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In Luca Aceto, Ivan Damgard, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *LNCS*, pages 398–409. Springer, 2008.

6 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

7 Yuri Gurevich and Leo Harrington. Trees, automata, and games. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *STOC*, pages 60–65. ACM, 1982.

8 Kosaburo Hashiguchi. Relative star height, star height and finite automata with distance functions. In Jean-Éric Pin, editor, *Formal Properties of Finite Automata and Applications*, volume 386 of *LNCS*, pages 74–88. Springer, 1988.

9 Karpiński. Personal communication, 2008.

10 Daniel Kirsten. Distance desert automata and the star height problem. *ITA*, 39(3):455–509, 2005.

11 Denis Kuperberg and Michael Vanden Boom. Quasi-weak cost automata: a new variant of weakness. In Supratik Chakraborty and Amit Kumar, editors, *FSTTCS*, volume 13 of *LIPIcs*, pages 66–77. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. Online at `http://www.liafa.jussieu.fr/~dkuperbe/`.

12 Orna Kupferman and Moshe Y. Vardi. The weakness of self- complementation. In Christoph Meinel and Sophie Tison, editors, *STACS*, volume 1563 of *LNCS*, pages 455–466. Springer, 1999.

13 Christop Löding. Logic and automata over infinite trees. Habilitation thesis, RWTH Aachen University, 2009.

14 Satoru Miyano and Takeshi Hayashi. Alternating finite automata on omega-words. *Theor. Comput. Sci.*, 32:321–330, 1984.

15 David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Alternating automata. The weak monadic theory of the tree, and its complexity. In Laurent Kott, editor, *ICALP*, volume 226 of *LNCS*, pages 275–283. Springer, 1986.

16 David E. Muller and Paul E. Schupp. Alternating automata on infinite trees. *Theor. Comput. Sci.*, 54:267–276, 1987.

17 David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theor. Comput. Sci.*, 141(1&2):69–107, 1995.

18 Damian Niwiński and Igor Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Electr. Notes Theor. Comput. Sci.*, 123:195–208, 2005.

19 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

20 Michael O. Rabin. Weakly definable relations and special automata. In *Mathematical Logic and Foundations of Set Theory (Proc. Internat. Colloq., Jerusalem, 1968)*, pages 1–23. North-Holland, Amsterdam, 1970.

21 Wolfgang Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 389–455. Springer, 1997.

22 Michael Vanden Boom. Weak cost monadic logic over infinite trees. In Filip Murlak and Piotr Sankowski, editors, *MFCS*, volume 6907 of *LNCS*, pages 580–591. Springer, 2011.

23 Michael Vanden Boom. *Weak Cost Automata over Infinite Trees*. PhD thesis, University of Oxford, 2012. Online at `http://www.cs.ox.ac.uk/people/michael.vandenboom/`.

A Appendix

There is another form of cost automata called S-automata that is dual to B-automata. We briefly describe the definition of S-automata and its relationship to the B-automata used in the body of the paper.

A.1 S-automata

S-automata have atomic actions increment $\mathtt{i}$, no change ε, reset $\mathtt{r}$, and check-reset $\mathtt{cr}$. Let $\mathbb{S} := \{\mathtt{i}, \varepsilon, \mathtt{cr}, \mathtt{r}\}$. Given a sequence of counter actions $u \in \mathbb{S}^\omega$, let $C(u)$ denote the values of the counter at the moment(s) when a check-reset $\mathtt{cr}$ occurs. For instance, if $u = \mathtt{i}^{100}\mathtt{cri}\varepsilon\mathtt{icr}(\mathtt{ir})^\omega$, then $C(u) = \{2, 100\}$. Likewise, for a set of counters Γ, and a word $u \in (\mathbb{S}^\Gamma)^\omega$, let $C(u) = \bigcup_{\gamma\in\Gamma} C(u_\gamma)$ where u_γ is the γ-projection of u. The S-value of such a sequence is $\inf C(u)$ (the minimum checked counter value).

An *(alternating) S-Büchi automaton* $\langle Q, \mathbb{A}, q_0, \Gamma, F, \delta\rangle$ on infinite trees has a finite set of states Q, alphabet $\mathbb{A}$, initial state $q_0 \in Q$, accepting states F, finite set Γ of counters, and transition function $\delta : Q \times \mathbb{A} \to \mathcal{B}^+(\{0,1\} \times \mathbb{S}^\Gamma \times Q)$. S-automata with other acceptance conditions can be defined as expected. The notion of plays and strategies is the same as in B-automata.

Given a play π from a S-automaton $\mathcal{A}$, the value of π is 0 if π is not accepting and $\inf C(u)$ otherwise, where u is the sequence of counter actions from π. For a strategy σ for Eve, value(σ) is the infimum over the values from all plays consistent with the strategy. For the corresponding game $\mathcal{A} \times t$, value$(\mathcal{A} \times t)$ is

$$\sup\{\text{value}(\sigma) : \sigma \text{ is a strategy for Eve in the game } \mathcal{A} \times t\}$$

and $[\![\mathcal{A}]\!](t) = \text{value}(\mathcal{A} \times t)$. The idea is that in an S-automaton Eve is trying to satisfy the acceptance condition while maximizing the values of the counters (whereas, in a B-automaton Eve is trying to satisfy the acceptance condition and minimize the values of the counters).

A.2 Duality and simulation

Switching between the B and S forms of cost automata takes the place of complementing classical automata.

Indeed, let L be the language recognized by some traditional automaton $\mathcal{A}$ without counters. If we view this automaton as a B-automaton then it defines the function χ_L (the characteristic function for the language), and if we view it as an S-automaton, then it defines the function $\chi_{\overline{L}}$ (the characteristic function for the complement of the language). This means that we can always find both a B and S form for cost functions of the form χ_L for L a regular language recognized by some automaton $\mathcal{A}$ (the S-form for χ_L can be obtained by complementing $\mathcal{A}$).

Using alternating cost automata, we can switch between B and S forms too.

▶ **Theorem 11** ([22]). *Alternating B-parity and alternating S-parity automata are effectively equivalent.*

A closer examination of the proof shows that in the conversion between B-parity and S-parity automata, the priorities are shifted by one (see [23] for more information), which parallels the classical complementation procedure for alternating parity automata. This means that Büchi automata become co-Büchi automata, and vice versa.

▶ **Corollary 12.** *Alternating B-Büchi and alternating S-co-Büchi automata are effectively equivalent.*

Alternating S-Büchi automata and alternating B-co-Büchi automata are effectively equivalent.

Alternating B-Büchi and alternating S-Büchi automata can also be simulated with nondeterministic versions. This parallels the classical result that alternating Büchi automata are equivalent to nondeterministic Büchi automata. The B-part of this result was stated in [22]. The S-part was not stated clearly there, but the proof technique is the same. The details of both of these proofs can be found in [23].

▶ **Theorem 13** ([22],[23])**.** *Alternating B-Büchi and nondeterministic B-Büchi automata are effectively equivalent.*

Alternating S-Büchi and nondeterministic S-Büchi automata are effectively equivalent.

The following corollary summarizes these results.

▶ **Corollary 14.** *Alternating B-co-Büchi automata, alternating S-Büchi automata, and nondeterministic S-Büchi automata are effectively equivalent.*

Alternating S-co-Büchi automata, alternating B-Büchi automata, and nondeterministic B-Büchi automata are effectively equivalent.

In [11], B-quasi-weak automata were characterized as precisely the cost functions that are definable by both nondeterministic B-Büchi and nondeterministic S-Büchi automata.

A.3 Decidability

The decidability of cost function equivalence over infinite trees was originally stated in terms of B and S automata.

▶ **Theorem 15** ([22])**.** *For cost functions f and g over infinite trees, the relation $f \preccurlyeq g$ is decidable if*

- *f is given by a nondeterministic S-parity automaton, and*
- *g is given by a nondeterministic B-parity automaton.*

Theorem 7 can be viewed as a restatement of these results, using Corollary 14 to write it in terms of B-automata only.

Innocent Game Semantics via Intersection Type Assignment Systems*

Pietro Di Gianantonio and Marina Lenisa

Dipartimento di Matematica e Informatica, Università di Udine, Italy
{pietro.digianantonio,marina.lenisa}@uniud.it

Abstract

The aim of this work is to correlate two different approaches to the semantics of programming languages: *game semantics* and *intersection type assignment systems* (ITAS). Namely, we present an ITAS that provides the description of the semantic interpretation of a typed lambda calculus in a *game* model based on innocent strategies. Compared to the traditional ITAS used to describe the semantic interpretation in domain theoretic models, the ITAS presented in this paper has two main differences: the introduction of a notion of labelling on moves, and the omission of several rules, *i.e.* the subtyping rules and some structural rules.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages. Denotational semantics

Keywords and phrases Game Semantics, Intersection Type Assignment System, Lambda Calculus.

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.231

1 Introduction

Game semantics has been proved a powerful and flexible tool to describe the semantics of programming languages. Its main idea is to define the behaviour of a program as a sequence of elementary interactions between the program and the environment. *Intersection types* have first been used to provide *logical descriptions* of domain theory models of λ-calculus [7, 8], but they can be applied to general programming languages. The approach can be outlined as follows. The semantics of the λ-calculus can be given in two forms: a term can be interpreted either *denotationally* by a point in a particular domain, or *logically* by a set of *properties*. *Stone-duality*, as presented in [1], establishes an equivalence between these two alternative descriptions for suitable categories of domains. In the ITAS approach, properties of terms are normally called "types". The logical semantics consists of the set of rules which allow to derive the properties satisfied by a term. ITAS can be used to provide concrete, *finitary* approximations of the semantics of a term.

The present work continues the line of research of [10], aiming at correlating the game semantics and ITAS. These two approaches to the semantics of programming languages seem, at first sight, quite distant one from the other, establishing a relation can enlighten a different perspective on them. Moreover, compared to game semantics, intersection types have a simpler and more direct presentation, so it is interesting to consider what aspects of game semantics can be described through them. In [10], the authors have considered a

* Work partially supported by the Italian MIUR PRIN Project CINA 2010LHT4KM, and by the ICT COST Action BETTY IC1201.

simply-typed λ-calculus, a game model for it, based on games à la Abramsky-Jagadeesan-Malacaria [2] (AJM games), and a corresponding ITAS. It has been shown that such ITAS gives a precise description of the interpretation of λ-terms in the AJM-game model. In particular, a type t for a term M describes a set of moves that the Proponent and the Opponent may exchange in some phases of the interaction of the term M with the environment, and the set of types assigned to a term gives a complete description of the history-free strategy of the term, seen as a partial function on moves. However, some aspects of game semantics are not captured by the ITAS description of AJM-games, for example in AJM-games the same strategies have several possible descriptions, differing by the use of indexes on moves. In order to capture the equivalence relation between strategies, a notion of equivariance between plays has been introduced. However, in the ITAS description, it is not clear how to capture this equivalence relation in a natural and simple way.

In the present work, the aim is to extend results presented in [10], by enlarging the aspects of game semantics that can be described through ITAS, and by considering alternative paradigms of game semantics. In particular, we obtain an ITAS description of *innocent games*, *i.e.* games based on innocent strategies. In this way, we show that the ITAS approach can be used for different paradigms of game semantics. Moreover, for innocent games, each strategy has a single representation, therefore a drawback of the ITAS description of AJM-games, namely the missing equivalence relation between alternative representations of the same strategy, is avoided. In more detail, we show that, by introducing in the ITAS a limited set of structural rules, it is possible to describe the interpretation of λ-terms in the framework of innocent games. The structural rules considered state that the $\wedge$ operator satisfies the associative and commutative properties, but *not* the idempotency. So, from an ITAS perspective, the difference between the two main paradigms of game semantics is reflected by the presence/absence of some structural rules. Technically, intersection types for innocent games carry more structure, since they represent sets of moves which are partitioned via a suitable labelling, and are (implicitly) endowed with justification pointers. Non-idempotent intersection types have been also considered in [15, 16, 9]. In particular, in [15, 16], it has be shown that in non-idempotent ITAS, any term t has a principal type τ that gives a complete description of the normal form of t.

In this paper, we chose, for the sake of simplicity, as target language unary PCF, and we provide an innocent game model for it. The ITAS presentation of this model exploit an alternative description of innocent strategies via *partitioned positions*, which we introduce in this paper. Given a play on a game A, by forgetting the linear order along with moves are played, one obtains a set of moves together with justification pointers. We call this kind of structure *position*. A partitioned position is a play where only part of the information concerning the order in which the moves have been laid down has been omitted. In the ITAS that we present, types correspond to partitioned positions. The induced type semantics turns out to provide the same theory of the game semantics. When simple positions without partitions are considered, the corresponding types and ITAS result simplified, however the theory induced by the game semantics is strictly included in that induced by the type semantics.

The idea to remove, partially or completely, the order information on games has been considered several times in the literature [3, 12, 14, 4, 6]. In [3], timeless games are used to build a model for classical linear logic. In timeless games the order on plays is completely forgotten as in the presentation of innocent strategies via simple positions. In [4], it has been shown that the operation of forgetting the time order can be described by a suitable functor. In [14], it has been shown that, for the particular class of asynchronous games, a

strategy can be completely characterized by the set of its *positions*. In [12, 6], a faithful functor from the category of games to a category of relations is presented, that forgets part of the time order along with moves are played.

Synopsis. In Section 2, we recall basic definitions and constructions on arenas and innocent strategies. In particular, we provide a characterisation of innocent strategies via partitioned positions. In Section 3, we present a game model of unary PCF. In Section 4, we introduce and study an ITAS giving a finitary description of the model of Section 3. In Section 5, we establish the connection between the ITAS and the game model of unary PCF. Finally, in Section 6, we discuss further developments.

Acknowledgements. The authors thank the anonymous referees for their comments, which allowed to improve the presentation of the paper.

2 The Category of Arenas and Innocent Strategies

In this section, we recall basic notions and constructions on arenas and innocent strategies in the style of [11]. Notice that we define justification sequences as containing exactly one initial move.

The following are the usual definitions of arena and strategy:

▶ **Definition 1** (Arena). An *arena* has two participants: the *Proponent* and the *Opponent*. An arena is specified by a triple $A = (M_A, \lambda_A, \vdash_A)$, where

- M_A is the set of *moves*.
- $\lambda_A : M_A \to \{OQ, OA, PQ, PA\}$ is the *labelling* function: it tells us if a move is taken by the Opponent or by the Proponent, and if it is a question or an answer. We denote by $^-$ the function which exchanges Proponent and Opponent.
- $\vdash_A$ is a relation between $M_A + \{\star\}$ and M_A, called *enabling*, which satisfies
  - $\star \vdash_A a \implies (b \vdash_A a \Leftrightarrow b = \star) \wedge \lambda_A(a) = OQ$, and
  - $a \vdash_A b \wedge a \neq \star \implies a$ is a question, *i.e.* $\pi_2 \circ \lambda_A(a) = Q$, and $\pi_1 \circ \lambda_A(a) \neq \pi_1 \circ \lambda_A(b)$.

The enabling relation tells us either that a move a is *initial* and needs no justification ($\star \vdash_A a$), or that it can be justified by another move b, if b has been played ($b \vdash_A a$).

▶ **Definition 2.**

- A *justified sequence* s of moves in an arena A is a sequence of moves together with *justification pointers* such that: *the first move is the only initial move*, and for each other move a in s there is a pointer to an earlier move b of s such that $b \vdash_A a$. We say that the move b *justifies* the move a, and we extend this terminology to say that a move b *hereditary justifies* a if the chain of pointers back from a passes through b.
- Given a justified sequence s, the *view* of s, $view(s)$, also called *P-view*, is defined as follows: $view(\epsilon) = \epsilon$, where ϵ denotes the empty sequence, $view(s \cdot a) = a$, if $s = \epsilon$ or a is an initial move, $view(s \cdot a \cdot t \cdot b) = view(s) \cdot a \cdot b$, if the move b is justified by a.
- If s is a justified sequence containing a move a, we say that a is *visible at* s if a appears in $view(s)$.
- A non-empty justified sequence s is a *play* iff
  - O moves first: $s = as'$ and $\pi_1 \circ \lambda(a) = O$
  - s is *alternating*: if $s = s_1 ab s_2$ then $\pi_1 \circ \lambda(a) \neq \pi_1 \circ \lambda(b)$
  - the *visibility condition* holds: if $s = s_1 a s_2$, and a is not initial, then the justifier of a is visible at s_1

- the *well-bracketing condition* holds: if $s = s_1 a s_2$, and a is an answer, then it must be justified by the most recent unanswered question.

The set of plays of an arena A is denoted by P_A.

Notice that the above definition is slightly different from the standard one: plays are *not* empty, and the *initial move* is *unique*. This presentation will provide a better correspondence with the intersection type assignment system.

▶ **Definition 3** (Strategy). A *strategy* for the Proponent on an arena A is a set $\sigma \subseteq P_A^{even}$ of plays of even length such that: $sab \in \sigma \implies s \in \sigma$ and $sab, sac \in \sigma \implies b = c$.

A strategy σ on an arena A is *innocent* if for all $sab, t \in \sigma$, if $ta \in P_A$ and $view(sa) = view(ta)$, then also $tab \in \sigma$, with b justified by the same element of $view(ta) = view(sa)$ as in sab.

A strategy can be seen as a set of rules which tells the Proponent which move to take after the last move by the Opponent. Innocent strategies are strategies which depend only on the *view*.

Constructions on Arenas

▶ **Definition 4** (Product). Given arenas A and B, the product $A \times B$ is the arena defined as follows:

- $M_{A\times B} = M_A + M_B$
- $\lambda_{A\times B} = [\lambda_A, \lambda_B]$
- $\star \vdash_{A\times B} m \iff \star \vdash_A m \vee \star \vdash_B m$ and $m \vdash_{A\times B} n \iff m \vdash_A n \vee m \vdash_B n$.

Here $+$ denotes disjoint union of sets, that is $A + B = \{(l, a) \mid a \in A\} \cup \{(r, b) \mid b \in B\}$, and $[-, -]$ is the usual (unique) decomposition of a function defined on disjoint unions.

The unit for $\times$ is $I = (\emptyset, \emptyset, \emptyset)$.

▶ **Definition 5** (Implication). Given games A and B, the compound game $A \to B$ is defined as follows:

- $M_{A\to B} = M_A + M_B$
- $\lambda_{A\to B} = [\overline{\lambda_A}, \lambda_B]$
- $\star \vdash_{A\to B} m \iff \star \vdash_B m$ and
  $m \vdash_{A\to B} n \iff m \vdash_A n \vee m \vdash_B n \vee (\star \vdash_B m \wedge \star \vdash_A n)$.

The Game Category $\mathcal{G}$

Objects: arenas.
Morphisms: a morphism between arenas A and B is an innocent strategy σ on $A \to B$.
Composition: given innocent strategies $\sigma : A \to B$ and $\tau : B \to C$, $\tau \circ \sigma : A \to C$ is defined by: $\tau \circ \sigma = \{s \restriction (A, C) \mid s \in \sigma || \tau\}^{even}$, where $\sigma || \tau = \{s \in (M_A + M_B + M_C)^* \mid s \restriction (A, B) \subseteq \sigma \ \& \ s \restriction (B, C) \in \overline{\tau}\}$, with $s \restriction (A, B)$ denoting the subsequence of s consisting of moves in A and B.
Identity: $id_A : A \to A$, $id_A = \{s \in P_A^{even} \mid \forall t$ even-length prefix of $s.\ t \restriction 1 = t \restriction 2\}$, where $t \restriction 1$ ($t \restriction 2$) denotes the restriction of s to the first (second) A component.

The arena constructions of product and implication can be made functorial, in such a way that

▶ **Proposition 6.** *The category $\mathcal{G}$ is cartesian closed with $\times$ as cartesian product and $\to$ as exponential. The arena I is the terminal object of the category $\mathcal{G}$.*

2.1 An Alternative Description of Innocent Strategies

The type assignment system we present describes the strategies associated to λ-terms in an indirect way. To establish the connection between ITAS and games semantics interpretation it is necessary to introduce an alternative description of strategies. Instead of describing an innocent strategy by a set of plays, we describe it by a set of *partitioned positions*. Given a play on a game A, by forgetting the linear order along with moves are played, one obtains a set of moves together with justification pointers for all moves but one (the initial move). We call this kind of structure *position*. For a particular class of games, *i.e.* the asynchronous games, Melliès [14] shows that a strategy is completely characterised by the set of its *positions*. This result is not anymore true for generic innocent games. We therefore introduce the new concept of partitioned position. A partitioned position is a play where only part of the information concerning the order in which the moves have been laid down has been omitted. The innocence condition on strategies assures that using the reduced information allows to reconstruct the original full description of the strategy.

▶ **Definition 7** (Partitioned Position)**.** Let A be an arena.

- We define a *position* on the arena A as an unordered tree, whose nodes are (instances) of moves on A and such that
  1. the root is an initial move,
  2. for any node n, all children of n are moves enabled by n.

  We denote by $(m, \{p_1, \ldots, p_n\})$ the position with root m and subtrees $p_1, \ldots, p_n$.
- A *partitioned position* is a pair (p, E_p), formed by a position p and a partition E_p on the nodes of p. On partitioned positions we consider the partial order $\subseteqq$ given by $(p, E_p) \subseteqq (p, E'_p)$ if E_p is a partition finer than E'_p, *i.e.* each equivalence class of E_p is contained in an equivalence class of E'_p. Since a partition E_p can be also seen as an equivalence relation, for convenience, in some definitions, we will treat E_p as an equivalence relation.

▶ **Definition 8** (Position from Play)**.**

- Given a play s on the arena A, we denote by $[s]^*$ the partitioned position (p, E_p), where
  - the position p is formed by the moves in s together with their justification pointers (a move n is a child of a move m in p if and only if n is justified by m in s);
  - two distinct moves m, n lie in the same set of the partition E_p if and only if m is an Opponent move, and n is the Proponent move immediately following m in the play s.
- Given a strategy σ on the arena A, we denote with $[\sigma]^*$ the set of partitioned positions $\{(p, E_p) \mid \exists s \in \sigma \,.\, [s]^* \subseteqq (p, E_p)\}$.

The function $[\,]^*$ on plays is not injective, that is there can be two distinct plays s and t generating the same partitioned position $[s]^*$ it is not possible to completely recover the linear order of moves in a play s. However, the function $[\,]^*$ is injective on P-views, in fact, given a P-view s and any move m in $[s]^*$, it is possible to define the predecessor of m in s: if m is a Proponent move then its predecessor is the Opponent move laying in the same partition, while if m is an Opponent move, by P-view definition, the predecessor of m is its parent in the tree. Since an innocent strategy is uniquely determined by the set of P-views that it contains ([11], Section 5.2), it follows that the function $[\,]^*$ is injective on innocent strategies. Moreover, from the set $[\sigma]^*$, it is possible to reconstruct the innocent strategy σ. In fact, given a partitioned position (p, E), it is decidable to check if (p, E) is the image of a P-view s along $[\,]^*$, and in this case to reconstruct the P-view s. Therefore, from $[\sigma]^*$, it is possible to define the set of P-views

of σ, and using the construction presented in [11] Section 5.2, from the set of P-views one can define the set of plays of σ.

On the sets of partitioned positions it is possible to define an operation of composition in the following way. A partitioned position (q, E_q) on the arena $A \to B$ can be decomposed in a partitioned position on B, denoted by $(q, E_q) \restriction B$, and in a multiset of partitioned positions in A, denoted by $(q, E_q) \restriction A$. In more detail, if $q = (m, \{p_1, \ldots, p_m, q_1, \ldots, q_n\})$ with $p_1, \ldots, p_m$ having moves in B and $q_1, \ldots, q_n$ having moves in A, $(p, E_p) \restriction B$ is the position $(m, \{p_1, \ldots, p_m\})$ with the inherited partition. The multiset $(p, E_p) \restriction A$ is composed of the multiset of positions $\{q_1, \ldots, q_n\}$ with the inherited partitions.

▶ **Definition 9** (Composition)**.**

- A finite multiset of partitioned positions $\{(q_1, E_{q_1}), \ldots, (q_n, E_{q_n})\}$ in $A \to B$ and a partitioned position (p, E_p) in $B \to C$ *compose* if $(p, E_p) \restriction B = \{(q_1, E_{q_1}) \restriction B, \ldots, (q_n, E_{q_n}) \restriction B\}$. In this case, the composition $\{(q_1, E_{q_1}) \ldots (q_n, E_{q_n})\} \circ (p, E_p)$ is defined as the position:

$$(m,\ \{p_1, \ldots, p_m\} \cup \bigcup_{i \in \{1,\ldots,n\}} \{q_{i,1}, \ldots, q_{i,n_i}\}\ )\ ,$$

  under the hypothesis that $(p, E_p) \restriction C = ((m, \{p_1, \ldots, p_m\}), E_{p'})$ and $(q_i, E_{q_i}) \restriction A = \{(q_{i,1}, E_{q_{i,1}}), \ldots, (q_{i,n_i}, E_{q_{i,n_i}})\}$.
  On the above position we define a partition E as follows: two nodes m, n are related by E iff one of the following conditions holds:
  – the nodes m, n are related either by E_p or by E_{q_i};
  – there exist an index i and a node m' in the arena B such that $(m, m') \in E_p$ and $(n, m') \in E_{q_i}$;
  – there exist indexes i, j and nodes m', n' in the arena B such that $(m, m') \in E_{q_i}$, $(n, n') \in E_{q_j}$, and $(m', n') \in E_p$.
- Given two sets of partitioned positions S in $B \to C$ and T in $A \to B$, the composition $S \circ T$ is defined by

$$\{\{(q_1, E_{q_1}), \ldots, (q_n, E_{q_n})\} \circ (p, E_p)\ \mid\ \{(q_1, E_{q_1}), \ldots, (q_n, E_{q_n})\} \subseteq S,\ (p, E_p) \in T\ \text{ compose}\}.$$

With the above definition of composition, arenas and sets of partitioned positions form the objects and the arrows of a category. It is possible to refine the notion of partitioned position by defining the notion of *well-formed* partitioned position characterizing those positions that are the image, along $[\ ]^*$, of a play. Then one can further define a subcategory having as arrows sets of well-formed partitioned positions. In this subcategory the function $[\ ]^*$ defines the arrow part of a faithful functor from the category of innocent strategies to the one of sets of partitioned positions. However, in the present work we omit this lengthy definition of the category, and we just prove the main property that will be used in the rest of the paper: the function $[\ ]^*$ on strategies preserves composition. The proof of the proposition below appears in the Appendix.

▶ **Proposition 10.** *For any pair of innocent strategies $\sigma : A \to B$ and $\tau : B \to C$, we have that $[\tau \circ \sigma]^* = [\tau]^* \circ [\sigma]^*$.*

2.2 Timeless Games

It is worthwhile to notice that the construction presented above can be repeated using the simpler notion of position, instead of partitioned position. Along this line, one can define a notion of composition between sets of positions, and a function $[\ ]^\bullet$ that associates to

a strategy the set of positions of its plays. As corollary of Proposition 10, one can show that also the function $[\ ]^\bullet$ preserves composition. Although presented in different form, the function $[\ ]^\bullet$ appears in [4]. In this work, positions are described as relations, and it has been shown that $[\ ]^\bullet$ constitutes the arrow part of a functor from the category of arenas and innocent strategies to a suitable category of sets and relations. It turns out that positions are not sufficient to describe innocent strategies, in that it can be the case that two different innocent strategies are mapped to the same set of positions, see at the end of Section 3 below for an example.

3 A Game Model of Unary PCF

In this section, we define a game model of unary PCF. We chose to consider unary PCF, since it is a simple language, with a minimal set of constants, and it allows for a concise presentation of our ITAS. However, the ideas presented in this paper can be immediately extended to more elaborated functional languages with call-by-name reduction.

Models of unary PCF have been extensively studied in the literature, especially extensional ones, see *e.g.* [13, 5]. Here we are interested in the intensional game model arising from the *Sierpinski arena*, which induces the theory of normal forms. In Section 4, we will provide a description of this model via a type assignment system.

We recall that unary PCF is a typed λ-calculus with two ground constants, $\bot, \top$, and a *sequential composition* constant $\&$[1], which takes two arguments of ground type: if its first argument is $\top$, then $\&$ returns its second argument, otherwise, if its first argument is $\bot$, then $\&$ returns $\bot$.

Unary PCF

▶ **Definition 11.** The class *SimType* of simple types over a ground type o is defined by:

$$(\text{SimType } \ni)\ A\ ::=\ o\ \mid\ A \to A\ .$$

Raw Terms are defined as follows:

$$\Lambda \ni\ M\ ::= \bot\ \mid\ \top\ \mid\ \&\ \mid\ x\ \mid\ \lambda x{:}A.M\ \mid\ MM\ ,$$

where $\bot, \top, \&$ are constants, and $x \in \text{Var}$. We denote by Λ^0 the set of closed λ-terms.

Well-typed terms. We introduce a *proof system* for deriving *typing judgements* of the form $\Delta \vdash M : A$, where Δ is a *type environment*, *i.e.* a finite set $x_1 : A_1, \ldots, x_k : A_k$. The rules of the proof system are the following:

$$\frac{}{\Delta \vdash \bot : o} \qquad \frac{}{\Delta \vdash \top : o} \qquad \frac{}{\Delta \vdash \& : o \to o \to o} \qquad \frac{}{\Delta, x : A \vdash x : A}$$

$$\frac{\Delta, x : A \vdash M : B}{\Delta \vdash \lambda x{:}A.M : A \to B} \qquad \frac{\Delta \vdash M : A \to B \quad \Delta \vdash N : A}{\Delta \vdash MN : B}$$

Conversion rules. The conversion relation between well-typed terms is the least relation generated by the following rules together with the rules for congruence closure (which we omit):

$\Delta \vdash (\lambda x{:}A.M)N = M[N/x]$

$\Delta \vdash \&\bot M = \bot \quad \Delta \vdash \&\top M = M \quad \Delta \vdash \&M\top = M$

[1] In the literature, this constant is usually denoted by $\wedge$; here we prefer to denote it by $\&$, since the symbol $\wedge$ is used in the intersection type assignment system.

Notice that the conversion rules for & include reductions where the first or the second argument is $\top$, but only the reduction where the first argument is $\bot$. The reduction in the case the second argument is $\bot$ is omitted, in order to keep the correspondence between normal forms and strategies (see Theorem 15 below).

Game Model

In the cartesian closed category $\mathcal{G}$, simple types are interpreted by the hierarchy of arenas over the following *Sierpinski arena*:

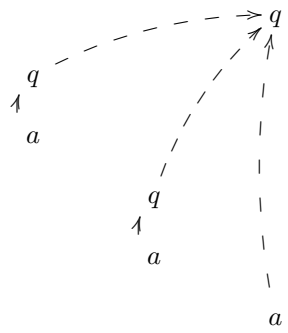
▶ **Definition 12** (Sierpinski Arena)**.** The arena $\mathcal{O}$ is defined as follows:

- $M_{\mathcal{O}} = \{q, a\}$
- $\lambda_{\mathcal{O}}(q) = OQ \quad \lambda_{\mathcal{O}}(a) = PA$
- $\star \vdash_{\mathcal{O}} q$ and $q \vdash_{\mathcal{O}} a$

In the game model, terms in contexts are interpreted as innocent strategies in the usual way, using standard categorical combinators, i.e. $x_1 : A_1, \ldots, x_k : A_k \vdash M : A$ is interpreted as a strategy on the arena $[\![A_1]\!]^{\mathcal{G}} \times \ldots \times [\![A_k]\!]^{\mathcal{G}} \to [\![A]\!]^{\mathcal{G}}$. Before giving the formal interpretation of terms, we first need to define the interpretation of constants.

Interpretation of the basic constants. The interpretation of the constants $\bot$, $\top$ is given by the two strategies on the Sierpinski arena: $[\![\bot]\!]^{\mathcal{G}}$ is the empty strategy, while $[\![\top]\!]^{\mathcal{G}} = \{qa\}$. The interpretation of the constant & is the strategy $[\![\&]\!]^{\mathcal{G}}$ on the arena $\mathcal{O} \to \mathcal{O} \to \mathcal{O}$, defined by the set of plays generated by the even-prefix closure of the play $(r,(r,q))(l,q)(l,a)(r,(l,q))(r,(l,a))(r,(r,a))$ (where justification pointers are omitted).

$[\![\&]\!]^{\mathcal{G}} \; : \; \mathcal{O} \longrightarrow \mathcal{O} \longrightarrow \mathcal{O}$



Given an arena A, we denote by $!_A$ the unique empty strategy from the arena A to the terminal arena I. With the obvious isomorphism, a strategy on the arena A can also be seen as a strategy on the arena $I \to A$.

The complete definition of the type and term interpretation in the model is the following:

▶ **Definition 13** (Type and Term Interpretation)**.**

Type interpretation:

$[\![o]\!]^{\mathcal{G}} = \mathcal{O} \qquad [\![A \to B]\!]^{\mathcal{G}} = [\![A]\!]^{\mathcal{G}} \to [\![B]\!]^{\mathcal{G}}$.

Term interpretation:

- $[\![x_1 : A_1, \ldots, x_k : A_k \vdash c : A]\!]^{\mathcal{G}} = [\![c]\!]^{\mathcal{G}} \circ !_{[\![A_1]\!]^{\mathcal{G}} \times \ldots \times [\![A_k]\!]^{\mathcal{G}}}$ if c is a constant.
- $[\![x_1 : A_1, \ldots, x_k : A_k \vdash x_i : A_i]\!]^{\mathcal{G}} = \pi_i : [\![A_1]\!]^{\mathcal{G}} \times \ldots \times [\![A_k]\!]^{\mathcal{G}} \to [\![A_i]\!]^{\mathcal{G}}$
- $[\![\Delta \vdash \lambda x{:}A.M : A \to B]\!]^{\mathcal{G}} = \Lambda([\![\Delta, x : A \vdash M : B]\!]^{\mathcal{G}})$
- $[\![\Delta \vdash MN : B]\!]^{\mathcal{G}} = ev \circ \langle [\![\Delta \vdash M : A \to B]\!]^{\mathcal{G}}, [\![\Delta \vdash N : A]\!]^{\mathcal{G}} \rangle$

where π_i denotes the i-th projection, ev denotes the natural transformation, and Λ denotes the functor characterizing $\mathcal{G}$ as cartesian closed category.

Using standard methods, one can prove that the theory induced by the game model is the theory of $\beta\eta$-normal forms. The notions of β-normal forms and $\beta\eta$-normal forms on unary PCF are the following:

▶ **Definition 14** (β-normal forms, $\beta\eta$-normal forms)**.**
(i) A typed term $\Delta \vdash M : A$ is in *β-normal form* if
- $M \equiv \lambda x_1 : A_1 \dots x_n : A_n.\ \bot$ or
- $M \equiv \lambda x_1 : A_1 \dots x_n : A_n.\top$ or
- $M \equiv \lambda x_1 : A_1 \dots x_n : A_n.\&MM'$, where M, M' are in β-normal form, $M \neq \bot, \top$, and $M' \neq \top$, or
- $M \equiv \lambda x_1 : A_1 \dots x_n : A_n.x_i M_1 \dots M_{q_i}$, where $M_1, \dots, M_{q_i}$ are in β-normal form.

(ii) A typed term $\Delta \vdash M : A$ is in *$\beta\eta$-normal form* if it is in β-normal form and each occurrence of a variable x of type $B_1 \to \dots \to B_k \to o$ in M appears applied to k arguments of types $B_1, \dots, B_k$, respectively.

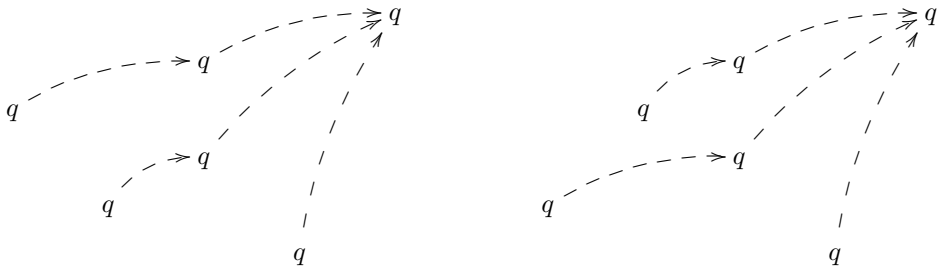
We omit the proof of the following theorem, which is standard:

▶ **Theorem 15.** *The theory $Th^{\mathcal{G}}$ induced by the game model $[\![\ ]\!]^{\mathcal{G}}$ is the $\beta\eta$-theory.*

In view of the results in [13], the extensional quotient of the above game model is universal for the observational equivalence on unary PCF (see [13] for more details).

▶ **Example 16.** We conclude this section by providing an example of two different innocent strategies with the same set of positions. Namely, let us consider the terms $P \equiv \lambda x : o \to o \to o.\lambda y : o.x(x\,\bot(\&\,y\,\bot))(x\,\bot\bot)$ and $Q \equiv \lambda x : o \to o \to o.\lambda y : o.x(x\,\bot\bot)(x(\&\,y\,\bot)\,\bot)$. Then, the strategies σ_P and σ_Q interpreting P and Q are different for only two plays:

$\sigma_P : (\mathcal{O} \to \mathcal{O} \to \mathcal{O}) \to \mathcal{O} \to \mathcal{O}$ $\qquad$ $\sigma_Q : (\mathcal{O} \to \mathcal{O} \to \mathcal{O}) \to \mathcal{O} \to \mathcal{O}$



The first play is contained in σ_P but not in σ_Q, while the second one is contained in σ_Q but not in σ_P. However, the two plays above induce the same position, so as all plays extending them, and hence the strategies interpreting P and Q have the same sets of positions.

4 The Type Assignment System

In this section, we introduce and study a type assignment system, which gives a finitary description of the game model of Section 3. The types involved are essentially the standard intersection types, where some structural rules are missing. Our approach to intersection types is "typed", *i.e.* intersection types are built inductively over arenas. The usual untyped intersection semantics (for the untyped λ-calculus) can be recovered as a special case of the typed case.

Intuitively, a type on an arena A represents a partitioned position induced by a play on A. Types on the Sierpinski arena are just sets of moves contained in the possible plays on this arena. As a further ingredient, moves in types are indexed on natural numbers. Indexes are used to describe partitions: two moves lie in the same partition if and only if they have the same index. A type $(t_1 \wedge \dots \wedge t_n) \to t$ on the arena $A \to B$ represents a partitioned position composed by a partitioned position on B, described by t, and several partitioned positions

on A, described by the types $t_1, \ldots, t_n$. In this approach, the intersection type constructor ($\wedge$) is used to build types on exponential arenas, possibly having multiple instances of the same move. Consequently, the $\wedge$ constructor is *not* idempotent.

The formal correspondence between the type semantics and the game semantics is established in Section 5.

We define a syntax for types that is more complex than the standard one for intersection types. The extra conditions we put on types reflect the alternating and well-bracketing conditions on plays. Namely, for each arena A, we define the set of corresponding intersection types, which divides into *P-types* (t^A_P) and *O-types* (t^A_O), *i.e.* types representing partitioned positions where the Proponent is next to move and types representing partitioned positions where the Opponent is next to move, respectively. Moreover, O-types are divided into "resolved types" (t^A_{Or}), which are intended to represent plays with no pending questions, and "pending types" (t^A_{Op}), which represent plays with pending questions. Notice that all P-types are pending types in this sense.

▶ **Definition 17** (Types). We define two families of *types*, *i.e. Proponent types* (P-types), $\{Type^A_P\}_A$, and *Opponent types* (O-types), $\{Type^A_O\}_A$, these latter are divided into *Opponent resolved types* and *Opponent pending types*, by induction on the structure of the arena A via the following abstract syntax:

- *Types on Sierpinski arena*:

$$(Type^{\mathcal{O}}_P \ni)\ t^{\mathcal{O}}_P ::= \{q_i\} \qquad (Type^{\mathcal{O}}_O \ni)\ t^{\mathcal{O}}_{Or} ::= \{q_i, a_j\} \qquad i,j \in N$$

- *Types on arrow arenas*:

$$\begin{aligned}
&(Type^{A\to B}_P \ni)\ t^{A\to B}_P ::= t^{!A}_{Or} \to t^B_P \ \mid\ t^{!A}_{Op} \to t^B_P \\
&(Type^{A\to B}_O \ni)\ t^{A\to B}_{Or} ::= t^{!A}_{Or} \to t^B_{Or} \\
&(Type^{A\to B}_O \ni)\ t^{A\to B}_{Op} ::= t^{!A}_{Or} \to t^B_{Op} \ \mid\ t^{!A}_{Op} \to t^B_{Op} \ \mid\ t^{!A}_P \to t^B_P
\end{aligned}$$

where

$$\begin{aligned}
&(MType^{!A}_O \ni)\ t^{!A}_{Or} ::= t^A_{Or} \ \mid\ \emptyset^A \ \mid\ t^{!A}_{Or} \wedge t^{!A}_{Or} \\
&(MType^{!A}_O \ni)\ t^{!A}_{Op} ::= t^A_{Op} \ \mid\ t^{!A}_{Op} \wedge t^{!A}_{Op} \ \mid\ t^{!A}_{Op} \wedge t^{!A}_{Or} \\
&(MType^{!A}_P \ni)\ t^{!A}_P ::= t^A_P \ \mid\ t^{!A}_{Or} \wedge t^{!A}_P \ \mid\ t^{!A}_{Op} \wedge t^{!A}_P
\end{aligned}$$

$\emptyset^A$ denotes the empty type on A, and $MType^{!A}_P$ ($MType^{!A}_O$) denotes the set of *Proponent multiple types* (*Opponent multiple types*).

Moreover, we define $Type^A = Type^A_P \cup Type^A_O$, $MType^{!A} = MType^{!A}_P \cup MType^{!A}_O$, $Type = \bigcup_A Type^A$, and $MType = \bigcup_{!A} MType^{!A}$. We use the symbols t^A, u^A, and $t^{!A}, u^{!A}$ to denote types and multiple types respectively, and the symbols t^A_P, u^A_P ($t^{!A}_P, u^{!A}_P$) and t^A_O, u^A_O ($t^{!A}_O, u^{!A}_O$) to denote P (multiple) types and O (multiple) types, respectively.

Finally, we endow the types with the equivalence relation induced by:

$$\emptyset^A \wedge t^{!A} = t^{!A}\ \text{(identity)} \qquad t^{!A}_1 \wedge t^{!A}_2 = t^{!A}_2 \wedge t^{!A}_1\ \text{(commutativity)}$$
$$(t^{!A}_1 \wedge t^{!A}_2) \wedge t^{!A}_3 = t^{!A}_1 \wedge (t^{!A}_2 \wedge t^{!A}_3)\ \text{(associativity)}\ .$$

In the definition of types, justification pointers are not explicitly represented, but they can be recovered from the structure of types.

▶ **Example 18.** The partitioned positions describing the copycat strategy on the arena $\mathcal{O} \to \mathcal{O}$ are induced by the types $\{q_0\} \to \{q_0\}$ and $\{q_0, a_1\} \to \{q_0, a_1\}$.

Notice that, since the type $\{q_0, a_1\} \to \{q_0, a_1\}$ contains as subexpression the type $\{q_0, a_1\}$, the grammar for types needs to generate also types where all indexes are distinct.

To make a more complex example, the two plays that differentiate the strategies σ_P and σ_Q in Example 16 are described by the types:

$$((\{q_1\} \to \emptyset^{\mathcal{O}} \to \{q_0\}) \wedge (\emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\})) \to \{q_2\} \to \{q_0\}$$
$$((\{q_2\} \to \emptyset^{\mathcal{O}} \to \{q_1\}) \wedge (\emptyset^{\mathcal{O}} \to \{q_1\} \to \{q_0\})) \to \{q_2\} \to \{q_0\}$$

Notice that types on the arena $\mathcal{O} \to \mathcal{O}$ containing a single move are P-types in the form $\emptyset \to \{q_i\}$, while types containing two moves are either Opponent resolved types in the form $\emptyset \to \{q_i, a_j\}$ or Opponent pending types in the form $\{q_j\} \to \{q_i\}$.

Since the grammar does not contain the production $t^{!A}_P \wedge t^{!A}_P$, the type $(\{q_0\} \wedge \{q_1\}) \to \{q_0\}$ does not belong to the grammar; this type describes a play not respecting the alternating condition.

Since the grammar does not contain the production $t^{!A}_{O_p} \to t^{B}_{O_r}$, the type $(\{q_1\} \to \{q_0\}) \to \{q_0, a_1\})$ does not belong to the grammar; this type describes a play not respecting the bracketing condition.

▶ **Definition 19** (Environments).

- *Environments* Γ are finite sets $\{x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k}\}$ with the variables $x_1, \ldots, x_k$ all distinct. For simplicity, we omit braces in writing the environments.
- The symbol $\Gamma_\emptyset$ stands for an environment in the form $x_1 : \emptyset^{A_1}, \ldots, x_k : \emptyset^{A_k}$.
- Given two environments Γ, Γ' in the form $\Gamma = x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k}$ and $\Gamma' = x_1 : t_1^{'!A_1}, \ldots, x_k : t_k^{'!A_k}$, we define $\Gamma \wedge \Gamma'$ as the environment $x_1 : t_1^{!A_1} \wedge t_1^{'!A_1}, \ldots, x_k : t_k^{!A_k} \wedge t_k^{'!A_k}$

We introduce a typing system for deriving judgements of the shape $x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k} \vdash M : t^A$, whose intended meaning is to represent a partitioned position in the strategy interpreting the term M in the game model of Section 3.

▶ **Definition 20** (Typing System). The *typing rules* for deriving judgements $x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k} \vdash M : t^A$ are the following:

$$\frac{i \in \mathbb{N}}{\Gamma_\emptyset \vdash \top : \{q_i, a_i\}} \quad (\top)$$

$$\frac{i \in \mathbb{N}}{\Gamma_\emptyset \vdash \& : \{q_i\} \to \emptyset^{\mathcal{O}} \to \{q_i\}} \quad (\&_1)$$

$$\frac{i, j \in \mathbb{N}}{\Gamma_\emptyset \vdash \& : \{q_i, a_j\} \to \{q_j\} \to \{q_i\}} \quad (\&_2)$$

$$\frac{i, j, k \in \mathbb{N}}{\Gamma_\emptyset \vdash \& : \{q_i, a_j\} \to \{q_j, a_k\} \to \{q_i, a_k\}} \quad (\&_3)$$

$$\frac{t^A \in \mathit{Type}^A}{\Gamma_\emptyset, x : t^A \vdash x : t^A} \quad (\mathrm{var})$$

$$\frac{\Gamma, x : u^{!A} \vdash M : t^B}{\Gamma \vdash \lambda x{:}A.M : u^{!A} \to t^B} \quad \text{(abs)}$$

$$\frac{\Gamma \vdash M : u_1^A \wedge \ldots \wedge u_n^A \to t^B \quad \Gamma_1 \vdash N : u_1^A \quad \ldots \quad \Gamma_n \vdash N : u_n^A}{\Gamma \wedge \Gamma_1 \wedge \ldots \wedge \Gamma_n \vdash MN : t^B} \quad \text{(app)}$$

$$\frac{\Gamma \vdash M : \emptyset^A \to t^B \quad \overline{\Gamma} \vdash N : A}{\Gamma \vdash MN : t^B} \quad \text{(app')}$$

where $\overline{\Gamma}$ denotes the simple type environment induced by Γ.

Notice that, in the judgements derivable in the typing system above there is a clear separation between types appearing in the left part (*i.e.* in the environment) and types appearing in the right part: namely, the types in the left part are multiple types, while in the right part only (arrow) types appear.

The extra rule for application (app') is necessary because the expression $\emptyset^A$ only belongs to the grammar of multiple types but not to the grammar of types.

▶ **Example 21.** By the axioms:
$x : \emptyset^{\mathcal{O}\to\mathcal{O}\to\mathcal{O}}, y : \emptyset^{\mathcal{O}} \vdash \& : \{q_2\} \to \emptyset^{\mathcal{O}} \to \{q_2\}$,
$x : \emptyset^{\mathcal{O}\to\mathcal{O}\to\mathcal{O}}, y : \{q_2\} \vdash y : \{q_2\}$,
$x : \emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\}, y : \emptyset^{\mathcal{O}} \vdash x : \emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\}$,
$x : \{q_1\} \to \emptyset^{\mathcal{O}} \to \{q_0\}, y : \emptyset^{\mathcal{O}} \vdash x : \{q_1\} \to \emptyset^{\mathcal{O}} \to \{q_0\}$,
using the rules (app) and (app'), we get $x : \emptyset, y : \{q_2\} \vdash \&y\bot : \{q_2\}$.
Again by the rules (app') and (app), $x : \emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\}, y : \{q_2\} \vdash x\bot(\&y\bot) : \{q_1\}$.
By the rules (app) and (app'),
$x : (\{q_1\} \to \emptyset^{\mathcal{O}} \to \{q_0\} \wedge \emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\}), y : \{q_2\} \vdash x(x\bot(\&y\bot))(x\bot\bot) : \{q_0\}$,
and by a double application of the rule (abs),
$\vdash \lambda x : o \to o \to o.\lambda y : o.x(x \bot (\& \, y \, \bot))(x \bot\bot) : ((\{q_1\} \to \emptyset^{\mathcal{O}} \to \{q_0\}) \wedge (\emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\})) \to \{q_2\} \to \{q_0\}$.

Notice that the following rule is admissible:

$$\frac{\Gamma \vdash M : t^A \quad \phi : \mathbb{N} \to \mathbb{N}}{\Gamma\phi \vdash M : t^A\phi} \quad \text{(sub)}$$

where ϕ is a generic a function on natural numbers and $t^A\phi$ denotes the type t^A where all indexes on moves are substituted according to the function ϕ. The rule (sub) can be usefully employed on the premises of the rule (app), in order to derive premises sharing identical indexes on the corresponding types. Notice that, to obtain this result, it can be necessary to identify different indexes, and so the function ϕ, used as parameter in sub, needs to be a general function and not simply a permutation.

The fact that the types in any derivable judgement are well-formed intersection types follows from Lemma 22 below. This lemma can be easily proved by induction on derivations.

▶ **Lemma 22.** *If $x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k} \vdash M : t^A$ is derivable, then:*
- *if t^A is a resolved O-type, then all $t_1^{!A_1}, \ldots, t_k^{!A_k}$ are resolved O-types;*
- *if t^A is a pending O-type, then all $t_1^{!A_1}, \ldots, t_k^{!A_k}$ are O-types;*
- *if t^A is a P-type, then at most one of the types in $t_1^{!A_1}, \ldots, t_k^{!A_k}$ is a P-type.*

As a consequence, we have:

▶ **Proposition 23.** *If* $x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k} \vdash M : t^A$ *is derivable, then* $(t_1^{!A_1} \to (t_2^{!A_2} \to \ldots (t_k^{!A_k} \to t^A))) \in \mathit{Type}^{(A_1 \to (A_2 \to \ldots (A_k \to A)))}$.

The type assignment system immediately induces a semantics of λ-calculus based on types, whereby any term in context is interpreted by a set of tuples of types as follows:

▶ **Definition 24** (Type Semantics)**.** Let $[\![\ ]\!]^{\mathcal{T}}$ be the interpretation function defined by: $[\![x_1 : A_1, \ldots, x_k : A_k \vdash M : A]\!]^{\mathcal{T}} = \{(t_1^{!A_1}, \ldots, t_k^{!A_k}, t^A) \mid x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k} \vdash M : t^A\}$.

5 From Types to Games

In this section, we show that the type semantics coincides with the game semantics. This result follows from the fact that the types appearing in judgements derivable in the intersection type system correspond to partitioned positions in the strategy interpreting the term.

In order to formally state this correspondence, it is useful to introduce the notion of *indexed position*, which is a position where moves are indexed. Clearly, any indexed position determines a partitioned position, where two moves belong to the same partition if and only if they have the same index; we denote by $\mathcal{U} : IP \to PP$ the natural map from indexed to partitioned positions. Vice versa, any partitioned position determines a class of indexed positions, differing by an injective renaming of indexes. Notice that it would have been possible to use only the notion of indexed position, but we have preferred to introduce also partitioned positions, which provide canonical representatives for strategies.

One can easily define a natural map $\mathcal{E}^A : \mathit{Type}^A \to IP^A$, for any set of types Type^A:

▶ **Definition 25.** For any set of intersection types Type^A, we define $\mathcal{E}^A : \mathit{Type}^A \to IP^A$, by induction on the arena A:

- $\mathcal{E}^{\mathcal{O}}(\{q_i\}) = (q_i, \emptyset) \qquad \mathcal{E}^{\mathcal{O}}(\{q_i, a_j\}) = (q_i, \{(a_j, \emptyset)\})$.
- $\mathcal{E}^{A \to B}(t^{!A} \to t^B) = (m', \{p'_1, \ldots, p'_k, \overline{\mathcal{E}^A(t_1^A)}, \ldots, \overline{\mathcal{E}^A(t_n^A)}\})$,
  where
  - $t^{!A} = t_1^A \wedge \ldots \wedge t_n^A$,
  - $\mathcal{E}^B(t^B) = (m', \{p'_1, \ldots, p'_k\})$,
  - $\overline{\mathcal{E}^A(t_i^A)}$ denotes the position where the polarity of moves has been reversed,
  - the move names in $(m', \{p'_1, \ldots, p'_k, \overline{\mathcal{E}^A(t_1^A)}, \ldots, \overline{\mathcal{E}^A(t_n^A)}\})$ are taken up to the obvious injection in $M_A + M_B$.

The maps $\mathcal{E}^A$ can be extended to $k+1$-tuple of types $(t_1^{!A_1}, \ldots, t_k^{!A_k}, t^A)$ as follows:

▶ **Definition 26.** For all $\mathit{MType}^{!A_1}, \ldots, \mathit{MType}^{!A_k}$, Type^A, for any $k \geq 0$, we define a map $\mathcal{E}^{A_1 \times \ldots \times A_k \to A} : \mathit{MType}^{!A_1} \times \ldots \times \mathit{MType}^{!A_k} \times \mathit{Type}^A \to IP^{A_1 \times \ldots \times A_k \to A}$ by induction on the arenas $A_1, \ldots, A_k, A$ as follows:

- for $k = 0$, $\mathcal{E}^A(t_A)$ is defined as in Definition 25;
- for $k > 0$, $\mathcal{E}^{A_1 \times \ldots \times A_k \to A}(t_1^{!A_1}, \ldots, t_k^{!A_k}, t^A) =$
  $$(m', \{p'_1, \ldots, p'_h, \overline{\mathcal{E}^{A_1}(t_{11}^{A_1})}, \ldots, \overline{\mathcal{E}^{A_1}(t_{1n_1}^{A_1})}, \ldots, \overline{\mathcal{E}^{A_k}(t_{k1}^{A_k})}, \ldots, \overline{\mathcal{E}^{A_k}(t_{kn_k}^{A_k})}\}) ,$$
  where
  - $t_i^{!A_i} = t_{i1}^{A_i} \wedge \ldots \wedge t_{in_i}^{A_i}$, for all i,
  - $\mathcal{E}^A(t^A) = (m', \{p'_1, \ldots, p'_h\})$,
  - $\overline{\mathcal{E}^{A_i}(t_{ij}^{A_i})}$, for all i, j, denotes the position where the polarity of moves has been reversed,

- move names in $(m', \{p'_1, \ldots, p'_h, \overline{\mathcal{E}^{A_1}(t^{A_1}_{11})}, \ldots, \overline{\mathcal{E}^{A_1}(t^{A_1}_{1n_1})}, \ldots, \overline{\mathcal{E}^{A_k}(t^{A_k}_{k1})}, \ldots, \overline{\mathcal{E}^{A_k}(t^{A_k}_{kn_k})}\})$ are taken up to the obvious injection in $M_{A_1} + \ldots + M_{A_k} + M_A$.

The maps $\mathcal{E}^A$ and $\mathcal{U}$ determine a correspondence between the type semantics and the game semantics, namely:

▶ **Definition 27.** We define
$\mathcal{F}([\![x_1 : A_1, \ldots, x_k : A_k \vdash M : A]\!]^{\mathcal{T}}) =$
$$\{\mathcal{U} \circ \mathcal{E}^{A_1 \times \ldots \times A_k \to A}(t_1^{!A_1}, \ldots, t_k^{!A_k}, t^A) \mid x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k} \vdash M : t^A\} \ .$$

Then, we have the following theorem (whose proof appears in the Appendix):

▶ **Theorem 28.**
(i) *For any well-typed term* $\Delta \vdash M : A$, $\quad \mathcal{F}([\![\Delta \vdash M : A]\!]^{\mathcal{T}}) = [[\![\Delta \vdash M : A]\!]^{\mathcal{G}}]^* \ .$
(ii) *The type semantics and the game semantics induce the same theory.*

5.1 ITAS without Indexes

A simplified model for unary PCF can be obtained by using an alternative version of ITAS where types are without indexes. In this alternative version the type semantics of a term M defines the set of positions (and not of partitioned positions) in the strategy $[\![M]\!]^{\mathcal{G}}$.

It turns out that the simplified model does *not* provide the theory of the game model. The terms P and Q considered in the Example 16 are interpreted in the game model by two different strategies, σ_P, σ_Q, containing the same set of positions. More precisely, the theory of the simplified model is intermediate between the theory of the game model and its extensional collapse.

Intersection types without idempotency and without indexes have been considered also in [15, 16]. In these works, it has been shown that two terms having the same set of types have also the same normal form. This result is in contrast with what happen in the above sketched ITAS without indexes, where terms P and Q have different normal forms but the same set of types. This difference can be explained by the fact that in our setting the set of types without indexes contains too few elements; in particular on the Sierpinski arena just three types are definable. In contrast, in [15], the untyped lambda calculus and types built over a countable set of type variables are considered. A posteriori, one can argue that, in order to precisely characterize the normal forms of terms, it is necessary to have a sufficiently rich set of types, and the introduction of indexes on types can be seen not only as a way to encode part of the time order on moves, but also as a way to obtain a richer set of types.

6 Conclusions and Further Work

In this work we have shown how a type assignment system can be used to determine the interpretation of λ-terms in an innocent game model. An interesting aspect that has emerged is that using very similar ITAS, essentially differing among them by the use of structural rules, it is possible to capture a large variety of denotational models: various domain theoretic and game models. It will be interesting to systematically investigate the relations between structural rules and the model counterpart.

As a further result, we hope to construct, in the type semantics setting, fully abstract models of programming languages. In game semantics, fully abstract models are obtained by extensional collapse, exploiting full definability results. To repeat the same construction in the type semantics, it is necessary to obtain an analogous result of full definability. This will imply to find a concrete characterisation of semantical objects, that is to characterise

those sets of types which are interpretations of terms, or, by the full definability of innocent strategies, to characterise those sets of types corresponding to innocent strategies. In a different setting, a similar result has been obtained by Mellies, in [14], for asynchronous games. There, it has been shown that an innocent strategy can be described by the set of its positions; moreover, it has been presented a direct characterisation of those sets of positions which correspond to innocent strategies. An analogous characterisation for partitioned positions could be studied. Since in our setting positions are described by types, this is the sort of result we are looking for.

In general, there are several other aspects in game semantics that arguably can be expressed in terms of intersection types. Game semantics is a quite sophisticated theory and so far we have formulated just one part of it in the ITAS approach. Thus, it is natural to investigate what will be a suitable translation of other game semantics concepts.

References

1 S. Abramsky. Domain theory in logical form. In *Annals of Pure and Applied Logic*, volume 51, pages 1–77, 1991.

2 S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 2000.

3 P. Baillot, V. Danos, T. Ehrhard, and L. Regnier. Timeless games. In M. Nielsen et al., editor, *CSL*, volume 1414 of *LNCS*, pages 56–77. Springer, 1997.

4 P. Boudes. Thick subtrees, games and experiments. In *Typed Lambda Calculi and Applications: Proc. 9th Int. Conf. TLCA 2009*, pages 65–79. Springer-Verlag, 2009. LNCS Vol. 5608.

5 A. Bucciarelli, B. Leperchey, and V. Padovani. Relative definability and models of unary PCF. In *TLCA'03*, volume 2701 of *LNCS*, pages 75–89. Springer, 2003.

6 A. Calderon and G. McCusker. Understanding game semantics through coherence spaces. *Electr. Notes Theor. Comput. Sci.*, 265:231–244, 2010.

7 M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. Formal Logic*, 21(4):685–693, 1980.

8 M. Coppo, M. Dezani-Ciancaglini, F. Honsell, and G. Longo. Extended type structure and filter lambda models. In G. Lolli, G. Longo, and A. Marcja, editors, *Logic Colloquium '82*, pages 241–262. Elsevier Science Publishers B.V. (North-Holland), 1984.

9 D. de Carvalho. *Execution Time of Lambda-Terms via Denotational Semantics and Intersection Types*. Mathematical Structure in Computer Science, 1991. To appear.

10 P. Di Gianantonio, F. Honsell, and M. Lenisa. A type assignment system for game semantics. *Theor. Comput. Sci.*, 398(1-3):150–169, 2008.

11 J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF. *Information and Computation*, 163:285–408, 2000.

12 M. Hyland and A. Schalk. Abstract games for linear logic. *ENTCS*, 29:127–150, 1999.

13 J. Laird. A fully abstract bidomain model of unary PCF. In *TLCA'03*, volume 2701 of *LNCS*, pages 211–225. Springer, 2003.

14 P.A. Melliès. Asynchronous games 2: The true concurrency of innocence. *Theor. Comput. Sci.*, 358(2-3):200–228, 2006.

15 P. M. Neergaard and H. G. Mairson. Types, potency, and idempotency: Why nonlinearity and amnesia make a type system work. In *Proc. 9th Int. Conf. Functional Programming*. ACM Press, 2004.

16 S. Salvati. On the membership problem for non-linear abstract categorical grammars. *Journal of Logic, Language and Information*, 19(2):163 – 183, 2010.

A Proofs

Proof of Proposition 10. First we prove that the first set of partitioned positions is included in the second one. Given a play $s \in \tau \circ \sigma$, by the definition of composition of strategies, there exist a set of plays $s_1, \ldots, s_n \in \sigma$ and a play $t \in \tau$, whose interleaving composition gives s. Let $(p, E_p) = [t]^*$, $(q_i, E_{q_i}) = [s_i]^*$. Starting from the partition E_p, we build a coarser partition E'_p by considering each pair of Opponent-Proponent moves m, n belonging to the arena B and forming a partition of E_{q_i}, the corresponding instances are contained in separated partitions of E_p, the two instances are equated in E'_p. In a symmetric fashion, we build a set of partitions $E'_{q_i} \supseteqq E_{q_i}$. It is not difficult to verify that the partitioned positions (p, E'_p) and $\{(q_i, E'_{q_i}) \mid i \in I\}$ compose and their composition coincides with $[s]^*$. This fact proves that $[s]^* \in [\tau]^* \circ [\sigma]^*$. Moreover any other partitioned position $(q, E_q) \supseteqq [s]^*$, can be shown to belong to $[\tau]^* \circ [\sigma]^*$ by repeating the above construction using suitable partitioned position $(p, E''_p) \supseteqq (p, E'_p)$ and $(q_i, E''_{q_i}) \supseteqq (q_i, E'_{q_i})$.

The proof of the reverse inclusion is more complex. From the hypothesis $(p, E_p) \in [\tau]^* \circ [\sigma]^*$, by definition, it follows that there exists a set of plays $s_1, \ldots, s_n \in \sigma$, a play $t \in \tau$ and a set of partitioned positions $(q_1, E_{q_1}), \ldots, (q_n, E_{q_n}), (p', E_{p'})$ such that: $[s_i]^* \subseteqq (q_i, E_{q_i})$, $[t]^* \subseteqq (p', E_{p'})$, $(p, E) \restriction C = (p', E_{p'}) \restriction C$, $(p, E) \restriction A = \bigcup_{i \in 1..n} (q_i, E_{q_i}) \restriction A$, and $(p', E_{p'}) \restriction B = \bigcup_{i \in 1..n} (q_i, E_{q_i}) \restriction B$.

Since the plays $s_1, \ldots s_n$, and t not necessarily have an interleaving composition, using the innocence hypothesis for the strategies σ and τ, we need to construct a second set of plays $s'_1, \ldots, s'_n$ and t', that do have an interleaving composition and such that $[s'_i]^* \subseteqq (q_i, E_{q_i})$, $[t']^* \subseteqq (p', E_{p'})$. Essentially $s'_1, \ldots, s'_n$ and t' coincide with $s_1, \ldots, s_n$ and t on the components A and C, while on B the move following a move m is determined by considering the behaviour of the Proponent in either the plays $s_1, \ldots, s_n$ or in the play t.

In more detail, the plays $s'_1, \ldots, s'_n$ and t' are defined incrementally as follows. First, one considers a bijection j between the B moves in $s_1, \ldots, s_n$ and the B moves in t induced by the equality $(p', E_{p'}) \restriction B = \bigcup_{i \in 1..n} (q_i, E_{q_i}) \restriction B$. Since $(p', E_{p'}) \restriction B$ is a multiset, the bijection j is not unique.

The initial sequence of t' coincides with the initial sequence t_1 of t till the first instance of a move b_1 in the arena B; b_1 must be a Proponent move. Then one considers the move $\overline{b_1}$ associated to b_1 by j; assume that $\overline{b_1}$ lies in the plays s_i, next, one considers the subsequence $s_{i,1}$ of s_i starting from $\overline{b_1}$ till the next move $\overline{b_2}$ in the arena B. The sequence $s_{i,1}$ forms the initial sequence of s'_i. Notice that $s_{i,1}$ can be composed by just two moves, but can also contain moves in A. Notice moreover that $\overline{b_1}$ is an Opponent move, while $\overline{b_2}$ must be a Proponent move. The construction goes on considering the move b_2 in t associated to $\overline{b_2}$ by j and the subsequence t_2 of t starting from b_2 till the next move b_3 in the arena B. The concatenation $t_1 t_2$ defines the initial sequence of t'. Notice that b_2 and b_3 are, respectively, Opponent and Proponent moves.

Repeating the steps presented above, one considers the move $\overline{b_3}$, associated to b_3 by j. If, by chance, $\overline{b_3}$ is contained in the play s_i, one considers the subsequence $s_{i,2}$ of s_1 starting from the move $\overline{b_3}$ to the next move $\overline{b_4}$ in the arena B, the concatenation $s_{i,1} s_{i,2}$ forms the initial sequence of s'_i. If $\overline{b_3}$ lies in a different sequence s_h, the subsequence from $\overline{b_3}$ to $\overline{b_4}$ defines the initial sequence of s'_h. The construction carries on in this way, moving continuously from the play t to the plays $s_1, \ldots, s_n$, till all moves have been considered. It is immediate to check that the constructed plays t' and $s'_1, \ldots, s'_n$ compose. It remains to prove that the the plays t' and $s'_1, \ldots, s'_n$ satisfy the visibility condition and belong to the innocent strategies τ, σ. The plays t' and $s'_1, \ldots, s'_n$ belong to the innocent strategies since

the Proponent view of a move in t' and $s'_1, \ldots, s'_n$ coincides with the Proponent view of the corresponding moves in t and $s_1, \ldots, s_n$. The visibility condition is satisfied since, for any B move, the Proponent view in t' coincides with the Opponent view in $s'_1, \ldots, s'_n$, and vice versa. To conclude the proof, from $t', s'_1, \ldots, s'_n$ one constructs a play s in the strategy $\tau \circ \sigma$ such that $(p, E_p) \supseteq [s]^*$. ◀

Proof of Theorem 15. Clearly, two $\beta\eta$-equivalent terms have the same interpretation in the model. Vice versa, if two terms at a given type have different $\beta\eta$-normal forms, then, by induction on the structure of them, one can show that the corresponding strategies are different. First of all notice that any normal form $\lambda\vec{x} : \vec{A}.\, \& MM'$ must be of the shape $\lambda\vec{x} : \vec{A}.\, \&(\ldots(\&(x_i\vec{M})M'_1)\ldots)M'_k$, *i.e.* there is a subterm $x_i\vec{M}$, and hence the strategy interpreting the whole normal form interrogates the variable x_i, and it is *not* constant. Therefore, the strategies interpreting the normal forms $\lambda\vec{x}{:}\vec{A}.\, \bot$ and $\lambda\vec{x}{:}\vec{A}.\, \top$, being constant, are different from all strategies interpreting other normal forms. Moreover, if the normal forms are of the shape $\lambda\vec{x}{:}\vec{A}.\, \&(\ldots(\&(x_i\vec{M})M'_1)\ldots)M'_k$ and $\lambda\vec{x}{:}\vec{A}.\, x_j\vec{N}$, then, if $i \neq j$, the corresponding strategies are extensionally different (*e.g.*, when x_i is $\bot$ and x_j is $\top$ of the appropriate types, they provide different results). If $i = j$, then, when x_i is $\top$, the strategy corresponding to the second term yields immediately $\top$, while the strategy corresponding to the first term would yield $\top$ immediately only if the strategy interpreting the second argument would be the $\top$-strategy. But, by induction hypothesis, this means that the second argument is $\top$, which cannot be by hypothesis. Now, if the two normal forms are of the shape $\lambda\vec{x}{:}\vec{A}.\, \&M_1M_2$ and $\lambda\vec{x}{:}\vec{A}.\, \&N_1N_2$, then the strategies interpreting them would be $\Lambda \circ ev \circ \langle ev \circ \langle [\![\&]\!]^{\mathcal{G}}, f_1\rangle, f_2\rangle$ and $\Lambda \circ ev \circ \langle ev \circ \langle [\![\&]\!]^{\mathcal{G}}, g_1\rangle, g_2\rangle$, where f_1, f_2, g_1, g_2 are the interpretations of M_1, M_2, N_1, N_2 in the appropriate environments. By induction hypothesis, $f_1 \neq g_1$ or $f_2 \neq g_2$. Notice that the strategy interpreting & starts by interrogating the first argument and, if this provides an answer, it interrogates the second one. But, since the first argument is different from $\bot$, it must provide an answer. Hence, from the fact that $f_1 \neq g_1$ or $f_2 \neq g_2$, we can conclude that the strategies interpreting the the two normal forms are different. Finally, if the normal forms are of the shape $\lambda\vec{x}{:}\vec{A}.x_iM_1 \ldots M_{q_i}$ and $\lambda\vec{x}{:}\vec{A}.x_jM'_1 \ldots M'_{q_j}$, and the head variable is different, then the strategies are different, because the first starts by interrogating the i-th argument, while the second starts by interrogating the j-th argument. If the head variable is the same in the two terms, but the strategies interpreting one of the arguments are different, *i.e.* $[\![\Delta \vdash \lambda\vec{x}{:}\vec{A}.x_iM_1 \ldots M_{q_i} : B]\!]^{\mathcal{G}} = \Lambda^n \circ ev \circ \langle \ldots \langle ev \circ \langle \pi^n_i, g_1\rangle, g_2\rangle \ldots, g_{q_i}\rangle$ and $[\![\Delta \vdash \lambda\vec{x}{:}\vec{A}.x_iM'_1 \ldots M'_{q_i} : B]\!]^{\mathcal{G}} = \Lambda^n \circ ev \circ \langle \ldots \langle ev \circ \langle \pi^n_i, g'_1\rangle, g'_2\rangle \ldots, g_{q'_i}\rangle$ with $g_j \neq g'_j$ for some j, then, by definition of ev, π^n_i and $\langle\ ,\ \rangle$, one can easily check that the overall strategies are also different. ◀

Proof of Theorem 28.

(i) The proof proceeds by induction on the derivation of $\Delta \vdash M : A$, by showing that $\mathcal{F}([\![\Delta \vdash M : A]\!]^{\mathcal{T}})$ is the set of partitioned positions of the strategy $[\![\Delta \vdash M : A]\!]^{\mathcal{G}}$. For M the ground constants $\bot, \top, \&$ or a variable, the thesis directly follows from the definitions of type assignment system and game semantics. For $\Delta \vdash \lambda x : A.M : A \to B$, the thesis easily follows by induction hypothesis. For $\Delta \vdash MN : B$, applying the induction hypothesis, we have that $\mathcal{F}([\![\Delta \vdash M : A \to B]\!]^{\mathcal{T}})$ is the set of partitioned positions of the strategy $[\![\Delta \vdash M : A \to B]\!]^{\mathcal{G}}$, while $\mathcal{F}([\![\Delta \vdash N : A]\!]^{\mathcal{T}})$ is the set of partitioned positions of the strategy $[\![\Delta \vdash N : A]\!]^{\mathcal{G}}$. Then, the thesis follows by rule (*app*) of the type assignment system, and by the characterisation of strategy application when strategies are viewed as sets of partitioned positions (see Section 2.1).

(ii) By item (i), since both $[\ ]^*$ and $\mathcal{F}$ are injective maps, $Th^{\mathcal{T}} = Th^{\mathcal{G}}$. ◀

Cuts for circular proofs: semantics and cut-elimination

Jérôme Fortier[1] and Luigi Santocanale[2]

1 **LaCIM, UQAM / LIF, AMU**
Montréal, Canada / Marseille, France
jerome.fortier@lif.univ-mrs.fr

2 **LIF, AMU**
Marseille, France
luigi.santocanale@lif.univ-mrs.fr

Abstract

One of the authors introduced in [16] a calculus of circular proofs for studying the computability arising from the following categorical operations: finite products, finite coproducts, initial algebras, final coalgebras. The calculus presented [16] is cut-free; even if sound and complete for provability, it lacked an important property for the semantics of proofs, namely fullness w.r.t. the class of intended categorical models (called μ-bicomplete categories in [18]).

In this paper we fix this problem by adding the cut rule to the calculus and by modifying accordingly the syntactical constraint ensuring soundness of proofs. The enhanced proof system fully represents arrows of the canonical model (a free μ-bicomplete category). We also describe a cut-elimination procedure as a a model of computation arising from the above mentioned categorical operations. The procedure constructs a cut-free proof-tree with possibly infinite branches out of a finite circular proof with cuts.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.1 Mathematical Logic

Keywords and phrases categorical proof-theory, fixpoints, initial and final (co)algebras, inductive and coinductive types

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.248

1 Introduction

Many researchers have studied fixed-point logics with, explicitly or implicitly, a proof-theoretic approach. Such a spread interest for the proof-theory of these logics stem from different fields of theoretical computer science: model-checking and the logics of computation such as modal μ-calculi [9, 12, 19, 20], logic programming and proof search [1, 3], computer aided verification via proof-assistants and its mathematical counterpart, mainly type theory with inductive and coinductive types [2, 7, 11], coalgebras [14] and categorical programming [6].

The calculus of circular proofs was introduced in [16] with, as main aim, that of lifting from provability to the level of proof-theory the game-theoretic machinery developed in the context of the lattice μ-calculus [17]. From a semantic and algebraic perspective, moving from provability to proof-theory meant sliding the focus from posetal structures to categorical structures; and as far as theoretical computer science is concerned, the reason for taking this step was to investigate fixed-point theory from the point of view of semantics of computation in the style of the Curry-Howard-Lawvere isomorphisms. We aim therefore with the present research at investigating the kind of computability arising from the following categorical operations: finite products and coproducts, initial algebras and final coalgebras. This can be

roughly rephrased in type theoretic terms, by saying that we aim at investigating product and sum types, as well as inductive and coinductive types.

In our first proposal [16] we exhibited a cut-free calculus. A circular proof was a finite pointed graph labeled by left or right introduction rules (the rules for additives of linear logic, as well as the fixed point rules, regenerating variables to their bindings) satisfying some constraints on cycles. The cut rule was no part of the calculus and the constraints on cycles were suggested from the theory of parity games in verification. A circular-proof could also be thought as a regular cut-free infinite proof-tree. The proposed calculus enjoyed some nice and non-obvious properties: a sound algebraic semantics and a way to compose two circular proofs into a new circular proof, a construction that could be assimilated to a cut-elimination procedure. However the calculus was not full—a fact of which we were conscious already in [16]—meaning that it was not expressive enough to denote all the arrows in the intended model, a free μ-bicomplete category, cf. [18]. Unfulness could be interpreted as evidence that we could not really dispense of the cut rule.

We fix, with the present work, the lack observed some years ago. We add the cut rule to the set of Gentzen-type rules; such a modification implies rethinking in some coherent way the condition on cycles for being a circular proof. A first main result presented here is the soundness of the proof-system: this amounts to rigorously define the semantics via a theorem asserting the existence and uniqueness of solutions for a certain class of systems of equations. We then prove the fullness of the refined calculus.

A second part of this work provides a cut-elimination procedure for circular proofs. The cut elimination procedure constructs a cut-free, infinite (not necessarily regular), finitely branching proof-tree, thus with infinite branches. This result can be read now as stating that we can actually dispense of the cut rule, but at the cost of giving away regularity of the infinite proof-tree.

2 Preliminaries and notation

Transition systems. A *transition system* is a tuple $G = \langle V, A, \varsigma \rangle$ where V is a set called the *support of* G, A is a set called the *alphabet* and $\varsigma \subseteq V \times A \times V$ is called the *transition relation*. By abuse language, the support V is thereby named G like the transition system itself and its elements are called *vertices*.

Transition systems are often seen as labelled oriented graphs. The notation $u \xrightarrow{a} v$ means $(u, a, v) \in \varsigma$. The *out-degree* of $u \in G$, denoted $\deg(u)$, is the cardinality of the set $\{v \in G : \exists a \in A \text{ such that } u \xrightarrow{a} v\}$. We say that G is *deterministic* if $u \xrightarrow{a} v$ and $u \xrightarrow{a} v'$ implies $v = v'$. In that case, we can also write $v = \varsigma_a u$. If, moreover, $\deg(u) = 1$, then we shall write $u \to v$ and $v = \varsigma u$ without ambiguity. Paths and cycles are defined in the obvious way and the composition of two paths Γ_0, Γ_1, when defined, is denoted $\Gamma_0 \cdot \Gamma_1$. For $u \in G$, let V_u denote the set of all targets of some path from u in G and let $\overline{G, u} = \langle V_u, A, \varsigma_{\restriction V_u \times A \times V_u} \rangle$. $\overline{G, u}$ is called the *reachable graph* from u.

Categories. The reader might consult [10] for basic notions about categories. For $f : a \to b$ and $g : b \to c$ arrows of some category, we shall mostly use $f \cdot g$ to denote their composition; notice however that, with respect to usual notation, we have $f \cdot g = g \circ f$.

3 The calculus of circular proofs

Terms are constructed from a fixed set of variables $\mathbb{V}$ using the binary function symbols $\times, +$ and the constants $1, 0$; the set of terms will be denoted by TERMS. The set of

Identity, Cut, Assumption	$\dfrac{}{t \vdash t}\,\mathtt{Id}$ $\qquad$ $\dfrac{s \vdash u \quad u \vdash t}{s \vdash t}\,\mathtt{Cut}$ $\qquad$ $\dfrac{}{s \vdash t}\,\mathtt{A}$	
	$\mathfrak{L}$	$\mathfrak{R}$
Products	$\dfrac{s_i \vdash t}{s_0 \times s_1 \vdash t}\,\mathtt{L}\times_i \;\; i = 0,1$	$\dfrac{}{t \vdash 1}\,\mathtt{RAx}$ $\dfrac{s \vdash t_0 \quad s \vdash t_1}{s \vdash t_0 \times t_1}\,\mathtt{R}\times$
Coproducts	$\dfrac{}{0 \vdash t}\,\mathtt{LAx}$ $\dfrac{s_0 \vdash t \quad s_1 \vdash t}{s_0 + s_1 \vdash t}\,\mathtt{L}+$	$\dfrac{s \vdash t_i}{s \vdash t_0 + t_1}\,\mathtt{R}+_i \;\; i = 0,1$
Fixpoints	$\dfrac{\tau(x) \vdash t}{x \vdash t}\,\mathtt{L}\mu x$ $\dfrac{\tau(x) \vdash t}{x \vdash t}\,\mathtt{L}\nu x$	$\dfrac{s \vdash \tau(x)}{s \vdash x}\,\mathtt{R}\mu x$ $\dfrac{s \vdash \tau(x)}{s \vdash x}\,\mathtt{R}\nu x$

Figure 1 Inference rules over a system S.

subterms of (resp. variables appearing in) a term t will be denoted $\mathsf{ST}(t)$ (resp. $\mathsf{VAR}(t)$). A *directed systems of equations* is a tuple $S = \langle X, \tau, \pi \rangle$, where X is a finite subset of $\mathbb{V}$, $\tau : X \to \mathsf{TERMS}$, and $\pi : X \to \mathbb{N}$. For such a system S, we let $\mathsf{BD}(S) := X$ and $\mathsf{FV}(S) := \bigcup_{x \in X} \mathsf{VAR}(\tau(x)) \setminus \mathsf{BD}(S)$.

Intuitively, we think of the tuple S as the system of equations $\{\, x =_{\theta(\pi(x))} \tau(x) \mid x \in X \,\}$ where $\theta(n) = \mu$ (least solution) if n is odd and $\theta(n) = \nu$ (greatest solution) otherwise. The priority function π shall also specify the order by which we solve this system of equations (cf. Section 4).

A sequent is here a pair (s, t) of terms. As usual we shall use the turnstile symbol, that is we write the sequent as $s \vdash t$, to separate its left part s from its right part t. We shall use SEQ to denote the set of sequents. For a fixed directed system of equations S, the *inference rules* over S are (instances of) the formal expressions appearing in Figure 1. Notice that the rules in the column $\mathfrak{L}$ act on the left part of a sequent while those $\mathfrak{R}$ act on the right. Accordingly, we group (labels of) the rules in two families $\mathfrak{L}$ and $\mathfrak{R}$ and set $\Sigma := \mathfrak{L} \cup \mathfrak{R} \cup \{\mathtt{Id}, \mathtt{Cut}, \mathtt{A}\}$.

▶ **Definition 1.** A *pre-proof* over S is a tuple $\Pi = \langle G, \rho, \sigma \rangle$ where G is a deterministic transition system over the alphabet $\{0, 1\}$, $\rho : G \to \Sigma$, and $\sigma = (\sigma_{\mathtt{L}}, \sigma_{\mathtt{R}}) : G \to \mathsf{SEQ}$; moreover, for each $v \in G$, $\deg(v) \leq 2$ and the expression (1) is an inference rule over S.

$$\frac{\sigma(\varsigma_0 v) \quad \dots \quad \sigma(\varsigma_{\deg(v)-1} v)}{\sigma(v)}\,\rho(v) \tag{1}$$

In order to be called a *proof*, a pre-proof must satisfy an additional syntactic constraint. The constraint is originally inspired by the theory of parity games: it actually codes, in a proof-theoretic setting, the winning condition on infinite paths. Scientists with a background in type theory might perceive the similarity with the *productivity* constraint of [7, §2.3]. The syntactic constraint turns out to be the key ingredient to ensure soundness of the proof system and local termination of the cut elimination procedure.

Let $\Pi = \langle G, \rho, \sigma \rangle$ be a pre-proof. We say that a path Γ in G is *left-traceable* if, for all n, if $\rho(\Gamma(n)) = \mathtt{Cut}$, then $\Gamma(n+1) = \varsigma_0(\Gamma(n))$. Similarly, Γ is *right-traceable* if, for all n, if $\rho(\Gamma(n)) = \mathtt{Cut}$, then $\Gamma(n+1) = \varsigma_1(\Gamma(n))$. We say that Γ has a *left μ-trace* if Γ is left-traceable, it contains a left fixpoint rule, and the highest priority of its left fixpoint rules is odd. Similarly, we say that Γ has a *right ν-trace* if Γ is right-traceable, it contains a right fixpoint rule, and the highest priority of its right fixpoint rules is even.

▶ **Condition 2** (The Guard condition on cycles)**.** Every cycle in G either has a left μ-trace or a right ν-trace.

▶ **Condition 3** (The Guard condition on infinite paths)**.** Every infinite path Γ in G can be written $\Gamma = \Gamma_0 \cdot \Gamma_1$ where Γ_0 is finite, Γ_1 either has a left μ-trace or a right ν-trace and every fixpoint rule in Γ_1 occurs infinitely often.

It is easily seen that if G is a finite graph, then conditions 2 and 3 are equivalent.

▶ **Definition 4.** A *circular proof* is a pre-proof $\Pi = \langle G, \rho, \sigma \rangle$ satisfying the guard conditions, with G a finite graph.

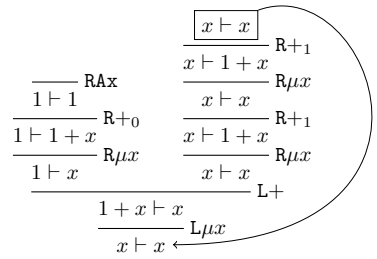
The assumption rule $\mathtt{A}$ is a technical tool that is needed to prove soundness of the system—the reader might have noticed that any sequent can be justified using this rule. Therfore, let $A_\Pi := \{v \in G : \rho(v) = \mathtt{A}\}$ and $C_\Pi := G \setminus A_\Pi$: A_Π is the set of assumptions of Π, while C_Π is the set of its conclusions. A circular proof is *ground* if $A_\Pi = \varnothing$. Even if we often draw a circular proof in the form of a tree with back-edges having a specified root (cf. Figure 3), we consider all the vertexes of the proof as potential conclusions. On the other hand, we can also easily define circular proofs with a root: a *rooted circular proof* is a pair $\langle \Pi, v \rangle$ where Π is a ground circular proof and $v \in G$.

4 Semantics of the calculus

The intended use of circular proofs is to describe functions between (possibly nested) inductive and coinductive types. Before going into the technical details, let us give a few examples.

Recall that a *natural numbers object* is the object part of an initial algebra of the functor $F(x) = 1 + x$ (such an object can be seen as the least categorical solution of the functorial equation $x = 1 + x$); of course, in Set, such an initial algebra is given by the usual data, $1 + \mathbb{N} \xrightarrow{\{0,\mathtt{suc}\}} \mathbb{N}$. The function $\mathtt{double} : \mathbb{N} \to \mathbb{N}$ that sends n to $2n$ is represented as the root of the proof in Figure 2. The $\mathtt{L}+$ instruction can be understood as destructively reading an input and branching according to its constructor (0 or $\mathtt{suc}$). The right rules, on the other hand, represent the choices of constructors for the output. The back edge marks the recursive call of the function to itself.

The fact that reading the input is destructive is a problem that can be partially dismissed with the cut rule. For instance, the interpretation of the circular proof in Figure 3 in Set is the diagonal mapping $\Delta : \mathbb{N} \to \mathbb{N}^2$ defined by $\Delta(n) = (n, n)$. It was constructed by the method given below (see *Fullness* and Figure 5) since Δ is the initial algebra morphism to the algebra $\{(0,0), \mathtt{suc} \times \mathtt{suc}\} : 1 + \mathbb{N}^2 \to \mathbb{N}^2$. It was mentioned in [15] that there is no cut-free circular proof with this interpretation.



$$\texttt{double}(n) = \begin{cases} 0 \text{ if } n = 0; \\ \texttt{suc}(\texttt{suc}(\texttt{double}(n'))) \text{ if } n = \texttt{suc}(n'). \end{cases}$$

Figure 2 Rooted circular proof denoting the function $\texttt{double} : \mathbb{N} \to \mathbb{N}$.

$$S := \left\{ \begin{array}{l} y =_1 x + y \\ x =_1 1 + x \end{array} \right\}$$

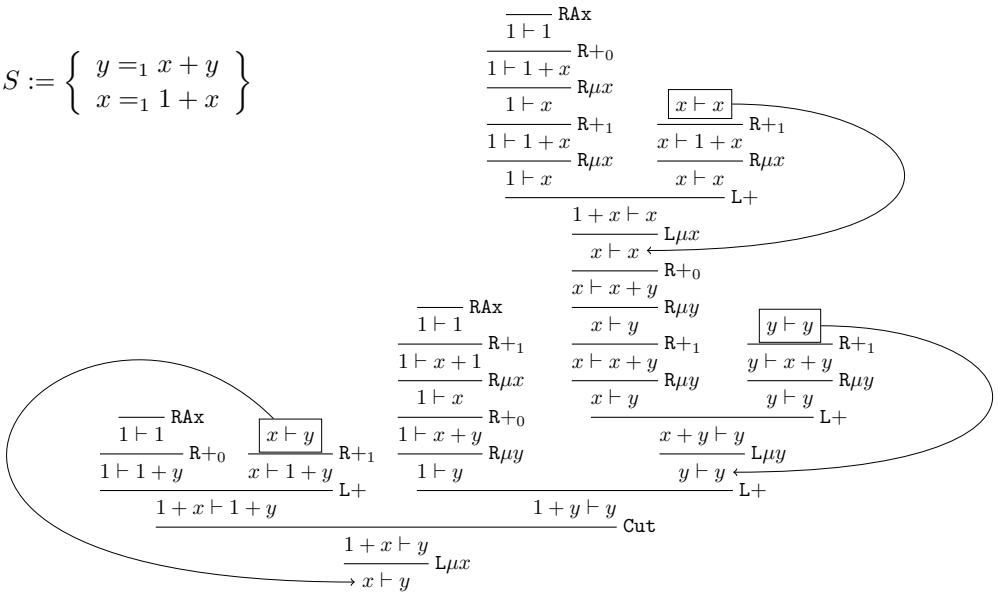


Figure 3 Rooted circular proof denoting the diagonal mapping $\Delta : \mathbb{N} \to \mathbb{N}^2$.

Interpreting the terms. μ-bicomplete categories were defined in previous work on the subject, mainly in [18]. In this paper it was also argued about the equivalence among possible definitions of this notion, via a scalar μ-calculus or via a vectorial one. Roughly speaking μ-bicomplete categories are categories with finite products and finite coproducts (i.e. bicartesian categories) with enough initial algebras and final coalgebras to solve directed systems of equations. Examples of μ-bicomplete categories include the locally presentable categories such as the category of sets, categories of algebras for a finitary signature, and categories of presheaves and sheaves.

Let $\mathcal{M}$ be a μ-bicomplete category and S be a directed system of equations. The definition of the semantics is achieved in three steps: first we define the obvious functorial semantics of terms; then we define the semantics of S as a canonical solution to the system of equations it represents; finally, we evalute terms in $\mathcal{M}$ by means of the bound variables of S.

Given $t \in \mathsf{TERMS}$ and a finite subset X with $\mathsf{VAR}(t) \subseteq X$, the *semantics of* t, denoted $|t|_X$, is a functor from $\mathcal{M}^X$ to $\mathcal{M}$. Let us denote by $\mathcal{M}_X$ the category of functors from $\mathcal{M}^X$ to $\mathcal{M}$ with natural transformations as arrows. Recall that this is a bicartesian category, limits being computed pointwise, see [10, Chapter V]. The formal definition of $|t|_X$ is by induction on the structure of t as follows:

- if $t = x \in X$, then $|x|_X$ is the projection functor on the x component (thus $|x|_X = \mathtt{pr}_x^X$);
- $|0|_X$ (resp. $|1|_X$) is the initial (resp. terminal) object in the category $\mathcal{M}_X$;

- if $t = t_1 \times t_2$ (resp. $t = t_1 + t_2$), then $|t|_X$ is the product $|t_1|_X \times |t_2|_X$ (resp. coproduct $|t_1|_X + |t_2|_X$) in the category $\mathcal{M}_X$.

Given $n \geq 0$ and a system S, let $X_n = \{ x \in \mathsf{BD}(S) \mid \pi(x) \leq n \}$ and let S_n be the restriction of S to X_n, namely $S_n = \langle X_n, \tau_{\restriction X_n}, \pi_{\restriction X_n} \rangle$. In particular, if $M = \max\{ \pi(x) \mid x \in \mathsf{BD}(S) \}$, then we define $\mathsf{MAX}(S) := \{ x \in \mathsf{BD}(S) \mid \pi(x) = M \}$, $\mathsf{LOW}(S) := X_{M-1}$, and let $P(S)$, the *predecessor system*, be S_{M-1}.

Assuming that X is finite and that $\mathsf{FV}(S) \subseteq X$ and $\mathsf{BD}(S) \cap X = \varnothing$, the *semantics of* S, noted by $[\![S]\!]_X$, is a functor from $\mathcal{M}^X$ to $\mathcal{M}^{\mathsf{BD}(S)}$. The definition is as follows:

▶ **Definition 5.** If $\mathsf{BD}(S) = \varnothing$, then $\mathcal{M}^{\mathsf{BD}(S)}$ is the terminal category (with just one object and its identity arrow), so that we let $[\![S]\!]_X$ be the unique functor from $\mathcal{M}^X$ to the terminal category. Otherwise, consider the predecessor system $P(S)$ and observe that $\mathsf{FV}(P(S)) \subseteq \mathsf{MAX}(S) \cup \mathsf{FV}(S) \subseteq \mathsf{MAX}(S) \cup X$ and $(\mathsf{MAX}(S) \cup X) \cap \mathsf{BD}(P(S)) = \varnothing$; as $\mathrm{card}(\mathsf{BD}(P(S))) < \mathrm{card}(\mathsf{BD}(S))$, $[\![P(S)]\!]_{X \cup \mathsf{MAX}(S)}$ is defined as a functor from $\mathcal{M}^{X \cup \mathsf{MAX}(S)}$ to $\mathcal{M}^{\mathsf{BD}(P(S))}$. Let G and H be the functors so defined:

$$G := \langle |\tau(x)|^{\mathsf{BD}(S) \cup X} \mid x \in \mathsf{MAX}(S) \rangle : \mathcal{M}^{\mathsf{BD}(S) \cup X} \to \mathcal{M}^{\mathsf{MAX}(S)},$$

$$H := \langle\, G\,,\ [\![P(S)]\!]_{\mathsf{MAX}(S) \cup X} \circ \mathtt{pr}^{\mathsf{BD}(S) \cup X}_{\mathsf{MAX}(S) \cup X} \,\rangle :$$

$$\mathcal{M}^{\mathsf{BD}(S)} \times \mathcal{M}^X = \mathcal{M}^{\mathsf{BD}(S) \cup X} \longrightarrow \mathcal{M}^{\mathsf{MAX}(S)} \times \mathcal{M}^{\mathsf{BD}(P(S))} = \mathcal{M}^{\mathsf{BD}(S)}.$$

If $\pi(\mathsf{MAX}(S))$ is odd, then we let $[\![S]\!]_X$ be the parametrized initial algebra of H; and if $\pi(\mathsf{MAX}(S))$ is even, then we let $[\![S]\!]_X$ be the parametrized final coalgebra of H.

▶ Remark. The existence of an initial algebra and of a final coalgebra in the previous definition is ensured by the assumption that $\mathcal{M}$ is a μ-bicomplete category.

Finally, given a system S, a term t, and a finite subset X with $\mathsf{FV}(S) \cup (\mathsf{VAR}(t) \setminus \mathsf{BD}(S)) \subseteq X$, the *value of* t *w.r.t.* S, noted $[\![t]\!]_X$, is the functor from $\mathcal{M}^X$ to $\mathcal{M}$ defined below:

$$[\![t]\!]_X := \left(\mathcal{M}^X \xrightarrow{\langle \mathtt{id}, [\![S]\!]_X \rangle} \mathcal{M}^X \times \mathcal{M}^{\mathsf{BD}(S)} = \mathcal{M}^{X \cup \mathsf{BD}(S)} \xrightarrow{|t|_{X \cup \mathsf{BD}(S)}} \mathcal{M} \right).$$

We leave the reader to verify that if $X \subseteq Y$, then $|t|_Y$ (resp. $[\![S]\!]_Y$, $[\![t]\!]_Y$) is obtained from $|t|_X$ (resp. $[\![S]\!]_X$, $[\![t]\!]_X$) by precomposing the latter with the projection from $\mathcal{M}^Y$ to $\mathcal{M}^X$. The previous observation allows us to be sloppy with the notation and to omit the subscript X, which shall be understood from the context as the least set of variables satisfying some required constraints. For example, if we are considering a set of terms $E = \{ t_1, \ldots, t_n \}$ with $t \in E$, then we shall have $[\![t]\!] := [\![t]\!]_X$ with $X = \mathsf{FV}(S) \cup (\bigcup_{i=1,\ldots,n} \mathsf{VAR}(t_i) \setminus \mathsf{BD}(S))$. It might be necessary, on the other hand, to evaluate a term with respect to different systems S and T; we shall then write the system in superscript, so to have $[\![t]\!]^S$ and $[\![t]\!]^T$.

Let $M = \pi(\mathsf{MAX}(S))$; let us remark that if M is odd, then for all $x \in \mathsf{BD}(S)$ there is (by definition of $[\![S]\!]$) a canonical invertible arrow $\zeta_x : [\![\tau(x)]\!] \to [\![x]\!]$; however, if $\pi(x) < M$, then we can assume that ζ_x is the identity while $[\![x]\!](Y) = [\![x]\!]^{S \restriction \pi(x)}([\![Z]\!], Y)$ where $Y = \mathsf{FV}(S)$ and $Z = \{ z \in \mathsf{BD}(S) \mid \pi(z) > \pi(z) \}$ (see Proposition 2.2 in [18] with $F := [\![P(S)]\!]$, $G := G$, $\mathsf{C} := \mathcal{M}^{MAX(S)}$, and $\mathsf{D} := \mathcal{M}^{\mathsf{LOW}(S)}$). Similarly, if $x \in \mathsf{BD}(S)$ and $\pi(\mathsf{MAX}(S))$ is odd, then there exists a canonical invertible arrow $\xi_x : [\![x]\!] \to [\![\tau(x)]\!]$; if $\pi(x) < M$, then we can assume that ξ_x is the identity and that $[\![x]\!](Y) = [\![x]\!]^{S \restriction \pi(x)}([\![Z]\!], Y)$. An easy induction shall therefore prove the following statement:

▶ **Proposition 6.** *For each $x \in \mathsf{BD}(S)$, if $\pi(x)$ is odd, then there exists a canonical invertible arrow $\zeta_x : [\![\tau(x)]\!] \to [\![x]\!]$; if $\pi(x)$ is even, then there exists a canonical invertible arrow $\xi_x : [\![x]\!] \to [\![\tau(x)]\!]$.*

Identity, Cut	$\dfrac{}{[\![t]\!] \xrightarrow{id_{[\![t]\!]}} [\![t]\!]}\,\mathtt{Id}$ $\qquad$ $\dfrac{[\![s]\!] \xrightarrow{f} [\![u]\!] \quad [\![u]\!] \xrightarrow{g} [\![t]\!]}{[\![s]\!] \xrightarrow{f\cdot g} [\![t]\!]}\,\mathtt{Cut}$	
Products	$\dfrac{[\![s_i]\!] \xrightarrow{f} [\![t]\!]}{[\![s_0 \times s_1]\!] \xrightarrow{\mathtt{pr}_i\cdot f} [\![t]\!]}\,\mathtt{L}\times_i \ i=0,1$	$\dfrac{}{[\![t]\!] \xrightarrow{!_{[\![t]\!]}} [\![1]\!]}\,\mathtt{RAx}$ $\quad$ $\dfrac{[\![s]\!] \xrightarrow{f} [\![t_0]\!] \quad [\![s]\!] \xrightarrow{g} [\![t_1]\!]}{[\![s]\!] \xrightarrow{\langle f,g\rangle} [\![t_0 \times t_1]\!]}\,\mathtt{R}\times$
Coproducts	$\dfrac{}{[\![0]\!] \xrightarrow{?_{[\![t]\!]}} [\![t]\!]}\,\mathtt{LAx}$ $\quad$ $\dfrac{[\![s_0]\!] \xrightarrow{f} [\![t]\!] \quad [\![s_1]\!] \xrightarrow{g} [\![t]\!]}{[\![s_0 + s_1]\!] \xrightarrow{\{f,g\}} [\![t]\!]}\,\mathtt{L}+$	$\dfrac{[\![s]\!] \xrightarrow{f} [\![t_i]\!]}{[\![s]\!] \xrightarrow{f\cdot \mathtt{in}_i} [\![t_0 + t_1]\!]}\,\mathtt{R}+_i \ i=0,1$
Fixpoints	$\dfrac{[\![\tau(x)]\!] \xrightarrow{f} [\![t]\!]}{[\![x]\!] \xrightarrow{\zeta_x^{-1}\cdot f} [\![t]\!]}\,\mathtt{L}\mu x$ $\quad$ $\dfrac{[\![\tau(x)]\!] \xrightarrow{f} [\![t]\!]}{[\![x]\!] \xrightarrow{\xi_x\cdot f} [\![t]\!]}\,\mathtt{L}\nu x$	$\dfrac{[\![s]\!] \xrightarrow{f} [\![\tau(x)]\!]}{[\![s]\!] \xrightarrow{f\cdot\zeta_x} [\![x]\!]}\,\mathtt{R}\mu x$ $\quad$ $\dfrac{[\![s]\!] \xrightarrow{f} [\![\tau(x)]\!]}{[\![s]\!] \xrightarrow{f\cdot\xi_x^{-1}} [\![x]\!]}\,\mathtt{R}\nu x$

Figure 4 Semantics of rules.

Interpreting the rules. From now on our goal shall be that of associating to a circular proof Π over a system S its semantics; this shall be a collection of arrows (one for each conclusion of Π) in some μ-bicomplete category. If Π is ground and does not have cycles, then this task is easily achieved using induction; namely, we start from the leaves and, by interpreting the rules of the calculus as specifying how to construct new arrows from given ones via the categorical operations, we build more complex arrows. Such interpretation of rules is given in Figure 4. For each vertex $v \in \Pi$ with $\sigma(v) = s \vdash t$, the construcion gives an arrow f_v from $[\![s]\!]_X$ to $[\![t]\!]_X$ in the category $\mathcal{M}_X$; that is, f_v is a natural transformation from $[\![s]\!]_X$ to the functor $[\![t]\!]_X$.

Thus, if we write $\mathcal{M}_X(F,G)$ for the set of arrows from F to G in the category $\mathcal{M}_X$, each rule `Rule`, with assumptions $s_i \vdash t_i$ and conclusion $s \vdash t$, can be intertpreted as a function from $\prod_{i=1,\dots,n} \mathcal{M}_X([\![s_i]\!],[\![t_i]\!])$ to $\mathcal{M}_X([\![s]\!],[\![t]\!])$. We notice, however, that for $F, G \in \mathcal{M}_X$, the similar expression $\mathcal{M}(F,G)$ denotes the following functor:

$$(\mathcal{M}^X)^{op} \times \mathcal{M}^X \xrightarrow{F^{op}\times G} \mathcal{M}^{op} \times \mathcal{M} \xrightarrow{\mathcal{M}(\_,\_)} \mathrm{Set}\,.$$

With exception of `Id` and `Cut`, all the rules can also be interpreted as defining natural transformations of these generalized hom-functors:

$$\lceil \mathtt{Rule} \rceil_{X,X'} : \prod_{i=1,\dots,n} \mathcal{M}([\![s_i]\!],[\![t_i]\!]) \to \mathcal{M}([\![s]\!],[\![t]\!]) : (\mathcal{M}^X)^{op} \times \mathcal{M}^X \to \mathrm{Set}\,. \tag{2}$$

The following Lemma relates the two possible semantical interpretations of rules.

▶ **Lemma 7.** *Let $\alpha : \prod_{i=1,\dots,n} \mathcal{M}(F_i, G_i) \to \mathcal{M}(F, G)$ be a natural transformation and suppose that, for each $i = 1, \dots n$, we are given a natural transformation $\beta^i \in \mathcal{M}_X(F_i, G_i)$. Then the collection of arrows $\alpha_{c,c}(\beta^1_c, \dots, \beta^n_c) : Fc \to Gc$, c an object of $\mathcal{M}^X$, is a natural transformation from F to G.*

We notice next that even the cut rule can be given a semantics as a natural transformation of hom-functors. Namely, *given a natural transformation $\beta : [\![u]\!] \to [\![t]\!]$*, we can define the semantics of a cut as the natural transformation

$$\lceil \mathtt{Cut}, \beta \rceil : \mathcal{M}([\![s]\!], [\![u]\!]) \to \mathcal{M}([\![s]\!], [\![t]\!]) \tag{3}$$

sending $f : [\![s]\!](c) \to [\![u]\!](d)$ to $f \cdot \beta_d : [\![s]\!](c) \to [\![t]\!](d)$. Similarly, given $\gamma : [\![s]\!] \to [\![s]\!]$, we can define the semantics of a cut as follows:

$$\lceil \gamma, \mathtt{Cut} \rceil : \mathcal{M}([\![u]\!], [\![t]\!]) \to \mathcal{M}([\![s]\!], [\![t]\!])\,, \qquad \lceil \gamma, \mathtt{Cut} \rceil (f) = \gamma_c \cdot f : [\![s]\!](c) \to [\![t]\!](d)\,. \tag{4}$$

Interpreting some derived inference rules. Motivated by the previous observations about the semantics of certain rules, we shall make sense of a large collection of circular-proofs as derived inference rules whose interpretation is a natural transformation between hom-set functors.

▶ **Definition 8.** A circular proof Π is *homogeneous* if it does not contain the rule $\mathtt{Id}$ and, for each $v \in \Pi$ with $\rho(v) = \mathtt{Cut}$, at least one among $\varsigma_0 v$ and $\varsigma_1 v$ is an assumption of Π.

It was argued in [16] that we can associate to an identity-free and cut-free circular proof Π a system of equations $\lceil \Pi \rceil$; it was then shown that Π has a unique solution $\lceil \Pi \rceil_\dagger$, thus defining the semantics of Π via this unique solution. We generalize here this result to homogeneous circular proofs. For each $v \in \Pi$ with $\rho(v) = \mathtt{Cut}$, let $\chi(v) \in \{ 0, 1 \}$ such that $\varsigma_{\chi(v)} v \in A_\Pi$; let therefore $A^{\mathtt{c}}_\Pi := \{ \varsigma_{\chi(v)} v \in A_\Pi \mid \rho(v) = \mathtt{Cut} \}$ and $A^{\mathtt{s}}_\Pi := A_\Pi \setminus A^{\mathtt{c}}_\Pi$ (w.l.o.g., we assume that if $v \in A_\Pi$, then v has just one predecessor in G). Given a collection of natural transformations $\beta = \{ \beta^v : [\![\sigma_{\mathtt{L}}(v)]\!] \to [\![\sigma_{\mathtt{R}}(v)]\!] \mid v \in A^{\mathtt{c}}_\Pi \}$, the system $\lceil \Pi_\beta \rceil$ is defined as

$$\lceil \Pi_\beta \rceil := \left\{ \; v \; = \; \lceil \rho(v)_\beta \rceil (C_\Pi, A^{\mathtt{s}}_\Pi) \; \right\}_{v \in C_\Pi}\,. \tag{5}$$

In the definition of $\lceil \Pi_\beta \rceil$ above, if $\rho(v) \neq \mathtt{Cut}$, then $\lceil \rho(v)_\beta \rceil := \lceil \rho(v) \rceil$ is as in (2); if $\rho(v) = \mathtt{Cut}$, then: if $\varsigma_1 v \in A_\Pi$, then $\lceil \rho(v)_\beta \rceil = \lceil \mathtt{Cut}, \beta_{\varsigma_1 v} \rceil$ as in (3); if $\varsigma_0 v \in A_\Pi$, then $\lceil \rho(v)_\beta \rceil = \lceil \beta_{\varsigma_0 v}, \mathtt{Cut} \rceil$ as in (4). Furthermore the defining equation (5) emphasizes that each $\lceil \rho(v)_\beta \rceil$ depends on two kinds of variables, those coming from C_Π and those coming from $A^{\mathtt{s}}_\Pi$. Therefore, the system has the conclusions of Π as bound variables and depends on parameters coming from $A^{\mathtt{s}}_\Pi$. We identify such a system with the natural transformation

$$\lceil \Pi_\beta \rceil : \prod_{v \in C_\Pi} \mathcal{M}([\![\sigma_{\mathtt{L}}(v)]\!], [\![\sigma_{\mathtt{R}}(v)]\!]) \times \prod_{v \in A^{\mathtt{s}}_\Pi} \mathcal{M}([\![\sigma_{\mathtt{L}}(v)]\!], [\![\sigma_{\mathtt{R}}(v)]\!]) \to \prod_{v \in C_\Pi} \mathcal{M}([\![\sigma_{\mathtt{L}}(v)]\!], [\![\sigma_{\mathtt{R}}(v)]\!])\,,$$

which is an arrow of the category of functors from $(\mathcal{M}^{op})^X \times \mathcal{M}^X$ to Set. Notice that, to a certain degree, we are abusing of language, as some circular proof might be considered to be over different systems S and T. If it we need to specify the system S, we can write the more explicit $\lceil \Pi \rceil^S$ (and $\lceil \Pi \rceil^S_\dagger$) in place of $\lceil \Pi \rceil$.

▶ **Theorem 9.** *For each homogeneous circular proof* Π *and each collection of natural transformations* $\{\, \beta_v : [\![\sigma_{\mathtt{L}}(v)]\!] \to [\![\sigma_{\mathtt{R}}(v)]\!] \mid v \in A^{\mathtt{c}}_{\Pi} \,\}$*, the system* $\lceil \Pi_\beta \rceil$ *admits a unique natural solution*

$$\lceil \Pi_\beta \rceil_\dagger : \prod_{v \in A^{\mathtt{s}}_{\Pi}} \mathcal{M}([\![\sigma_{\mathtt{L}}(v)]\!], [\![\sigma_{\mathtt{R}}(v)]\!]) \longrightarrow \prod_{v \in C_\Pi} \mathcal{M}([\![\sigma_{\mathtt{L}}(v)]\!], [\![\sigma_{\mathtt{R}}(v)]\!])\,.$$

The proof of the Theorem mostly depends on the Bekić Lemma, as well as on the following kind of categorical fixed point Lemma. The Lemma can be understood as giving a categorical interpretation to Mendler's style recursion, see [11, §2].

▶ **Lemma 10.** *Let* $\mathsf{W}, \mathsf{C}, \mathsf{D}$ *be three categories, of which* C *has products; let* $F : \mathsf{C} \times \mathsf{W} \to \mathsf{C}$*,* $G : \mathsf{D} \to \mathsf{C}$*,* $Q : \mathsf{C}^{op} \times \mathsf{W}^{op} \times \mathsf{D} \to \mathrm{Set}$ *be functors; let* $\zeta_w : F(\boldsymbol{x}_w, w) \to \boldsymbol{x}_w$ *be a parametrized initial algebra of* F*. Consider an arbitrary natural transformation*

$$\alpha : \mathsf{C}(\_, G) \times Q \to \mathsf{C}(F, G) : \mathsf{C}^{op} \times \mathsf{W}^{op} \times \mathsf{D} \to \mathrm{Set}\,.$$

For each $w \in W$*,* $d \in \mathsf{D}$ *and* $q \in Q(\boldsymbol{x}_w, w, d)$*, there exists a unique* $f_{w,d} : \boldsymbol{x}_w \to Gd$ *which is a solution of the equation*

$$f = \zeta_w^{-1} \cdot \alpha_{\boldsymbol{x}_w, w, d}(f, q)\,.$$

Moreover, the map sending $q \in Q(\boldsymbol{x}_w, w, d)$ *to* $f_{w,d} \in \mathsf{C}(F(\boldsymbol{x}_w, w), Gd)$ *is natural in* w *and* d*.*

Semantics of rooted circular proofs (soundness). Let $\langle \Pi, v \rangle$ be a rooted circular proof with $\sigma(v) = s \vdash t$; we can now define $[\![\Pi, v]\!] : [\![s]\!] \to [\![t]\!]$, the *natural transformation interpreting* $\langle \Pi, v \rangle$ (with respect to a system S), by induction, almost as usual. The induction is now on the well-founded structure of maximal strongly connected components of the underlying graph of Π. The key observation is that if $\mathcal{C}$ is such a non trivial component of Π (i.e. there exists $v, u \in \mathcal{C}$ and a non-null path from v to u), then the restriction of Π to $\mathcal{C}$ can be made into an homogeneous circular proof. More formally, we can define $\Pi \restriction \mathcal{C}$ by choosing $v_0 \in \mathcal{C}$ and putting

$$A_{\mathcal{C}} := \{\, \varsigma_i v \mid v \in \mathcal{C}, \varsigma_i v \notin \mathcal{C} \,\}\,, \qquad \Pi \restriction \mathcal{C} := \overline{\overline{\Pi, v_0}^{A_{\mathcal{C}}}, v_0}\,.$$

The inductive definition is as follows. If v belongs to a trivial component, then we can define the semantics of $\langle \Pi, v \rangle$ as in the non-circular case. Otherwise, we dispose by induction of two collections $\beta = \{\, [\![\Pi, v]\!] \mid v \in A^{\mathtt{c}}_{\Pi \restriction \mathcal{C}} \,\}$ and $\gamma = \{\, [\![\Pi, v]\!] \mid v \in A^{\mathtt{s}}_{\Pi \restriction \mathcal{C}} \,\}$; by Theorem 9 we dispose of a natural transformation $\lceil (\Pi \restriction \mathcal{C})_\beta \rceil_\dagger$ between the appropriate hom-functors; Lemma 7 ensure that we can pointwise apply $\lceil (\Pi \restriction \mathcal{C})_\beta \rceil_\dagger$ to the natural transformations in γ to obtain a new collection of natural transformations indexed by elements of $\mathcal{C}$. We define therefore

$$[\![\Pi, v]\!] := \big(\lceil (\Pi \restriction \mathcal{C})_\beta \rceil_\dagger(\gamma) \big) \cdot \mathtt{pr}_v\,.$$

Fullness. We show that this interpretation of rooted circular proods is *full*. That means that if $\mathcal{M}$ is a free μ-bicomplete category over a set of generators, then, for every arrow f of $\mathcal{M}$, there is a rooted circular proof $\langle \Pi, v \rangle$ such that $[\![\Pi, v]\!] = f$.

The proof of this fact is a lengthy induction. However, the only non-trivial case is the closure of the class of definable arrows under canonical maps from initial algebras (and dually, behaviour maps to final coalgebras); let us exemplify this point. Suppose S is a system with just one bound variable x of maximal priority; if this priority is odd, then

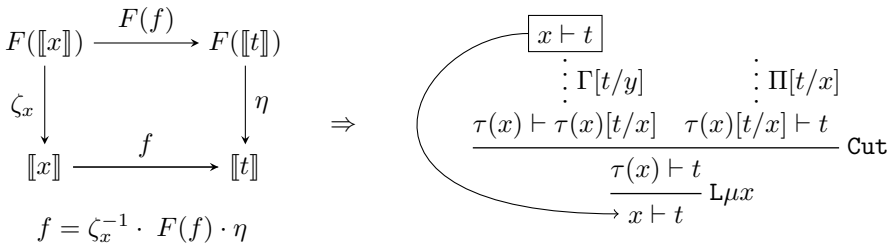


Figure 5 Construction of maps from the initial algebra.

$F = [\![\tau(x)]\!]^{P(S)} : \mathcal{M} \to \mathcal{M}$ while $[\![x]\!]^S$ is the initial F-algebra. Suppose now that we are given a system T which contains an exact copy of $P(S)$, with the exception that the variable x is bound to some term t. If we dispose of a rooted circular proof $\langle \Pi, v\rangle$ on T with $\sigma(v) = \tau(x) \vdash t$, then $[\![\tau(x)]\!]^T = [\![\tau(x)]\!]^{P(S)}([\![t]\!]^T) = F([\![t]\!]^T)$, that is $\eta := [\![\Pi, v]\!] : F([\![t]\!]^T) \to [\![t]\!]^T$ is an F-algebra. We notice that the canonical natural transformation $F_{x,y} : \mathcal{M}(x,y) \to \mathcal{M}(Fx, Fy)$ is definable via a cut-free circular proof Γ (using a language of game theory, Γ is the copycat strategy). Let T' be a system which is a disjoint copy of S and T, with the variable x of T being renamed to y. The circular proof on the right of Figure 5, on the system T', shall then denote the unique algebra morphism from the initial one.

5 Cut elimination

We give in this section an algorithm that, given as input a pointed circular proof $\langle \Pi, v\rangle$, outputs a cut-free pre-proof with possibly infinite branches (yet, a finitely branching tree). A refinement of the technique used to prove Theorem 12 below can also be used to prove that the output tree satisfies the Guard condition 3. This justifies saying that the output tree is an *infinite proof-tree*.

Just like in the classical case for Gentzen's system (see [8], for instance) the procedure consists in "pushing" every cut away from the root. However, in our case, the output tree must be computed with a lazy (outermost) rather than eager (innermost) strategy; this is because not every path in Π leads to a leaf, so that we have to eliminate cuts by performing a breadth-first search of Π from the root. A problem that arises by using this strategy is that we might need to permute a cut with another cut. We temporarily dismiss the problem by merging consecutive cuts together into a sort of n-ary cut.

$$\frac{t_0 \vdash t_1 \quad t_1 \vdash t_2 \quad \dots \quad t_{n-1} \vdash t_n}{t_0 \vdash t_n}\,\texttt{Cut}$$

Therefore, the algorithm grows an output tree whose pending leaves contain objects that can be thought of as n-ary cuts between vertices of Π, waiting to be pushed forward. We call these objects *tapes*.

▶ **Definition 11.** A *tape* is a finite list $M := [u_1, \dots, u_n]$ of vertices of Π such that for all $i = 1 \dots n-1$, $\sigma_{\mathtt{R}}(u_i) = \sigma_{\mathtt{L}}(u_{i+1})$.

Analogously to the behaviour of higher order pushdown automata, the algorithm can also be understood as a non-deterministic automaton disposing of a tape as its internal data structure; the tape can be thought of as a generalized stack. The automaton tries to build up a branch of the proof-tree; when the proof-tree branches, the automaton forks into several automata so to construct all the branches; equivalently, we can think that the automaton

undeterministically chooses which branch to continue constructing. The automaton grows up the branch by means of commutative cut reductions (called here *flips*) at the extremities of the tape; if all the cuts in the tape are principal, the automaton undeterministically chooses one and reduces it, without constructing a new node on the branch. While in principle the construction of a complete branch might fail, due to the fact that we cannot operate flips, we shall see that this does not actually happen.

5.1 Primitive operations

Internal operations. What we call *internal operations* on tapes are functions that take (M, i) as input (with some assumptions on M and i) and return a new tape.

- **Elimination of identities.** The first thing we can do is to eliminate identities since they bring nothing to the semantics. Thus, if $\rho(u_i) = \mathtt{Id}$, we define $\textsc{IdElim}(M, i)$ as the tape obtained by removing u_i from M.

$$\dfrac{\dots \quad t_{i-1} \vdash s \qquad \dfrac{}{s \vdash s}\,\mathtt{Id} \qquad s \vdash t_{i+2} \quad \dots}{t_0 \vdash t_n}\,\mathtt{Cut} \overset{\textsc{IdElim}}{\Longrightarrow} \dfrac{\dots \quad t_{i-1} \vdash s \quad s \vdash t_{i+2} \quad \dots}{t_0 \vdash t_n}\,\mathtt{Cut}$$

- **Merging cuts.** Since the tape represents the fusion of some consecutive cuts, we need an operation for merging new cuts into the tape, thus expanding its size. So if $M = [\dots, u_i, \dots]$ and $\rho(u_i) = \mathtt{Cut}$, we define $\textsc{Merge}(M, i) = [\dots, \varsigma_0 u_i, \varsigma_1 u_i, \dots]$. Schematically:

$$\dfrac{\dots \qquad \dfrac{t_i \vdash s \quad s \vdash t_{i+1}}{t_i \vdash t_{i+1}}\,\mathtt{Cut} \qquad \dots}{t_0 \vdash t_n}\,\mathtt{Cut} \overset{\textsc{Merge}}{\Longrightarrow} \dfrac{\dots \quad t_i \vdash s \quad s \vdash t_{i+1} \quad \dots}{t_0 \vdash t_n}\,\mathtt{Cut}$$

- **Essential reductions.** The last internal operation is a bit more subtle. Suppose $\rho(u_i) \in \mathfrak{R}$ and $\rho(u_{i+1}) \in \mathfrak{L}$ for some i. Note that $t_i := \sigma_{\mathtt{R}}(u_i) = \sigma_{\mathtt{L}}(u_{i+1})$ and that this common term is either a product, a sum, a bound variable or a constant. But $\rho(u_i) \in \mathfrak{R}$ implies $t_i \neq 0$ and $\rho(u_i) \in \mathfrak{R}$ implies $t_i \neq 1$, so t_i is not a constant. Hence there are actually three possible scenarios for the values of $\rho(u_i)$ and $\rho(u_{i+1})$: they can be both product rules, both coproduct rules or both fixpoint rules of the same variable. In each case, we can find successors of u_i and u_{i+1} with compatible sequents. We can then *reduce* u_i with u_{i+1} and substitute them in M with their appropriate successors. More precisely:
  - If $\rho(u_i) = \mathtt{R}\times$, $\rho(u_{i+1}) = \mathtt{L}\times_k$, $k \in \{0, 1\}$, then $\textsc{Reduce}(M, i) = [\dots, \varsigma u_i, \varsigma_k u_{i+1}, \dots]$.

$$\dfrac{\dfrac{t_{i-1} \vdash s_0 \quad t_{i-1} \vdash s_1}{\dots \quad t_{i-1} \vdash s_0 \times s_1}\,\mathtt{R}\times \qquad \dfrac{s_k \vdash t_{i+1}}{s_0 \times s_1 \vdash t_{i+1} \quad \dots}\,\mathtt{L}\times_k}{t_0 \vdash t_n}\,\mathtt{Cut} \overset{\textsc{Reduce}}{\Longrightarrow} \dfrac{\dots \; t_{i-1} \vdash s_k \quad s_k \vdash t_{i+1} \; \dots}{t_0 \vdash t_n}\,\mathtt{Cut}$$

  - If $\rho(u_i) = \mathtt{R}+_k$, $\rho(u_{i+1}) = \mathtt{L}+$, $k \in \{0, 1\}$, then $\textsc{Reduce}(M, i) = [\dots, \varsigma_k u_i, \varsigma u_{i+1}, \dots]$.

$$\dfrac{\dfrac{t_{i-1} \vdash s_k}{\dots \quad t_{i-1} \vdash s_0 + s_1}\,\mathtt{R}+_k \qquad \dfrac{s_0 \vdash t_{i+1} \quad s_1 \vdash t_{i+1}}{s_0 + s_1 \vdash t_{i+1} \quad \dots}\,\mathtt{L}+}{t_0 \vdash t_n}\,\mathtt{Cut} \overset{\textsc{Reduce}}{\Longrightarrow} \dfrac{\dots \; t_{i-1} \vdash s_k \quad s_k \vdash t_{i+1} \; \dots}{t_0 \vdash t_n}\,\mathtt{Cut}$$

  - If $\rho(u_i) = \mathtt{R}\theta x$, $\rho(u_{i+1}) = \mathtt{L}\theta x$, $x \in \mathsf{BD}(S)$, then $\textsc{Reduce}(M, i) = [\dots, \varsigma u_i, \varsigma u_{i+1}, \dots]$.

$$\dfrac{\dfrac{t_{i-1} \vdash \tau(x)}{\dots \quad t_{i-1} \vdash x}\,\mathtt{R}\theta x \qquad \dfrac{\tau(x) \vdash t_{i+1}}{x \vdash t_{i+1} \quad \dots}\,\mathtt{L}\theta x}{t_1 \vdash t_n}\,\mathtt{Cut} \overset{\textsc{Reduce}}{\Longrightarrow} \dfrac{\dots \; t_{i-1} \vdash \tau(x) \quad \tau(x) \vdash t_{i+1} \; \dots}{t_0 \vdash t_n}\,\mathtt{Cut}$$

Productions (external operations, flips). We call *productions* functions that take a tape M as input and return a tuple (r, s, L) where r is a rule name, s is a sequent that will be used to create a new vertex of the output tree, and L is a list of tapes that will be the successors of that new vertex.

The simplest case is when $M = [u]$ with $\rho(u) = \mathtt{Id}$. In that case, let $\textsc{IdOut}(M) = (\mathtt{Id}, \sigma(u), [])$. Otherwise, productions can only happen when there is a left rule on the left of M or a right rule on its right. In these cases, we can perform a *commutative reduction*, or *flip* just like in the classical case.

- If $\rho(u_0) = \mathtt{LAx}$, then $\textsc{LFlip}(M) = (\mathtt{LAx}, 0 \vdash t_n, [])$.

$$\dfrac{\dfrac{}{0 \vdash t_1}\,\mathtt{LAx} \qquad t_1 \vdash t_2 \quad \dots}{0 \vdash t_n}\,\mathtt{Cut} \quad \overset{\textsc{LFlip}}{\Longrightarrow} \quad \dfrac{}{0 \vdash t_n}\,\mathtt{LAx}$$

- If $\rho(u_0) = \mathtt{L}\times_k$ for $k \in \{0, 1\}$, then $\textsc{LFlip}(M) = (\mathtt{L}\times_k, t_0 \vdash t_n, [[\varsigma u_0, u_1 \dots]])$.

$$\dfrac{\dfrac{s_k \vdash t_1}{s_0 \times s_1 \vdash t_1}\,\mathtt{L}\times_k \qquad t_1 \vdash t_2 \quad \dots}{s_0 \times s_1 \vdash t_n}\,\mathtt{Cut} \quad \overset{\textsc{LFlip}}{\Longrightarrow} \quad \dfrac{s_k \vdash t_1 \quad t_1 \vdash t_2 \quad \dots}{\dfrac{s_k \vdash t_n}{s_0 \times s_1 \vdash t_n}\,\mathtt{L}\times_k}\,\mathtt{Cut}$$

- If $\rho(u_0) = \mathtt{L}+$, then $\textsc{LFlip}(M) = (\mathtt{L}+, t_0 \vdash t_n, [[\varsigma_0 u_0, u_1, \dots], [\varsigma_1 u_0, u_1 \dots]])$.

$$\dfrac{\dfrac{s_0 \vdash t_1 \;\; s_1 \vdash t_1}{s_0 + s_1 \vdash t_1}\,\mathtt{L}+ \qquad t_1 \vdash t_2 \;\dots}{s_0 + s_1 \vdash t_n}\,\mathtt{Cut} \quad \overset{\textsc{LFlip}}{\Longrightarrow} \quad \dfrac{s_0 \vdash t_1 \;\; t_1 \vdash t_2 \;\dots}{s_0 \vdash t_n}\,\mathtt{Cut} \quad \dfrac{s_1 \vdash t_1 \;\; t_1 \vdash t_2 \;\dots}{s_1 \vdash t_n}\,\mathtt{Cut}$$
$$\dfrac{s_0 \vdash t_n \qquad\qquad s_1 \vdash t_n}{s_0 + s_1 \vdash t_n}\,\mathtt{L}+$$

- If $\rho(u_0) = \mathtt{L}\theta x$, $x \in \mathsf{BD}(S)$, $\theta \in \{\mu, \nu\}$, then $\textsc{LFlip}(M) = (\mathtt{L}\theta x, x \vdash t_n, [[\varsigma u_0, u_1 \dots]])$.

$$\dfrac{\dfrac{\tau(x) \vdash t_1}{x \vdash t_1}\,\mathtt{L}\theta x \qquad t_1 \vdash t_2 \quad \dots}{x \vdash t_n}\,\mathtt{Cut} \quad \overset{\textsc{LFlip}}{\Longrightarrow} \quad \dfrac{\tau(x) \vdash t_1 \quad t_1 \vdash t_2 \quad \dots}{\dfrac{\tau(x) \vdash t_n}{x \vdash t_n}\,\mathtt{L}\theta x}\,\mathtt{Cut}$$

Right flips are defined dually to left flips, so we present them without the schemas.

- If $\rho(u_n) = \mathtt{RAx}$, then $\textsc{RFlip}(M) = (\mathtt{RAx}, t_0 \vdash 1, [])$.
- If $\rho(u_n) = \mathtt{R}\times$, then $\textsc{RFlip}(M) = (\mathtt{R}\times, t_0 \vdash t_n, [[\dots, u_{n-1}, \varsigma_0 u_n], [\dots u_{n-1}, \varsigma_1 u_n]])$.
- If $\rho(u_n) = \mathtt{R}+_k$ for $k \in \{0, 1\}$, then $\textsc{RFlip}(M) = (\mathtt{R}+_k, t_0 \vdash t_n, [[\dots u_{n-1}, \varsigma u_n]])$.
- If $\rho(u_n) = \mathtt{R}\theta x$, $x \in \mathsf{BD}(S)$, $\theta \in \{\mu, \nu\}$, then $\textsc{RFlip}(M) = (\mathtt{R}\theta x, t_0 \vdash x, [[\dots u_{n-1}, \varsigma u_n]])$.

5.2 The cut-elimination algorithm

In order to produce a segment of the output tree, we need a tape with a left rule on the left or a right rule on the right (call such a tape *reduced*). The *treatment* phase of a tape M consists in executing internal operations until the tape is reduced.

Algorithm 1 defines a tree with transitions labelled by $\{0, 1\}$ along with two mappings $\rho : \Lambda \to \Sigma$, and $\sigma = \Lambda \to \mathsf{SEQ}$ that make it into a possibly infinite proof-tree. In the algorithm, $v \in \Pi$ is fixed and Q is a queue whose elements are of the form (w, M) where $w \in \Lambda$ was previously computed and M is a tape.

It may not be clear that the computation of $\textsc{Treat}(M)$ always halts. After all, once a pair of consecutive nodes in the tape is reduced, it is replaced by another pair of nodes that could, in principle, still be labelled by right and left rules, respectively. Even worse: when a new cut is encountered, the tape grows, thus the number of pairs left to be reduced may enlarge. So why does it stop? It is a consequence of the fact that Π satisfies the Guard condition.

Function $\textsc{Treat}(M)$

while $\rho(\mathrm{fst}(M)) \notin \mathfrak{L}$ ***and*** $\rho(\mathrm{lst}(M)) \notin \mathfrak{R}$ **do**
- **if** $|M| > 1$ ***and*** $\exists i : \rho(u_i) = \texttt{Id}$ **then**
  - $M \leftarrow \textsc{IdElim}(M, i)$;
- **else if** $\exists i : \rho(u_i) = \texttt{Cut}$ **then**
  - $M \leftarrow \textsc{Merge}(M, i)$;
- **else if** $\exists i : \rho(u_{i-1}) \in \mathfrak{R}$ ***and*** $\rho(u_i) \in \mathfrak{L}$ **then**
  - $M \leftarrow \textsc{Reduce}(M, i)$;

return M;

Algorithm 1: Cut-elimination

Initialization: $\Lambda \leftarrow \varnothing$; $Q \leftarrow [(\varepsilon, [v])]$;
while $Q \neq []$ **do**
- $(w, M) \leftarrow \mathrm{pull}(Q)$;
- $\Lambda \leftarrow \Lambda \cup \{w\}$;
- $M \leftarrow \textsc{Treat}(M)$;
- **if** $|M| = 1$ ***and*** $\rho(\mathrm{fst}(M)) = \texttt{Id}$ **then**
  - $(\rho(w), \sigma(w), L) \leftarrow \textsc{IdOut}(M)$;
- **else if** $\rho(\mathrm{fst}(M)) \in \mathfrak{L}$ **then**
  - $(\rho(w), \sigma(w), L) \leftarrow \textsc{LFlip}(M)$;
- **else if** $\rho(\mathrm{lst}(M)) \in \mathfrak{R}$ **then**
  - $(\rho(w), \sigma(w), L) \leftarrow \textsc{RFlip}(M)$;
- **if** $L = [M']$ **then**
  - $\mathrm{push}((w0, M'), Q)$;
- **else if** $L = [M'_0, M'_1]$ **then**
  - $\mathrm{push}((w0, M'_0), Q)$;
  - $\mathrm{push}((w1, M'_1), Q)$;

Figure 6 The cut-elimination algorithm.

▶ **Theorem 12.** *For every input tape M, the computation of* $\textsc{Treat}(M)$ *halts.*

Proof. We suppose, for a contradiction, that there is an entry tape M on which the computation of $\textsc{Treat}(M)$ loops forever. For all $i \geq 1$, let M_i be the tape in memory before the i-th turn of the loop (so that $M = M_1$). Consider that tapes are words and that they are generated from one another according to some context dependent grammar; we wish therefore to consider a sort of infinite parse tree. To that end, we define the *full trace* of the algorithm as the reachable graph $T = \overline{T', (0,0)}$, where T' is a transition system over the alphabet $\mathbb{N} \cup \{\bot\}$ with support $\mathbb{N} \times \mathbb{N}$ and the following transitions:

- For $1 \leq i \leq |M_1|$, $(0,0) \xrightarrow{i} (1,i)$.
- If $M_{n+1} = \textsc{IdElim}(M_n, i)$, then for $k < i$, $(n,k) \xrightarrow{\bot} (n+1,k)$ and for $k > i$, $(n,k) \xrightarrow{\bot} (n+1,k-1)$.
- If $M_{n+1} = \textsc{Merge}(M_n, i)$, then for $k < i$, $(n,k) \xrightarrow{\bot} (n+1,k)$ and for $k > i$, $(n,k) \xrightarrow{\bot} (n+1,k+1)$. Moreover $(n,i) \xrightarrow{1} (n+1,i)$ and $(n,i) \xrightarrow{2} (n+1,i+1)$.
- If $M_{n+1} = \textsc{Reduce}(M_n, i)$, then for $k \in \{i, i+1\}$, $(n,k) \xrightarrow{0} (n+1,k)$ and otherwise $(n,k) \xrightarrow{\bot} (n+1,k)$.

For $(n,k) \in T \setminus (0,0)$, let $g(n,k) \in \Pi$ denote the k-th element of M_n. Basically, the paths in T represent the history of the vertices of Π that occur in the tapes. Transitions labelled by $\bot$ mean that such a vertex has not evolved at a given stage, while the other labels encode the operation that made them evolve. In order to exploit the Guard condition, we shall collapse the transitions labelled by $\bot$ to get a correspondence with paths in Π. We then get the *real trace* Ψ of the algorithm.

It should be clear that Ψ is an infinite, finitely branching labelled tree. The prefix order on such a tree is denoted $\sqsubseteq$ and the lexicographical order is denoted $\preceq$ (see [13]). A maximal (finite or infinite) path in Ψ is called a *branch* and it can be shown that the set of branches

of Ψ ordered lexicographically is a complete lattice. Note that by Kőnig's lemma, the set of infinite branches is nonempty and it is easy to see that its infimum is an infinite branch itself.

Given an infinite branch β, we say that β is a μ-branch (resp. *ν-branch*) if the path Γ in Π formed by the transitions between vertices of β can be written $\Gamma = \Gamma_0 \cdot \Gamma_1$ where Γ_0 is finite, Γ_1 has a left μ-trace (resp. right ν-trace) and every fixpoint rule in Γ_1 occurs infinitely often. Since Π satisfies the Guard condition 3, it follows that every infinite branch β is either a μ-branch or a ν-branch. We shall use the lexicographical order to compare them. Note that by the Guard condition, a μ-branch (resp. ν-branch) can only admit finitely many right (resp. left) cuts.

▶ **Lemma 13.**

1. *The least infinite branch of Ψ is a ν-branch.*
2. *Let E be a nonempty collection of ν-branches and let $\gamma = \bigvee E$. Then γ is a ν-branch.*
3. *If β is a ν-branch, then there exists another ν-branch $\beta' \succ \beta$.*

The key observation to prove Lemma 13 is the following. Recall that each time $M_{n+1} = \textsc{Reduce}(M_n, i)$, a pair (u, v) of elements of Ψ is created, such that $u \prec v$ and $\rho(g(u)) \in \mathfrak{R}$, $\rho(g(v)) \in \mathfrak{L}$ are rules of the same nature (product, coproduct or fixpoint on the same variable). u and v are then called *twins* of each other. Conversely, every $u \in \Psi$ that is not the root, a leaf or a cut occurs in such a pair.

Now, let β_0 be the least infinite branch of Ψ. If β_0 were a μ-branch, then it would admit infinitely many left rules. The twins of those rules would then generate an infinite subtree on the left β_0, contradicting the minimality and proving part 1. For part 2, the case $\gamma \in E$ is trivial, and otherwise, one must analyze how the supremum is computed to observe that infinitely many right cuts are necessary. Therefore, γ must be a ν-branch. Finally, for part 3, it suffices to prove that if β is a ν-branch and β' is a μ-branch, such that $\beta \prec \beta'$ are consecutive, then after a finite time, all the forementioned pairs (u, v) are such that $u \sqsubset \beta$ if and only if $v \sqsubset \beta'$. Therefore, the variables $x \in \mathsf{BD}(S)$ for which the fixpoint rule is applied infinitely often in those two branches are the same. But then, the highest priority of such a variable should be both even and odd, a contradiction.

To conclude, we reach the fundamental contradiction of the proof. Let E be the collection of all the ν-branches. By part 1 of Lemma 13, E is nonempty. Let $\gamma = \bigvee E$. By part 2 of the same Lemma, γ is a ν-branch. Hence by part 3 of the Lemma, there is another ν-branch $\gamma' \succ \gamma$. But then, by definition of E, we should have $\gamma' \in E$ and therefore $\gamma' \preceq \bigvee E = \gamma$. ◀

6 Conclusions and perspectives

We consider this work as a starting point for future research—the more we develop it, the more are the questions. A main motivation for this work was the following problem about the computability by means of initial and final coalgebras: since that all the primitive recursive functions are definable by circular proofs [5] and some function space can be constructed via final coalgebras, can we also define more complex set-theoretic functions such as the Ackermann function? The growing knowledge about hierarchies of infinite trees [4] suggested a concrete way to tackle the problem and encouraged us to pursue this work. A concrete step to be taken is now to compare the expressive power of the calculus with respect to these existing hierarchies. For example, can we simulate higher order pushdown automata by means of our tape automaton?

References

1 D. Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Log.*, 13(1):2, 2012.

2 Y. Bertot and E. Komendantskaya. Inductive and coinductive components of corecursive functions in Coq. *Electr. Notes Theor. Comput. Sci.*, 203(5):25–47, 2008.

3 J. Brotherston and A. Simpson. Sequent calculi for induction and infinite descent. *J. Log. Comput.*, 21(6):1177–1216, 2011.

4 D. Caucal. On infinite transition graphs having a decidable monadic theory. *Theor. Comput. Sci.*, 290(1):79–115, 2003.

5 J. R. B. Cockett and L. Santocanale. Induction, coinduction, and adjoints. *Electr. Notes Theor. Comput. Sci.*, 69:101–119, 2002.

6 J. R. B. Cockett and D. Spencer. Strong categorical datatypes II: A term logic for categorical programming. *Theor. Comput. Sci.*, 139(1&2):69–113, 1995.

7 T. Coquand. Infinite objects in type theory. In H. Barendregt and T. Nipkow, editors, *TYPES*, volume 806 of *Lecture Notes in Computer Science*, pages 62–78. Springer, 1993.

8 R. David, K. Nour, and C. Raffalli. *Introduction à la logique*. Dunod, 2nd edition, 2004.

9 C. Dax, M. Hofmann, and M. Lange. A proof system for the linear time μ-calculus. In S. Arun-Kumar and N. Garg, editors, *FSTTCS*, volume 4337 of *Lecture Notes in Computer Science*, pages 273–284. Springer, 2006.

10 S. Mac Lane. *Categories for the working mathematician*. Springer-Verlag, New York, second edition, 1998.

11 N. P. Mendler. Inductive types and type constraints in the second-order lambda calculus. *Ann. Pure Appl. Logic*, 51(1-2):159–172, 1991.

12 D. Niwinski and I. Walukiewicz. Games for the mu-calculus. *Theor. Comput. Sci.*, 163(1&2):99–116, 1996.

13 D. Perrin and J.-E. Pin. *Infinite Words, Automata, Semigroups, Logic and Games*, volume 141. Elsevier, 2004.

14 G. Rosu and D. Lucanu. Circular coinduction: A proof theoretical foundation. In A. Kurz, M. Lenisa, and A. Tarlecki, editors, *CALCO*, volume 5728 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 2009.

15 L. Santocanale. A calculus of circular proofs and its categorical semantics. Technical Report RS-01-15, BRICS, daimi, May 2001. 30 pp.

16 L. Santocanale. A calculus of circular proofs and its categorical semantics. In M. Nielsen and U. Engberg, editors, *FoSSaCS*, volume 2303 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 2002.

17 L. Santocanale. Free μ-lattices. *J. of Pure and Appl. Algebra*, 168(2-3):227–264, Mar. 2002.

18 L. Santocanale. μ-bicomplete categories and parity games. *Theoretical Informatics and Applications*, 36:195–227, Sept. 2002.

19 T. Studer. On the proof theory of the modal mu-calculus. *Studia Logica*, 89(3):343–363, 2008.

20 I. Walukiewicz. Completeness of Kozen's Axiomatisation of the Propositional μ-Calculus. *Inf. Comput.*, 157(1-2):142–182, 2000.

Hierarchies in independence logic*

Pietro Galliani, Miika Hannula, and Juha Kontinen

University of Helsinki, Department of Mathematics and Statistics,
P.O. Box 68, 00014 Helsinki, Finland
pgallian@gmail.com, {miika.hannula,juha.kontinen}@helsinki.fi

Abstract

We study the expressive power of fragments of inclusion and independence logic defined either by restricting the number of universal quantifiers or the arity of inclusion and independence atoms in formulas. Assuming the so-called lax semantics for these logics, we relate these fragments of inclusion and independence logic to familiar sublogics of existential second-order logic. We also show that, with respect to the stronger strict semantics, inclusion logic is equivalent to existential second-order logic.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Existential second-order logic, Independence logic, Inclusion logic, Expressiveness hierarchies

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.263

1 Introduction

Independence logic [15] and inclusion logic [11] are recent variants of dependence logic. Dependence logic [20] extends first-order logic by dependence atomic formulas

$$=(x_1, \ldots, x_n) \tag{1}$$

the meaning of which is that the value of x_n is completely determined by the values of $x_1, \ldots, x_{n-1}$. The semantics of dependence logic is defined using sets of assignments rather than a single assignment as in first-order logic. Independence logic replaces the dependence atoms by independence atoms $\vec{y} \perp_{\vec{x}} \vec{z}$, the intuitive meaning of which is that, with respect to any fixed value of $\vec{x}$, the variables $\vec{y}$ are totally independent of the variables $\vec{z}$. In inclusion logic dependence atoms are replaced by inclusion atoms $\vec{x} \subseteq \vec{y}$, meaning that all the values of $\vec{x}$ appear also as values for $\vec{y}$. We study the expressive power of the syntactic fragments of these logics defined either by restricting the number of universal quantifiers or the arity of the independence and inclusion atoms in sentences. These results are proved with respect to lax semantics. We also show that, under strict semantics, inclusion logic is equivalent to existential second-order logic ESO while, by a recent result of Hella and Galliani [3], with lax semantics inclusion logic is equivalent to greatest fixed point logic, and hence to LFP (and PTIME) over finite (ordered) structures.

Since the introduction of dependence logic ($\mathcal{D}$) in 2007 many interesting variants of it have been introduced. In fact the team semantics of dependence logic has turned into a general framework for logics in which various notions of dependence and independence can be formalized. Dependence logic has a very intimate and well understood connection to ESO dating back to the results of [17, 8, 22] on Henkin quantifiers. For some of the new variants

* The authors were supported by grant 264917 of the Academy of Finland.

and concepts in this area the correspondence to ESO does not hold. We briefly mention some related work on the complexity theoretic aspects of these logics:

- The extension of dependence logic by so-called intuitionistic implication $\to$ (introduced in [1]) increases the expressive power of dependence logic to full second-order logic [23].
- The model checking problem of full dependence logic, and many of its variants, was recently shown to be NEXPTIME-complete. In fact, for any variant of dependence logic whose atoms are PTIME-computable, the corresponding model checking problem is contained in NEXPTIME [13].
- The non-classical interpretation of disjunction in dependence logic has the effect that the model checking problems of $\phi_1 := =(x,y) \vee =(u,v)$ and $\phi_2 := =(x,y) \vee =(u,v) \vee =(u,v)$ are already NL-complete and NP-complete, respectively [18].

While dependence logic and independence logic are both equivalent to ESO in expressive power [20, 15], for inclusion logic only containment in ESO has been shown [11]. Furthermore, the expressive power of various natural syntactic fragments of independence and inclusion logics is not understood at the moment. The starting point of our work were the results of [7] on the fragments $\mathcal{D}(k\forall)$ and $\mathcal{D}(k\text{-dep})$ of dependence logic. The fragment $\mathcal{D}(k\forall)$ contains those $\mathcal{D}$-formulas in which at most k variables have been universally quantified, and in the formulas of $\mathcal{D}(k\text{-dep})$ dependence atoms of arity at most k may appear (atoms of the form $=(x_1,\dots,x_n)$ satisfying $n \leq k+1$). The following results were shown in [7]:

1. $\mathcal{D}(k\text{-dep}) = \mathrm{ESO}_f(k\text{-ary})$,
2. $\mathcal{D}(k\forall) \leq \mathrm{ESO}_f(k\forall) \leq \mathcal{D}(2k\forall)$

where $\mathrm{ESO}_f(k\text{-ary})$ is the fragment of ESO in which the quantified functions and relations have arity at most k, and $\mathrm{ESO}_f(k\forall)$ consists of ESO-sentences that are in Skolem Normal Form and contain at most k universal first-order quantifiers. The equivalence in (1) was used to show that in $\mathcal{D}(k\text{-dep})$ even cardinality of a $k+1$-ary relation cannot be expressed using the result of Ajtai [2]. On the other hand, since

$$\mathrm{ESO}_f(k\forall) = \mathrm{NTIME}_{\mathrm{RAM}}(n^k) < \mathrm{NTIME}_{\mathrm{RAM}}(n^{k+1})$$

by [14] and [6], an infinite expressivity hierarchy for the fragments $\mathcal{D}(k\forall)$ was shown using 2. Above $\mathrm{NTIME}_{\mathrm{RAM}}(n^k)$ denotes the family of classes of τ-structures that can be recognized by a non-deterministic RAM in time $O(n^k)$.

In [11] it was observed that independence logic and inclusion logic can be given two alternative semantics called strict and lax semantics. For dependence logic these two semantics coincide in the sense that the meaning of any $\mathcal{D}$-formula is the same under both interpretations. For independence and inclusion logic formulas this is not the case as shown in [11]. In fact, we will show that, with respect to strict semantics, inclusion logic is equivalent to ESO, while by a recent result of Hella and Galliani [3], with lax semantics inclusion logic is equivalent to greatest fixed point logic. In the rest of the article we consider the expressive power of fragments of independence logic and inclusion logic with lax semantics. First we look at fragments defined analogously to $\mathcal{D}(k\text{-dep})$ of dependence logic. We let $\mathrm{FO}(\perp_{\mathrm{c}})(k\text{-ind})$ contain those independence logic sentences in which independence atoms with at most $k+1$ different variables may appear. Similarly in the sentences of $\mathrm{FO}(\subseteq)(k\text{-inc})$ only inclusion atoms of the form $\vec{a} \subseteq \vec{b}$, where $|\vec{a}| = |\vec{b}| \leq k$ may appear. Our results show that

$$\mathrm{FO}(\subseteq)(k\text{-inc}) \leq \mathrm{ESO}_f(k\text{-ary}) = \mathrm{FO}(\perp_{\mathrm{c}})(k\text{-ind}).$$

Then we consider the analogoues of $\mathcal{D}(k\forall)$ in the case of $\mathrm{FO}(\perp) = \mathrm{FO}(\perp_{\mathrm{c}})$ [21], which is the sublogic of independence logic allowing only so-called pure atoms $\vec{y} \perp \vec{z}$, and $\mathrm{FO}(\perp, \subseteq)$. We show that

- $\mathrm{FO}(\bot)(2\forall) = \mathrm{FO}(\bot)$,
- $\mathrm{FO}(\bot, \subseteq)(1\forall) = \mathrm{FO}(\bot, \subseteq)$.

This article is organized as follows. In Section 2 we review some basic properties and results regarding dependence logic and its variants. In Section 3 we compare the strict and lax semantics and in Section 4.1 relate the arity fragments of independence logic and inclusion logic with that of ESO. Finally, in Section 4.2 we consider fragments defined by restricting the number of universally quantified variables.

2 Preliminaries

2.1 Team Semantics

Team semantics is a generalization of Tarski semantics in which formulas are interpreted by *sets* of assignments, called *teams*, rather than by single assignments. In this subsection, we will recall the definition of team semantics for first order logic. We will assume that all our formulas are in negation normal form. Also, all structures considered in the paper are assumed to have at least two elements.

▶ **Definition 2.** *Let $\mathcal{M}$ be a first-order model and V a finite set of variables. Then*

- *a* team *X over $\mathcal{M}$ with domain $Dom(X) = V$ is a finite set of assignments from V to the domain M of $\mathcal{M}$;*
- *for a tuple $\vec{v}$ of variables in V, we write $X(\vec{v})$ for the set $\{s(\vec{v}) : s \in X\}$ of all values that $\vec{v}$ takes in X, where $s(\vec{v}) := (s(v_1), \ldots, s(v_n))$;*
- *for a subset W of V, we write $X \upharpoonright W$ for the team obtained by restricting all assignments of X to the variables in W.*
- *For a formula ϕ, the set of free variables of ϕ is denoted by* $\mathrm{Fr}(\phi)$.

There exist two variants of team semantics, called respectively *strict* and *lax*, which differ with respect to the interpretation of disjunction and existential quantification. Informally speaking, the choice between strict and lax semantics corresponds to the choice between disallowing or allowing *nondeterministic strategies* in the corresponding semantic games.[1]

We first give the definition of the lax version of team semantics; later, we will discuss some of the ways in which strict semantics differs from it.

▶ **Definition 3** (Team Semantics). *Let $\mathcal{M}$ be any first-order model and let X be any team over it. Then*

TS-lit: *For all first-order literals α, $\mathcal{M} \models_X \alpha$ if and only if, for all $s \in X$, $\mathcal{M} \models_s \alpha$ in the usual Tarski semantics sense;*

TS-$\vee$: *For all ψ and θ, $\mathcal{M} \models_X \psi \vee \theta$ if and only if $X = Y \cup Z$ for two subteams Y and Z such that $\mathcal{M} \models_Y \psi$ and $\mathcal{M} \models_Z \theta$;*

TS-$\wedge$: *For all ψ and θ, $\mathcal{M} \models_X \psi \wedge \theta$ if and only if $\mathcal{M} \models_X \psi$ and $\mathcal{M} \models_X \theta$;*

TS-$\exists$: *For all ψ and all variables v, $\mathcal{M} \models_X \exists v\psi$ if and only if there exists a function $H : X \to \mathcal{P}(M)\backslash\{\emptyset\}$ such that $\mathcal{M} \models_{X[H/v]} \psi$, where $X[H/v] = \{s[m/v] : s \in X, m \in H(s)\}$;*

TS-$\forall$: *For all ψ and all variables v, $\mathcal{M} \models_X \forall v\psi$ if and only if $\mathcal{M} \models_{X[M/v]} \psi$, where $X[M/v] = \{s[m/v] : s \in X, m \in M\}$.*

[1] See [10] and [13] for details.

If $\mathcal{M} \models_X \phi$, we say that X satisfies *ϕ in $\mathcal{M}$; and if a sentence (that is, a formula with no free variables) ϕ is satisfied by the team $\{\emptyset\}$,*[2] *we say that ϕ is* true *in $\mathcal{M}$ and we write $\mathcal{M} \models \phi$.*

In the team semantics setting, formulas ϕ and ψ are said to be logically equivalent, $\phi \equiv \psi$, if for all models $\mathcal{M}$ and teams X, with $\mathrm{Fr}(\phi) \cup \mathrm{Fr}(\psi) \subseteq \mathrm{Dom(X)}$, $\mathcal{M} \models_X \phi \Leftrightarrow \mathcal{M} \models_X \psi$. Logics $\mathcal{L}$ and $\mathcal{L}'$ are said to be equivalent, $\mathcal{L} = \mathcal{L}'$, if every $\mathcal{L}$-sentence ϕ is equivalent to some $\mathcal{L}'$-sentence ψ, and vice versa.

The following result can be proved by structural induction on the formula ϕ:

▶ **Theorem 4** (Flatness). *For all first order formulas ϕ and all suitable models $\mathcal{M}$ and teams X, the following are equivalent:*

1. *$\mathcal{M} \models_X \phi$;*
2. *For all $s \in X$, $\mathcal{M} \models_{\{s\}} \phi$;*
3. *For all $s \in X$, $\mathcal{M} \models_s \phi$ according to Tarski semantics.*

2.2 Dependencies in Team Semantics

The advantage of team semantics, and the reason for its development, is that it allows us to extend first-order logic by new atoms and operators. For the purposes of this paper, the following atoms, inspired by database-theoretic *dependency notions*[3], are of particular interest:

▶ **Definition 5.** ▬ *Let $\vec{x}$ be a tuple of variables and let y be another variable. Then $=(\vec{x}, y)$ is a* dependence atom*, with the semantic rule*

TS-dep: *$\mathcal{M} \models_X =(\vec{x}, y)$ if and only if any two $s, s' \in X$ which assign the same value to $\vec{x}$ also assign the same value to y;*

▬ *Let $\vec{x}$, $\vec{y}$, and $\vec{z}$ be tuples of variables (not necessarily of the same length). Then $\vec{y} \perp_{\vec{x}} \vec{z}$ is a* conditional independence atom*, with the semantic rule*

TS-ind: *$\mathcal{M} \models_X \vec{y} \perp_{\vec{x}} \vec{z}$ if and only if for any two $s, s' \in X$ which assign the same value to $\vec{x}$ there exists a $s'' \in X$ which agrees with s with respect to $\vec{x}$ and $\vec{y}$ and with s' with respect to $\vec{z}$.*

Furthermore, we will write $\vec{x} \perp \vec{y}$ as a shorthand for $\vec{x} \perp_\emptyset \vec{y}$, and call it a pure independence atom*;*

▬ *Let $\vec{x}$ and $\vec{y}$ be two tuples of variables of the same length. Then $\vec{x} \subseteq \vec{y}$ is an* inclusion atom*, with the semantic rule*

TS-inc: *$\mathcal{M} \models_X \vec{x} \subseteq \vec{y}$ if and only if $X(\vec{x}) \subseteq X(\vec{y})$;*

Given a collection $\mathcal{C} \subseteq \{=(\ldots), \perp_c, \subseteq\}$ of atoms, we will write $\mathrm{FO}(\mathcal{C})$ (omitting the set parenthesis of $\mathcal{C}$) for the logic obtained by adding them to the language of first-order logic. With this notation dependence logic, independence logic and inclusion logic are denoted by $\mathrm{FO}(=(\ldots))$, $\mathrm{FO}(\perp_c)$ and $\mathrm{FO}(\subseteq)$, respectively. We will also write $\mathrm{FO}(\perp)$ for the fragment of independence logic containing only pure independence atoms.

---

[2] $\{\emptyset\}$ is the team containing the empty assignment. Of course, this is different from the *empty team* $\emptyset$, containing no assignments.

[3] More precisely, dependence atoms correspond to functional dependencies [4], independence atoms to embedded multivalued dependencies and conditional dependency conditions as in [12, 19], and inclusion atoms to inclusion dependencies [9, 5].

All formulas of all the above-mentioned logics satisfy the two following properties:

▶ **Proposition 6** (Empty Team Property). *For all models $\mathcal{M}$ and $\phi \in \mathrm{FO}(=(\ldots), \bot_c, \subseteq)$ over the signature of $\mathcal{M}$, $\mathcal{M} \models_\emptyset \phi$.*

▶ **Proposition 7** (Locality). *Let ϕ be a formula of $\mathrm{FO}(=(\ldots), \bot_c, \subseteq)$ whose free variables $\mathrm{Fr}(\phi)$ are contained in V. Then, for all models $\mathcal{M}$ and teams X, $\mathcal{M} \models_X \phi$ if and only if $\mathcal{M} \models_{X\restriction V} \phi$.*

Furthermore, we have the two following results for dependence logic:

▶ **Proposition 8** (Downwards Closure). *For all models $\mathcal{M}$, dependence logic formulas ϕ and teams X, if $\mathcal{M} \models_X \phi$ then $\mathcal{M} \models_Y \phi$ for all $Y \subseteq X$.*

▶ **Theorem 9** ([22, 8, 20]). *Any dependence logic sentence ϕ is logically equivalent to some* ESO *sentence ϕ^*, and vice versa.*

What about independence logic? As shown in [15], a dependence atom $=(\vec{x}, y)$ is logically equivalent to the independence atom $y \bot_{\vec{x}} y$, and, since independence logic is clearly contained in ESO, we have at once that

▶ **Theorem 10** ([15]). *Any independence logic sentence ϕ is logically equivalent to some* ESO *sentence ϕ^*, and vice versa.*

Furthermore,

▶ **Theorem 11** ([21]). *Any independence logic formula is equivalent to some pure independence logic formula.*

For inclusion logic the following is known.

▶ **Theorem 12.**

1. *An inclusion atom $\vec{x} \subseteq \vec{y}$ is equivalent to the* $\mathrm{FO}(\bot)$ *expression*

$$\forall v_1 v_2 \vec{z}((\vec{z} \neq \vec{x} \wedge \vec{z} \neq \vec{x}) \vee (v_1 \neq v_2 \wedge \vec{z} \neq \vec{y}) \vee ((v_1 = v_2 \vee \vec{z} = \vec{y}) \wedge \vec{z} \bot v_1 v_2))$$

   where v_1, v_2 and $\vec{z}$ are new variables [11].
2. *Any inclusion logic sentence ϕ is logically equivalent to some* positive greatest fixpoint logic *sentence ϕ^*, and vice versa [3].*

We conclude this subsection with two novel results, a characterization of *dependence* in terms of pure independence and a *prenex normal form theorem* for formulas of our logics.

▶ **Theorem 13.** *For all models $\mathcal{M}$ and teams X*

$$\mathcal{M} \models_X =(\vec{x}, y) \Leftrightarrow \mathcal{M} \models_X \forall \vec{z} \exists w((\vec{z} = \vec{x} \to w = y) \wedge \vec{x} y \bot \vec{z} w).$$

▶ **Lemma 14.** *Let $\phi, \psi \in \mathrm{FO}(=(\ldots), \bot_c, \subseteq)$ and let x be a variable not occurring free in ψ. Then the following equivalences hold:*

1. $\exists x\phi \wedge \psi \equiv \exists x(\phi \wedge \psi)$,
2. $\exists x\phi \vee \psi \equiv \exists x(\phi \vee \psi)$,
3. $\forall x\phi \wedge \psi \equiv \forall x(\phi \wedge \psi)$,
4. $\forall x\phi \vee \psi \equiv \exists a \exists b \forall x((\phi \wedge a = b) \vee (\psi \wedge a \neq b))$ *where a and b are new variables.*

Lemma 14 allows us to show the following.

▶ **Theorem 15.** *Any formula $\phi \in \mathrm{FO}(=(\ldots), \bot_c, \subseteq)$ is logically equivalent to some formula ϕ' such that*

1. *ϕ' is of the form $Q_1 x_1 \ldots Q_k x_k \psi$, where ψ is quantifier-free;*
2. *Any literal or non-first-order atom which occurs in ϕ' occurred already in ϕ;*
3. *The number of universal quantifiers in ϕ' is the same as the number of universal quantifiers in ϕ.*

3 Comparing strict and lax semantics

As we mentioned, there exists an alternative variant of lax semantics, called strict semantics. It differs from lax semantics in the definition of the semantic rules for disjunction and existential quantification, which are replaced respectively by

STS-∨: For all ψ and θ, $\mathcal{M} \models_X \psi \vee \theta$ if and only if Y and Z exist such that $Y \cup Z = X$, $Y \cap Z = \emptyset$, $\mathcal{M} \models_Y \psi$ and $\mathcal{M} \models_Z \theta$;

STS-∃: For all ψ and all variables v, $\mathcal{M} \models_X \exists v\psi$ if and only if there exists a function $F : X \to M$ such that $\mathcal{M} \models_{X[F/v]} \psi$, where $X[F/v] = \{s[F(s)/v] : s \in X\}$.

In the original version of dependence logic lax disjunction and strict existential quantification were used [20]. However, since dependence logic is downwards closed, it does not make any difference whether strict or lax version of disjunction (or existential quantification) is used. In general the following holds.

▶ **Proposition 16.** *If $\mathcal{M} \models_X \phi$ according to strict team semantics, then $\mathcal{M} \models_X \phi$ according to lax team semantics.*

For downwards closed logics, such as dependence logic, the converse is then also true.

▶ **Proposition 17** ([11])**.** *For all dependence logic formulas ϕ, models $\mathcal{M}$ and teams X, $\mathcal{M} \models_X \phi$ holds wrt strict team semantics if and only if it holds wrt lax team semantics.*

However, the same is false for both inclusion logic and independence logic. In particular, as we will now see, inclusion logic with strict semantics is equivalent to full existential second order logic, in contrast with the second item of Theorem 12.

By Theorem 9, it suffices to show that every dependence logic sentence is equivalent to some inclusion logic sentence (with strict semantics). In order to do so, we will use the following *normal form theorem* from [20]:

▶ **Theorem 18** ([20])**.** *Every dependence logic sentence is equivalent to some sentence of the form*

$$\phi := \forall\vec{x}\exists\vec{y}\left(\bigwedge_{y_i \in \vec{y}} =(\vec{v_i}, \vec{y_i}) \wedge \theta\right) \tag{19}$$

where for all i, $\vec{v_i}$ is contained in $\vec{x}$ and where θ is a quantifier-free first-order formula.

As we will now show, in strict semantics the dependence atoms in (19) can be replaced by equivalent inclusion logic subformulas; and, therefore, it follows at once that (strict) inclusion logic is equivalent to dependence logic (and, therefore, to ESO) over sentences.

▶ **Definition 20.** *Let $\mathcal{M}$ be a model and X a team, and let $\vec{x}$ be a tuple of variables in its domain. We say that X is* $\vec{x}$-universal *if for all tuples of elements $\vec{m}$ with $|\vec{m}| = |\vec{x}|$, there exists one and only one $s \in X$ with $s(\vec{x}) = \vec{m}$.*

▶ **Lemma 21.** *If X is of the form $\{\emptyset\}[M/\vec{x}][\vec{F}/\vec{y}]$ then X is $\vec{x}$-universal.*

Proof. Obvious (but note that if the $\vec{F}$ were replaced by *nondeterministic* choice functions $\vec{H}$, as in the case of the lax semantics, this would not hold). ◀

▶ **Proposition 22.** *Let $\mathcal{M}$ be a model and X a $\vec{x}$-universal team. Suppose also that $y \notin \vec{x}$, $\vec{v} \subseteq \vec{x}$, and $\vec{w} = \vec{x} \backslash \vec{v}$ (that is, $\vec{w}$ lists, without repetitions, all variables occurring in $\vec{x}$ but not in $\vec{v}$). Then*

$$\mathcal{M} \models_X =(\vec{v}, y) \Leftrightarrow \mathcal{M} \models_X \forall\vec{q}(\vec{q}\vec{v}y \subseteq \vec{w}\vec{v}y).$$

Proof. Suppose that $\mathcal{M} \models_X =(\vec{v}, y)$, and let $h = s[\vec{m}/\vec{q}] \in X[M/\vec{q}]$, where $s \in X$. Since X is $\vec{x}$-universal and $\vec{x} = \vec{v} \cup \vec{w}$, there exists an assignment $s' \in X$ such that $s'(\vec{w}) = \vec{m}$ and $s'(\vec{v}) = s(\vec{v})$. Since y is a function of $\vec{v}$ alone, this implies that $s'(y) = s(y)$. Finally, $h' = s'[\vec{m}/\vec{q}] \in X[M/\vec{q}]$, and $h'(\vec{w}\vec{v}y) = \vec{m}s(\vec{v}y) = h(\vec{q}\vec{v}y)$, as required.

Conversely, suppose that $\mathcal{M} \models_X \forall\vec{q}(\vec{q}\vec{v}y \subseteq \vec{w}\vec{v}y)$, and let $s, s' \in X$ be such that $s(\vec{v}) = s'(\vec{v})$. Now let $\vec{m} = s'(\vec{w})$, and consider $h = s[\vec{m}/\vec{q}] \in X[M/\vec{q}]$. By hypothesis, there exists a $h' \in X[M/\vec{q}]$ such that $h'(\vec{w}) = h(\vec{q}) = \vec{m}$ and $h'(\vec{v}y) = h(\vec{v}y) = s(\vec{v}y)$. This h' is of the form $s''[\vec{m'}/\vec{q}]$ for some $s'' \in X$; and for this s'', we have that $s''(\vec{v}) = s(\vec{v}) = s'(\vec{v})$, $s''(\vec{w}) = \vec{m} = s'(\vec{w})$ and $s''(y) = s(\vec{y})$. Now, $\vec{x} = \vec{v} \cup \vec{w}$, and s'' coincides with s' over it, and X is $\vec{x}$-universal; therefore, we have to conclude that $s'' = s'$. But then $s'(y) = s''(y) = s(y)$, and therefore s'' and s coincide over y too. ◀

▶ **Corollary 23.** *With strict semantics inclusion logic is equivalent to* ESO.

Proof. By Lemma 21 and the Proposition 22, any sentence of the form (19) can be expressed in inclusion logic as

$$\forall\vec{x}\exists\vec{y}\left(\bigwedge_{y_i \in \vec{y}} (\forall\vec{q}_i(\vec{q}_i\vec{v}_i y \subseteq \vec{w}_i\vec{v}_i y)) \wedge \theta\right) \tag{24}$$

where for all i, $\vec{w}_i = \vec{x}\backslash\vec{v}_i$; and this implies our result. ◀

The analogue of Theorem 7 (locality) for inclusion logic with strict semantics fails. As an especially surprising example of such an failure we now show that one can find inclusion logic sentences that count the number of assignments in a team:

▶ **Theorem 25.** *For each natural number n there is a sentence $\phi \in \mathrm{FO}(\subseteq)$ such that for all models $\mathcal{M}$ and teams X where $X \neq \emptyset$ and the variables in $Dom(X)$ do not appear in ϕ,*

$\mathcal{M} \models_X \phi$ *if and only if* $|X| \geq n$.

The failure of locality in non-downwards closed logics with strict semantics is somewhat problematic, as it causes the interpretation of a formula to depend on the values that our assignments take on variables which do *not* occur in it. As a consequence, in the rest of this work we will focus on logics with lax semantics.

4 The expressive power of fragments

The purpose of this section is to generalize the classification of the expressive power of fragments of dependence logic of [7] to the case of other variants (with respect to lax semantics). We will consider the following fragments.

▶ **Definition 26.** *Let $\mathcal{C}$ be a subset of $\{=(\ldots), \perp_c, \perp, \subseteq\}$ and let $k \in \mathbb{N}$. Then*

1. $\mathrm{FO}(\mathcal{C})(k{-}dep)$ *is the class of sentences of* $\mathrm{FO}(\mathcal{C})$ *in which dependence atoms of the form $=(\vec{z}, y)$, where $\vec{z}$ is of length at most k, may appear.*
2. $\mathrm{FO}(\mathcal{C})(k{-}ind)$ *is the class of sentences of* $\mathrm{FO}(\mathcal{C})$ *in which independence atoms of the form $\vec{y}\perp_{\vec{x}}\vec{z}$, where $\vec{x}\vec{y}\vec{z}$ has at most $k+1$ distinct variables, may appear.*
3. $\mathrm{FO}(\mathcal{C})(k{-}inc)$ *is the class of sentences of* $\mathrm{FO}(\mathcal{C})$ *in which inclusion atoms of the form $\vec{a} \subseteq \vec{b}$, where $\vec{a}$ and $\vec{b}$ are of length at most k, may appear.*
4. $\mathrm{FO}(\mathcal{C})(k\forall)$ *is the class of sentences of* $\mathrm{FO}(\mathcal{C})$ *in which at most k universal quantifiers occur.*

As in [7], we will write $\mathcal{D}(k\text{-dep})$ and $\mathcal{D}(k\forall)$ for $\mathrm{FO}(=(\ldots))(k{-}\mathrm{dep})$ and $\mathrm{FO}(=(\ldots))(k\forall)$, respectively.

4.1 Arity hierarchies

In this section we will prove that $\mathrm{FO}(\perp_c)(k\text{-ind}) = \mathrm{ESO}_f(k\text{-ary})$. In particular this also implies that $\mathrm{FO}(\perp_c)(k\text{-ind}) = \mathcal{D}(k\text{-dep})$ [7]. We will also prove that $\mathrm{FO}(\subseteq)(k\text{-inc}) \leq \mathrm{ESO}_f(k\text{-ary})$. The direction from $\mathrm{ESO}_f(k\text{-ary})$ to $\mathrm{FO}(\perp_c)(k\text{-ind})$ is straightforward.

▶ **Proposition 27.** $\mathrm{ESO}_f(k\text{-ary}) \leq \mathrm{FO}(\perp_c)(k\text{-ind})$.

Proof. Let $\phi \in \mathrm{ESO}_f(k\text{-ary})$. By [7] there exists a $\phi' \in \mathcal{D}(k\text{-dep})$ equivalent to ϕ and of the form

$$Q^1x_1 \dots Q^m x_m \exists y_1 \dots \exists y_n(\bigwedge_{1\leq j\leq n} =(\vec{z}_j, y_j) \wedge \theta)$$

where $\vec{z}_j$, for $1 \leq j \leq n$, is a sequence of length at most k. By [15] each dependence atom $=(\vec{z}, y)$ is equivalent to the independence atom $y \perp_{\vec{z}} y$. Therefore we can present ϕ' in the following independence logic form

$$Q^1x_1 \dots Q^m x_m \exists y_1 \dots \exists y_n(\bigwedge_{1\leq j\leq n} y_j \perp_{\vec{z}_j} y_j \wedge \theta)$$

where $\vec{z}_j y_j$, for $1 \leq j \leq n$, is a sequence of at most $k+1$ different variables. ◀

We will next show the other direction.

▶ **Lemma 28.** *Let $\vec{b} \perp_{\vec{a}} \vec{c}$ be an independence atom where $\vec{a}$, $\vec{b}$ and $\vec{c}$ are tuples of variables. If $\vec{b_0}$ lists the variables in $\vec{b} - \vec{a} \cup \vec{c}$, $\vec{c_0}$ lists the variables in $\vec{c} - \vec{a} \cup \vec{b}$, and $\vec{d}$ lists the variables in $\vec{b} \cap \vec{c} - \vec{a}$, then*

$$\vec{b} \perp_{\vec{a}} \vec{c} \equiv \vec{b_0} \perp_{\vec{a}} \vec{c_0} \wedge \bigwedge_{d\in\vec{d}} =(\vec{a}, d).$$

Now we can prove the following proposition. In the proof we will present a translation from independence logic to ESO, where independence atoms are coded by relation variables preserving the arity of the atoms. Note that the translation presented in [15] does not preserve this property.

▶ **Proposition 29.** $\mathrm{FO}(\perp_c)(k\text{-ind}) \leq \mathrm{ESO}_f(k\text{-ary})$.

Proof. Let $\phi \in \mathrm{FO}(\perp_c)(k\text{-ind})$. By Theorem 15 we may assume that ϕ is in prenex normal form $Q^1x_1 \dots Q^n x_n \theta$ where θ is a quantifier-free formula. By Lemma 28 we may assume that each independence atom in θ is either of the form $=(\vec{z}, y)$ or $\vec{b} \perp_{\vec{a}} \vec{c}$ where

- y is not listed in $\vec{z}$,
- $\vec{a}$, $\vec{b}$ and $\vec{c}$ do not share any variables,
- $|\vec{z}| \leq k$ and $|\vec{a}\vec{b}\vec{c}| \leq k+1$.

Let us next consider the subformulas of θ. We will enumerate the subformulas of θ by $\theta_{\vec{i}}$ where $\vec{i}$ is a binary sequence encoding the location of the subformula in θ. Let $\theta_\lambda := \theta$ where λ is the empty sequence. If $\theta_{\vec{i}}$ is a conjunction (or a disjunction), then we denote its conjuncts (or the disjuncts) as $\theta_{\vec{i}0}$ and $\theta_{\vec{i}1}$. Now let $S := \{\vec{i} \mid \theta_{\vec{i}} \text{ is a subformula of } \theta\}$, and let D and I be the subsets of S consisting of sequences $\vec{i}$ for which $\theta_{\vec{i}}$ is a dependence atom or an independence atom, respectively. Let $\leq$ be a partial order in S where $\vec{i} \leq \vec{j}$ if $\vec{i}\vec{k} = \vec{j}$ for some binary $\vec{k}$. Then $\vec{i} \leq \vec{j}$ if and only if $\theta_{\vec{j}}$ is a subformula of $\theta_{\vec{i}}$.

Next we will define a $\Phi \in \mathrm{ESO}_f(k\text{-ary})$ equivalent to ϕ. First we define $\varphi_{\vec{i}}$ for each $\vec{i} \in S$ inductively as follows:

- $\varphi_{\vec{i}} := \theta_{\vec{i}}$ if $\theta_{\vec{i}}$ is a first-order atom,

- $\varphi_{\vec{i}} := S_{\vec{i}}(\vec{a}\vec{b}) \wedge T_{\vec{i}}(\vec{a}\vec{c})$ if $\theta_{\vec{i}}$ is $\vec{b} \perp_{\vec{a}} \vec{c}$,
- $\varphi_{\vec{i}} := f_{\vec{i}}(\vec{z}) = y$ if $\theta_{\vec{i}}$ is $=(\vec{z}, y)$,
- $\varphi_{\vec{i}} := \varphi_{\vec{i}0} \wedge \varphi_{\vec{i}1}$ if $\theta_{\vec{i}}$ is $\theta_{\vec{i}0} \wedge \theta_{\vec{i}1}$,
- $\varphi_{\vec{i}} := \varphi_{\vec{i}0} \vee \varphi_{\vec{i}1}$ if $\theta_{\vec{i}}$ is $\theta_{\vec{i}0} \vee \theta_{\vec{i}1}$.

Now let $\varphi := \varphi_\lambda$. Then φ is a quantifier-free first-order formula sharing the structure of θ where the dependence and independence atoms are interpreted using new function symbols $f_{\vec{i}}$ and relation symbols $S_{\vec{i}}$ and $T_{\vec{i}}$, respectively. Let $\vec{z}_{\vec{i}}$, for $\vec{i} \in I$, list the variables in $\{x_1, \ldots, x_n\} \setminus \mathrm{Fr}(\theta_{\vec{i}})$. In the following, for example, $\exists(S_{\vec{i}})_{\vec{i}\in I}$ denotes the prefix $\exists S_{\vec{i}_1} \ldots \exists S_{\vec{i}_m}$ where $\vec{i}_1, \ldots, \vec{i}_m$ enumerates I. So let us define Φ as

$$\exists(S_{\vec{i}})_{\vec{i}\in I}(T_{\vec{i}})_{\vec{i}\in I}(f_{\vec{i}})_{\vec{i}\in D}(Q^1 x_1 \ldots Q^n x_n \varphi \wedge \Omega) \tag{30}$$

where

$$\Omega := \bigwedge_{\vec{i}\in I} [\forall \vec{a}\vec{b}\vec{c}(S_{\vec{i}}(\vec{a}\vec{b}) \wedge T_{\vec{i}}(\vec{a}\vec{c})) \to \exists \vec{z}_{\vec{i}}(\bigwedge_{\vec{j}\leq\vec{i}} \varphi_{\vec{j}} \wedge Q^1 x'_1 \ldots Q^n x'_n(\varphi' \wedge \chi))] \tag{31}$$

where $\varphi' := \varphi(x'_1/x_1) \ldots (x'_n/x_n)$ and

$$\chi := \bigwedge_{\substack{1\leq k\leq n \\ Q^k=\exists}} (x_1 = x'_1 \wedge \ldots \wedge x_{k-1} = x'_{k-1}) \to x_k = x'_k. \tag{32}$$

The idea behind Φ is that the relation variables $S_{\vec{i}}$ and $T_{\vec{i}}$, for $\vec{i} \in I$, encode a subteam $X_{\vec{i}}$ that satisfies $\vec{b} \perp_{\vec{a}} \vec{c}$. Then Ω will ensure that for each $s, s' \in X_{\vec{i}}$ with $s(\vec{a}) = s'(\vec{a})$ there is s'' corresponding to the values of $\vec{a}\vec{b}\vec{c}$ and $\vec{z}_{\vec{i}}$ such that $s''(\vec{a}\vec{b}\vec{c}) = s(\vec{a}\vec{b})s'(\vec{b})$. The variables x'_i and χ will ensure that $s'' \in X_{\vec{i}}$. We will now prove that

$$\mathcal{M} \models \phi \Leftrightarrow \mathcal{M} \models \Phi.$$

Only if-part: Assume that $\mathcal{M} \models \phi$. Then there are functions

$$F_i : X[F_1/x_1] \ldots [F_{i-1}/x_{i-1}] \to \mathcal{P}(M) \setminus \{\emptyset\},$$

for $1 \leq i \leq n$, such that

$$\mathcal{M} \models_Y \theta$$

when $Y := \{\emptyset\}[F_1/x_1] \ldots [F_n/x_n]$. Note that $F_i(s) = M$ if $Q^i = \forall$.

Let us then construct teams $Y_{\vec{i}}$, for $\vec{i} \in S$, such that $\mathcal{M} \models_{Y_{\vec{i}}} \theta_{\vec{i}}$, as follows. Let $Y_\lambda := Y$.

- Assume that $\mathcal{M} \models_{Y_{\vec{i}}} \theta_{\vec{i}}$ where $\theta_{\vec{i}} = \theta_{\vec{i}0} \wedge \theta_{\vec{i}1}$. Then $Y_{\vec{i}0} := Y_{\vec{i}}$ and $Y_{\vec{i}1} := Y_{\vec{i}}$.
- Assume that $\mathcal{M} \models_{Y_{\vec{i}}} \theta_{\vec{i}}$ where $\theta_{\vec{i}} = \theta_{\vec{i}0} \vee \theta_{\vec{i}1}$. Then choose $Y_{\vec{i}0} \cup Y_{\vec{i}1} = Y_{\vec{i}}$ so that $\mathcal{M} \models_{Y_{\vec{i}0}} \theta_{\vec{i}0}$ and $\mathcal{M} \models_{Y_{\vec{i}1}} \theta_{\vec{i}1}$.

We then note that

$$\mathcal{M} \models_{Y_{\vec{i}}} \vec{b} \perp_{\vec{a}} \vec{c} \quad \text{if} \quad \theta_{\vec{i}} \text{ is } \vec{b} \perp_{\vec{a}} \vec{c}, \tag{33}$$

$$\mathcal{M} \models_{Y_{\vec{i}}} =(\vec{z}, y) \quad \text{if} \quad \theta_{\vec{i}} \text{ is } =(\vec{z}, y). \tag{34}$$

Now, for $\theta_{\vec{i}}$ of the form $\vec{b} \perp_{\vec{a}} \vec{c}$, the interpretations of $S_{\vec{i}}$ and $T_{\vec{i}}$ will be the following:

$$S_{\vec{i}}^{\mathcal{M}} := \{s(\vec{a}\vec{b}) \mid s \in Y_{\vec{i}}\},$$
$$T_{\vec{i}}^{\mathcal{M}} := \{s(\vec{a}\vec{c}) \mid s \in Y_{\vec{i}}\}.$$

For $\theta_{\vec{i}}$ of the form $=(\vec{z}, y)$ we interpret $f_{\vec{i}}$ as follows:

$$f_{\vec{i}}^{\mathcal{M}}(\vec{a}) := \begin{cases} b & \text{if } s(\vec{z}y) = \vec{a}b \text{ for some } s \in Y_{\vec{i}}, \\ 0 & \text{otherwise} \end{cases}$$

where $0 \in M$ is arbitrary. Now $f_{\vec{i}}$ is well defined by (34). Let then $\mathcal{M}^* := (\mathcal{M}, \vec{S}^{\mathcal{M}}, \vec{T}^{\mathcal{M}}, \vec{f}^{\mathcal{M}})$. We will show that

$$\mathcal{M}^* \models Q^1 x_1 \dots Q^n x_n \varphi \wedge \Omega.$$

Consider the first conjunct. For each x_i with $Q^i = \exists$ we can choose a value for it so that the values of $x_1, \dots, x_i$ agree with some $s \in Y$. Thus it suffices to show that, for $s \in Y$,

$$\mathcal{M}^* \models_s \varphi.$$

Since φ is a first-order formula, by Theorem 4 it suffices to show that $\mathcal{M}^* \models_Y \varphi$. This can be done inductively: For each atomic $\varphi_{\vec{i}}$, $\mathcal{M}^* \models_{Y_{\vec{i}}} \varphi_{\vec{i}}$ by the definitions. If $\mathcal{M}^* \models_{Y_{\vec{i}0}} \varphi_{\vec{i}0}$ and $\mathcal{M}^* \models_{Y_{\vec{i}1}} \varphi_{\vec{i}1}$, and $\varphi_{\vec{i}}$ is either disjunction or conjunction of $\varphi_{\vec{i}0}$ and $\varphi_{\vec{i}1}$, then $\mathcal{M}^* \models_{Y_{\vec{i}}} \varphi_{\vec{i}}$ by the construction of $Y_{\vec{i}}$. This concludes the claim and thus the first conjunct part.

Next we will to show that $\mathcal{M}^* \models \Omega$ where Ω is the formula

$$\bigwedge_{\vec{i} \in I} [\forall \vec{a}\vec{b}\vec{c}(S_{\vec{i}}(\vec{a}\vec{b}) \wedge T_{\vec{i}}(\vec{a}\vec{c})) \to \exists \vec{z}_{\vec{i}}(\bigwedge_{\vec{j} \le \vec{i}} \varphi_{\vec{j}} \wedge Q^1 x_1' \dots Q^n x_n'(\varphi' \wedge \chi))].$$

Let $\vec{i} \in I$ and assume that $\theta_{\vec{i}} = \vec{b} \perp_{\vec{a}} \vec{c}$. Let $\vec{\alpha}\vec{\beta}\vec{\gamma}$ be such that $\vec{\alpha}\vec{\beta} \in S_{\vec{i}}^{\mathcal{M}}$ and $\vec{\alpha}\vec{\gamma} \in T_{\vec{i}}^{\mathcal{M}}$. Then there are $s, s' \in Y_{\vec{i}}$ such that $s(\vec{a}\vec{b}) = \vec{\alpha}\vec{\beta}$ and $s'(\vec{a}\vec{c}) = \vec{\alpha}\vec{\gamma}$. By (33) we can choose $s'' \in Y_{\vec{i}}$ such that $s''(\vec{a}\vec{b}\vec{c}) = s(\vec{a}\vec{b})s'(\vec{c})$. Let us then choose the values for $\vec{z}_{\vec{i}}$ according to s''. Then all the values of $x_1, \dots, x_n$ agree with s''. Now, since $\mathcal{M}^* \models_{Y_{\vec{j}}} \varphi_{\vec{j}}$ for all $\vec{j}$, and $s'' \in Y_{\vec{j}}$ for $\vec{j} \le \vec{i}$, it follows by Theorem 4

$$\mathcal{M}^* \models_{s''} \bigwedge_{\vec{j} \le \vec{i}} \varphi_{\vec{j}}.$$

Now it suffices to show that

$$\mathcal{M}^* \models_{s''} Q^1 x_1' \dots Q^n x_n'(\varphi' \wedge \chi).$$

For each x_i' with $Q^i = \exists$ we choose a value for it so that, for some $t \in Y$, the values of $x_1', \dots, x_i'$ are $t(x_1), \dots, t(x_i)$. In particular, if the values of $x_1', \dots, x_{i-1}'$ agree with s'', then we choose x_i' according to s'' also. Let s^* be an extension of s'' which is constructed according to these rules. Now using the fact that $\mathcal{M}^* \models_t \varphi$ for all $t \in Y$, and the way s^* was chosen, we get

$$\mathcal{M}^* \models_{s^*} \varphi' \wedge \chi.$$

Hence $\mathcal{M} \models \Phi$. This concludes the *only if-part*.

If-part: Assume that $\mathcal{M} \models \Phi$. Then we can find interpretations $\vec{S}^{\mathcal{M}}$, $\vec{T}^{\mathcal{M}}$ and $\vec{f}^{\mathcal{M}}$ such that

$$\mathcal{M}^* \models Q^1 x_1 \dots Q^n x_n \varphi \wedge \Omega$$

when $\mathcal{M}^* := (\mathcal{M}, \vec{S}^{\mathcal{M}}, \vec{T}^{\mathcal{M}}, \vec{f}^{\mathcal{M}})$. Consider the usual semantic game for first-order logic where player $\exists$ plays the role of verifier and player $\forall$ plays the role of falsifier. Then there is a winning strategy for player $\exists$ in the semantic game for $Q^1 x_1 \dots Q^n x_n \varphi \wedge \Omega$ over $\mathcal{M}^*$. Let Y consist of assignments $s : \{x_1, \dots, x_n\} \to M$ corresponding to every possible play of $x_1, \dots, x_n$ where player $\exists$ follows her winning strategy. Analogously, let Y' consist of

assignments $s : \{x_1, \dots, x_n\} \to M$ that correspond to every possible play of $x'_1, \dots, x'_n$ where player $\exists$ follows her winning strategy. Let $X := Y \cup Y'$. We will show that

$$\mathcal{M} \models_X \theta.$$

We know that $\mathcal{M}^* \models_X \varphi$. Let us now define $X_{\vec{i}}$, for $\vec{i} \in S$, as follows. Recall that $\varphi_\lambda = \varphi$ where λ is the empty sequence. We also let $X_\lambda := X$.

- If $\mathcal{M}^* \models_{X_{\vec{i}}} \varphi_{\vec{i}}$ where $\varphi_{\vec{i}} = \varphi_{\vec{i}0} \wedge \varphi_{\vec{i}1}$, then we let $X_{\vec{i}0} := X_{\vec{i}}$ and $X_{\vec{i}1} := X_{\vec{i}}$.
- If $\mathcal{M}^* \models_{X_{\vec{i}}} \varphi_{\vec{i}}$ where $\varphi_{\vec{i}} = \varphi_{\vec{i}0} \vee \varphi_{\vec{i}1}$, then we let $X_{\vec{i}0} := \{s \in X_{\vec{i}} \mid \mathcal{M}^* \models_s \varphi_{\vec{i}0}\}$ and $X_{\vec{i}1} := \{s \in X_{\vec{i}} \mid \mathcal{M}^* \models_s \varphi_{\vec{i}1}\}$.

From the construction it follows that $\mathcal{M}^* \models_{X_{\vec{i}}} \varphi_{\vec{i}}$, for $\vec{i} \in S$, and that $X_{\vec{i}0} \cup X_{\vec{i}1} = X_{\vec{i}}$ if $\varphi_{\vec{i}}$ is a disjunction. We will now show that for each atomic $\theta_{\vec{i}}$, $\mathcal{M} \models_{X_{\vec{i}}} \theta_{\vec{i}}$:

1. If $\theta_{\vec{i}}$ is a first-order atom, then the claim follows from $\theta_{\vec{i}} = \varphi_{\vec{i}}$.
2. If $\theta_{\vec{i}}$ is $=(\vec{z}, y)$, then the claim follows from $\mathcal{M}^* \models_{X_{\vec{i}}} f_{\vec{i}}(\vec{z}) = y$.
3. Assume that $\theta_{\vec{i}}$ is $\vec{b} \perp_{\vec{a}} \vec{c}$. Then $\mathcal{M}^* \models_{X_{\vec{i}}} S_{\vec{i}}(\vec{a}\vec{b}) \wedge T_{\vec{i}}(\vec{a}\vec{c})$. Let $s, s' \in X_{\vec{i}}$ be such that $s(\vec{a}) = s'(\vec{a})$. We have to show that there is $s'' \in X_{\vec{i}}$ such that $s''(\vec{a}\vec{b}\vec{c}) = s(\vec{a})s(\vec{b})s'(\vec{c})$. Now $\mathcal{M}^* \models \Omega$, so consider a play in the semantic game where player $\forall$ chooses first the conjunct with index $\vec{i}$ from Ω, and then chooses $s(\vec{a})s(\vec{b})s'(\vec{c})$ as values for $\vec{a}\vec{b}\vec{c}$. Since $s(\vec{a})s(\vec{b}) \in S^{\mathcal{M}}_{\vec{i}}$ and $s(\vec{a})s'(\vec{c}) \in T^{\mathcal{M}}_{\vec{i}}$, then player $\exists$ plays according to her strategy and chooses values for $\vec{z}_{\vec{i}}$ so that

$$\mathcal{M}^* \models_{s''} \bigwedge_{\vec{j} \leq \vec{i}} \varphi_{\vec{j}} \wedge Q^1 x'_1 \dots Q^n x'_n (\varphi' \wedge \chi)$$

where s'' is the assignment agreeing with the chosen values for $\vec{a}\vec{b}\vec{c}$ and $\vec{z}_{\vec{i}}$. Now we let player $\forall$ play each x'_i with $Q^i = \forall$ as $s''(x_i)$. Then because of χ (defined in (32)) player $\exists$ must also play each x'_i with $Q^i = \exists$ as $s''(x_i)$. Hence s'' corresponds to a play of $x'_1, \dots, x'_n$, and thus $s'' \in X$.

Since $\mathcal{M}^* \models_{s''} \bigwedge_{\vec{j} \leq \vec{i}} \varphi_{\vec{j}}$, it is a straightforward induction to show that $s'' \in X_{\vec{i}}$. This concludes the step 3.

Now using the previous, a straightforward backward induction shows that $\mathcal{M} \models_X \theta$. It then suffices to show that there are functions

$$F_i : \{\emptyset\}[F_1/x_1] \dots [F_{i-1}/x_{i-1}] \to \mathcal{P}(M) \setminus \{\emptyset\}$$

such that $F_i(s) = M$ if $Q^i = \forall$, and that

$$X = \{\emptyset\}[F_1/x_1] \dots [F_n/x_n].$$

We define these functions inductively so that $\{\emptyset\}[F_1/x_1] \dots [F_i/x_i] = X \upharpoonright \{x_1, \dots, x_i\}$, for $1 \leq i \leq n$. Assume that we have defined $F_1, \dots, F_i$ successfully. We will define F_{i+1} as wanted. Assume first that $Q^{i+1} = \exists$. Then for $s \in \{\emptyset\}[F_1/x_1] \dots [F_i/x_i]$, we let

$$F_{i+1}(s) = \{t(x_{i+1}) \mid t \in X, t \upharpoonright \{x_1, \dots, x_i\} = s\}.$$

By the induction assumption $F_{i+1}(s)$ is non-empty, though it may not be singleton in case there are multiple plays where values of $x_1, \dots, x_i$ (or $x'_1, \dots, x'_i$) agree with s. We note that

$$\{\emptyset\}[F_1/x_1] \dots [F_{i+1}/x_{i+1}] = X \upharpoonright \{x_1, \dots, x_{i+1}\}.$$

Assume then that $Q^{i+1} = \forall$. For $s \in \{\emptyset\}[F_1/x_1] \dots [F_i/x_i]$, we let $F_{i+1}(s) = \mathcal{P}(M)$ and note that

$$X \upharpoonright \{x_1, \dots, x_{i+1}\} \subseteq \{\emptyset\}[F_1/x_1] \dots [F_{i+1}/x_{i+1}].$$

For the other direction, assume that $s \in \{\emptyset\}[F_1/x_1]\dots[F_i/x_i]$ and let $a \in M$. We show that $s(a/x) \in X \upharpoonright \{x_1, \dots, x_{i+1}\}$. By the induction assumption $s \in X \upharpoonright \{x_1, \dots, x_i\}$, and thus there is a play of $x_1, \dots, x_n$ (or $x'_1, \dots, x'_n$) that agrees with s in the first i variables. Let s' be the assignment corresponding to this play. Now instead of choosing $s'(x_{i+1})$ (or $s'(x'_{i+1})$) at move $i+1$, player $\forall$ can choose a for x_{i+1} (or for x'_{i+1}). Let t be an assignment that corresponds to some play with these moves for the first $i+1$ variables. Then $t \in X$ and $t \upharpoonright \{x_1, \dots, x_{i+1}\} = s(a/x_{i+1})$. This concludes the proof, and thus also the *only if-part.*

Note that in Φ each function or relation variable has an arity at most k. This concludes the proof. ◀

▶ **Theorem 35.** $\mathrm{ESO}_f(k\text{-ary}) = \mathrm{FO}(\perp_c)(k\text{-ind})$.

Proof. Follows from Propositions 27 and 29. ◀

This gives us immediately a corollary regarding inclusion logic. Recall that $\mathrm{FO}(\subseteq)(k\text{-inc})$ denotes the class of inclusion logic sentences in which inclusion atoms of width at most k (i.e. atoms of the form $\vec{a} \subseteq \vec{b}$ where $|\vec{a}| = |\vec{b}| \leq k$) may appear.

▶ **Theorem 36.** *Assume $k \geq 2$. Then* $\mathrm{FO}(\subseteq)(k\text{-inc}) \leq \mathrm{ESO}_f(k\text{-ary})$.

Proof. Using item 1 of Theorem 12, we first translate inclusion logic sentences to independence logic, and then apply Proposition 29. It is easy to check that this translation takes us to $\mathrm{ESO}_f(k'\text{-ary})$, where $k' = \max\{k, 2\}$. ◀

There is no hope of proving the other direction, since, e.g., even cardinality cannot be expressed in $\mathrm{FO}(\subseteq)$ [3], but it is expressible in $\mathrm{ESO}_f(1\text{-ary})$. Next we will show that $\mathrm{ESO}_f(k\text{-ary}) \leq \mathrm{FO}(\perp)(2k+2\text{-ind})$.

▶ **Theorem 37.** $\mathrm{ESO}_f(k\text{-ary}) \leq \mathrm{FO}(\perp)(2k+1\text{-ind}) \leq \mathrm{ESO}_f(2k+1\text{-ary})$.

Proof. For the first inequality, note that $\mathrm{ESO}_f(k\text{-ary}) = \mathcal{D}(k\text{-dep})$ by [7], and $\mathcal{D}(k\text{-dep}) \leq \mathrm{FO}(\perp)(2k+1\text{-ind})$ by Theorem 13. The second inequality follows from Theorem 35. ◀

4.2 $\forall$-hierarchies

In this section, we will examine the fragments $\mathrm{FO}(\mathcal{C})(k\forall)$. We will prove that, contrary to the case of the fragments $\mathcal{D}(k\forall)$ [7], the following holds:

1. If $\{\perp, \subseteq\} \subseteq \mathcal{C}$ then the hierarchy collapses at level 1: $\mathrm{FO}(\mathcal{C}) = \mathrm{FO}(\mathcal{C})(1\forall)$;
2. If $\perp \in \mathcal{C}$ then it collapses at level 2: $\mathrm{FO}(\mathcal{C}) = \mathrm{FO}(\mathcal{C})(2\forall)$.

We will use the following result from [21]:

▶ **Proposition 38.** *Let ϕ be a* $\mathrm{FO}(\perp)$ *sentence. Then ϕ is equivalent to an formula of the form $\forall\vec{x}\exists\vec{y}(\theta \wedge \chi)$, where θ is a conjunction of pure independence atoms and χ is first-order and quantifier-free.*

Since, as we saw in the Preliminaries, we can define inclusion atoms and conditional independence atoms in terms of pure independence atoms, it follows at once that any sentence of $\mathrm{FO}(=(\dots), \perp_c, \subseteq)$ is equivalent to some sentence of the above form.

Using this, we will prove that

▶ **Theorem 39.** $\mathrm{FO}(\perp, \subseteq)(1\forall) = \mathrm{FO}(=(\dots), \perp_c, \subseteq)$.

Proof. Let $\phi \in \mathrm{FO}(=(\ldots), \perp_c, \subseteq)$. We will show that there exists a $\phi' \in \mathrm{FO}(\perp, \subseteq)(1\forall)$ equivalent to it. As we said, we can assume that ϕ is of the form $\forall x_1 \ldots \forall x_m \exists x_{m+1} \ldots \exists x_{m+n}(\theta \wedge \chi)$, where θ is a conjunction of pure independence atoms and χ is first-order and quantifier-free. Let us then define ϕ' as

$$\forall x_1 \exists x_2 \ldots \exists x_m \exists x_{m+1} \ldots \exists x_{m+n}(\bigwedge_{2 \leq i \leq m} (x_1 \subseteq x_i \wedge x_1 \ldots x_{i-1} \perp x_i) \wedge \theta \wedge \chi).$$

We claim that ϕ' is equivalent to ϕ. Assume first that $\mathcal{M} \models \phi$. Then there are, for $m+1 \leq i \leq m+n$, functions

$$F_i : \{\emptyset\}[M/x_1] \ldots [M/x_n][F_{m+1}/x_{m+1}] \ldots [F_{i-1}/x_{i-1}] \to \mathcal{P}(M) \setminus \{\emptyset\}$$

such that $\mathcal{M} \models_X \theta \wedge \chi$ when $X := [M/x_1] \ldots [M/x_n][F_{m+1}/x_{m+1}] \ldots [F_{m+n}/x_{m+n}]$. Let F_i, for $2 \leq i \leq m$, be the constant function mapping each assignment to M. Then

$$X = \{\emptyset\}[M/x_1][F_2/x_2] \ldots [F_{m+n}/x_{m+n}].$$

Clearly $\mathcal{M} \models_X \bigwedge_{2 \leq i \leq m}(x_1 \subseteq x_i \wedge x_1 \ldots x_{i-1} \perp x_i)$, and hence $\mathcal{M} \models \phi'$.

For the other direction, assume that $\mathcal{M} \models \phi'$. Then there are, for $2 \leq i \leq m+n$, functions

$$F_i : \{\emptyset\}[M/x_1][F_2/x_2] \ldots [F_{i-1}/x_{i-1}] \to \mathcal{P}(M) \setminus \{\emptyset\}$$

such that $\mathcal{M} \models_X \bigwedge_{2 \leq i \leq m}(x_1 \subseteq x_i \wedge x_1 \ldots x_{i-1} \perp x_i)$ when

$$X := \{\emptyset\}[M/x_1][F_2/x_2] \ldots [F_{m+n}/x_{m+n}].$$

Define, for $2 \leq i \leq m$,

$$X_i := \{\emptyset\}[M/x_1][F_2/x_2] \ldots [F_i/x_i]$$

and

$$Y_i := \{\emptyset\}[M/x_1][M/x_2] \ldots [M/x_i].$$

It suffices to show that $X_i = Y_i$ for $2 \leq i \leq m$.

First let us prove the claim for $i = 2$. Let $s \in Y_2$. It suffices to show that $s \in X_2$. By Proposition 7, $\mathcal{M} \models_{X_2} x_1 \subseteq x_2 \wedge x_1 \perp x_2$. Let $s' \in X_2$ be such that $s'(x_1) = s(x_2)$. Since $\mathcal{M} \models_{X_2} x_1 \subseteq x_2$, we can find a $t \in X_2$ such that $t(x_2) = s'(x_1)$. Now let $t' \in X_2$ be such that $t'(x_1) = s(x_1)$. Because $\mathcal{M} \models_{X_2} x_1 \perp x_2$, we can find a $t'' \in X_2$ such that $t''(x_1) = t'(x_1)$ and $t''(x_2) = t(x_2)$. Then $t'' = s$ which concludes the claim for $i = 2$.

The induction step is proved analogously. This concludes the claim and the proof. ◀

Let us now prove our second claim.

▶ **Theorem 40.** $\mathrm{FO}(\perp)(2\forall) = \mathrm{FO}(=(\ldots), \perp_c, \subseteq)$.

Proof. Let $\phi \in \mathrm{FO}(=(\ldots), \perp_c, \subseteq)$. Again, we can assume that ϕ is of the form $\forall \vec{x} \psi$, where $\vec{x} = x_1 \ldots x_n$ and ψ is of the form $\exists \vec{y} \theta$ for θ quantifier-free and in $\mathrm{FO}(\perp)$. Let now p, q be two variables not occurring in ϕ. We state that ϕ is equivalent to

$$\phi^* = \forall p \forall q \exists \vec{x} \left(\left(p = q \to \bigwedge_{i=1}^{n} x_i = p \right) \wedge \bigwedge_{i=1}^{n-1} (x_1 \ldots x_i \perp x_{i+1}) \wedge \psi \right).$$

Indeed, let $\mathcal{M}$ be a model and $X = \{\emptyset\}[M/p][M/q]$, and let the tuple of (nondeterministic) choice functions $\vec{U}$ for $\vec{x}$ be such that

$$\vec{U}(s) = \begin{cases} (m, \ldots, m) & \text{if } s(p) = s(q) = m; \\ M^n & \text{otherwise} \end{cases}$$

and let $Y = X[\vec{U}/\vec{x}]$. It is obvious that $\mathcal{M} \models_Y (p = q \to \bigwedge_{i=1}^{n} x_i = p)$; and $\mathcal{M} \models_Y \psi$, because $Y(\vec{x}) = M^n$ and p, q do not occur in ψ. Finally, it is also true that Y satisfies all independence atoms $x_1 \dots x_i \perp x_{i+1}$, since $Y(x_1 \dots x_i x_{i+1}) = M^{i+1}$ (assuming that our model contains two distinct elements). Therefore $\mathcal{M} \models \phi^*$, as required.

Conversely, suppose $\mathcal{M} \models \phi^*$ when there exists a $\vec{U}$ such that, for $Y = \{\emptyset\}[M/pq][\vec{U}/\vec{x}]$, $\mathcal{M} \models_Y (p = q \to \bigwedge_{i=1}^{n} x_i = p) \wedge \bigwedge_{i=1}^{n-1}(x_1 \dots x_i \perp x_{i+1}) \wedge \psi$. We will show that $Y(x_1 \dots x_n)$ is M^n, that is, that all possible tuples $m_1 \dots m_n$ of elements of our models are possible values for $x_1 \dots x_n$ in Y.

First of all, let us observe that for all $m \in M$ there exists a $h^m \in Y$ such that $h^m(x_i) = m$ for all i. Indeed, we can find a $s^m \in X$ such that $s^m(p) = s^m(q) = m$ and then pick an arbitrary $h^m \in s^m[\vec{U}/\vec{x}] \subseteq Y$. Since $\mathcal{M} \models_Y p = q \to \bigwedge_i x_i = p$, we have at once that $h^m(x_i) = h^m(p) = m$, as required.

Now we prove, by induction on $i = 1 \dots n$, that there exists a $h_i \in Y$ such that $h_i(x_1 \dots x_i) = m_1 \dots m_i$.

Base Case: Let h_1 be $h^{m_1} \in Y$. Then $h^{m_1}(x_1) = m_1$, as required.

Induction Case: Suppose that $h_i(x_1 \dots x_i) = m_1 \dots m_i$, and consider $h^{m_{i+1}}$. As we saw, $h^{m_{i+1}} \in Y$ and $h^{m_{i+1}}(x_{i+1}) = m_{i+1}$. But $\mathcal{M} \models_Y x_1 \dots x_i \perp x_{i+1}$; and therefore there exists a $h_{i+1} \in Y$ with $h_{i+1}(x_1 \dots x_i) = h_i(x_1 \dots x_i) = m_1 \dots m_i$ and $h_{i+1}(x_{i+1}) = h^{m_{i+1}}(x_{i+1}) = m_{i+1}$. Hence, $h_{i+1}(x_1 \dots x_{i+1}) = m_1 \dots m_{i+1}$.

In particular, this implies that $h_n(x_1 \dots x_n) = m_1 \dots m_n$; and since we started from an arbitrary choice of $m_1 \dots m_n$, we can conclude that $Y(\vec{x}) = M^{|\vec{x}|}$. But then the restriction of Y to $\vec{x}$ is precisely $\{\emptyset\}[M/\vec{x}]$; and since $\mathcal{M} \models_Y \psi$, by locality we have that $\mathcal{M} \models \forall\vec{x}\psi$, as required. ◀

Conclusion

In this paper, we examined the expressive power of fragments of inclusion and independence logic obtained by restricting the arity of non first-order atoms or the number of universal quantifiers. For the first kind of restriction, we adapted and extended the hierarchy theorems of [7] to this new setting; but for the second kind of restriction, we showed that the hierarchy collapses at a very low level if our logic contains at least pure independence atoms.

A question which is still open is whether the fragments $\mathrm{FO}(\subseteq)(k\forall)$ of inclusion logic give rise to an infinite expressivity hierarchy. Another issue that requires further investigation is to which degree our results can be adapted to the case of strict semantics. The exact nature of the relationship between strict and lax semantics is a matter which is of no small interest for the further development of the area, and a comparison of the properties of our fragments in these two settings might prove itself of great value.

References

1 Samson Abramsky and Jouko Väänänen. From IF to BI. *Synthese*, 167:207–230, 2009. 10.1007/s11229-008-9415-6.

2 Miklos Ajtai. Σ_1^1-formulae on finite structures. *Ann. Pure Appl. Logic*, 4(1):1 – 48, 1983.

3 Pietro Galliani Lauri Hella. Inclusion logic and fixpoints. Manuscript, 2013.

4 William W. Armstrong. Dependency Structures of Data Base Relationships. In *Proc. of IFIP World Computer Congress*, pages 580–583, 1974.

5 Marco A. Casanova, Ronald Fagin, and Christos H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. In *Proceedings of the 1st ACM SIGACT-*

SIGMOD symposium on Principles of database systems, PODS '82, pages 171–176, New York, NY, USA, 1982. ACM.

6 Stephen A. Cook. A hierarchy for nondeterministic time complexity. In *Conference Record, Fourth Annual ACM Symposium on Theory of Computing*, pages 187–192. ACM, 1972.

7 Arnaud Durand and Juha Kontinen. Hierarchies in dependence logic. *ACM Transactions on Computational Logic (TOCL)*, 13(4):31, 2012.

8 Herbert B. Enderton. Finite partially-ordered quantifiers. *Mathematical Logic Quarterly*, 16(8):393–397, 1970.

9 Ronald Fagin. A normal form for relational databases that is based on domains and keys. *ACM Transactions on Database Systems*, 6:387–415, September 1981.

10 Pietro Galliani. *The Dynamics of Imperfect Information*. PhD thesis, University of Amsterdam, September 2012.

11 Pietro Galliani. Inclusion and exclusion dependencies in team semantics: On some logics of imperfect information. *Annals of Pure and Applied Logic*, 163(1):68 – 84, 2012.

12 Dan Geiger, Azaria Paz, and Judea Pearl. Axioms and algorithms for inferences involving probabilistic independence. *Information and Computation*, 91(1):128–141, 1991.

13 Erich Grädel. Model-checking games for logics of imperfect information. *Theoretical Computer Science (to appear)*, 2012.

14 Etienne Grandjean and Frédéric Olive. Graph properties checkable in linear time in the number of vertices. *J. Comput. Syst. Sci.*, 68(3):546–597, 2004.

15 Erich Grädel and Jouko Väänänen. Dependence and independence. *Studia Logica*, 101(2):399–410, 2013.

16 Miika Hannula. Axiomatizing first-order consequences in independence logic. Manuscript, 2012.

17 Leon Henkin. Some Remarks on Infinitely Long Formulas. In *Infinitistic Methods. Proc. Symposium on Foundations of Mathematics*, pages 167–183. Pergamon Press, 1961.

18 Jarmo Kontinen. Coherence and computational complexity of quantifier-free dependence logic formulas. *Studia Logica*, 101(2):267–291, 2013.

19 Pavel Naumov and Brittany Nicholls. R.E. axiomatization of conditional independence. To appear, 2013.

20 Jouko Väänänen. *Dependence Logic*. Cambridge University Press, 2007.

21 Jouko Väänänen and Pietro Galliani. On dependence logic. In Preparation, 2013.

22 Wilbur John Walkoe. Finite partially-ordered quantification. *The Journal of Symbolic Logic*, 35(4):pp. 535–555, 1970.

23 Fan Yang. Expressing Second-order Sentences in Intuitionistic Dependence Logic. *Studia Logica*, 101(2):323–342, 2013.

5 Appendix

This secrtion contains the proof omitted in the main part of the paper.

▶ **Theorem 13.** *For all models $\mathcal{M}$ and teams X*

$$\mathcal{M} \models_X =(\vec{x}, y) \Leftrightarrow \mathcal{M} \models_X \forall \vec{z} \exists w((\vec{z} = \vec{x} \to w = y) \wedge \vec{x} y \perp_{\vec{z}} w).$$

Proof. Suppose first that $\mathcal{M} \models_X =(\vec{x}, y)$. Then there exists a function $f : M^{|\vec{x}|} \to M$ such that $f(s(\vec{x})) = s(y)$ for all $s \in X$. Then for $Y = X[M/\vec{z}]$, define the choice function $H : Y \to \mathcal{P}(M) \setminus \{\emptyset\}$ so that

$$H(s) = \{f(s(\vec{z}))\}$$

for all $s \in Y$, and let $Z = Y[H/w]$. If we can verify that $\mathcal{M} \models_Z \vec{z} = \vec{x} \to w = y$ and that $\mathcal{M} \models_Z \vec{x}y \perp \vec{z}w$, the left-to-right direction of our proof is done. Now, if $h \in Z$ then $h(w) = f(h(\vec{z}))$ and $h(y) = f(h(\vec{x}))$, and therefore $\mathcal{M} \models_Z \vec{z} = \vec{x} \to w = y$. Furthermore, for $h, h' \in Z$, we have that $h'' = h[h'(\vec{z})/\vec{z}][h'(w)/w] \in Z$, since our choice of w depends only on $\vec{z}$, and therefore $\mathcal{M} \models_Z \vec{x}y \perp \vec{z}w$.

Conversely, suppose that there exists a function $H : X[M/\vec{z}] \to \mathcal{P}(M)\backslash\{\emptyset\}$ such that, for $Z = X[M/\vec{z}][H/w]$, $\mathcal{M} \models_Z \vec{z} = \vec{x} \to w = y \wedge \vec{x}y \perp \vec{z}w$. Now let $s, s' \in X$ be such that $s(\vec{x}) = s'(\vec{x}) = \vec{m}$, let $a = s(y)$ and let $b = s'(y)$: we need to prove that $a = b$.

Take $h \in s[\vec{m}/\vec{z}][H/w] \subseteq Z$: since $\mathcal{M} \models_Z \vec{z} = \vec{x} \to w = y$, we must have that $h(w) = s(y) = a$. Similarly, for $h' \in s'[\vec{m}/\vec{z}][H/w] \subseteq Z$, we must have that $h'(w) = s'(y) = b$. But $\mathcal{M} \models_Z \vec{x}y \perp \vec{z}w$, so there exists a $h'' \in Z$ such that $h''(\vec{x}y) = h(\vec{x}y) = \vec{m}a$ and $h''(\vec{z}w) = h'(\vec{z}w) = \vec{m}b$. Since, again, $\mathcal{M} \models_Z \vec{z} = \vec{x} \to w = y$, the only possibility is that $a = b$, as required. ◀

▶ **Lemma 14.** *Let $\phi, \psi \in \mathrm{FO}(=(\ldots), \perp_c, \subseteq)$ and let x be a variable not occurring free in ψ. Then the following equivalences hold:*

1. $\exists x\phi \wedge \psi \equiv \exists x(\phi \wedge \psi)$,
2. $\exists x\phi \vee \psi \equiv \exists x(\phi \vee \psi)$,
3. $\forall x\phi \wedge \psi \equiv \forall x(\phi \wedge \psi)$,
4. $\forall x\phi \vee \psi \equiv \exists a \exists b \forall x((\phi \wedge a = b) \vee (\psi \wedge a \neq b))$ *where a and b are new variables.*

Proof. The cases 1, 2 and 3 are proved as in Lemma 12 in [16]. We prove number 4. By Proposition 7 it is enough to prove the equivalence for teams X with $\mathrm{Dom}(X) = \mathrm{Fr}(\forall x\phi \vee \psi)$.

Assume first that $\mathcal{M} \models_X \forall x\phi \vee \psi$ and x does not occur free in ψ. Then there are $Y \cup Z = X$ such that $\mathcal{M} \models_{Y[M/x]} \phi$ and $\mathcal{M} \models_Z \psi$. Let $0, 1 \in M$ be distinct. We extend each $s \in X$ with $a \mapsto 0$ and $b \mapsto 0$, for $s \in Y$, and with $a \mapsto 0$ and $b \mapsto 1$, for $s \in Z$, and we let X' consist of these extended assignments. So each $s \in X$ has either one or two extensions in X'. Let $Y' := \{s \in X'[M/x] \mid s(a) = s(b)\}$ and $Z' := \{s \in X'[M/x] \mid s(a) \neq s(b)\}$. Then by Proposition 7, $\mathcal{M} \models_{Y'} \phi \wedge a = b$ and $M \models_{Z'} \psi \wedge a \neq b$. Hence $M \models_{X'[M/x]} (\phi \wedge a = b) \vee (\psi \wedge a \neq b)$, and we conclude that $\mathcal{M} \models_X \exists a \exists b \forall x((\phi \wedge a = b) \vee (\psi \wedge a \neq b))$.

Assume then that $\mathcal{M} \models_X \exists a \exists b \forall x((\phi \wedge a = b) \vee (\psi \wedge a \neq b))$. Let $F_a : X \to \mathcal{P}(M)$ and $F_b : X[F_a/a] \to \mathcal{P}(M)$ be such that if $X' := X[F_a/a][F_b/b][M/x]$, then $\mathcal{M} \models_{X'} (\phi \wedge a = b) \vee (\psi \wedge a \neq b)$. Let $Y' \cup Z' = X'$ be such that $\mathcal{M} \models_{Y'} \phi \wedge a = b$ and $\mathcal{M} \models_{Z'} \psi \wedge a \neq b$. Let $Y := Y' \upharpoonright \mathrm{Dom}(X)$ and $Z := Z' \upharpoonright \mathrm{Dom}(X)$. Then $Y[M/x] = Y' \upharpoonright (\mathrm{Dom}(X) \cup \{x\})$, and thus by Proposition 7 $\mathcal{M} \models_{Y[M/x]} \phi$. Also by Proposition 7 $\mathcal{M} \models_Z \psi$. Since $Y \cup Z = X$, we conclude that $\mathcal{M} \models \forall x\phi \vee \psi$. ◀

▶ **Theorem 25.** *For each natural number n there is a sentence $\phi \in \mathrm{FO}(\subseteq)$ such that for all models $\mathcal{M}$ and teams X where $X \neq \emptyset$ and the variables in $Dom(X)$ do not appear in ϕ,*

$\mathcal{M} \models_X \phi$ *if and only if* $|X| \geq n$.

Proof. Let n be a natural number. We may assume that $n \geq 2$ because in the case $n = 1$ we can just choose $\phi := \top$. Let $\vec{x}_i$, for $0 \leq i \leq n-1$, list variables $x_{i,0}, \ldots, x_{i,l}$ where $l = \log(n)$. Let

$$\phi := \exists \vec{x}_0 \ldots \exists \vec{x}_{n-1}\Big(\bigwedge_{0 \leq i \leq n-1} \vec{x}_i \subseteq \vec{x}_0 \wedge \bigwedge_{0 \leq i < j \leq n-1} \vec{x}_i \neq \vec{x}_j\Big)$$

where

$$\vec{x}_i \neq \vec{x}_j := \bigvee_{0 \leq k \leq l} x_{i,k} \neq x_{j,k}.$$

Now ϕ is as wanted:

Assume first that $\mathcal{M} \models_X \phi$. Then there are, for $0 \leq i \leq n-1$, functions

$$F_i : X[F_0/\vec{x}_0] \dots [F_{i-1}/\vec{x}_{i-1}] \to M^{l+1}$$

such that

$$M \models_{X'} \bigwedge_{0 \leq i \leq n-1} \vec{x}_i \subseteq \vec{x}_0 \wedge \bigwedge_{0 \leq i < j \leq n-1} \vec{x}_i \neq \vec{x}_j \tag{41}$$

when $X' := X[F_0/\vec{x}_0] \dots [F_{n-1}/\vec{x}_{n-1}]$. Let $s \in X'$ be some arbitrary assignment. From (41) it follows that X' must include assignments s_i, for $0 \leq i \leq n-1$, such that $s_i(\vec{x}_0) = s(\vec{x}_i)$. Also from (41) it follows that $s(\vec{x}_i) \neq s(\vec{x}_j)$, for $0 \leq i < j \leq n-1$. Thus the assignments s_i are distinct and therefore $|X'| \geq n$. Because existential quantification of new variables in strict semantics preserves the cardinality of a team we deduce that $X \geq n$.

Suppose then $X \geq n$. By the assumption $n \geq 2$, and thus we may deduce that $|M| \geq 2$. Let 0 and 1 be two different members of M, and let $\bar{i}$ be the binary representation (of length $l+1$) of i, for $0 \leq i \leq n-1$, in terms of these 0 and 1. Choose then n different assignments $s_0, \dots, s_{n-1}$ from X. We define, for $0 \leq i \leq n-1$, $F_i : X[F_0/\vec{x}_0] \dots [F_{i-1}/\vec{x}_{i-1}] \to M^{l+1}$ as follows:

$$F_i(s) := \begin{cases} \overline{j+i} & \text{if } s \upharpoonright \mathrm{Dom}(X) = s_j, \text{ for } 0 \leq j \leq n-1, \\ \bar{i} & \text{otherwise} \end{cases}$$

where $j+i$ is mod n. By the assumption, the variables in $\mathrm{Dom}(X)$ are not listed in $\vec{x}_0 \dots \vec{x}_{n-1}$, and thus the functions F_i are consistent with the definition of existential quantification for strict semantics. Without the assumption it could be the case that different s_i and s_j would collapse into one assignment in the quantification procedure. Let $X' := X[F_0/\vec{x}_0] \dots [F_{n-1}/\vec{x}_{n-1}]$. Then s_j, for $0 \leq j \leq n-1$, is extended in X' to

$$s_j(\bar{j}/\vec{x}_{\bar{0}})(\overline{j+1}/\vec{x}_1) \dots (\overline{j-2}/\vec{x}_{n-2})(\overline{j-1}/\vec{x}_{n-1}),$$

and each $t \in X \setminus \{s_j \mid 0 \leq j \leq n-1\}$ is extended in X' analogously to s_0. So for each $s \in X'$ and $0 \leq i < j \leq n-1$ it holds that $s(\vec{x}_i) \neq s(\vec{x}_j)$. Also

$$\{s(\vec{x}_0) \mid s \in X'\} = \{\bar{i} \mid 0 \leq i \leq n-1\} = \bigcup_{0 \leq i \leq n-1} \{s(\vec{x}_i) \mid s \in X'\},$$

and thus

$$M \models_{X'} \bigwedge_{0 \leq i \leq n-1} \vec{x}_i \subseteq \vec{x}_0 \wedge \bigwedge_{0 \leq i < j \leq n-1} \vec{x}_i \neq \vec{x}_j$$

which concludes the proof. ◀

▶ **Lemma 28.** *Let $\vec{b} \perp_{\vec{a}} \vec{c}$ be an independence atom where $\vec{a}$, $\vec{b}$ and $\vec{c}$ are tuples of variables. If $\vec{b_0}$ lists the variables in $\vec{b} - \vec{a} \cup \vec{c}$, $\vec{c_0}$ lists the variables in $\vec{c} - \vec{a} \cup \vec{b}$, and $\vec{d}$ lists the variables in $\vec{b} \cap \vec{c} - \vec{a}$, then*

$$\vec{b} \perp_{\vec{a}} \vec{c} \equiv \vec{b_0} \perp_{\vec{a}} \vec{c_0} \wedge \bigwedge_{d \in \vec{d}} {=}(\vec{a}, d).$$

Proof. Assume that $\mathcal{M} \models_X \vec{b} \perp_{\vec{a}} \vec{c}$. Then clearly $\vec{b_0} \perp_{\vec{a}} \vec{c_0}$. For $\bigwedge_{d \in \vec{d}} {=}(\vec{a}, d)$, let $d \in \vec{d}$ and $s, s' \in X$ be such that $s(\vec{a}) = s'(\vec{a})$. Then by the assumption there is $s'' \in X$ such that $s''(\vec{a}\vec{b}\vec{c}) = s(\vec{a}\vec{b})s'(\vec{c})$. Because d is listed in both $\vec{b}$ and $\vec{c}$, it follows that $s(d) = s'(d)$.

Suppose then $\mathcal{M} \models_X \vec{b_0} \perp_{\vec{a}} \vec{c_0} \wedge \bigwedge_{d \in \vec{d}} =(\vec{a}, d)$. Let $s, s' \in X$ be such that $s(\vec{a}) = s'(\vec{a})$. By the assumption there is $s'' \in X$ such that $s''(\vec{a}\vec{b_0}\vec{c_0}) = s(\vec{a}\vec{b_0})s'(\vec{c_0})$. We want to show that $s''(\vec{a}\vec{b}\vec{c}) = s(\vec{a}\vec{b})s'(\vec{c})$. Consider first variables x listed in $\vec{b} - \vec{b_0}$. If x is listed in $\vec{a}$, then $s''(x) = s(x)$ as wanted. Assume that x is listed in $\vec{c} - \vec{a}$. Then $x \in \vec{d}$, and thus $s''(x) = s(x)$ follows from $s''(\vec{a}) = s(\vec{a})$.

For variables x is listed in $\vec{c} - \vec{c_0}$ the proof of $s''(x) = s'(x)$ is analogous because $s(\vec{a}) = s'(\vec{a})$. This concludes the proof. ◀

Inclusion Logic and Fixed Point Logic

Pietro Galliani[1] and Lauri Hella[2]

1 **University of Helsinki**
`pgallian@gmail.com`
2 **University of Tampere**
`lauri.hella@uta.fi`

Abstract

We investigate the properties of Inclusion Logic, that is, First Order Logic with Team Semantics extended with inclusion dependencies. We prove that Inclusion Logic is equivalent to Greatest Fixed Point Logic, and we prove that all union-closed first-order definable properties of relations are definable in it. We also provide an Ehrenfeucht-Fraïssé game for Inclusion Logic, and give an example illustrating its use.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Dependence Logic, Team Semantics, Fixpoint Logic, Inclusion

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.281

1 Introduction

Inclusion Logic [10], FO($\subseteq$), is a novel logical formalism designed for expressing inclusion dependencies between variables. It is closely related to Dependence Logic [24], FO(D), which is the extension of First Order Logic by functional dependencies between variables. Dependence Logic initially arose as a variant of *Branching Quantifier Logic* [13] and of *Independence-Friendly Logic* [14, 22], and its study has sparked the development of a whole family of logics obtained by adding various dependency conditions to First Order Logic.

All these logics are based on Team Semantics [16, 24] which is a generalization of Tarski Semantics. In Team Semantics, formulas are satisfied or not satisfied by *sets* of assignments, called *teams*, rather than by single assignments. This semantics was introduced in [16] for the purpose of defining a compositional equivalent for the Game Theoretic Semantics of Independence-Friendly Logic [14, 22], but it was soon found out to be of independent interest. See [9] for a, mostly up-to-date, account of the research on Team Semantics.

Like Branching Quantifier Logic and Independence-Friendly Logic, Dependence Logic has the same expressive power as Existential Second Order Logic Σ^1_1: every FO(D)-sentence is equivalent to some Σ^1_1-sentence, and vice versa [24]. The semantics of Dependence Logic is downwards closed in the sense that if a team X satisfies a formula ϕ in a model M, then all subteams $Y \subseteq X$ also satisfy ϕ in M. The equivalence between FO(D) and Σ^1_1 was extended to formulas in [19], where it was proved that FO(D) captures exactly the downwards closed Σ^1_1-definable properties of teams.

Other variants of Dependence Logic that have been studied are Conditional Independence Logic FO($\perp_c$) [12], Independence Logic FO($\perp$) [12, 25], Exclusion Logic FO($|\,$) [10] and Inclusion/Exclusion Logic FO($\subseteq$, $|\,$) [10]. All the logics in this family arise from dependency notions that have been studied in Database Theory. In particular, FO(D) is based on *functional dependencies* introduced by Armstrong [1], FO($\subseteq$) is based on *inclusion dependencies* [8, 3], FO($|\,$) is based on *exclusion dependencies* [4], and FO($\perp$) is based on *independence conditions* [11].

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 281–295

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The expressive power of all these logics, with the exception of FO($\subseteq$), is well understood. It is known that, with respect to sentences, they are all equivalent with Σ^1_1. With respect to formulas, FO($|$) is equivalent with FO(D) [10]; and FO($\subseteq$, $|$), FO($\perp_c$) and FO($\perp$) are all equivalent to each other [10, 25]. Moreover, FO($\perp_c$) (and hence also FO($\subseteq$, $|$) and FO($\perp$)) captures all Σ^1_1-definable properties of teams [10].

On the other hand, relatively little is known about the expressive power of Inclusion Logic, and the main purpose of the present work is precisely to remedy this. What little is known about this formalism can be found in [10], and amounts to the following: With respect to formulas, FO($\subseteq$) is strictly weaker than $\Sigma^1_1 \equiv$ FO($\perp_c$) and incomparable with FO(D) $\equiv$ FO($|$). This is simply because the semantics of FO($\subseteq$) is not downwards closed, but is closed under unions: if both teams X and Y satisfy a formula ϕ in a model M, then $X \cup Y$ also satisfies ϕ in M. Moreover, FO($\subseteq$) is stronger than First Order Logic over sentences, and it is contained in Σ^1_1; but it was unknown whether it is equivalent to Σ^1_1, or whether FO($\subseteq$)-formulas could define all union closed Σ^1_1-definable properties of teams.

In this paper we show that the answer to both of these problems is negative. In fact, we give a complete characterization for the expressive power of FO($\subseteq$) in terms of Positive Greatest Fixed Point Logic GFP$^+$: We prove that every FO($\subseteq$)-sentence is equivalent to some GFP$^+$-sentence, and vice versa (Corollary 17).

Fixed point logics have a central role in the area of Descriptive Complexity Theory. By the famous result of Immerman [17] and Vardi [26], Least Fixed Point Logic LFP captures PTIME on the class of ordered finite models. Furthermore, it is well known that on finite models, LFP is equivalent to GFP$^+$. Thus, we obtain a novel characterization for PTIME: a class of ordered finite models is in PTIME if and only if it is definable by a sentence of FO($\subseteq$).

In addition to the equivalence with GFP$^+$, we prove that all union-closed first-order definable properties of teams are definable in Inclusion Logic (Corollary 26). Thus, it is not possible to increase the expressive power of FO($\subseteq$) by adding first-order definable union-closed dependencies. On the other hand, it is an interesting open problem, whether FO($\subseteq$) can be extended by some natural set $\mathbf{D}$ of union-closed dependencies such that the extension FO($\subseteq, \mathbf{D}$) captures all union-closed Σ^1_1-definable properties of teams.

We also introduce a new Ehrenfeucht-Fraïssé game that characterizes the expressive power of Inclusion Logic (Theorem 29). Our game is a modification of the EF game for Dependence Logic defined in [24]. Although the EF game has a clear second order flavour, it is still more manageable than the usual EF game for Σ^1_1; we illustrate this by describing a concrete winning strategy for Duplicator in the case of models with empty signature (Proposition 30). Due to the equivalence between FO($\subseteq$) and GFP$^+$ we see that the EF game for Inclusion Logic is also a novel EF game for GFP$^+$; it is quite different in structure from the one introduced in [2]. It may be hoped that this new game and its variants could be of some use for studying the expressive power of fixed point logics.

2 Preliminaries

2.1 Team Semantics

In this section, we will recall the definition of the Team Semantics for First Order Logic. For simplicity reasons, we will assume that all our expressions are in negation normal form.

▶ **Definition 1.** Let M be a first order model and let V be a set of variables. A *team* X over M with *domain* $\mathtt{Dom}(X) = V$ is a set of assignments $s : V \to \mathtt{Dom}(M)$. Given a tuple

$\vec{t} = (t_1, \ldots, t_n)$ of terms with variables in V and an assignment $s \in X$, we write $\vec{t}\langle s \rangle$ for the tuple $(t_1\langle s \rangle, \ldots, t_n\langle s \rangle)$, where $t\langle s \rangle$ denotes the value of the term t with respect to s in the model M. Furthermore, we write $X(\vec{t})$ for the relation $\{\vec{t}\langle s \rangle : s \in X\}$.

A (non-deterministic) *choice function* for a team X over a set A is a function $H : X \to \mathcal{P}(A) \setminus \{\emptyset\}$. The set of all choice functions for X over A is denoted by $\mathcal{C}(X, A)$.

▶ **Definition 2** (Team Semantics for First Order Logic[1])**.** Let M be a first order model and let X be a team over it. Then, for all first-order literals α, variables v, and formulas ϕ and ψ over the signature of M and with free variables in $\mathtt{Dom}(X)$,
TS-lit: $M \models_X \alpha$ iff for all $s \in X$, $M \models_s \alpha$ in the usual Tarski Semantics sense;
TS-$\vee$: $M \models_X \phi \vee \psi$ iff $X = Y \cup Z$ for some Y and Z such that $M \models_Y \phi$ and $M \models_Z \psi$;
TS-$\wedge$: $M \models_X \phi \wedge \psi$ iff $M \models_X \phi$ and $M \models_X \psi$;
TS-$\exists$: $M \models_X \exists v\phi$ iff there exists a function $H \in \mathcal{C}(X, \mathtt{Dom}(M))$ such that $M \models_{X[H/v]} \psi$, where $X[H/v] = \{s[m/v] : s \in X, m \in H(s)\}$;
TS-$\forall$: $M \models_X \forall v\phi$ iff $M \models_{X[M/v]} \phi$, where $X[M/v] = \{s[m/v] : s \in X, m \in \mathtt{Dom}(M)\}$.

The next theorem can be proved by structural induction on ϕ:

▶ **Theorem 3** (Team Semantics and Tarski Semantics)**.** *For all first order formulas $\phi(\vec{v})$, all models M and all teams X, $M \models_X \phi$ if and only if for all $s \in X$, $M \models_s \phi$ with respect to Tarski Semantics.*

Thus, in the case of First Order Logic it is possible to reduce Team Semantics to Tarski Semantics. What is then the point of working with the technically more complicated Team Semantics? As we will see in the next subsection, the answer is that Team Semantics allows us to extend First Order Logic in novel and interesting ways.

Note that on every model M, there are two teams with empty domain: the empty team $\emptyset$, and the team $\{\emptyset\}$ containing the empty assignment $\emptyset$. All the logics that we consider in this paper have the *empty team property*: $M \models_\emptyset \phi$ for every formula ϕ and model M. Thus, we say that a *sentence ϕ is true* in a model M if $M \models_{\{\emptyset\}} \phi$. If this is the case, we drop the subscript $\{\emptyset\}$, and write just $M \models \phi$.

2.2 Dependencies in Team Semantics

As we saw, in Team Semantics formulas are satisfied or not satisfied by sets of assignments, called *teams*; and a team corresponds in a natural way to a relation over the domain of the model. Therefore, any property of relations can be made to correspond to some property of teams, which we can then add to our language as a new atomic formula. In particular, we can do so for database-theoretic dependency notions, thus obtaining the following *generalized atoms*:[2]

▶ **Definition 4** (Dependence Atoms)**.** Let $\vec{t}_1$, $\vec{t}_2$, $\vec{t}_3$ be tuples of terms over some vocabulary. Then, for all models M and all teams X over M whose domain contains the variables of $\vec{t}_1\vec{t}_2\vec{t}_3$,
TS-fdep: $M \models_X =(\vec{t}_1, \vec{t}_2)$ if and only if, for all $s, s' \in X$, $\vec{t}_1\langle s \rangle = \vec{t}_1\langle s' \rangle \Rightarrow \vec{t}_2\langle s \rangle = \vec{t}_2\langle s' \rangle$;
TS-exc: For $|\vec{t}_1| = |\vec{t}_2|$, $M \models_X \vec{t}_1 \mid \vec{t}_2$ if and only if $X(\vec{t}_1) \cap X(\vec{t}_2) = \emptyset$;

[1] What we present here is the so-called *lax* version of Team Semantics. There also exists a *strict* version, with somewhat different rules for disjunction and existential quantification. As discussed in [10], the lax semantics has more convenient properties for the case of Inclusion Logic.
[2] The notion of "generalized atom" is defined formally in [20].

TS-inc: For $|\vec{t}_1| = |\vec{t}_2|$, $M \models_X \vec{t}_1 \subseteq \vec{t}_2$ if and only if $X(\vec{t}_1) \subseteq X(\vec{t}_2)$;
TS-ind: $M \models_X \vec{t}_1 \perp \vec{t}_2$ if and only if for all $s, s' \in X$ there exists an $s'' \in X$ with $\vec{t}_1\langle s''\rangle = \vec{t}_1\langle s\rangle$ and $\vec{t}_2\langle s''\rangle = \vec{t}_2\langle s'\rangle$;
TS-cond-ind: $M \models_X \vec{t}_2 \perp_{\vec{t}_1} \vec{t}_3$ if and only if for all $s, s' \in X$ with $\vec{t}_1\langle s\rangle = \vec{t}_1\langle s'\rangle$ there exists an $s'' \in X$ with $(\vec{t}_1\vec{t}_2)\langle s''\rangle = (\vec{t}_1\vec{t}_2)\langle s\rangle$ and $(\vec{t}_1\vec{t}_3)\langle s''\rangle = (\vec{t}_1\vec{t}_3)\langle s'\rangle$.

These atoms correspond respectively to *functional dependencies* [1], to *exclusion dependencies* [4], to *inclusion dependencies* [8, 3], to *independence conditions* [11], and to *conditional independence conditions*[3]; and by adding them to the language of First Order Logic we can obtain various logics, whose principal known properties we will now briefly recall.

Dependence Logic FO(D) is obtained by adding functional dependence atoms to the language of First Order Logic. It is the oldest and the most studied among the logics that we will discuss in this work, having been introduced in the seminal book [24] as an alternative approach to the study of *Branching* [13] and *Independence-Friendly* [14, 22] Quantification. It is *downwards closed*, in the sense that, for all models M, Dependence Logic formulas ϕ and teams X, if $M \models_X \phi$ then $M \models_Y \phi$ for all subsets Y of X.

On the level of sentences, Dependence Logic has the same expressive power as Existential Second Order Logic Σ^1_1.

▶ **Theorem 5** ([27, 6, 24]). *Every* FO(D)*-sentence is equivalent to some* Σ^1_1*-sentence, and vice versa. In particular,* FO(D) *captures* NP *on finite models.*

The equivalence between FO(D) and Σ^1_1 was extended to formulas by Kontinen and Väänänen, who proved the following characterization:

▶ **Theorem 6** ([19]). *Let* ϕ *be a* FO(D)*-formula with free variables in* $\vec{v}$*. Then there exists a* Σ^1_1*-sentence* $\Phi(R)$*, where* R *is a* $|\vec{v}|$*-ary relation symbol which occurs only negatively in* Φ*, such that*

$$M \models_X \phi \iff (M, X(\vec{v})) \models \Phi(R) \text{ for all models } M \text{ and teams } X \neq \emptyset.$$

Conversely, for any such $\Phi(R)$ *there exists an* FO(D)*-formula* ϕ *such that the above holds.*

Thus, FO(D) is the strongest logic that can be obtained by adding Σ^1_1-definable downwards-closed dependence conditions to First-Order Logic. Indeed, any such condition will be expressible as $\exists S(X(\vec{v}) \subseteq S \wedge \Phi(S))$ for some Φ in Σ^1_1, and therefore it will be equivalent to some FO(D)-formula.

Exclusion Logic FO(|), on the other hand, is the logic obtained by adding exclusion atoms to First-Order Logic. It was introduced in [10], where it was shown to be equivalent to Dependence Logic with respect to formulas.

Conditional Independence Logic FO($\perp_c$), which was introduced in [12], adds conditional independence atoms $\vec{t}_2 \perp_{\vec{t}_1} \vec{t}_3$ to the language of First Order Logic. Like FO(D), FO($\perp_c$) is equivalent to Σ^1_1 with respect to sentences, and also with respect to formulas:

▶ **Theorem 7** ([12]). *Every* FO($\perp_c$)*-sentence is equivalent to some* Σ^1_1*-sentence, and vice versa.*

▶ **Theorem 8** ([10]). *A class of relations is definable in Conditional Independence Logic if and only if it contains the empty relation and it is* Σ^1_1*-definable.*

[3] As observed in [7], conditional independence atoms also correspond to *embedded multivalued dependencies.*

Therefore, Conditional Independence Logic is the strongest logic that can be obtained by adding Σ^1_1-definable dependencies which are true of the empty relation to First Order Logic. In particular, this implies that every FO(D) formula (and, therefore, every FO(|) formula) is equivalent to some FO($\perp_c$) formula.[4] However, the converse is not true, since FO($\perp_c$) formulas are not, in general, downwards closed.

Furthermore, **Inclusion/Exclusion Logic** FO($\subseteq$, |) – that is, the logic obtained by adding inclusion *and* exclusion dependencies to First Order Logic – was proved in [10] to be equivalent with FO($\perp_c$) with respect to formulas.

Finally, **Independence Logic** FO($\perp$) is the logic obtained by adding only non-conditional dependence atoms $\vec{t}_1 \perp \vec{t}_2$ to First Order Logic. As proved in [25], Independence Logic and Conditional Independence Logic are also equivalent with respect to formulas.

Inclusion Logic FO($\subseteq$) is obtained by adding inclusion atoms to First Order Logic. It is not downwards closed, but it is *closed under unions* in the following sense: if ϕ is an FO($\subseteq$)-formula, M is a model, and X_i, $i \in I$, are teams on M such that $M \models_{X_i} \phi$ for all $i \in I$, then $M \models_X \phi$, where $X = \bigcup_{i \in I} X_i$. (For a proof, see [10]).

Relatively little is known about the expressive power of FO($\subseteq$), and the main purpose of the present work is precisely to remedy this. Here we recall the following results from [10]:

1. On the level of formulas, FO($\subseteq$) is strictly weaker than FO($\perp_c$) $\equiv$ FO($\perp$) $\equiv \Sigma^1_1$, and incomparable with FO(D) $\equiv$ FO(|).
2. The complement of the transitive closure of any first-order formula $\phi(\vec{x}, \vec{y})$ is definable in FO($\subseteq$); hence, FO($\subseteq$) is strictly stronger than First Order Logic on sentences.
3. On the level of sentences, FO($\subseteq$) is contained in Σ^1_1.

We give next a couple of further examples of the expressive power of FO($\subseteq$).

▶ **Example 9.** (a) Consider the sentence $\phi := \exists x \exists y (y \subseteq x \wedge Exy)$. Let $M = (\mathtt{Dom}(M), E^M)$ be a finite model. Then $M \models \phi$ if and only if E^M contains a cycle, i.e., there are $a_0, \ldots, a_{n-1} \in \mathtt{Dom}(M)$ such that $(a_i, a_{i+1}) \in E^M$ for all $i < n-1$, and $(a_{n-1}, a_0) \in E^M$.

The idea here is the following: the first existential quantifier gives a set C of values for x, and the formula $\exists y(y \subseteq x \wedge Exy)$ then says that for every $a \in C$ there is a $b \in C$ such that $(a, b) \in E^M$.

(b) Let ψ be the FO($\subseteq$)-sentence $\exists w(\exists u(Pu \wedge u \subseteq w) \wedge \forall u(Ewu \to \exists v(Euv \wedge v \subseteq w)))$. Then $M \models \psi$ if and only if player I has a winning strategy in the following game $G(M)$: Player I starts by choosing some element $a_0 \in P^M$. In each odd round $i+1$, player II chooses an element a_{i+1} such that $(a_i, a_{i+1}) \in E^M$. In each even round $i+1$, player I chooses an element a_{i+1} such that $(a_i, a_{i+1}) \in E^M$. The first player unable to move according to the rules, loses the game. Player I wins all infinite plays of the game.

The class K of all finite models M such that player II has a winning strategy in $G(M)$ is an equivalent to Immerman's *alternating graph accessibility problem*, AGAP. It is well known that AGAP is a complete problem for PTIME with respect to quantifier free reductions ([18]).

2.3 Greatest Fixed Point Logic

Let $\psi(R, \vec{x})$ be a first-order formula such that the arity of R, $\mathtt{ar}(R)$, is equal to the length $k = |\vec{x}|$ of the tuple $\vec{x}$. If M is a model, then ψ defines an operation $\Gamma = \Gamma_{M,\psi}$ on the set

[4] This was already shown in [12], in which it was shown that any dependence atom $=(\vec{t}_1, \vec{t}_2)$ is equivalent to the conditional independence atom $\vec{t}_2 \perp_{\vec{t}_1} \vec{t}_2$.

$\mathcal{P}(\mathtt{Dom}(M)^k)$ of k-ary relations on $\mathtt{Dom}(M)$ as follows:

$$\Gamma(P) := \{\vec{a} : (M, P) \models_{s[\vec{a}/\vec{x}]} \psi(R, \vec{x})\} \text{ for each } P \in \mathcal{P}(\mathtt{Dom}(M)^k).$$

A relation P is a *fixed point* of the operation $\Gamma_{M,\psi}$ on M if $\Gamma(P) = P$. Furthermore, P is the *greatest fixed point* (*least fixed point*) of $\Gamma_{M,\psi}$ if $Q \subseteq P$ ($P \subseteq Q$, respectively) for all fixed points Q of $\Gamma_{M,\psi}$.

It is well known that if R occurs only positively in ψ, then for every model M, $\Gamma_{M,\psi}$ has a greatest fixed point (and a least fixed point). Moreover, the greatest fixed point P of $\Gamma_{M,\psi}$ has the following characterization: $P = \bigcup\{Q \subseteq \mathtt{Dom}(M)^k : Q \subseteq \Gamma_{M,\psi}(Q)\}$ (see, e.g. [21]).

▶ **Definition 10.** *Greatest Fixpoint Logic*, GFP, is obtained by adding to First Order Logic the *greatest fixed point operator* $[\mathsf{gfp}_{R,\vec{x}}\psi(R, \vec{x})]\vec{t}$, where R is a relation variable with $\mathtt{ar}(R) = |\vec{x}|$, $\psi(R, \vec{x})$ is a formula in which R occurs only positively, and $\vec{t}$ is a tuple of terms with $|\vec{t}| = |\vec{x}|$. The semantics of the operator gfp is defined by the clause:

- $M \models_s [\mathsf{gfp}_{R,\vec{x}}\psi(R, \vec{x})]\vec{t}$ if and only if $\vec{t}\langle s\rangle$ is in the greatest fixed point of $\Gamma_{M,\psi}$.

Positive Greatest Fixed Point Logic, GFP^+, is the fragment of Greatest Fixed Point Logic in which fixed point operators occur only positively.

Least Fixpoint Logic, LFP, similarly, introduces an operator $[\mathsf{lfp}_{R,\vec{x}}\psi(R, \vec{x})]\vec{t}$, again for R occurring only positively in ψ, such that $M \models [\mathsf{lfp}_{R,\vec{x}}\psi(R, \vec{x})]\vec{t}$ if and only if $\vec{t}\langle s\rangle$ is in the least fixed point of $\Gamma_{M,\psi}$.

Fixed point logics have been the object of a vast amount of research, especially because of their applications in Finite Model Theory and Descriptive Complexity Theory. In particular, Least Fixed Point Logic captures the complexity class PTIME that consists of all problems that are solvable in polynomial time:

▶ **Theorem 11** ([17, 26]). *A class of linearly ordered finite models is definable in* LFP *if and only if it can be recognized in* PTIME.

Another important result is that on finite models, Greatest Fixed Point Logic has the same expressive power as Least Fixed Point Logic.

▶ **Theorem 12** ([17]). *Over finite models,* GFP^+ *(as well as* GFP*) is equivalent to* LFP.

We will also make use of the following normal form result for Positive Greatest Fixed Point Logic:

▶ **Theorem 13** ([23, 17]). *Every* GFP^+*-sentence* ϕ *is equivalent to a* GFP^+*-sentence of the form* $\exists\vec{z}[\mathsf{gfp}_{R,\vec{x}}\psi(R, \vec{x})]\vec{z}$*, where* ψ *is a first-order formula.*

3 Inclusion Logic captures GFP^+

We will now prove that Inclusion Logic has exactly the same expressive power as Positive Greatest Fixed Point Logic. Since the semantics of GFP^+ is defined in terms of single assignments instead of teams, the equivalence of $\mathrm{FO}(\subseteq)$ and GFP^+ on formulas has to be formulated in a bit indirect way; see Theorems 15 and 16 below.

We start with a lemma that connects teams and the greatest fixed point operator:

▶ **Lemma 14.** *Let* $\psi(S, \vec{x})$ *be a* GFP^+*-formula with free variables in* $\vec{x} = (x_1, \ldots, x_n)$ *such that* S *is* n*-ary and occurs only positively in* ψ*, let* M *be a model, and let* Y *a team on* M*.*
(a) *If* $(M, Y(\vec{x})) \models_s \psi(S, \vec{x})$ *for all* $s \in Y$*, then* $M \models_s [\mathsf{gfp}_{S,\vec{x}}\psi(S, \vec{x})]\vec{x}$ *for all* $s \in Y$*.*

(b) *If Y is a maximal team such that $M \models_s [\mathsf{gfp}_{S,\vec{x}}\, \psi(S,\vec{x})]\vec{x}$ for all $s \in Y$, then $(M, Y(\vec{x})) \models_s \psi(S,\vec{x})$ for all $s \in Y$.*

Proof. Note that $(M, Y(\vec{x})) \models_s \psi(S,\vec{x})$ for all $s \in Y$ if and only if $Y(\vec{x}) \subseteq \Gamma_{M,\psi}(Y(\vec{x}))$. Thus, claim (a) follows from the fact that the greatest fixed point of $\Gamma_{M,\psi}$ is the union of all relations Q such that $Q \subseteq \Gamma_{M,\psi}(Q)$. Claim (b) follows from the observation that if Y is a maximal team such that $M \models_s [\mathsf{gfp}_{S,\vec{x}}\, \psi(S,\vec{x})]\vec{x}$ for all $s \in Y$, then $Y(\vec{x})$ is the greatest fixed point of $\Gamma_{M,\psi}$. ◀

We will next prove that every FO($\subseteq$)-formula can be expressed in GFP$^+$.

▶ **Theorem 15.** *For every* FO($\subseteq$)*-formula $\phi(\vec{x})$ with free variables in $\vec{x} = (x_1, \ldots, x_n)$ there is a* GFP$^+$*-formula $\phi^* = \phi^*(R,\vec{x})$ such that $\mathtt{ar}(R) = |\vec{x}|$, R occurs only positively in ϕ^*, and*

$$M \models_X \phi(\vec{x}) \iff (M, X(\vec{x})) \models_s \phi^*(R,\vec{x}) \text{ for all } s \in X$$

holds for all models M and teams X with $\mathtt{Dom}(X) = \{x_1, \ldots, x_n\}$.

Proof. The proof is by structural induction on ϕ.

1. If $\phi(\vec{x})$ is a first-order literal, let $\phi^*(R,\vec{x})$ be just $\phi(\vec{x})$. Then we have
$$\begin{aligned} M \models_X \phi(\vec{x}) &\iff M \models_s \phi(\vec{x}) \text{ for all } s \in X \\ &\iff (M, X(\vec{x})) \models_s \phi(\vec{x}) \text{ for all } s \in X. \end{aligned}$$
2. If $\phi(\vec{x})$ is an inclusion atom $\vec{t}_1 \subseteq \vec{t}_2$, let $\phi^*(R,\vec{x})$ be $\exists\vec{z}(R\vec{z} \wedge \vec{t}_1(\vec{x}) = \vec{t}_2(\vec{z}))$, where $\vec{z}$ is a tuple of new variables. Note that $(M, X(\vec{x})) \models_h \vec{t}_1(\vec{x}) = \vec{t}_2(\vec{z})$ for an assignment h defined on $\vec{x}\vec{z}$ if and only if there are two assignments s, s' defined on $\vec{x}$ such that $\vec{t}_1\langle s\rangle = \vec{t}_2\langle s'\rangle$ and $h = s \cup (s' \circ f)$, where f is the function $f(z_i) = x_i$. Thus, we see that $(M, X(\vec{x})) \models_s \phi^*(R,\vec{x})$ for all $s \in X$ if and only if for every $s \in X$ there is an $s' \in X$ such that $\vec{t}_1\langle s\rangle = \vec{t}_2\langle s'\rangle$, as desired.
3. Assume next that $\phi(\vec{x})$ is of the form $\psi(\vec{x}) \vee \theta(\vec{x})$. Then we define
$$\phi^*(R,\vec{x}) := [\mathsf{gfp}_{S,\vec{x}}\, (R\vec{x} \wedge \psi^*(S,\vec{x}))]\vec{x} \vee [\mathsf{gfp}_{T,\vec{x}}\, (R\vec{x} \wedge \theta^*(T,\vec{x}))]\vec{x}.$$
If $M \models_X \phi(\vec{x})$, then there exist teams Y and Z such that $X = Y \cup Z$, $M \models_Y \psi(\vec{x})$ and $M \models_Z \theta(\vec{x})$. By induction hypothesis, $(M, Y(\vec{x})) \models_s \psi^*(S,\vec{x})$, and consequently $(M, X(\vec{x}), Y(\vec{x})) \models_s R\vec{x} \wedge \psi^*(S,\vec{x})$, holds for all $s \in Y$. Hence, by Lemma 14, $(M, X(\vec{x})) \models_s [\mathsf{gfp}_{S,\vec{x}}\, (R\vec{x} \wedge \psi^*(S,\vec{x}))]\vec{x}$ holds for all $s \in Y$.
In the same way we see that $(M, X(\vec{x})) \models_s [\mathsf{gfp}_{T,\vec{x}}\, (R\vec{x} \wedge \theta^*(T,\vec{x}))]\vec{x}$ holds for all $s \in Z$. Thus, we conclude that $(M, X(\vec{x})) \models_s \phi^*(R,\vec{x})$ for all $s \in X$.
To prove the converse, assume that $(M, X(\vec{x})) \models_s \phi^*(R,\vec{x})$ for all $s \in X$. Let Y be the set of all assignments $s \in X$ that satisfy the first disjunct of $\phi^*(R,\vec{x})$, and let Z be the set of assignments $s \in X$ that satisfy the second disjunct. Then Y is the maximal team such that, for all $s \in Y$, $(M, X(\vec{x})) \models_s [\mathsf{gfp}_{S,\vec{x}}\, (R\vec{x} \wedge \psi^*(S,\vec{x}))]\vec{x}$. It follows from Lemma 14 that $(M, X(\vec{x}), Y(\vec{x})) \models_s R\vec{x} \wedge \psi^*(S,\vec{x})$ for all $s \in Y$. Thus, $(M, Y(\vec{x})) \models_s \psi^*(S,\vec{x})$ for all $s \in Y$, and by induction hypothesis, $M \models_Y \psi(\vec{x})$. In the same way we see that $M \models_Z \theta(\vec{x})$. Finally, since $X = Y \cup Z$, we conclude that $M \models_X \phi(\vec{x})$.
4. If $\phi(\vec{x}) = \psi(\vec{x}) \wedge \theta(\vec{x})$, we define simply $\phi^*(R,\vec{x}) := \psi^*(R,\vec{x}) \wedge \theta^*(R,\vec{x})$. The claim follows then directly from the induction hypothesis.
5. If $\phi(\vec{x})$ is of the form $\exists v\, \psi(\vec{x}v)$, let $\phi^*(R,\vec{x})$ be $\exists v[\mathsf{gfp}_{S,\vec{x}v}\, (R\vec{x} \wedge \psi^*(S,\vec{x}v))]\vec{x}v$. Then $M \models_X \phi(\vec{x})$ if and only if there is a function $H \in \mathcal{C}(X, \mathtt{Dom}(M))$ such that $M \models_Y \psi(\vec{x}v)$,

where $Y = X[H/v]$. By the induction hypothesis, this is equivalent to $(M, Y(\vec{x}v)) \models_h \psi^*(S, \vec{x}v)$ being true for all $h \in Y$. This, in turn, is equivalent with the condition

$$(M, X(\vec{x}), Y(\vec{x}v)) \models_h R\vec{x} \wedge \psi^*(S, \vec{x}v) \text{ for all } h \in Y. \tag{1}$$

If condition (1) holds, then by Lemma 14, $(M, X(\vec{x})) \models_h [\mathsf{gfp}_{S,\vec{x}v}(R\vec{x} \wedge \psi^*(S, \vec{x}v))]\vec{x}v$ holds for all $h \in Y$. Since every $s \in X$ has an extension $h \in Y$, it follows that $(M, X(\vec{x})) \models_s \phi^*(R, \vec{x})$ for all $s \in X$.
On the other hand, if $(M, X(\vec{x})) \models_s \phi^*(R, \vec{x})$ for all $s \in X$, we define $H \in \mathcal{C}(X, \mathtt{Dom}(M))$ to be the function such that

$$H(s) := \{a \in \mathtt{Dom}(M) : (M, X(\vec{x})) \models_{s[a/v]} [\mathsf{gfp}_{S,\vec{x}v}(R\vec{x} \wedge \psi^*(S, \vec{x}v))]\vec{x}v\},$$

and let $Y = X[H/v]$. Then Y is the maximal team such that

$$(M, X(\vec{x})) \models_h [\mathsf{gfp}_{S,\vec{x}v}(R\vec{x} \wedge \psi^*(S, \vec{x}v))]\vec{x}v$$

for all $h \in Y$, whence condition (1) follows from Lemma 14.

6. If $\phi(\vec{x})$ is of the form $\forall v\, \psi(\vec{x}v)$, let $\phi^*(R, \vec{x})$ be $\forall v[\mathsf{gfp}_{S,\vec{x}v}(R\vec{x} \wedge \psi^*(S, \vec{x}v))](\vec{x}v)$. The proof of the claim is similar to the case of existential quantification.

◀

In proving that GFP$^+$-sentences can be expressed in FO($\subseteq$) we will use the normal form given in Theorem 13. Thus, it suffices to find translations for first-order formulas, and formulas obtained by a single application of the gfp-operator to first-order formulas.

▶ **Theorem 16.** *Let $\eta(R, \vec{x}, \vec{y})$ be a first-order formula such that R occurs only positively in η, $\mathtt{ar}(R) = |\vec{x}| = n$, and the free variables of η are in $\vec{x}\vec{y}$.*

(a) *There exists an* FO($\subseteq$)*-formula $\eta^+(\vec{x}, \vec{y})$ such that for all models M and teams X on M*

$$M \models_X \eta^+(\vec{x}, \vec{y}) \iff (M, X(\vec{x})) \models_s \eta(R, \vec{x}, \vec{y}) \text{ for every } s \in X$$

(b) *If $\vec{y}$ is empty, and $\vec{z}$ is an n-tuple of variables not occurring in η, then there exists an* FO($\subseteq$)*-formula $\tilde{\eta}(\vec{z})$ such that for all models M and teams X on M*

$$M \models_X \tilde{\eta}(\vec{z}) \iff M \models_s [\mathsf{gfp}_{R,\vec{x}}\, \eta(R, \vec{x})]\vec{z} \text{ for every } s \in X$$

Proof. (a) We prove the claim by structural induction on η.

1. If $\eta(R, \vec{x}, \vec{y})$ is a first-order literal not containing the relation symbol R, we define $\eta^+ := \eta$. Then $M \models_X \eta^+$ if and only if $M \models_s \eta$ for every $s \in X$. Since R does not occur in η, this is equivalent with $(M, X(\vec{x})) \models_s \eta$ for all $s \in X$, as required.

2. If η is of the form $R\vec{t}$, we define $\eta^+(\vec{x}, \vec{y}) := \vec{t} \subseteq \vec{x}$. Then we have

$$\begin{aligned} M \models_X \eta^+(\vec{x}, \vec{y}) &\iff \forall s \in X\, \exists s' \in X : \vec{t}\langle s\rangle = \vec{x}\langle s'\rangle \\ &\iff \forall s \in X : \vec{t}\langle s\rangle \in X(\vec{x}) \\ &\iff \forall s \in X : (M, X(\vec{x})) \models_s R\vec{t}. \end{aligned}$$

3. If η is of the form $\alpha(R, \vec{x}, \vec{y}) \vee \beta(R, \vec{x}, \vec{y})$, let $\vec{u} = (u_1, \ldots, u_n)$ be a tuple of new variables and let $\eta^+(\vec{x}, \vec{y})$ be the formula $\exists\vec{u}\big((\vec{u} \subseteq \vec{x}) \wedge (\alpha^+(\vec{u}, \vec{x}\vec{y}) \vee \beta^+(\vec{u}, \vec{x}\vec{y}))\big)$. Here we assume as induction hypothesis that $M \models_Y \alpha^+(\vec{u}, \vec{x}\vec{y})$ if and only if $(M, Y(\vec{u})) \models_h \alpha(R, \vec{x}, \vec{y})$ for all $h \in Y$, and similarly for $\beta^+(\vec{u}, \vec{x}\vec{y})$ and $\beta(R, \vec{x}, \vec{y})$.
Suppose first that $M \models_X \eta^+(\vec{x}, \vec{y})$. Then there is a function $H \in \mathcal{C}(X, \mathtt{Dom}(M)^n)$ such that $X[H/\vec{u}](\vec{u}) \subseteq X(\vec{x})$, and furthermore, $X[H/\vec{u}]$ can be split into two subteams Y

and Z such that $M \models_Y \alpha^+(\vec{u}, \vec{x}\vec{y})$ and $M \models_Z \beta^+(\vec{u}, \vec{x}\vec{y})$. Now take any $s \in X$ and let $h \in X[H/\vec{u}]$ be an extension of s. If $h \in Y$ then $(M, Y(\vec{u})) \models_h \alpha(R, \vec{x}, \vec{y})$. Since $Y(\vec{u}) \subseteq X[H/\vec{u}](\vec{u}) \subseteq X(\vec{x})$, $\vec{x}\vec{y}\langle h \rangle = \vec{x}\vec{y}\langle s \rangle$ and R occurs only positively in α, we have $(M, X(\vec{x})) \models_s \alpha(R, \vec{x}, \vec{y})$. Similarly, if $h \in Z$ then $(M, X(\vec{x})) \models_s \beta(R, \vec{x}, \vec{y})$. Thus, $(M, X(\vec{x})) \models_s \alpha(R, \vec{x}, \vec{y}) \vee \beta(R, \vec{x}, \vec{y})$ for all $s \in X$, as required.

Conversely, suppose that for any $s \in X$, $(M, X(\vec{x})) \models_s \alpha(R, \vec{x}, \vec{y}) \vee \beta(R, \vec{x}, \vec{y})$. Now let $H \in \mathcal{C}(X, \mathtt{Dom}(M)^n)$ be the function such that $H(s) = X(\vec{x})$ for all $s \in X$. Note first that clearly $M \models_{X[H/\vec{u}]} \vec{u} \subseteq \vec{x}$. Let $Y = \{h \in X[H/\vec{u}] : (M, X(\vec{x})) \models_h \alpha(R, \vec{x}, \vec{y})\}$ and $Z = \{h \in X[H/\vec{u}] : (M, X(\vec{x})) \models_h \beta(R, \vec{x}, \vec{y})\}$. By hypothesis, $X[H/\vec{u}] = Y \cup Z$.
If $Y \neq \emptyset$, then $Y(\vec{u}) = X[H/\vec{u}](\vec{u}) = X(\vec{x})$: indeed, if $(M, X(\vec{x})) \models_h \alpha(R, \vec{x}, \vec{y})$ then the same holds for all h' which differ from h only with respect to $\vec{u}$, since $\vec{u}$ is not free in α. Therefore $(M, Y(\vec{u})) \models_h \alpha(R, \vec{x}, \vec{y})$ for all $h \in Y$, and thus $M \models_Y \alpha^+(\vec{u}, \vec{x}\vec{y})$. If instead $Y = \emptyset$, then $M \models_Y \alpha^+(\vec{u}, \vec{x}\vec{y})$ trivially. Similarly, $M \models_Z \beta^+(\vec{u}, \vec{x}\vec{y})$, and therefore $M \models_{X[H/\vec{u}]} \alpha^+(\vec{u}, \vec{x}\vec{y}) \vee \beta^+(\vec{u}, \vec{x}\vec{y})$, whence the function H witnesses that $M \models_X \eta^+$.

4. If η is $\alpha(R, \vec{x}, \vec{y}) \wedge \beta(R, \vec{x}, \vec{y})$, let $\eta^+(\vec{x}, \vec{y})$ be $\alpha^+(\vec{x}, \vec{y}) \wedge \beta^+(\vec{x}, \vec{y})$. Then the claim follows directly from the induction hypothesis.
5. If $\eta(R, \vec{x}, \vec{y})$ is $\exists v\, \alpha(R, \vec{x}, \vec{y}v)$, let $\eta^+(\vec{x}, \vec{y})$ be $\exists v\, \alpha^+(\vec{x}, \vec{y}v)$; here we assume w.l.o.g. that v is not among the variables in $\vec{x}\vec{y}$. Then $M \models_X \eta^+(\vec{x}, \vec{y})$ if and only if there is a function $H \in \mathcal{C}(X, \mathtt{Dom}(M))$ such that $M \models_{X[H/v]} \alpha^+(\vec{x}, \vec{y}v)$. Since $X[H/v](\vec{x}) = X(\vec{x})$, by induction hypothesis this is equivalent with the condition

$$(M, X(\vec{x})) \models_h \alpha(R, \vec{x}, \vec{y}v) \text{ holds for all } h \in X[H/v]. \tag{2}$$

If condition (2) is true, then clearly $(M, X(\vec{x})) \models_s \eta(R, \vec{x}, \vec{y})$ for all $s \in X$. Conversely, if $(M, X(\vec{x})) \models_s \eta(R, \vec{x}, \vec{y})$ holds for all $s \in X$, then (2) is true for the function H such that $H(s) = \{a \in \mathtt{Dom}(M) : (M, X(\vec{x})) \models_{s[a/v]} \alpha(R, \vec{x}, \vec{y}v)\}$.
6. If $\eta(\vec{R}, \vec{x}, \vec{y})$ is $\forall v\, \alpha(\vec{R}, \vec{x}, \vec{y}v)$, let $\eta^+(\vec{x}, \vec{y})$ be $\forall v\, \alpha^+(\vec{x}, \vec{y}v)$. The proof of the claim is similar as in the previous case.

(b) Let $\vec{z}$ be an n-tuple of variables not occurring in η. We define $\tilde{\eta}(\vec{z})$ to be the formula $\exists \vec{x}(\vec{z} \subseteq \vec{x} \wedge \eta^+(\vec{x}))$, where η^+ is the FO($\subseteq$)-formula corresponding to $\eta(R, \vec{x})$, as given in claim (a). Suppose first that $M \models_X \tilde{\eta}(\vec{z})$. Then there is a function $H \in \mathcal{C}(X, \mathtt{Dom}(M)^n)$ such that $M \models_Y \eta^+(\vec{x})$, and $\vec{z}\langle h \rangle \in Y(\vec{x})$ for all $h \in Y$, where $Y = X[H/\vec{x}]$. Thus, by claim (a), $(M, Y(\vec{x})) \models_h \eta(R, \vec{x})$ holds for all $h \in Y$. It follows now from Lemma 14 that $M \models_h [\mathsf{gfp}_{R,\vec{x}}\, \eta(R, \vec{x})]\vec{x}$ for all $h \in Y$. Since every $s \in X$ has an extension $h \in Y$, and $\vec{z}\langle s \rangle = \vec{z}\langle h \rangle \in Y(\vec{x})$, we conclude that $M \models_s [\mathsf{gfp}_{R,\vec{x}}\, \eta(R, \vec{x})]\vec{z}$ for all $s \in X$.

To prove the converse, assume that $M \models_s [\mathsf{gfp}_{R,\vec{x}}\, \eta(R, \vec{x})]\vec{z}$ for all $s \in X$. Let P be the greatest fixed point of the formula $\eta(R, \vec{x})$ (with respect to R and $\vec{x}$) on the model M, and let $H \in \mathcal{C}(X, \mathtt{Dom}(M)^n)$ be the function such that $H(s) = P$ for every $s \in X$. Let $Y = X[H/\vec{x}]$. Then $(M, Y(\vec{x})) \models_h \eta(R, \vec{x})$ for all $h \in Y$, whence by claim (a), we have $M \models_Y \eta^+(\vec{x})$. Moreover, $\vec{z}\langle h \rangle \in Y(\vec{x}) = P$ for all $h \in H$, whence $M \models_Y \vec{z} \subseteq \vec{x}$. Thus, the function H witnesses that $M \models_X \exists \vec{x}(\vec{z} \subseteq \vec{x} \wedge \eta^+(\vec{x}))$. ◀

Note that in the case of disjunction above, it was necessary to "store" the possible values of $\vec{x}$ into the values of a new tuple $\vec{u}$ of variables: otherwise, by splitting the team X into two subteams we could have lost information about $X(\vec{x})$.

The equivalence of FO($\subseteq$) and GFP$^+$ follows now from the two theorems above:

▶ **Corollary 17.** *For any* FO($\subseteq$)*-sentence* ϕ *there exists an equivalent* GFP$^+$*-sentence* θ*, and vice versa.*

Proof. If ϕ is an FO($\subseteq$)-sentence, then by Theorem 15, there is a formula $\phi^*(R,x)$ such that for all models M and teams X, $M \models_X \phi$ if and only if $(M, X(x)) \models_s \phi^*(R,x)$ for all $s \in X$. Thus, $M \models \phi$ if and only if $M \models \forall x\,[\mathsf{gfp}_{R,x}\,\phi^*(R,x)]x$.

On the other hand, if ψ is a GFP$^+$-sentence, then by Theorem 13, we can assume that it is of the form $\exists\vec{z}\,[\mathsf{gfp}_{R,\vec{x}}\,\eta(R,\vec{x})]\vec{z}$, where η is a first-order formula. It follows now from Theorem 16(b) that ψ is equivalent to the FO($\subseteq$)-sentence $\exists\vec{z}\,\tilde{\eta}(\vec{z})$. ◀

▶ **Corollary 18.** *A class of linearly ordered finite models is definable in* FO($\subseteq$) *if and only if it can be recognized in* PTIME.

This connection between Inclusion Logic, Fixed Point Logic and descriptive complexity may be of great value for the further development of the area. In particular, it implies that fragments and extensions of FO($\subseteq$) can be made to correspond to various fragments and extensions of PTIME. Hence, results concerning their relationships may lead to insights which may be valuable in complexity theory, and vice versa.

4 First-Order Union Closed Properties

From Corollary 17 it follows immediately that Inclusion Logic is strictly weaker than Σ^1_1. As an immediate consequence, not all Σ^1_1-definable union-closed properties of relations can be expressed in Inclusion Logic. For example, consider the atom

TS-$\mathcal{R}$: $M \models \mathcal{R}(xyzw)$ if and only if there exist two functions $f, g : \mathtt{Dom}(M) \to \mathtt{Dom}(M)$ such that, for all $a, b \in \mathtt{Dom}(M)$, $(a, f(a), b, g(b)) \in X(xyzw)$.

It is easy to see that the atom $\mathcal{R}$ is union-closed. On the other hand, it can be seen that that the sentence $\forall x\exists y\forall z\exists w(\mathcal{R}(xyzw) \wedge (x = z \leftrightarrow y = w) \wedge (y = z \to x = w) \wedge x \neq y)$ holds in a finite model if and only if it contains an even number of elements.[5] Since even cardinality is not definable in GFP, it follows that $\mathcal{R}$ is not definable in FO($\subseteq$).

But what about first order definable union-closed properties? As we will now see, all such properties are indeed definable in Inclusion Logic; and, therefore, it is not possible to increase the expressive power of Inclusion Logic by adding any first order definable union-closed dependency.

▶ **Definition 19.** A sentence $\phi(R)$ is *myopic* if it is of the form $\forall\vec{x}(R\vec{x} \to \theta(R,\vec{x}))$ for some first-order formula θ in which R occurs only positively.

It follows at once from Theorem 16 that myopic sentences correspond to Inclusion Logic-definable properties:

▶ **Proposition 20.** *Let $\phi(R) = \forall\vec{x}(R\vec{x} \to \theta(R,\vec{x}))$ be a myopic sentence. Then there exists an* FO($\subseteq$)*-formula $\phi^+(\vec{x})$ such that, for all models M and teams X,*

$M \models_X \phi^+(\vec{x})$ *if and only if* $(M, X(\vec{x})) \models \phi(R)$.

Proof. Consider $\theta(R,\vec{x})$: by Theorem 16(a), there exists an FO($\subseteq$)-formula $\theta^+(\vec{x})$ such that for all models M and teams X,

$$\begin{aligned} M \models_X \theta^+(\vec{x}) &\iff \forall s \in X : (M, X(\vec{x})) \models_s \theta(R,\vec{x}) \\ &\iff (M, X(\vec{x})) \models \forall\vec{x}(R\vec{x} \to \theta(R,\vec{x})), \end{aligned}$$

as required. ◀

[5] The proof of this fact mirrors that of the example in [24], §4.1. In brief, the sentence asserts that the function f mapping x to y is the same as the function g mapping z to w, that this function is an involution, and that this function has no fixed points.

It is also easy to see that all myopic properties are union-closed. We will now prove the converse implication: if $\phi(R)$ is a first order sentence that defines a union-closed property of relations, then it is equivalent to some myopic sentence. From this preservation theorem it will follow at once that all union-closed first-order properties of relations are definable in Inclusion Logic.

First, let us recall some model-theoretic machinery:

▶ **Definition 21** (ω-big models)**.** A model A of signature Σ is ω-big if for all finite tuples $\vec{a}$ of elements of it and for all models $(B, \vec{b}, S)$ such that $(A, \vec{a}) \equiv (B, \vec{b})$ there exists a relation P over A such that $(A, \vec{a}, P) \equiv (B, \vec{b}, S)$.

▶ **Definition 22** (ω-saturated models)**.** A model A is ω-saturated if for every finite set C of elements of A, all complete 1-types over C with respect to A are realized in A.

The proofs of the following model-theoretic results can be found in [15].

▶ **Theorem 23** ([15], Theorem 8.2.1)**.** *Let A be a model. Then A has an ω-big elementary extension.*

▶ **Theorem 24** ([15], Lemma 8.3.4)**.** *Let A and B be ω-saturated structures over a finite signature and such that, for all sentences $\chi(R)$ in which R occurs only positively,*

$$A \models \chi(R) \implies B \models \chi(R).$$

Then there are elementary substructures C and D of A and B and a bijective homomorphism $f : C \to D$ which fixes all relation symbols except R.

▶ **Theorem 25** (Essentially [15], Theorem 8.1.2)**.** *Suppose that A is ω-big and $\vec{a}$ is a finite tuple of elements. Then $(A, \vec{a})$ is ω-saturated.*

Using these results, we can prove our representation theorem:

▶ **Theorem 26.** *Let $\phi(R)$ be a first order sentence that defines a union-closed property of R. Then ϕ is equivalent to some myopic sentence. Consequently, every first-order definable union-closed property of relations is definable in* FO($\subseteq$).

Proof. Let $T = \{\phi'(R) : \phi'(R) \text{ is myopic}, \phi(R) \models \phi'(R)\}$. If we can show that $T \models \phi(R)$, we are done: indeed, by compactness this implies that ϕ is equivalent to a finite conjunction $\forall\vec{x}(R\vec{x} \to \theta_1(R, \vec{x})) \wedge \ldots \wedge \forall\vec{x}(R\vec{x} \to \theta_n(R, \vec{x}))$ of myopic sentences, which of course is equivalent to $\forall\vec{x}(R\vec{x} \to (\theta_1(R, \vec{x}) \wedge \ldots \wedge \theta_n(R, \vec{x})))$.

So, let B' be a model satisfying T, and let B be an ω-big elementary extension of B'. We need to show that $B \models \phi(R)$ (and, therefore, $B' \models \phi(R)$).

Now choose an arbitrary tuple $\vec{b}$ of elements such that $B \models R\vec{b}$, and let Γ be the theory

$$\Gamma = \{R\vec{a}, \phi(R)\} \cup \{\psi(R, \vec{a}) : R \text{ only negative in } \psi, B \models \psi(R, \vec{b})\}.$$

Γ is satisfiable: indeed, if it were not then by compactness there would be formulas $\psi_1(R, \vec{x}), \ldots, \psi_n(R, \vec{x})$ in which R occurs only negatively such that

$$\phi(R) \models \forall\vec{x}\Big(R\vec{x} \to \bigvee_{1 \le i \le n} \neg\psi_i(R, \vec{x})\Big).$$

But this is a myopic formula, and therefore it would have to hold in B, which is a contradiction since $B \models \psi_i(R, \vec{b})$ for all $1 \le i \le n$.

Now let $(A, \vec{a})$ be an ω-saturated model of Γ. If R occurs only positively in $\chi(R, \vec{x})$ and $A \models \chi(R, \vec{a})$, then $B \models \chi(R, \vec{b})$; otherwise $\neg\chi(R, \vec{a})$ would be in Γ. Furthermore, since B is ω-big, $(B, \vec{b})$ is ω-saturated. Thus, there are elementary substructures $(C, \vec{a})$ and $(D, \vec{b})$ of $(A, \vec{a})$ and $(B, \vec{b})$ and a bijective homomorphism $f : C \to D$ that fixes all relations except R.

Let $S = f(R^C)$. Then $S \subseteq R^D$, since f is an homomorphism; and f is actually an isomorphism between $(C, \vec{a})$ and $(D[S/R], \vec{b})$, since f fixes even R between these two models. Now, $C \models R\vec{a} \wedge \phi(R)$, whence $D \models S\vec{b} \wedge \phi(S)$. Furthermore, since $S \subseteq R$ we have that $D \models \forall\vec{x}(S\vec{x} \to R\vec{x})$.

Now, $(D, \vec{b})$ is an elementary substructure of $(B, \vec{b})$ and B is a ω-big model: therefore, there exists a relation P over B such that $(D, \vec{b}, S) \equiv (B, \vec{b}, P)$. In particular, this implies that $B \models P\vec{b} \wedge \phi(P) \wedge P \subseteq R$: there is a subset of R^B which contains $\vec{b}$ and satisfies ϕ.

But we chose $\vec{b}$ as an arbitrary tuple in R^B. So we have that R^B is the union of a family of relations $P_{\vec{b}}$, where $\vec{b}$ ranges over R^B; and $B \models \phi(P_{\vec{b}})$ for all such $\vec{b}$. Since $\phi(R)$ is closed under unions, this implies that $B \models \phi(R)$, as required. ◀

5 An EF Game for Inclusion Logic

We will now define an Ehrenfeucht-Fraïssé game for Inclusion Logic. This game is an obvious variant of the one defined in [24] for Dependence Logic:

▶ **Definition 27.** Let A and B be two models over the same signature, let $n \in \mathbb{N}$, and let X and Y be two teams with the same domain over A and B, respectively. Then the two-player game $G_n(A, X, B, Y)$ is defined as follows:

1. The initial position p_0 is (X, Y);
2. For each $i \in \{1, \ldots, n\}$, let p_{i-1} be (X_{i-1}, Y_{i-1}). Then Spoiler makes a move of one of the following types:
   - **Splitting:** Spoiler chooses two teams X', X'' such that $X_{i-1} = X' \cup X''$. Then Duplicator chooses two teams Y', Y'' such that $Y_{i-1} = Y' \cup Y''$. Then Spoiler chooses whether the next position p_i is (X', Y') or (X'', Y'').
   - **Supplementing:** Spoiler chooses a variable v and a function $H : X_{i-1} \to \mathcal{P}(\mathtt{Dom}(A))\backslash\{\emptyset\}$. Then Duplicator chooses a function $K : Y_{i-1} \to \mathcal{P}(\mathtt{Dom}(B))\backslash\{\emptyset\}$, and the new position p_i is $(X_{i-1}[H/v], Y_{i-1}[K/v])$.
   - **Duplication:** Spoiler chooses a variable v. The next position p_i is $(X_{i-1}[A/v], Y_{i-1}[B/v])$.
3. The final position $p_n = (X_n, Y_n)$ is *winning for Spoiler* if and only if there exists a formula α which is either a first-order literal, or an inclusion atom, such that $A \models_{X_n} \alpha$, but $B \not\models_{Y_n} \alpha$. Otherwise, the final position is winning for Duplicator.

The rank of an Inclusion Logic formula is also defined much in the same way as the rank of a Dependence Logic formula:

▶ **Definition 28.** Let ϕ be an FO($\subseteq$)-formula. Then we define its *rank* $\mathtt{rk}(\phi) \in \mathbb{N}$ by structural induction on ϕ, as follows:

1. If ϕ is a first-order literal or an inclusion atom, $\mathtt{rk}(\phi) = 0$;
2. $\mathtt{rk}(\psi \wedge \theta) = \max(\mathtt{rk}(\psi), \mathtt{rk}(\theta))$;
3. $\mathtt{rk}(\psi \vee \theta) = \max(\mathtt{rk}(\psi), \mathtt{rk}(\theta)) + 1$;
4. $\mathtt{rk}(\exists v\psi) = \mathtt{rk}(\forall v\psi) = \mathtt{rk}(\psi) + 1$.

The next theorem shows that our games behave as required with respect to our notion of rank. Its proof is practically the same as for the EF game for FO(D) in [24].

▶ **Theorem 29.** *Let A and B be models and X and Y teams on A and B. Then Duplicator has a winning strategy in $G_n(A, X, B, Y)$ if and only if*

$$A \models_X \phi \implies B \models_Y \phi$$

holds for all FO($\subseteq$)*-formulas ϕ with* $\mathtt{rk}(\phi) \leq n$.

Due to the equivalence between FO($\subseteq$) and GFP$^+$ we can conclude at once that the EF game for Inclusion Logic is also a novel EF game for GFP$^+$, rather different in structure from the one introduced in [2]. It may be hoped that this new game and its variants could be of some use for studying the expressive power of fixed point logics.

Although the EF game for Inclusion Logic has a clear second order flavour, it is still manageable: we will next show that Duplicator has a concrete winning strategy, when the models are simple enough.

▶ **Proposition 30.** *Let $A = \{1, \ldots, n\}$ and $B = \{1, \ldots, n+1\}$ be two finite models over the empty signature. Then for all* FO($\subseteq$)*-sentences ϕ of rank $\leq n$,*

$$A \models \phi \implies B \models \phi.$$

Proof. It suffices to specify a winning strategy for Duplicator in the game $G_n(A, \{\emptyset\}, B, \{\emptyset\})$. Our aim for such a strategy is to preserve the following property for n turns:

- If the current position is (X, Y) then

$$Y = \bigcup \{\pi[X] : \pi \in I(A, B)\}, \tag{3}$$

where $I(A, B)$ is the set of all 1-1 functions $A \to B$, $\pi[X] = \{\pi(s) : s \in X\}$ and $\pi(s)$ denotes the assignment $\pi \circ s$.

The property (3) is trivially true for $(\{\emptyset\}, \{\emptyset\})$. Furthermore, as long as (3) holds, Spoiler does not win. Indeed, if α is a first-order literal such that $A \models_s \alpha$ for all $s \in X$, then, since all $s' \in Y$ are of the form $\pi(s)$ for some $s \in X$ and the signature is empty, we have $B \models_{s'} \alpha$ for all $s' \in Y$. Similarly, suppose that $A \models_X \vec{u} \subseteq \vec{w}$, and let $s' \in Y$. Then $s' = \pi(s)$ for some $s \in X$ and some $\pi \in I(A, B)$, and there exists a $h \in X$ such that $\vec{u}\langle s \rangle = \vec{w}\langle h \rangle$. But then $\pi(h) \in Y$, and $\vec{w}\langle \pi(h) \rangle = \vec{u}\langle \pi(s) \rangle = \vec{u}\langle s' \rangle$, as required.

Thus, we only need to verify that Duplicator can maintain property (3) for n rounds. Suppose that at round $i < n$ the current position (X, Y) has property (3), and let us consider the possible moves of Spoiler:

Splitting: Suppose that Spoiler splits X into X_1 and X_2. Then let Duplicator reply by splitting Y into $Y_j = \{s' \in Y : \exists \pi \in I(A, B) \exists s \in X_j \text{ such that } \pi(s) = s'\}$ for $j \in \{1, 2\}$. Then $Y = Y_1 \cup Y_2$, and it is straightforward to check that both possible successors (X_1, Y_1) and (X_2, Y_2) have property (3).

Supplementing: Suppose that Spoiler chooses a function $H \in \mathcal{C}(X, A)$. Then let Duplicator reply with the function $K \in \mathcal{C}(Y, B)$ defined as

$$K(s') = \{\pi(a) : \exists \pi \in I(A, B) \exists s \in X \text{ such that } \pi(s) = s' \text{ and } a \in H(s)\}$$

for each $s' \in Y$. We leave it to the reader to verify that the next position $(X[H/v], Y[K/v])$ has property (3).

Duplication: If Spoiler chooses a duplication move, the next position is $(X[M/v], Y[M/v])$. We check that this new position satisfies property (3).

Let $s[a/v] \in X[A/v]$ and let $\pi \in I(A, B)$. Since $s \in X$, we have that $\pi(s) \in Y$, and therefore $\pi(s)[\pi(a)/v] = \pi(s[a/v]) \in Y[B/v]$.
Conversely, let $s' \in Y$ and let b be any element of B. We need to show that $s'[b/v] = \pi(s[a/v])$ for some $\pi \in I(A, B)$, $s \in X$ and $a \in \mathtt{Dom}(A)$.
By induction hypothesis, there exists $\pi \in I(A, B)$ and $s \in X$ such that $\pi(s) = s'$. If b is in the range of π, then $s'[b/v] = \pi(s[a/v])$, where $a = \pi^{-1}(b)$. On the other hand, if b is not in the range of π, then since $i < n$, there is an element $a \in A$ which is not in the range of s. Now $s[a/v] \in X[A/v]$, and $s'[b/v] = \pi'(s[a/v])$, where $\pi' \in I(A, B)$ is a function such that $\pi'(a) = b$ and $\pi'(c) = \pi(c)$ for all c in the range of s. ◀

From Proposition 30 it immediately follows that *even cardinality* (and other similar cardinality properties) of finite models is not definable in Inclusion Logic. This, of course, follows already from the equivalence of FO($\subseteq$) and GFP$^+$, as it is well-known that non-trivial cardinality properties are not definable in fixed point logics.

6 Conclusions and Further Work

In this work, we proved a number of results concerning the expressive power of inclusion Logic. We showed that this logic is strictly weaker than Σ^1_1, and corresponds in fact to Positive Greatest Fixed Point Logic. Furthermore, we showed that all union-closed first-order properties of relations correspond to the satisfaction conditions of Inclusion Logic formulas, and we also defined a new Ehrenfeucht-Fraïssé game for it.

Due to the connection between Inclusion Logic and fixed point logics, the study of this formalism may have interesting applications in descriptive complexity theory. In [5], Durand and Kontinen established some correspondences between fragments of Dependence Logic and fragments of NP; in the same way, one may hope to find correspondences between fragments of Inclusion Logic and fragments of PTIME.

Furthermore, we may inquire about extensions of Inclusion Logic. For example, is there any natural union-closed dependency notion $\mathbf{D}$ such that FO($\subseteq, \mathbf{D}$) defines all Σ^1_1 union-closed properties of relations? By the results in Section 4, we know that if this is the case, then $\mathbf{D}$ is not first-order.

Acknowledgements. Pietro Galliani gratefully acknowledges the support of grant 264917 of the Academy of Finland. We thank Erich Grädel, Miika Hannula, Juha Kontinen and Jouko Väänänen for a number of highly useful suggestions and comments. We especially thank Miika Hannula for pointing out an error in a previous version of the paper. Finally, we thank the referees for a number of useful suggestions and comments.

References

1 William W. Armstrong. Dependency Structures of Data Base Relationships. In *Proc. of IFIP World Computer Congress*, pages 580–583, 1974.

2 Uwe Bosse. An "Ehrenfeucht-Fraïssé game" for fixpoint logic and stratified fixpoint logic. In *Computer science logic*, pages 100–114. Springer, 1993.

3 Marco A. Casanova, Ronald Fagin, and Christos H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. In *Proceedings of the 1st ACM SIGACT-SIGMOD symposium on Principles of database systems*, PODS '82, pages 171–176, New York, NY, USA, 1982. ACM.

4 Marco A. Casanova and Vânia M. P. Vidal. Towards a sound view integration methodology. In *Proceedings of the 2nd ACM SIGACT-SIGMOD symposium on Principles of database systems*, PODS '83, pages 36–47, New York, NY, USA, 1983. ACM.

5 Arnaud Durand and Juha Kontinen. Hierarchies in dependence logic. *ACM Trans. Comput. Log.*, 13(4), 2012, 31 pages.

6 Herbert B. Enderton. Finite partially-ordered quantifiers. *Mathematical Logic Quarterly*, 16(8):393–397, 1970.

7 Fredrik Engström. Generalized quantifiers in dependence logic. *Journal of Logic, Language and Information*, 21(3):299–324, 2012.

8 Ronald Fagin. A normal form for relational databases that is based on domains and keys. *ACM Transactions on Database Systems*, 6:387–415, September 1981.

9 Pietro Galliani. *The Dynamics of Imperfect Information*. PhD thesis, University of Amsterdam, September 2012.

10 Pietro Galliani. Inclusion and exclusion dependencies in team semantics: On some logics of imperfect information. *Annals of Pure and Applied Logic*, 163(1):68 – 84, 2012.

11 Dan Geiger, Azaria Paz, and Judea Pearl. Axioms and algorithms for inferences involving probabilistic independence. *Information and Computation*, 91(1):128–141, 1991.

12 Erich Grädel and Jouko Väänänen. Dependence and Independence. *Studia Logica*, 101(2):399–410, 2013.

13 Leon Henkin. Some Remarks on Infinitely Long Formulas. In *Infinitistic Methods. Proc. Symposium on Foundations of Mathematics*, pages 167–183. Pergamon Press, 1961.

14 Jaakko Hintikka and Gabriel Sandu. Informational independence as a semantic phenomenon. In J.E Fenstad, I.T Frolov, and R. Hilpinen, editors, *Logic, methodology and philosophy of science*, pages 571–589. Elsevier, 1989.

15 Wilfrid Hodges. *A Shorter Model Theory*. Cambridge University Press, 1997.

16 Wilfrid Hodges. Compositional Semantics for a Language of Imperfect Information. *Journal of the Interest Group in Pure and Applied Logics*, 5 (4):539–563, 1997.

17 Neil Immerman. Relational queries computable in polynomial time. *Information and control*, 68(1):86–104, 1986.

18 Neil Immerman. Languages That Capture Complexity Classes. *SIAM Journal of Computing*, 16:760–778, 1987.

19 Juha Kontinen and Jouko Väänänen. On definability in dependence logic. *Journal of Logic, Language and Information*, 3(18):317–332, 2009.

20 Antti Kuusisto. Defining a double team semantics for generalized quantifiers. URN:ISBN:978-951-44-8882-5, 2012.

21 Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

22 Allen L. Mann, Gabriel Sandu, and Merlijn Sevenster. *Independence-Friendly Logic: A Game-Theoretic Approach*. Cambridge University Press, 2011.

23 Yannis Moschovakis. *Elementary Induction on Abstract Structures*. North Holland, 1974.

24 Jouko Väänänen. *Dependence Logic*. Cambridge University Press, 2007.

25 Jouko Väänänen and Pietro Galliani. On dependence logic. ArXiv:1305.5948, 2013.

26 Moshe Y. Vardi. The complexity of relational query languages. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 137–146. ACM, 1982.

27 Wilbur John Walkoe. Finite partially-ordered quantification. *The Journal of Symbolic Logic*, 35(4):pp. 535–555, 1970.

Theories for Subexponential-size Bounded-depth Frege Proofs*

Kaveh Ghasemloo and Stephen A. Cook

Department of Computer Science, University of Toronto,
Toronto, ON M5S 3G4, Canada
{[first name],sacook}@cs.toronto.edu

Abstract

This paper is a contribution to our understanding of the relationship between uniform and nonuniform proof complexity. The latter studies the lengths of proofs in various propositional proof systems such as Frege and bounded-depth Frege systems, and the former studies the strength of the corresponding logical theories such as VNC^1 and V^0 in [7]. A superpolynomial lower bound on the length of proofs in a propositional proof system for a family of tautologies expressing a result like the pigeonhole principle implies that the result is not provable in the theory associated with the propositional proof system.

We define a new class of bounded arithmetic theories n^{ε}-ioV^{∞} for $\varepsilon < 1$ and show that they correspond to complexity classes $\mathsf{AltTime}(O(1), O(n^{\varepsilon}))$, uniform classes of subexponential-size bounded-depth circuits $\mathsf{DepthSize}(O(1), 2^{O(n^{\varepsilon})})$. To accomplish this we introduce the novel idea of using types to control the amount of composition in our bounded arithmetic theories. This allows our theories to capture complexity classes that have weaker closure properties and are not closed under composition. We show that the proofs of Σ_0^B-theorems in our theories translate to subexponential-size bounded-depth Frege proofs.

We use these theories to formalize the complexity theory result that problems in uniform NC^1 circuits can be computed by uniform subexponential bounded-depth circuits in [1]. We prove that our theories contain a variation of the theory VNC^1 for the complexity class NC^1. We formalize Buss's proof in [4] that the (unbalanced) Boolean Formula Evaluation problem is in NC^1 and use it to prove the soundness of Frege systems. As a corollary, we obtain an alternative proof of [10] that polynomial-size Frege proofs can be simulated by subexponential-size bounded-depth Frege proofs.

Our results can be extended to theories corresponding to other nice complexity classes inside $\mathsf{NTimeSpace}(n^{O(1)}, n^{o(1)})$ such as NL. This is achieved by essentially formalizing the containment

$$\mathsf{NTimeSpace}(n^{O(1)}, n^{o(1)}) \subseteq \mathsf{AltTime}(O(1), O(n^{\varepsilon}))$$

for all $\varepsilon > 0$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems - Complexity of proof procedures, F.4.1 Mathematical Logic – Proof theory

Keywords and phrases Computational Complexity Theory, Proof Complexity, Bounded Arithmetic, NC^1-Frege, AC^0-Frege

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.296

* Supported by NSERC. The research leading to these results has received funding from the European Union's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 238381.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca ; pp. 296–315

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

In [7] a general method is presented for associating a theory VC of bounded arithmetic and a (quantified) propositional proof system C-Frege with a complexity subclass C of the functions computable in polynomial time FP. The base complexity class is AC^0 of functions computable by a uniform family of polynomial-size bounded-depth Boolean circuits. The associated theory is V^0 and the associated proof system is bounded-depth Frege (bdFrege)[1]. Another important example is the complexity class NC^1 of functions computable by a uniform family of polynomial-size $O(\lg n)$-depth circuits (equivalently polynomial-size Boolean formulas), where the theory is VNC^1 and a Frege system serves as the associated proof system[2].

In general VC is obtained by adding a comprehension axiom for a function complete for C to the base theory V^0 [7]. Similarly, the proof system C-Frege is obtained by adding the cut rule for formulas in nonuniform C to the base proof system bdFrege [7, 11]. Moreover there are witnessing theorems stating that certain proofs in VC can be witnessed using C functions, and proofs in polynomial-size quantified C-Frege can be witnessed by nonuniform C functions. The witnessing theorems can be used to relate the functions in C and nonuniform C to the definable functions in VC and C-Frege [7, 8].

There is a two-fold connection between the theory VC and the propositional proof system C-Frege:

1. VC proves the soundness of C-Frege.
2. Any Σ_0^B theorem (essentially a universal theorem) of VC, translates into a polynomial-size family of tautologies with polynomial-size C-Frege proofs. For example the pigeonhole principle is provable in VNC^1, therefore its propositional translation has polynomial-size Frege proofs.

The second connection provides a universality condition that complements the first connection: C-Frege is maximal among propositional proof systems whose soundness can be proven in VC. More formally, C-Frege can efficiently prove any tautology family which is efficiently provable in any propositional proof system whose soundness is provable in VC [16, 6, 7]. Here a tautology family is efficiently provable in a proof system iff it has a polynomial-size proof family in the proof system, and polynomial size means polynomial size in the length of formulas being proven.

With this perspective, we consider the Σ_0^B fragment of the theory VC as a uniform version of the associated propositional proof system C-Frege.

One motivation for the present paper is the [FPS'11] result that Frege proofs can be simulated by subexponential-size bdFrege proofs. We wish to generalize this result and prove it in a uniform setting. The first step is to find a theory whose provably total functions are those computable by uniform families of bounded-depth subexponential-size circuits.

But here we run into a fundamental obstacle. A function f is subexponential[3] iff $f(n) = 2^{O(n^\varepsilon)}$ for some $\varepsilon < 1$. But the class of subexponential functions is not closed under composition, and not closed even under composition with polynomials. For example, if we compose the subexponential function $n \mapsto 2^{O(n^{\frac{1}{2}})}$ with the polynomial function $n \mapsto n^2$ the resulting function is $n \mapsto 2^n$, which is not subexponential. On the other hand, the provably

[1] bdFrege is also referred to as AC^0-Frege.
[2] Frege is also referred to as NC^1-Frege.
[3] There are other definitions of subexponential functions in the literature. The definition given here is the largest class among them. Using this version is required since even in computational complexity theory it is not known if the bounded-depth circuit classes of smaller size contain NC^1.

total functions in theories which extend the base theory V^0 are closed under composition[4].

To deal with this issue, we introduce two types of variables: input type and output type. The idea is that for a fast growing function f, the arguments of f have input types and are small, while the value of f has output type and might be large. This allows us to control the compositions of the provably total functions of theories. We refer to these typed theories as *io-typed* theories. We can then explicitly allow a limited amount of composition if we want. For example, since subexponential functions are closed under composition with linear functions, we can allow this limited amount of composition in a theory corresponding to subexponential size circuits by conversion axioms defined below in section 2.3.

The propositional proof translations from V^0 to bdFrege in [7] can be adapted for translating proofs from its io-typed version ioV^0 to bdFrege. In addition, it is possible to prove the soundness of these proof systems in ioV^0: we have enough comprehension to define the truth of a sequent in a propositional proof, and we have enough induction to show that sequents in a proof are true under an arbitrary assignment.

Next, we define an extension n^ε-ioV^∞ of ioV^0 (for any $\varepsilon < 1$ of the form $1/d$) whose provably total functions are those computed by $\mathsf{AltTime}(O(1), O(n^\varepsilon))$, a uniform succinct subclass of size $2^{O(n^\varepsilon)}$ bounded-depth circuits. We provide a translation from n^ε-ioV^∞ to polynomial size proofs in the quantified proof system n^ε-bdG_∞, and also to subexponential-size bdFrege proofs. In addition, n^ε-ioV^∞ can prove the soundness of n^ε-bdG_∞ proofs.

We observe that in general, for proving the soundness of line-based propositional proof systems like C-Frege we use the following two ingredients:

- Evaluate the sequents in a proof, and state their truth under a given truth assignment.
- Use induction to prove the sequents in proofs are true under an arbitrary truth assignment.

The comprehension and induction axioms in our theories provide these ingredients.

The central role of soundness tautologies for proof complexity is similar to the central role of complete problems for computational complexity theory. The complete problems for complexity classes provide universal problems that can be used to solve any problem in those classes (over a class of reductions). A similar universal role is played by soundness tautologies in proof complexity: the soundness tautologies for that propositional proof system can be made to serve as axioms for the proof system.

Finally, we show that n^ε-ioV^∞ contains ioVNC^1 and ioVNL, the io-typed versions of VNC^1 and VNL. It follows that n^ε-ioV^∞ proves the soundness of Frege, and this gives us a uniform version of result in [10] that we can simulate Frege proofs with subexponential-size bdFrege proofs. The inclusion $\mathsf{ioVNC}^1 \subseteq n^\varepsilon$-$\mathsf{ioV}^\infty$ (and similar results) is proven essentially by formalizing inside n^ε-ioV^∞ the following theorem[5] from computational complexity theory (for all $\varepsilon < 1$):

$$\mathsf{NTimeSpace}(n^{O(1)}, n^{o(1)}) \subseteq \mathsf{AltTime}(O(1), O(n^\varepsilon)).$$

The rest of the paper is organized as follows:

Section 2 We define our io-typed theories. We start with the base theory ioV^0, the io-typed version of V^0. We then extend ioV^0 to ioVNC^1, the io-typed version of VNC^1, which corresponds to the complexity class NC^1. Our theories n^ε-ioV^∞ corresponding to (uniform) subexponential-size bounded-depth circuits are also defined in this section.

Section 3 We provide the background information about propositional proof systems. We introduce proof classes and define the proof class polynomial-size n^ε-bdG_∞.

---

[4] This obstacle also arises in attempts to design a theory that corresponds to small classes like fixed-depth circuits and propositional proof systems like Resolution.

[5] The result is a similar to Nepomnjaščij's theorem [15, 5].

Section 4 We provide a propositional translation from n^{ε}-ioV^{∞} to n^{ε}-bdG_{∞} and subexponential-size $\mathsf{bdFrege}$.

Section 5 We discuss the soundness of $\mathsf{bdFrege}$, Frege, and n^{ε}-bdG_{∞} in io-typed theories. We discuss the provability of Buss's result [4] that Boolean Formula Evaluation can be computed in NC^1 in our theory ioVNC^1 and use it to show that ioVNC^1 can prove the soundness of Frege.

Section 6 We show that n^{ε}-ioV^{∞} contains ioVNC^1. Theorem 28 is our proof complexity version of the computational complexity theory result that the uniform NC^1 can be computed by $\mathsf{AltTime}(O(1), O(n^{\varepsilon}))$, a uniform complexity class corresponding to subexponential-size bounded-depth circuits [1]. It is also a uniform version of [10] that polynomial-size Frege proofs can be simulated by subexponential-size $\mathsf{bdFrege}$ proofs.

	Nonuniform	Uniform
Computational Complexity	$\mathsf{NC}^1/\mathsf{poly} \subseteq \mathsf{DepthSize}(O(1/\varepsilon), 2^{O(n^{\varepsilon})})$ (Folklore)	$\mathsf{NC}^1 \subseteq \mathsf{AltTime}(O(1), O(n^{\varepsilon}))$ [1], (Omitted Appendix)
Proof Complexity	$\mathsf{FregeSize}(n^{O(1)}) \subseteq O(1/\varepsilon)\text{-}\mathsf{FregeSize}(2^{O(n^{\varepsilon})})$ [10]	$\mathsf{ioVNC}^1 \subseteq n^{\varepsilon}$-$\mathsf{ioV}^{\infty}$ (Theorem 28)

Section 7 We combine the results in the previous sections to show that polynomial-size Frege proofs are simulated by subexponential-size $\mathsf{bdFrege}$ proofs and to provide an alternative proof of [10].[6]

2 A Theory for Subexponential-Size Bounded-Depth Circuits

We start by defining an io-typed version of LK and the base theory $\mathsf{2Basic}$ for two-sorted bounded arithmetic in [7].

2.1 Two-Sorted io-Typed Bounded Arithmetics

Our language $\mathcal{L}_2$ have two sorts: num for (unary) numbers, and str for (binary) strings[7]. In addition, we will syntactically type the terms of the language as[8]: i for input type, and o for output type. The input types are subtypes of the output types, i.e. every object of input type is also an object of output type: $\mathsf{num}^{\mathsf{i}} \subseteq \mathsf{num}^{\mathsf{o}} = \mathsf{num}$, $\mathsf{str}^{\mathsf{i}} \subseteq \mathsf{str}^{\mathsf{o}} = \mathsf{str}$.

We refer to the free variables of a formula as its *parameters*. The idea here is that the input-type terms are going to be *small* (of linear size in the parameters) while the output-type terms can be *large* (of polynomial size in the parameters, like the original two-sorted theories of bounded arithmetic).

Lower-case letters denote numbers: a, b, c denote input-type numbers, and x, y, z denote output-type numbers. Upper-case letters denote strings: A, B, C denote input-type strings, and X, Y, Z denote output-type strings. We use f, g for number-valued functions; D, F, G for string-valued functions; s, t for number-values terms; T for string-valued terms; φ, ψ, etc. for formulas; $\Sigma, \Gamma, \Delta, \Pi$ for set of formulas; S for sequents; and T for sets of sequents, i.e. theories.

[6] A similar result is obtained independently in [14] using model theoretic methods.

[7] Or equivalently, finite sets of natural numbers with explicit (strict) upper bounds on the their members. For example, the string 01100 is equivalent to the finite set $\{2, 3\}$ with the explicit upper bound 5.

[8] We can view our theory as having a two-sorted theory with two types. Each object has a sort and a type. The sorts are: (unary) numbers and (binary) strings. The types are: input and output. Finally, the input type is a subtype of the output type. Alternatively, we can define it as a four-sorted language with appropriate axioms expressing the relation between them.

The language $\mathcal{L}_2$ has function symbols 0, 1, + (addition), · (multiplication), pd (predecessor), | | (length), and relation symbols = (equality), ≤ (comparison), and ∈ (membership/bit). All of these symbols except ∈ and | | apply to terms of number sort and have the obvious intended interpretations. The membership relation $y \in X$ means that the yth bit of the string X is one[9]. The length function $|X|$ returns the length of the string X as a number. We consider the size of a number to be the number itself. The size of a string is its length.

An $\mathcal{L}_2$-stricture has four sets for interpreting num, $\mathsf{num}^{\mathsf{i}}$, str, and $\mathsf{str}^{\mathsf{i}}$, where $\mathsf{num}^{\mathsf{i}} \subseteq \mathsf{num}$ and $\mathsf{str}^{\mathsf{i}} \subseteq \mathsf{str}$. The standard model $\mathbb{N}_2$ for $\mathcal{L}_2$ is given by interpreting num and $\mathsf{num}^{\mathsf{i}}$ as 0^*10^* and str and $\mathsf{str}^{\mathsf{i}}$ as $\{0,1\}^*$. Note that we encode unary numbers using binary strings with a single 1 bit whose location index from right determines the number [10]. We use m, and n for numbers and M and N for strings in the standard model. The operations and relations of $\mathcal{L}_2$ has the usual interpretations as explained above. Note that $m \in N$ is 0 for $|N| \leq m$.

A *linear term* is a term built from constants 0 and 1, variables, and +. A function has *provably linear growth* if the size of its output is provably bounded with a linear term in the size of its inputs.

Every term in the language is an output-type term. The input-type terms of the language are a subset of output-type terms and have provably linear growth. Input-type variables are input-type terms, and when functions 0, 1, | |, +, pd are applied to input-type terms, the result is also of input-type. Formally, the input-type terms are defined inductively: input-type terms include input-type variables and constants 0 and 1; input-type terms are closed under | |, +, and pd.

Unless stated otherwise, by quantifiers we mean quantifiers of both types. In one-sorted theories, the bounded formulas with at most i alternations of number quantifiers comprise the union of the classes Σ_i^b and Π_i^b. In two-sorted theories, these classes are defined similarly and do not have any string quantifiers, but can have free string variables. We will often be interested in the class of number bounded formulas Σ_0^B. A formula is Σ_0^B if it does not have any string quantifiers and all number quantifiers in it are bounded by terms in the language. We abbreviate $|X| = y$ and $|X| \leq y$ by $X = y$ and $X \leq y$. A bounded string quantifier is a string quantifier with an explicitly given size e.g. $\exists X = y$. Bounded formulas with at most i alternations of string quantifiers comprise the union of the classes Σ_i^B and Π_i^B. We define $\Sigma_\infty^B = \bigcup_i \Sigma_i^B$. Let Φ be a class of formulas. The formula class $\exists^B\Phi$ consists of formulas starting with bounded existential string quantifiers followed by a formula in Φ. We will be using a subclass of bounded formulas which we call $\Sigma_\infty^B(t(n))$.

▶ **Definition 1** ($\Sigma_\infty^B(t(n))$)**.** We call a formula $\Sigma_\infty^B(t(\vec{n}))$ iff it is Σ_∞^B and all of its string quantifiers are bounded by number terms of size $O(t(\vec{n}))$ where $\vec{n}$ is the size of its free variables. We often refer to this class simply as $\Sigma_\infty^B(t(n))$ in place of $\Sigma_\infty^B(t(\vec{n}))$. In such cases n can be considered to be the total size of free variables.

Note that every formula Σ_0^B represents a relation $R(\vec{x}, \vec{X})$ in its free variables in the standard model. These relations comprise the complexity class $\mathsf{AC}^0 = \mathsf{AltTime}(O(1), O(\lg n))$ [7].

---

[9] Our semantics differs slightly from [7] where the second sort objects are finite subsets of $\mathbb{N}$, and technically are binary strings starting with 1. Our second sort objects are finite binary strings, where the most significant bit does not need to be 1.

[10] This more complex encoding is needed since otherwise we cannot consider nonconstant number-valued functions in nonuniform models of computation. We need to be able to represent different unary number with the same number of bits. The reason for choosing this particular encoding is its efficiency for performing operations like checking the value of a given unary number. We discard the leading zeros when considering these strings as numbers. For example, 00010 and 010 both represent number 1 and are equal.

We adopt the sequent calculus LK of [7] with quantifier introduction rules to respect the types. In the quantifier introduction rules for input types the target term must be an input-type term. Similarly, if the quantifier variable is of output type, the eigenvariable must be an output-type variable. The intuition here is that if we are deriving the existence of a small object with some property, we must have a small object satisfying the property; or if we are deriving that a property holds for all objects, the property must hold for an arbitrary object, not just small ones.

More formally, in ∃R and ∀L rules, if the quantification variable is of input type then the target term must be also of input type. Similarly, in ∀R and ∃L rules, if the quantification variable is of output type, then the eigenvariable must be also of output type. These restrictions make sure that we *cannot* derive $\forall a\ \ \varphi(a) \Rightarrow \forall x\ \ \varphi(x)$ and $\exists x\ \ \varphi(x) \Rightarrow \exists a\ \ \varphi(a)$. The implications in the other direction are still provable as expected: input types are subsets of output types, so $\forall x\ \ \varphi(x) \Rightarrow \forall a\ \ \varphi(a)$ and $\exists a\ \ \varphi(a) \Rightarrow \exists x\ \ \varphi(x)$ are valid.

We write $\pi : \mathrm{T} \vdash \varphi$ for "π is a LK-proof of φ in the theory T", and $\mathrm{T} \vdash \varphi$ for "φ has an LK-proof in the theory T". We refer to the free variables of the end-sequent of an LK proof as its *parameters*.

Let Φ be a set of formulas (for example take $\Phi = \exists^B\Sigma^B_0$). We say that a set is Φ-definable (over the standard model) iff there is a formula $\varphi \in \Phi$ which defines the set over the standard model. The graph of a function f is defined as $\{(\vec{n}, \vec{N}, m) \in \mathbb{N}_2 \mid f(\vec{n}, \vec{N}) = m\}$. We say that a function f is Φ-definable in T iff its graph is Φ-definable using a formula $\varphi(\vec{x}, \vec{X}, z) \in \Phi$ and T proves that φ defines a function, i.e. $\mathrm{T} \vdash \exists! y\ \ \varphi(\vec{x}, \vec{X}, y)$.

Let F be string-valued. The bit-graph of F is defined as $\{(\vec{n}, \vec{N}, m) \in \mathbb{N}_2 \mid m \in F(\vec{n}, \vec{N})\}$. We say a string-valued function F is Φ-bit-definable in a theory T iff there is a formula $\varphi(\vec{x}, \vec{X}, z) \in \Phi$ and a number term $p(\vec{x}, \vec{y})$ such that[11]

- $|F(\vec{n}, \vec{N})| = p(\vec{n}, |\vec{N}|)$,
- φ defines the bit-graph of the function F (over the standard model), and
- T proves that φ and p define a total function, i.e.

$$\mathrm{T} \vdash \exists! Y = p(\vec{x}, |\vec{X}|)\ \ \forall z < p(\vec{x}, |\vec{X}|)\ \ \Big(z \in Y \leftrightarrow \varphi(\vec{x}, \vec{X}, z)\Big).$$

Notice that the size of F depends only on the size of its arguments. We say that a function is *provably total* in a theory T if the function is Φ-definable in T, for the appropriate choice of the function class Φ. The choice of Φ depends on the theory T: We want the provably total functions in T to be those in the complexity class associated with T. For two-sorted theories associated with complexity classes contained in polynomial time (such as ioV^0 defined below), the right choice is $\Phi = \exists^B\Sigma^B_0$ [7]. For the theory $t(n)$-ioV^∞ defined in Section 2.4 we choose Φ to be a larger class.

When a function is provably total in a theory T, we can add a new function symbol to the language for it and include its definition as an axiom to our theory to obtain a conservative extension of T. This extends the language and possible formulas. We are interested in whether axiom schemas like comprehension continue to hold for the extended class of formulas. For cases of interest in this paper the techniques presented in [7] suffice to show this.

When we extend the language by adding a new provably total function symbol, if we can prove that the function has linear growth, then we can extend input type terms to be closed under the new function symbol.

[11] Note that in circuit complexity, the size of output of a function must only depend on the size of its inputs.

Table 1 io2Basic.

B1	$x+1 \neq 0$	B7	$x \leq y \wedge y \leq x \rightarrow x = y$
B2	$x+1 = y+1 \rightarrow x = y$	B8	$x \leq x+y$
B3	$x+0 = x$	B9	$0 \leq x$
B4	$x+(y+1) = (x+y)+1$	B10	$x \leq y \vee y \leq x$
B5	$x \cdot 0 = 0$	B11	$x \leq y \leftrightarrow x < y+1$
B6	$x \cdot (y+1) = x \cdot y + x$	B12	$\mathsf{pd}(0) = 0 \wedge (x \neq 0 \rightarrow \mathsf{pd}(x)+1 = x)$
L	$y \in X \rightarrow y < \lvert X \rvert$		

The class of io-typed provably total functions of a theory T are those functions that T can prove to be total when the inputs to the functions are of input type. More formally, the free variables in φ corresponding to inputs have input type. In other words, it is sufficient for the function to be total over input-type objects. This is the class of functions we associate with a theory. Note that this class is a possibly larger class than the usual class of provably total functions of a theory since every input-type object is also an output-type object.

Equality for strings, $X = Y$, is an abbreviation for $|X| = |Y| \wedge \forall x \leq |X|\ \ x \in X \leftrightarrow x \in Y$. The notations $X[y,z]$ and $X[i;l]$ abbreviate the substring of X starting from bit y up of length z, and ith substring block of X of length l:

$$x \in X[y,z] := x < z \wedge y + x \in X, \qquad |X[y,z]| := z, \qquad X[i;l] := X[i \cdot l, l].$$

2.2 Theory io2Basic

The axioms of io2Basic are given in table 1.

Note that unlike the original 2Basic in [7], our length function $|\ |$ gives only an upper bound on the size of binary numbers. In this sense our axioms are similar to the second-order theories in [2]. A binary string is determined by its length and its bits. This change doesn't make any essential difference in the presence of number induction for Σ^B_0 formulas: the original version of the length function is definable.

Let d be a fixed positive integer. The *fractional power* $\lfloor x^{\frac{1}{d}} \rfloor$, which we will write simply as $x^{\frac{1}{d}}$, is definable using $x^{\frac{1}{d}} = y \leftrightarrow y^d \leq x < (y+1)^d$.

2.3 Theory ioV^0 for AC^0

Our theory ioV^0 is an io-typed version of the base theory V^0 of [7]. Besides the axioms for io2Basic, we need an io-typed axiom for induction, an io-typed axiom scheme for comprehension, and two conversion axioms (one for each sort). All free variables in the axioms below which are not displayed are of input type[12].

- Ind: $0 \in X, \forall y < z\ \ (y \in X \rightarrow y+1 \in X) \Rightarrow z \in X$
- φ-CA: $\Rightarrow \exists Y = z\ \ \forall x < z\ \ \ (x \in Y \leftrightarrow \varphi(x, A))$

Although we do not want to allow arbitrary compositions, we may want to allow some under specific conditions. The main condition of interest for us here is to avoid increasing the size of the input strings. Therefore, we will add conversion axioms that would allow us to create composition when the size of the computed intermediate values are small[13].

[12] We could have used a stronger version of Σ^B_0-CA where the free variables have output type. In that case the input-type variable free part of the theory will be the same as V^0. The simpler axiom is sufficient and allows a conceptually and technically cleaner treatment.

[13] There are other ways of expressing this axiom e.g. a comprehension axiom from output types to input types, however these variations do not affect our results and we take this simpler from.

- $\mathsf{oiConv}_{\mathsf{num}}$: $\Rightarrow \exists b \leq a \ \ (b = x \leq a) \vee (b = a \leq x)$
- $\mathsf{oiConv}_{\mathsf{str}}$: $\Rightarrow \exists B = a \ \ \forall z < a \ \ (z \in B \leftrightarrow y + z \in X)$

We refer to these two axioms together as oiConv. The first axiom tells us that the minimum of two numbers is small when at least one of them is small. The second axiom tells us that a small substring of an output type string is small. These axioms allow us to compose definable functions if the intermediate results are small in some input variables.

The theory ioV^0 is obtained from $\mathsf{io2Basic}$ by adding the comprehension axiom for Σ^B_0 formulas, the induction axiom, and the conversion axioms.

▶ **Definition 2.** $\mathsf{ioV}^0 = \mathsf{io2Basic} + \mathsf{Ind} + \Sigma^B_0\text{-}\mathsf{CA} + \mathsf{oiConv}$.

As noted earlier the sets in $\mathsf{AltTime}(O(1), O(\lg n)) = \mathsf{LH} = \mathsf{FO} = \mathsf{AC}^0$ are precisely the sets definable by Σ^B_0 formulas. Since ioV^0 has comprehension for these sets, p-bounded functions with bit graphs in these sets are $\exists^B\Sigma^B_0$ definable functions in the theory. By a witnessing theorem, they are the only $\exists^B\Sigma^B_0$ definable functions in the theory. Thus the provable total functions in ioV^0 coincide with the AC^0 functions, where we take the class Φ associated with this theory to be $\exists^B\Sigma^B_0$. As a general rule, the comprehension axiom of any of our theories will determine its computational power.

2.4 Theory ioVC

We can define io-typed versions of other theories built upon V^0. However, simply adding the same comprehension axiom used for VC in [7] might not be sufficient. The io-typed version of these theories can be weaker than their original version. The io-types do not allow arbitrary compositions of the provably total functions of a theory. Therefore, adding the comprehension axiom for a problem complete with respect to AC^0 reductions might not capture the complexity class. This is intentional and necessary since we are going to deal with complexity classes which are not closed under composition (they are not closed even under composition with AC^0 reductions from the right, e.g. consider the subexponential-size bounded-depth circuits where their AC^0 closure contains all functions via padding). Therefore, we cannot define an io-typed theory assuming that the provably total functions of the theory are closed under composition. For example, the theory VTC^0 captures TC^0 because every TC^0 function can be built by composing a finite number of MAJ[14] and AC^0 functions, $\mathsf{TC}^0 = \overline{\mathsf{MAJ} + \mathsf{AC}^0}$. This result is not useful for defining the io-typed version of the theory. Or the theory VNC^1 captures NC^1 because $\mathsf{NC}^1 = \mathsf{MBBFE} \circ \mathsf{AC}^0$ where MBBFE is the Monotone Balanced Boolean Formula Evaluation problem.

With this in mind we have to be careful about the representations of complexity classes we use in the comprehension axiom. The main requirement for a reasonable theory ioVC for computational complexity class C is that ioVC has enough comprehension to evaluate problems in C and the provably total functions of ioVC are exactly the functions in C.

2.5 Theory ioVNC^1

▶ **Definition 3.** The io-typed version of the theory VNC^1, ioVNC^1 is defined as $\mathsf{ioV} + \Sigma^B_0(\mathsf{MBBFE})\text{-}\mathsf{CA}$, where $\Sigma^B_0(\mathsf{MBBFE})\text{-}\mathsf{CA}$ is

$$\exists Y = 2s \ \ \exists Z = 2s \ \ [\forall x < 2s \ \ (x \in Z \leftrightarrow \varphi(x, A)) \wedge \text{“}Y \text{ is the computation of } Z\text{”}]$$

[14] The function MAJ computes the majority for the rows of a given matrix.

We think of Z as an instance of MBBFE and its second half gives inputs to the formula. "Y is the computation of Z" is a shorthand for

$$\forall z < s \ [(z \in Z \to (z \in Y \leftrightarrow 2z \in Y \wedge 2z+1 \in Y)) \wedge$$

$$(z \notin Z \to (z \in Y \leftrightarrow 2z \in Y \vee 2z+1 \in Y))]$$

The theory ioVNC^1 corresponds to NC^1. We will look at $\exists^B\Sigma^B_0$ theorems where the free variables have input type and existentially quantified string variables have output-type. For example, consider the identity function that maps an input string to an output string of the same value. This can be expressed by the formula $\exists X \ X = A$. This formula is provable in ioV^0 using the comprehension axiom. We have the following theorem:

▶ **Theorem 4.** *The $\exists^B\Sigma^B_0$ definable functions of* ioVNC^1 *are precisely* NC^1 *functions.*

Proof. The proof is similar to the proof for VNC^1 in [7]. For one direction we note that the witnessing theorem still applies.

For the other direction any NC^1 function is can be obtained by composing MBBFE with an AC^0 function. We can express this using a $\exists^B\Sigma^B_0$ formula: Σ^B_0(MBBFE) without the leftmost quantifier and with a suitable φ defining the AC^0 function. By comprehension axiom for Σ^B_0(MBBFE) the function is provably total. ◀

2.6 Theory $t(n)$-ioV$^\infty$

Next, we define our theory for the complexity classes AltTime$(O(1), O(t(n)))$. We are interested in $t(n) = n^\varepsilon$ where $\varepsilon = \frac{1}{d} < 1$ for some fixed d. Functions in AltTime$(O(1), O(t(n)))$ can be computed by uniform families of subexponential-size bounded-depth circuits. Note that every function in AltTime$(O(1), O(t(n)))$ is definable by a $\Sigma^B_\infty(t(n))$ formula, i.e. a formula with string quantifiers bounded by a term of size $O(t(n))$ where n is a bound on the size of inputs. The complexity class AltTime$(O(1), O(n^\varepsilon))$ corresponds to $\Sigma^B_\infty(n^\varepsilon)$ in a similar way that the complexity class AltTime$(O(1), O(\lg n)) =$ AC^0 corresponds to $\Sigma^B_\infty(\lg n) = \Sigma^B_0$. The complexity class AltTime$(O(1), O(n^\varepsilon))$ is a nice uniform version of subexponential-size bounded-depth circuits. We will include the comprehension axiom $\Sigma^B_\infty(t(n))$-CA for this class of functions to provide the necessary computational power. Note that $t(n)$ is a term of number sort and input-type which bounds the quantified string variables in φ. The term $t(n)$ cannot contain any output-type variable. We will consider cases where $t(n)$ is not a term in the original language but an AC^0 function definable in the base theory ioV^0 with at most linear growth. In these cases we can easily extend the language to contain the new function and add the defining axiom of the function to the theory, e.g. $t(n) = n^{\frac{1}{d}}$ where $n = |A|$.

We define our theory $t(n)$-ioV$^\infty$ as follows:

▶ **Definition 5.** $t(n)$-ioV$^\infty$ = ioV^0 + $\Sigma^B_\infty(t(n))$-CA

It is easy to see that ioV^0 is equivalent to $\lg n$-ioV$^\infty$ since we can convert unary numbers to binary numbers of logarithmic size and vice versa in AC^0. Using Σ^B_0-CA and definability of Bit in ioV^0 we can prove $\Sigma^B_\infty(\lg n) = \Sigma^B_0$.

We take the provably total functions in $t(n)$-ioV$^\infty$ to be the Φ-definable functions, where $\Phi = \exists^B\Sigma^B_\infty(t(n))$.

▶ **Theorem 6** (Provably Total Functions of $t(n)$-ioV$^\infty$)**.** *The provably total functions of the theory $t(n)$-ioV$^\infty$ are exactly those in* AltTime$(O(1), O(t(n)))$ *of size $n^{O(1)}$, where n is the size of the arguments.*

Proof Outline. The proof is similar to the proof for V^0: functions in $\mathsf{AltTime}(O(1), O(t(n)))$ are definable and provably total using the comprehension axiom for $\Sigma^B_\infty(t(n))$. On the other hand, by a witnessing theorem every provably total function of $t(n)$-ioV^∞ is in $\mathsf{AltTime}(O(1), O(t(n)))$. ◀

3 Proof Systems and n^ε-bdG_∞ Proofs

We use lower-case letters like p and q for propositional variables; and α, β, γ for propositional function symbols. We use φ, ψ, etc. for formulas (propositional, quantified propositional, and first-order).

We say that a term is free in a formula iff all of its variables are free in the formula. Expressions like $\varphi[p/q]$ denote the formula resulting from substituting q for p in φ. The usual restrictions on substitution apply, e.g. only free occurrences are replaced, and new variables must not become bound after substitution.

We allow unbounded $\bigwedge$ and $\bigvee$ connectives in propositional formulas. The (logical) depth of a propositional formula denoted by $\mathsf{ldepth}()$ is defined as the depth of the formula tree. The size of a propositional formula denoted by $\mathsf{size}()$ is the number of nodes in its tree[15]. Quantified propositional formulas are defined by allowing quantification over propositional variables of propositional formulas. We allow a single quantifier to quantify over multiple propositional variables. For quantified propositional formulas the depth is defined as the depth of their quantifier-free part. The quantifier depth of a quantified propositional formula is defined as the maximum depth of formula tree counting only quantifier nodes and is denoted by $\mathsf{qdepth}()$. The size of a proof is the total size of its formulas. The depth of a proof is the maximum depth of its formulas. The quantifier depth of a proof is the maximum quantifier depth of its formulas. The class of quantified propositional formulas with quantifier depth i starting with an existential quantifier is denoted by Σ^q_i. Σ^q_∞ is their union.

3.1 Proof Systems and Proof Classes

Let L denote a class of formulas, e.g. propositional formulas or quantified propositional formulas. Let τ be a truth assignment for free variables, i.e. a function from free variables to $\{0, 1\}$. We sometimes call τ an evaluation context. We use $\tau \vDash \varphi$ to express that a formula $\varphi \in L$ is true under the truth assignment τ. When φ is true under all truth assignments we write $\vDash \varphi$ and say that φ is valid. We refer to the set of valid formulas in L as L-tautologies and denoted it by $TAUT_L$.

Let Q be a relation with two inputs. We say π is a Q-proof for φ iff $\mathrm{Q}(\pi, \varphi)$ accepts, in which case we write $\pi : \mathrm{Q} \vdash \varphi$. We say φ is provable in Q iff φ has a Q-proof. An relation Q is a(n) (efficient) proof system for L iff

- efficiency: Q is computable in polynomial time,
- completeness: every L-tautology is provable in Q,
- soundness: every L-formula provable in Q is an L-tautology.

In propositional proof complexity we want to study *families* of proofs for *families* of formulas. We say a proof family $\{\pi_n\}_n$ is a Q-proof for a formula family $\{\varphi_n\}_n$ iff for all n, $\pi_n : \mathrm{Q} \vdash \varphi_n$. We define *proof classes* in a similar way to nonuniform computational complexity classes. Let Q be a proof system. A Q-proof class is a set of Q-proof families.

[15] If we encode formulas as binary strings there will be a constant factor of the number of nodes in the tree.

Similar to bounded-depth circuits, bounded-depth Frege (bdFrege) proof class is defined as the set of Frege-proof families where the depth of cut formulas is $O(1)$. Polynomial-size bdFrege is the subclass of bdFrege where the size of the proofs are polynomial in the size of the proven formula. If F is a proof class, we write $\mathsf{F} \vdash \{\varphi_n\}_n$ to state that $\{\varphi_n\}_n$ has a F-proof. For example, the pigeon-hole principle $PHP = \{PHP_n\}_n$ has polynomial-size Frege proofs [7, 12] but it does not have polynomial-size bdFrege proofs [12].

The proof class bdFrege is sometimes defined as the union of proof systems d-Frege for $d \in \mathbb{N}$, where d-Frege is a subsystem of Frege obtained by restricting cuts to depth d formulas. It is easy to see that this gives the same bdFrege proof class we defined above. Note that bdFrege is not a proof system, The proof system obtained from taking the union of proof systems d-Frege for $d \in \mathbb{N}$ is not bdFrege but Frege. In fact, it doesn't make much sense to say a single proof has a bounded depth.

3.2 Standard Proof Systems and Proof Classes: bdFrege, Frege, and G

Our reference is [7]. PK is the classical sequent calculus propositional proof system. For $d \in \mathbb{N}$, the propositional proof system d-PK is PK with cuts restricted to depth d formulas. bdPK is the proof class resulting from taking the union of d-PKs, i.e. proof families with the depth $O(1)$ cut formulas. Frege denotes any Frege proof system, e.g. PK. Bounded-depth Frege denoted by bdFrege (a.k.a. AC^0-Frege) is the Frege where cuts are restricted to bounded-depth formulas.

Next, we define proof systems for quantified propositional calculus. The proof system G is obtained from PK by adding the Boolean quantifier introduction rules. In the rules $\forall$R and $\exists$L, p is a free variable called *eigenvariable* and does not appear in the bottom sequent. In the rules $\forall$L and $\exists$R, $\varphi[q/\psi]$ is the result of substituting ψ for q in φ. The formula ψ is called the *target* formula of the rule and may be any quantifier-free formula[16]. The formulas $\exists q\ \varphi$ and $\forall q\ \varphi$ are called the *principal* formulas and the corresponding $\varphi[q/\psi]$ or $\varphi[q/p]$ formulas on top are called the *auxiliary* formulas. $\mathsf{G_i}$ is a subsystem of G where the cuts are restricted to formulas of quantifier depth i. For $d \in \mathbb{N}$, the proof class d-$\mathsf{G_i}$ is $\mathsf{G_i}$ where the depth of the quantifier free part of cut formulas is bounded by d. $\mathsf{bdG_i}$ is the union of these proof systems, i.e. the depth of the cut formulas in the proofs are bounded by a constant. $\mathsf{G_0 0}$ is a conservative extension of Frege and $\mathsf{bdG_0}$ is a conservative extension of bdFrege [7, 12].

In our systems, we allow the introduction of a quantifier over multiple propositional variables in a single step. For example, we can derive $\exists \vec{p}\ \varphi(\vec{p})$ is a single step from $\varphi(\vec{\psi})$. Similarly, we allow the introduction of conjunction/disjunction of multiple formulas in a single step. These modifications do not change the power of the proof systems, but will be convenient to assume to obtain a nicer correspondence with first-order proofs.

In general, a proof can be a DAG and does not need to be a tree. We use a superscript * to denote proofs which are trees [7, p. 195].

3.3 Proof Systems n^{ε}-bdG_∞ and H

Let $\mathsf{bd}\Sigma^q_\infty(t(n))$ denote the class of those Σ^q_∞ formula families where the number of quantified propositional variables is bounded by $O(t(n))$ and the depth of quantifier-free part is bounded.

[16] The exact class of formulas for these rules is not important. By [7, VII.3.6, p.176], we can assume that the target formulas are arbitrary formulas and need not to be quantifier free. Similarly, we can restrict them to be only $\top$ or $\bot$. These modifications does not change the power of the proof system. Following [7], we use the definition of G (and its subsystems) that does not restrict all formulas but only the cut formulas.

▶ **Definition 7** ($t(n)$-bdG_∞)**.** The proof class $t(m)$-bdG_∞ is the class of bdG_∞ proofs for formula families where cuts are restricted to $\mathsf{bd}\Sigma^q_\infty(t(m))$ formulas where m is the size of proven formula. In addition we will assume that[17] the total number of eigenvariables in each sequent of in $t(m)$-bdG_∞ proofs must not exceed $t(m)$.

▶ **Definition 8** (H)**.** The proof system H is an extension of G obtained by allowing propositional function symbols (denoted by α, β, γ) as atomic formulas and including the following extensionality axiom for them[18] Ext: $\vec{p} \leftrightarrow \vec{q} \Rightarrow \alpha(\vec{p}) \leftrightarrow \alpha(\vec{q})$

We will use the propositional function symbols to remove quantifiers from the axioms and obtain Skolemized axioms. We will consider proofs with non-logical axioms.

4 From Uniform to Nonuniform

4.1 Translation of Terms and Formulas

In this section, we define a translation from first-order terms and bounded formulas to families of sequences of propositional formulas and quantified propositional formulas, respectively.

We use $(w)_i$ to refer to the ith item in the sequence w. For example, if $\vec{p} = (p_0, \ldots, p_{k-1})$ is a sequence of propositional variables, then $(\vec{p})_i$ is p_i for $i < k$ (and is $\bot$ if $i \geq k$). Recall that s and t denote number terms and T denotes a string term.

Let Var be the set of variables. Let σ be a function that determines the size of variables, i.e. $\sigma : Var \to \mathbb{N}$ assigns a numeric value to each variable. We refer to σ as a translation context. The notation $\sigma[x \mapsto n]$ is used for the function obtained from σ by mapping x to n. The propositional translation of a term t and a formula φ under translation parameters σ are denoted by $[\![t]\!]_\sigma$ and $[\![\varphi]\!]_\sigma$. We sometimes use $[\![t]\!]_{\vec{n}}$ and $[\![\varphi]\!]_{\vec{n}}$ in place of $[\![t]\!]_{[\vec{x}\mapsto\vec{n}]}$ and $[\![\varphi]\!]_{[\vec{x}\mapsto\vec{n}]}$ when it is clear which variables $\vec{x}$ are being mapped.

The terms are translated recursively. The main difference from the usual propositional translations is that we have function symbols that we translate to propositional function symbols. We translate a function symbol into a sequence of propositional function symbols. Each of these propositional function symbols will correspond to a bit of the original function symbol. The number of propositional function symbols is the length of the original function symbol. We can translate functions of the theory that are AC^0 computable into reasonable AC^0 circuits computing them (sequences of AC^0 formulas).

We first need to extend the translation context σ to all terms in the language. We will use σ to determine the size of sequences used for the translation of the terms. The number of bits used for translating a term may only depend on the size of its variables. Note that every function symbol in our languages has an explicit size in terms of the size of its inputs. We denote the size of a function symbol by adding a superscript $^\sigma$. The translation context σ distributes over sequences, i.e. $\sigma(t_k, \ldots, t_0) = \sigma(t_k), \ldots, \sigma(t_0)$.

▶ **Definition 9** (Extended translation context)**.** Let σ be a translation context. The extended translation context is given in table 2. For any term t, $\sigma(t)$ is bounded by a polynomial in σ. If a variable x occurs in t, then $\sigma(t) \geq \sigma(x)$.

[17] This assumption simplifies some arguments significantly. However, we think that the results hold also without this assumption.

[18] A similar system is presented in [9].

Table 2 Extended Translation Context σ and Translation of Terms.

	$[\![x]\!]_\sigma = (\top, \overbrace{\bot, \ldots, \bot}^{\sigma(x) \text{ times}})$
	$[\![X]\!]_\sigma = (p^X_{\sigma(X)-1}, \ldots, p^X_0)$
$\sigma(0) = 0$	$[\![0]\!]_\sigma = (\top)$
$\sigma(1) = 1$	$[\![1]\!]_\sigma = (\top, \bot)$
$\sigma(t+s) = \sigma(t) + \sigma(s)$	$[\![s+t]\!]_\sigma = (o_{\sigma(s+t)}, \ldots, o_0)$ where $o_k = ([\![s+t]\!]_\sigma)_k = \bigvee_{\substack{i \le \sigma(s), j \le \sigma(t) \\ i+j=k}} ([\![s]\!]_\sigma)_i \wedge ([\![t]\!]_\sigma)_j$
$\sigma(t \cdot s) = \sigma(t) \cdot \sigma(s)$	$[\![s \cdot t]\!]_\sigma = (o_{\sigma(s \cdot t)}, \ldots, o_0)$ where $o_k = ([\![s \cdot t]\!]_\sigma)_k = \bigvee_{\substack{i \le \sigma(s), j \le \sigma(t) \\ i \cdot j=k}} ([\![s]\!]_\sigma)_i \wedge ([\![t]\!]_\sigma)_j$
$\sigma(\mathsf{pd}(t)) = \sigma(t)$	$[\![\mathsf{pd}(s)]\!]_\sigma = (o_{\sigma(\mathsf{pd}(t))}, \ldots, o_0)$ where $o_k = ([\![\mathsf{pd}(t)]\!]_\sigma)_k = \begin{cases} ([\![t]\!]_\sigma)_0 \vee ([\![t]\!]_\sigma)_1 & k = 0 \\ ([\![t]\!]_\sigma)_{k+1} & o.w. \end{cases}$
$\sigma(\lvert T \rvert) = \sigma(T)$	$[\![\lvert T \rvert]\!]_\sigma = [\![\sigma(T)]\!]_\sigma$
$\sigma(f(\vec{t}, \vec{T})) = f^\sigma(\sigma(\vec{t}), \sigma(\vec{T}))$	$[\![f(\vec{t}, \vec{T})]\!]_\sigma = (f_{\sigma(f(\vec{t}, \vec{T}))}([\![\vec{t}]\!]_\sigma, [\![\vec{T}]\!]_\sigma), \ldots, f_0([\![\vec{t}]\!]_\sigma, [\![\vec{T}]\!]_\sigma))$
$\sigma(F(\vec{t}, \vec{T})) = F^\sigma(\sigma(\vec{t}), \sigma(\vec{T}))$	$[\![F(\vec{t}, \vec{T})]\!]_\sigma = (F_{\sigma(F(\vec{t}, \vec{T}))-1}([\![\vec{t}]\!]_\sigma, [\![\vec{T}]\!]_\sigma), \ldots, F_0([\![\vec{t}]\!]_\sigma, [\![\vec{T}]\!]_\sigma))$

Terms are translated recursively: a number term t is translated to a sequence of size $\sigma(t)+1$, and a string terms T is translated to a sequence of size $\sigma(T)$. A unary number n is represented by $\top\bot^n$, i.e. the a sequence of size $n+1$ where only the nth bit[19] is $\top$. We can use any reasonable AC^0 formula for the translation of the functions of $\mathcal{L}_2$. The main requirement is that the translation of the axioms in io2Basic must have simple propositional proofs. The translation distributes over sequences, i.e. $[\![t_k, \ldots, t_0]\!]_\sigma = [\![t_k]\!]_\sigma, \ldots, [\![t_0]\!]_\sigma$.

▶ **Definition 10** (Translation of terms)**.** Let σ be a translation context. The translation for terms is given in table 2. For any term t, $\mathsf{size}([\![t]\!]_\sigma)$ is bounded by a polynomial in σ. If a variable x occurs in t, then $\mathsf{size}([\![t]\!]_\sigma) \ge \sigma(x)$. In addition, $\mathsf{ldepth}([\![t]\!]_\sigma) = O(1)$.

We can translate terms containing a string-valued function F which is defined by a monotone non-decreasing term p and a formula φ (see Section 2.2) as[20] $|F(\vec{x}, \vec{X})| = p(\vec{x}, |\vec{X}|)$, and $y \in F(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, \vec{X}, y)$, using $\sigma(F(\vec{t}, \vec{T})) = p(\sigma(\vec{t}), \sigma(\vec{T}))$, and $([\![F(\vec{t}, \vec{T})]\!]_\sigma)_k = [\![\varphi(\vec{t}, \vec{T}, y)]\!]_{\sigma[y \mapsto k]}$. Similarly, for number-valued functions f defined by $f(\vec{x}) \le p(\vec{x})$, and $f(\vec{x}) = y \leftrightarrow \varphi(\vec{x}, y)$ we can use $\sigma(f(\vec{t})) = p(\sigma(\vec{t}))$, $([\![f(\vec{t})]\!]_\sigma)_k = [\![\varphi(\vec{t}, y)]\!]_{\sigma[y \mapsto k]}$.

For example, if we include substring function $T[s,t]$ in the language defined in Section 2.1, we can translate it using $\sigma(T[s,t]) = \sigma(t)$ and $([\![T[s,t]]\!]_\sigma)_k = [\![x < t \wedge s + x \in X]\!]_{\sigma[x \mapsto k]}$.

Formulas are also translated recursively: atomic formulas are translated directly to AC^0 formulas such that the axioms about them have simple propositional proofs. Logical connective are translated to themselves. Bounded number quantifiers are translated to $\bigwedge$ and $\bigvee$. Bounded string quantifiers are translated to propositional quantifiers. The number of quantified propositional variables will be equal to the bound.

▶ **Definition 11** (Translation of formulas)**.** Let σ be a translation context. The translation of bounded formulas is given in table 3. For any formula φ, $\mathsf{size}([\![\varphi]\!]_\sigma)$ is bounded by a polynomial in σ. If a variable x occurs freely in φ, then $\mathsf{size}([\![\varphi]\!]_\sigma) \ge \sigma(x)$. Also $\mathsf{ldepth}([\![\varphi]\!]_\sigma) = O(1)$ and $\mathsf{qdepth}([\![\varphi]\!]_\sigma) = O(1)$. The number of quantified propositional variables is determined by the translation of the bounding terms for the string quantifiers.

[19] The index of the rightmost bit is 0.

[20] For the translation to work we need the size of the functions to depend only on the size of their inputs. If we want to include in our translation functions like msb whose size is not determined by the size of its inputs we need to use a language that has a length operation for numbers.

Table 3 Translation of Formulas

$[\![s = t]\!]_\sigma = \bigvee_{i \le \sigma(s),\sigma(t)} ([\![s]\!]_\sigma)_i \wedge ([\![t]\!]_\sigma)_i$ $[\![s \le t]\!]_\sigma = \bigvee_{i \le \sigma(s) i \le j \le \sigma(t)} ([\![s]\!]_\sigma)_i \wedge ([\![t]\!]_\sigma)_j$ $[\![t \in T]\!]_\sigma = \bigvee_{i \le \sigma(T)} ([\![T]\!]_\sigma)_i \wedge ([\![t]\!]_\sigma)_i$	$[\![\bot]\!]_\sigma = \bot$ $[\![\top]\!]_\sigma = \top$ $[\![\neg\varphi]\!]_\sigma = \neg[\![\varphi]\!]_\sigma$ $[\![\psi \wedge \varphi]\!]_\sigma = [\![\psi]\!]_\sigma \wedge [\![\varphi]\!]_\sigma$ $[\![\psi \vee \varphi]\!]_\sigma = [\![\psi]\!]_\sigma \vee [\![\varphi]\!]_\sigma$ $[\![\exists x \le t\ \ \varphi]\!]_\sigma = \bigvee_{i \le \sigma(t)} [\![x \le t \wedge \varphi]\!]_{\sigma[x \mapsto i]}$ $[\![\forall x \le t\ \ \varphi]\!]_\sigma = \bigwedge_{i \le \sigma(t)} [\![x \le t \to \varphi]\!]_{\sigma[x \mapsto i]}$ $[\![\exists X = t\ \ \varphi]\!]_\sigma = \exists [\![X]\!]_\tau\ \ [\![X = t \wedge \varphi]\!]_\tau$ $[\![\forall X = t\ \ \varphi]\!]_\sigma = \forall [\![X]\!]_\tau\ \ [\![X = t \to \varphi]\!]_\tau.$ where $\tau = \sigma[X \mapsto \sigma(t)]$

Recall that bounded string quantifiers of the form $\exists X \le t\ \ \varphi$ and $\forall X \le t\ \ \varphi$ are equivalent to $\exists y \le t\ \ \exists X = y\ \ \varphi$ and $\forall y \le t\ \ \forall X = y\ \ \varphi$ and can be translated as such.

Using induction on the structure of formulas we can prove that $[\![\varphi]\!]_\sigma$ is a tautology iff $\vec{x} = \sigma(\vec{x}), |\vec{X}| = \sigma(\vec{X}) \Rightarrow \varphi$ is true in the standard model.

4.2 Translation of Proofs

In this section, we provide a translation from proofs in the theory n^ε-ioV^∞ to polynomial-size n^ε-bdG_∞ and subexponential-size bdG_0 proofs. We will consider proofs in n^ε-ioV^∞ where $\varepsilon = \frac{1}{d} < 1$ is fixed.

▶ **Theorem 12** (Propositional Translation). *If $\varphi \in \Sigma^B_0$ is provable in n^ε-ioV^∞, then $\{[\![\varphi]\!]_{\vec{n}}\}_{\vec{n}}$ has polynomial-size n^ε-bdG_∞.*

Assume that n^ε-$\mathsf{ioV}^\infty \vdash \varphi$. If φ has no free variables its translation is a fixed formula and has a size $O(1)$ depth $O(1)$ proof and we are done. So assume that φ has at least one free variable.

Without loss of generality, we can assume that all free variables in φ have input type: if φ is provable then so is $\varphi[\vec{x}, \vec{X}/\vec{a}, \vec{A}]$, which has the same translation. We refer to $\vec{n} = \sigma(\vec{a}, \vec{A})$ as translation parameters where $\vec{a}$ and $\vec{A}$ are φ's free variables. Let $m_{\vec{n}} = \mathsf{size}([\![\varphi]\!]_{\vec{n}})$.

Since φ has at least one free variable we have $m_{\vec{n}} = \mathsf{size}([\![\varphi]\!]_{\vec{n}}) = \Omega(\vec{n})$. Therefore, if prove that some entity like the size of a proof for φ is bounded by a monotone non-decreasing function of $\vec{n}$, the bound will also apply with $\vec{n}$ replaced by $\vec{m}$. With this in mind, we will study proof size, proof depth, etc. in terms of the translation parameters $\vec{n}$.

The propositional translation has three steps. In the first step, we Skolemize conversion and comprehension axioms to remove the initial existential string quantifiers from them. We denote the Skolemized version of these axioms by adding a * superscript. The Skolem functions for comprehension axioms return output-type values while the Skolem function for the conversion axiom returns input-type values. In other words, the type of value returned by a Skolem function matches the type of quantified variable it is witnessing. Note that axioms determine the size of the functions symbols.

- $\mathsf{oiConv}_{\mathsf{str}}$: $\Rightarrow \exists B = a\ \ \forall z < a\ \ (z \in B \leftrightarrow y + z \in X)$,
- $\mathsf{oiConv}^*_{\mathsf{str}}$: $\Rightarrow |F^{\mathsf{oiConv}}(a, y, X)| = a \wedge \forall z < a\ \ (z \in F^{\mathsf{oiConv}}(a, y, X) \leftrightarrow y + z \in X)$.
- φ-CA: $\Rightarrow \exists Y = z\ \ \forall x < z\ \ (x \in Y \leftrightarrow \varphi(x, A))$,
- φ-CA^*: $\Rightarrow |F^{\varphi\text{-}\mathsf{CA}}(z, A)| = z \wedge \forall x < z\ \ \big(x \in F^{\varphi\text{-}\mathsf{CA}}(z, A) \leftrightarrow \varphi(x, A)\big)$.

Let's call the resulting theory $\widehat{n^\varepsilon\text{-}\mathsf{ioV}^\infty}$. Note that the theory still has the nice properties of the original theory. In particular, the size of an input-type term is still bounded by a linear function of the size of its input-type variables. We have:

▶ **Lemma 13** (Step 1). *If* $n^{\varepsilon}\text{-ioV}^{\infty} \vdash \varphi$ *then* $\widehat{n^{\varepsilon}\text{-ioV}^{\infty}} \vdash \varphi$.

Proof of Step 1. We only need to derive the axioms of $n^{\varepsilon}\text{-ioV}^{\infty}$ in $\widehat{n^{\varepsilon}\text{-ioV}^{\infty}}$. The original axioms are derivable from the Skolemized versions using a single application of the $\exists\mathsf{R}$. ◀

Next, we translate proofs in $\widehat{n^{\varepsilon}\text{-ioV}^{\infty}}$ to proof families in H with non-logical axioms.

▶ **Lemma 14** (Step 2). *If* $\pi : \widehat{n^{\varepsilon}\text{-ioV}^{\infty}} \vdash \varphi$, *then there is a* H*-proof* $\{[\![\pi]\!]_{\vec{n}}\}_{\vec{n}}$ *of* $\{[\![\varphi]\!]_{\vec{n}}\}_{\vec{n}}$ *using the translation of the axioms in* $\widehat{n^{\varepsilon}\text{-ioV}^{\infty}}$ *as non-logical axioms. Moreover,* $\mathsf{size}([\![\pi]\!]_{\vec{n}}) = O(\mathsf{poly}(\vec{n}))$, $\mathsf{ldepth}([\![\pi]\!]_{\vec{n}}) = O(1)$, $\mathsf{qdepth}([\![\pi]\!]_{\vec{n}}) = O(1)$, *the proof contains only cuts over* $\Sigma^{q}_{\infty}(n^{\varepsilon})$ *formulas, and the number of eigenvariables in each sequent is* $O(n^{\varepsilon})$.

Proof of Step 2. We first convert the proof π into a proof in free-variable free-cut free normal form. The resulting proof only contains subformulas of φ and the axioms. Since φ does not have any string quantifier, all cut formulas with string quantifiers are subformulas of the Skolemized comprehension axiom. By lemma 15 below, we can bound the size of free input-type string variables in the proof by linear size terms in parameters of the proof $\vec{n}$. Therefore, all formulas in the proof are $\Sigma^{B}_{\infty}(n^{\varepsilon})$.

We translate the proof to a propositional proof in H recursively starting from the end-sequent. The translation is straightforward. The rules in H correspond to rules in LK. The only interesting cases are the number quantifier rules which need to be replaced by rules for $\bigvee$ and $\bigwedge$. For $\exists\mathsf{L}$ and $\forall\mathsf{R}$, we extend the translation context to assign values to the eigen variable for all possible values less than the bound. All terms have at most polynomial size in their free variables. Therefore, we will construct a polynomial number of them.

For example, consider $\forall\mathsf{R}$. The sequent $\Gamma \Rightarrow \Delta, \forall x \leq t\ \psi$ is translated to $[\![\Gamma]\!]_{\sigma} \Rightarrow [\![\Delta]\!]_{\sigma}, \bigwedge_{i\leq\sigma(t)} [\![x \leq t \to \psi]\!]_{\sigma[x\mapsto i]}$. We recursively obtain the proofs for the translations of $\Gamma \Rightarrow \Delta, x \leq t \to \psi$ under translation contexts $\sigma[x \mapsto i]$ for all $i \leq \sigma(t)$, and use $\bigwedge\mathsf{R}$ to obtain

$$\frac{\Gamma \Rightarrow \Delta, x \leq t \to \psi}{\Gamma \Rightarrow \Delta, \forall x \leq t\ \psi}\,\forall\mathbf{R} \qquad \frac{\{[\![\Gamma]\!]_{\sigma} \Rightarrow [\![\Delta]\!]_{\sigma}, [\![x \leq t \to \psi]\!]_{\sigma[x\mapsto i]}\}_{i\leq\sigma(t)}}{[\![\Gamma]\!]_{\sigma} \Rightarrow [\![\Delta]\!]_{\sigma}, \wedge_{i\leq\sigma(t)} [\![x \leq t \to \psi]\!]_{\sigma[x\mapsto i]}}\,\bigwedge\mathbf{R}$$

The axioms are translated to non-logical quantified propositional axioms. It is easy to check that the depth and quantifier depth of the proof are $O(1)$ and size of the proof is $O(\mathsf{poly}(\vec{n}))$. Formulas in $\Sigma^{B}_{\infty}(n^{\varepsilon})$ are translated to $\Sigma^{q}_{\infty}(n^{\varepsilon})$ formulas, so cuts formulas are $\Sigma^{q}_{\infty}(n^{\varepsilon})$. Each sequent in the translated proof is a translation of a first-order sequent. Therefore, the number of formulas in each sequent is constant and the total number of eigenvariables in each sequent is $O(n^{\varepsilon})$. ◀

▶ **Lemma 15** (Linear Type Bounds). *The size of free input-type string variables in the proof can be bounded by linear terms in* $\vec{n}$.

Proof Idea. The proof is similar to Parikh's theorem. ◀

In the third step, we remove the function symbols and non-logical axioms from the proof to obtain a polynomial-size $n^{\varepsilon}\text{-bdG}_{\infty}$ proof.

▶ **Lemma 16** (Step 3). *If* $\widehat{n^{\varepsilon}\text{-ioV}^{\infty}} \vdash \varphi$ *then* $\{[\![\varphi]\!]_{\vec{n}}\}_{\vec{n}}$ *has a polynomial-size* $n^{\varepsilon}\text{-bdG}_{\infty}$ *proof.*

Proof Step 3. Consider the proof obtained in step 2. To obtain a $n^{\varepsilon}\text{-bdG}_{\infty}$-proof we need to provide

- polynomial-size explicit witnessing formulas for the function symbols, and
- polynomial-size $n^{\varepsilon}\text{-bdG}_{\infty}$ proofs for the non-logical axioms.

Note that the translations of the axioms of io2Basic have simple polynomial-size bdG_0 proofs. The the translation of the induction axiom becomes

$$[\![0 \in X]\!]_\sigma, \bigwedge_{i \le \sigma(z)} [\![y \le z]\!]_{\sigma[y \mapsto i]} \to ([\![y \in X]\!]_{\sigma[y \mapsto i]} \to [\![y+1 \in X]\!]_{\sigma[y \mapsto i]}) \Rightarrow [\![z \in X]\!]_\sigma$$

which has a simple proof of polynomial size and bounded depth: We provide proofs for $[\![y+1 \in X]\!]_{\sigma[y \mapsto i]} \Rightarrow [\![y \in X]\!]_{\sigma[y \mapsto i+1]}$ and then combine these using $\vee\mathsf{L}$ to obtain the proof.

We use substring functions to witness the conversion axiom. To remove the conversion axiom, we replace $[\![F^{oiConv_{str}}(a, y, X)]\!]_n$ with $[\![X[y, a]]\!]_n$. The axiom becomes

$$\Rightarrow [\![|X[y, a]| = a]\!]_n \wedge \bigwedge_{i \le \sigma(a)} [\![x \in X[y, a] \leftrightarrow y + x \in X]\!]_{n,[x \mapsto i]}$$

which has a bdG_0-proof of polynomial size and bounded depth.

We will use the defining formula of the comprehension axioms to witness the comprehension function symbols. To remove the comprehension function symbols, we replace $([\![F^{\varphi\text{-}CA}(A)]\!]_\sigma)_n$ with $[\![\varphi(x, A)]\!]_{n,[x \mapsto i]}$. The comprehension axioms become

$$\Rightarrow \bigwedge_{i \le \sigma(t)} \left([\![\varphi(x, A)]\!]_{n,[x \mapsto i]} \leftrightarrow [\![\varphi(x, A)]\!]_{n,[x \mapsto i]}\right)$$

which have bdG_0 cut-free proofs of polynomial size. ◀

Finally, we expand the $\Sigma^B_\infty(n^\varepsilon)$ formulas to bounded-depth formulas of size $2^{O(n^\varepsilon)}$. As a result, we get size $2^{O(n^\varepsilon)}$ bdG_0 proofs.

▶ **Theorem 17** (Subexpoential-Size Bounded-Depth G_0 Proofs)**.** *If* $\{[\![\varphi]\!]_{\vec{n}}\}_{\vec{n}}$ *has a polynomial-size* n^ε-bdG_∞ *proof then* $\{[\![\varphi]\!]_{\vec{n}}\}_{\vec{n}}$ *has a size* $2^{O(n^\varepsilon)}$ bdG_0 *proof.*

Proof of Corollary 17. We convert the proof by replacing propositional quantifiers with $\bigwedge$ and $\bigvee$ and quantifier introduction rules by their $\bigwedge$ and $\bigvee$ counterparts. The result is a valid proof with no quantifiers. The depth of the formulas in the proof is still $O(1)$. Sine the number of quantified variables in any formula was $O(n^\varepsilon)$ the size of new formulas is $2^{O(n^\varepsilon)}$. Moreover, since the number of eigen variables in each sequent is $O(n^\varepsilon)$, we only need to make $2^{O(n^\varepsilon)}$ copies of them in total for replacing the quantifier introduction rules. Therefore, the size of the resulting proof is $2^{O(n^\varepsilon)}$. ◀

5 From Nonuniform to Uniform

The other half of the relation between a bounded arithmetic theory and a propositional proof system is given by the provability of the soundness of the propositional proof system (or proof class) inside the corresponding theory. The soundness statement for a proof system Q states that for every φ, π, and τ, if π is a Q-proof of the formula φ, and τ is a truth assignment for φ, then τ satisfies φ: $\forall\varphi, \pi, \tau \ \ (\pi : \mathrm{Q} \vdash \varphi \Rightarrow \tau \vDash \varphi)$. For a proof class that is obtained from taking the union of an indexed family of proof systems the soundness statement is defined as the set of soundness statements for each proof system in the family. For example, we say that a theory proves the the soundness of bdFrege iff it proves $\{Sound(d\text{-}\mathsf{Frege}) \mid d \in \mathbb{N}\}$.

The importance of soundness statements are their universal role in proof complexity similar to complete problems in complexity theory, i.e. proof classes can be characterized as the set of tautology families derivable from the soundness tautology families in a weak propositional proof system like resolution.

Let T be a theory extending ioV^0 and F be a proof class containing polynomial-size $\mathsf{bdFrege}$ and closed under cuts over $\mathsf{bd}\Sigma_0^q$ formulas. Assume that F proves the translation of Σ_0^B theorems of T. We have

▶ **Theorem 18.** *If* T *proves the soundness of* F′*, then* F′ ⊆ F*.*

Proof. First, if $\{\pi_n\}_n : \mathrm{F}' \vdash \{\varphi_n\}_n$, then this family has a polynomial-size $\mathsf{bdFrege}$ proof. Second, $\{(\tau \vDash \varphi_n) \Rightarrow \varphi_n(\tau)\}_n$ has a polynomial-size $\mathsf{bdFrege}$ proof. Now, since the soundness of F′ is provable in T, the translation of the soundness of F′, $\{\pi_n \vdash \varphi_n \Rightarrow \tau \vDash \varphi_n\}_n$, is provable in F. F is closed under $\mathsf{bd}\Sigma_0^q$ cuts, and proves $\{\pi_n : \mathrm{F}' \vdash \varphi_n\}_n$, therefore $\{\tau \vDash \varphi_i\}_i$ is provable in F. Which means $\{\varphi_n(\tau)\}_n$ is provable in F. To make the argument complete, we need discuss the accepting computations of the proof system containing F and the evaluation of formulas. See [13, 12] for details. ◀

▶ **Theorem 19.** *Let* F *be a* Q*-proof class satisfying conditions mentioned above. If the soundness of a proof system* Q′ *is provable in* F*, then* Q *simulates* Q′ *with* F *proofs. If the* F*-proofs are soundness of* Q′ *are effectively given, then the simulation is effective.*

Proof. The proof is similar to the proof of Theorem 18. ◀

5.1 ioV^0 Proves Soundness of $\mathsf{bdFrege}$

▶ **Theorem 20** ([7])**.** *The soundness of the proof class* $\mathsf{bdFrege}$ *is provable in* V^0*.*

In soundness statements, the proof is given to us as an input. Since the size of the formulas in the proof are bounded by the size of the proof, we can easily evaluate these formulas in ioV^0 using the comprehension axiom. Similarly, the size of the proof is an input-type number, so we can use the induction axiom to prove that all sequents in the proof are true under a given assignment. At no point in the argument do we need to compute large values, so the provability of soundness of $\mathsf{bdFrege}$ works in ioV^0. Therefore

▶ **Theorem 21.** *The soundness of* $\mathsf{bdFrege}$ *is provable in* ioV^0*.*

5.2 ioVNC^1 Proves Soundness of (Unbalanced) Frege

The comprehension axiom of ioVNC^1 can be used to evaluate balanced formulas and therefore ioVNC^1 can prove the soundness of Frege proofs where formulas in the proof are balanced.

▶ **Theorem 22.** $\mathsf{ioVNC}^1 \vdash \mathit{Sound}(\mathsf{BalancedFrege})$.

But that does not necessarily imply that ioVNC^1 can prove the soundness of (unbalanced) Frege proofs. We need to balance formulas and provably so in ioVNC^1. It turns out that ioVNC^1 can prove the Buss's result [3, 4] that (unbalanced) Boolean formulas can be evaluated in $\mathsf{ALogTime}$ (which is equivalent to uniform NC^1). The Buss's proof [4] is formalized in VNC^1, see [7, pp. 410-424].

▶ **Theorem 23.** ioVNC^1 *proves the totality and correctness of Buss's* $\mathsf{ALogTime}$ *algorithm [4] for unbalanced Boolean formula evaluation problem.*

Proof. The goal is to prove that we can evaluate unbalanced formulas, i.e. for a formula and a truth assignment given in A and B, there is a Y which is a computation of A on B. Note that the computation doesn't need to encode the values obtained for gates, the evaluation is correct, i.e. it distributes over logical operations and correctly computes the value of $\top$

and $\bot$. We can use an AC^0 function to build a balanced formula Z from A using Buss's algorithm, and then apply MBBFE to Z and obtain a computation Y of it. Note that the $\Sigma_0^B(\mathsf{MBBFE})$-CA axiom allows this. The game tree of Buss's algorithm only depends on the size of the formula. The part of the balanced formula that depends on the formula is a TC^0 functions that given a game play and a formula decides the winner. We attach a balanced Boolean formula computing this TC^0 function to the leaves of the game tree.

Note that for correctness we don't need to compute any global function of output-type objects. So the argument in [7] still works. ◀

▶ **Corollary 24.** *The (unbalanced) Boolean formula evaluation is provably total in* ioVNC^1.

Now, following the standard argument we can prove the soundness of (unbalanced) Frege in ioVNC^1.

▶ **Theorem 25.** ioVNC^1 *proves the soundness of (unbalanced)* Frege

Proof. Let $\pi : \mathsf{Frege} \vdash \varphi$ be a Frege proof of φ. We show that φ is true. Let τ be an arbitrary truth assignment for the formulas in π. We show by induction on the size of π that the sequents in π are true under τ. For the base case, we have to verify that the axioms are true which is straight forward. For the induction step, we have to show that the rules preserver truth of sequents. The correctness of all rules can be verified by case analysis. We use Theorem 24 to show the correctness of the cut rule. ◀

5.3 n^ε-ioV^∞ Proves Soundness of n^ε-bdG_∞

First note that as a corollary of Theorem 21 we have

▶ **Corollary 26.** *The soundness of* bdFrege *is provable in* n^ε-ioV^∞.

▶ **Theorem 27.** *The soundness of* n^ε-bdG_∞ *is provable in* n^ε-ioV^∞.

Proof Idea. The argument follows the same structure of provably of soundness results. ◀

6 $\mathsf{ioVNC}^1 \subseteq n^\varepsilon$-$\mathsf{ioV}^\infty$

In this section, we prove that n^ε-$\mathsf{ioV}^\infty \vdash \mathsf{ioVNC}^1$ which is essentially formalizing and proving correctness of $\mathsf{NC}^1 \subseteq \mathsf{AltTime}(\mathsf{O}(1), \mathsf{O}(\mathsf{n}^\varepsilon))$. The argument also applies to other nice classes in $\mathsf{NTimeSpace}(\mathsf{n}^{\mathsf{O}(1)}, \mathsf{n}^{\mathsf{o}(1)})$ like NL.

▶ **Theorem 28.** *The theories* n^ε-ioV^∞ *contain the theory* ioVNC^1.

Proof of Theorem 28. We only need to derive $\Sigma_0^B(\mathsf{MBBFE})$-CA in n^ε-ioV^∞. Our goal is to show that there is a $\Sigma_\infty^B(n^\varepsilon)$ formula $\psi(x, A)$ which provably gives the bit graph of the computation Y of circuit Z in $\Sigma_0^B(\mathsf{MBBFE})$-CA. In other words, $\psi(x, A)$ iff the value computed for gate x of the circuit Z is one. Therefore by $\Sigma_\infty^B(n^\varepsilon)$-CA the computation of circuit Z exists and we are done. Let's fix φ and s in $\Sigma_0^B(\mathsf{MBBFE})$-CA Note that s is bounded by a polynomial term in the size of free variables of the formula.

We think of φ as representing the graph of an AC^0 function. Abusing the notation, we use $\varphi(A)$ to denote this function. Consider a given A of length n. We can prove the existence of the $Z = \varphi(A)$ using the comprehension axiom in ioV^0. Let;s assume that Z and x are given.

Formula ψ is similar to Buss's algorithm. We describe it as a game between two players with k rounds. The first player claims that the correct value of gate x is 1 while the second player claims that is not the case. We refer to them as P (Prover) and C (Challenger).

Let Z be a balanced Boolean formula of size s and x a gate in Z. We divide Z into k levels. This results in subformulas of size $O(s^{1/k})$. We look at each of these small subformulas as a possible round in the game. The game tree has depth k and branching $O(s^{1/k})$. Each of these small formulas can be described by a path from the root to it. We index the subformulas, their inputs, and their computation using sequences of number $w = (i_1, i_2, \cdots, i_l)$ where $0 \leq l < k$ and each number is less than $s^{1/k}$. For example, the subformula in the root is indexed as $Z_{()}$. The subformulas below it are index as $Z_{(1)}$, $Z_{(2)}$, ... Each round is played in one of these subformulas, starting at the root subformula. After each step we will move to one of the subformulas below the current one. The game is finished when we reach a leave.

The game starts by the P giving a computation of the top subformula including the inputs to that subformula. If the computation is not correct P loses. Otherwise C challenges one of the inputs to the subformula whose output is the challenged input for the previous subformula. The games continues by moving to that subformula. The game ends when we reach the original input bits. At which point we can check if all claims (the claimed values for gates in each subcircuit are consistent and the value of the output gate of each subcircuit is equal to the value of the challenged input bit of the upper subcircuit) by P are correct, in which case P wins. Otherwise C wins.

The player P represents existential string quantifiers of size $O(s^{1/k})$. The player C represents universal number quantifiers of size $O(s^{1/k})$. Note that given a circuit, an input, and a computation, the fact that computation is correct is expressible as a Σ_0^B formula. We have k blocks of these quantifier followed by a Σ_0^B formula that checks if the claims are correct: 1. for each subformula in a game; the computation given for that subformula is compatible with the gates for that subformula; 2. the value of root for each subformula is equal the value of the leaf challenged in the previous round by C; 3. gate x belongs to one of the subformulas in the game; 4. the value of gate x is 1. If we pick $k = \frac{\lg s}{\varepsilon \lg n}$ then $s^{1/k} \leq n^{\varepsilon}$ and ψ is a $\Sigma_\infty^B(n^{\varepsilon})$ formula. Note that ϕ is a σ_0^B formula giving the bit graph of Z and we can easily replace membership in and length of Z by φ and s.

Now that we have defined our Σ_0^B formula for φ and s, we have to show that it gives the bits of the computation of Z according to the definition in $\Sigma_0^B(\mathsf{MBBFE})$-CA, i.e. the value for each gate should be compatible with the type of the gate and the values for its children. This is mainly case analysis: either the gate is inside a subformula, in which case it is the case, or it is on the boarder between two level, in which case it is assigned the same values in both subformula computations. This completes the proof. ◀

7 Simulation of Frege proofs with bdFrege proofs

In this section, we combine the results from previous sections to provide an alternative proof of [10] that Frege proofs can be simulated by bdFrege proofs with a $2^{O(n^{\varepsilon})}$ increase in proof size , i.e. size $2^{O(n^{\varepsilon})}$bdFrege simulates polynomial-size Frege. These results also hold for size $2^{O(n^{\varepsilon})}$bdFrege replaced with size $2^{O(n^{\varepsilon})}\mathsf{bdG}_0$.

Combining n^{ε}-$\mathsf{ioV}^{\infty} \vdash \mathsf{ioVNC}^1$ from theorem 28 and $\mathsf{ioVNC}^1 \vdash \mathit{Sound}(\mathsf{Frege})$ from theorem 25 we obtain the following corollary:

▶ **Corollary 29.** *The proof class n^{ε}-ioV^{∞} proves the soundness of the* Frege *proof system.*

Now, by Theorems 18 and 19 we have:

▶ **Corollary 30.** *The proof class polynomial size n^{ε}-bdG_{∞} contains the proof class polynomial-size* Frege. *The proof system* G *effectively simulates proof system* Frege *by polynomial-size n^{ε}-bdG_{∞} proofs.*

As a corollary of Theorem 17 that size $2^{O(n^\varepsilon)}$ bdFrege contains n^ε-bdG$_\infty$ we get

▶ **Corollary 31.** *The proof class size* $2^{O(n^\varepsilon)}$bdFrege *contains the proof class polynomial-size* Frege. *Size* $2^{O(n^\varepsilon)}$bdFrege *simulates polynomial-size* Frege. *The proof system* bdFrege *effectively simulates* Frege *by proofs of size* $2^{O(n^{n^\varepsilon})}$.

Acknowledgements. We would like to thank Phuong The Nguyen for many helpful discussions during the last two years. The idea for a uniform version of [10] came up during a discussion with Yuval Filmus, Toniann Pitassi, and Rahul Santhanam. We would like to thank the anonymous referees for their pointing out several sections which were unclear, and for their detailed and constructive comments and suggestions. The proofs of some lemmas are omitted because of length restrictions. Some sections of the current version may need further clarification. An earlier version of this work was presented in Fall 2011 by the first author while attending MALOA's special semester on Logic and Complexity in Prague. He would like to thank the organizers and participants, in particular Jan Krajíček, for creating a lovely environment for research and exchange of ideas.

References

1 Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1 – 14:36, 2010.
2 Samuel R. Buss. *Bounded Arithmetic*. Bibliopolis, 1986.
3 Samuel R. Buss. The boolean formula value problem is in alogtime. In Alfred V. Aho, editor, *STOC*, pages 123–131. ACM, 1987.
4 Samuel R. Buss. Algorithms for boolean formula evaluation and for tree-contraction. In P. Clote and J. Krajicek, editors, *Proof Theory, Complexity, and Arithmetic*, pages 95–115. Oxford University Press, 1993.
5 Samuel R. Buss and Ryan Williams. Limits on alternation-trading proofs for time-space lower bounds. In *IEEE Conference on Computational Complexity*, pages 181–191. IEEE, 2012.
6 Stepehn A. Cook. Feasibly constructive proofs and propositional calculus. In *Annual ACM Symposium on Theory of Computing*, volume 7, pages 83–97, 1975.
7 Stepehn A. Cook and Phoung Nguyen. *Logical Fouondations of Proof Complexity*. Cambridge University Press, 2010.
8 Stephen Cook and Tsuyoshi Morioka. Quantified propositional calculus and a second-order theory for NC1. *Archive for Mathematical Logic*, 44(6):711–749, 2005.
9 Stephen A. Cook. Relativized propositional calculus. Manuscript, 2012.
10 Y. Filmus, T. Pitassi, and R. Santhanam. Exponential lower bounds for ac-frege imply superpolynomial frege lower bounds. *Proceedings ICALP*, 2011:618–629, 1.
11 Emil Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129:1–37, 2004.
12 Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Cambridge University Press, 1995.
13 Jan Krajíček. A note on sat algorithms and proof complexity. *Information Processing Letters*, 112(12):490–493, June 2012.
14 Sebastian Müller. Polylogarithmic cuts in models of V^0. *LMCS*, 9:2013, 2013.
15 V.A. Nepomnjascij. Rudimentary predicates and turing calculations. *Doklady AN SSSR*, 195, 1970.
16 J. Paris and A. Wilkie. Counting problems in bounded arithmetic. In CarlosAugusto Prisco, editor, *Methods in Mathematical Logic*, volume 1130 of *Lecture Notes in Mathematics*, pages 317–340. Springer Berlin Heidelberg, 1985.

The Structure of Interaction*

Stéphane Gimenez and Georg Moser

Institute of Computer Science
University of Innsbruck, Austria
{stephane.gimenez,georg.moser}@uibk.ac.at

Abstract

Interaction nets form a local and strongly confluent model of computation that is per se parallel. We introduce a Curry–Howard correspondence between well-formed interaction nets and a deep-inference deduction system based on linear logic. In particular, linear logic itself is easily expressed in the system and its computational aspects materialise though the correspondence. The system of interaction nets obtained is a typed variant of already well-known sharing graphs. Due to a strong confluence property, strong normalisation for this system follows from weak normalisation. The latter is obtained via an adaptation of Girard's reducibility method. The approach is modular, readily gives rise to generalisations (e.g. second order, known as polymorphism to the programmer) and could therefore be extended to various systems of interaction nets.

1998 ACM Subject Classification F.1.2 Modes of Computation, F.4.1 Mathematical Logic

Keywords and phrases Interaction Nets, Linear Logic, Curry–Howard Correspondence, Deep Inference, Calculus of Structures, Strong Normalisation, Reducibility

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.316

1 Introduction

We introduce a deep-inference deduction system based on multiplicative-exponential linear logic (MELL for short) [4] and provide a direct correspondence with interaction nets systems [11], among which sharing graphs [12, 5, 6] play a key role. Our calculus was directly inspired by, and is in large parts identical to, an earlier presentation of MELL in the calculus of structures given by Guglielmi and Straßburger [16, 15]. We thus unveil a Curry–Howard correspondence between deep-inference formalisms of linear logic and the simple, almost canonical, parallel computation model portrayed by interaction nets. On the one hand, the deep-inference deduction system fulfills the role of a long-awaited enhanced type system for nets: the additional structure conferred to nets through typing ensures correctness, and can furthermore, under some reasonable assumptions, guarantee termination. On the other hand, interaction nets provide an answer to the unsettled topic of a computational interpretation for proof normalization steps in deep-inference systems.

Interaction nets, introduced by Lafont in [11], form an abstract model of computation based on graph rewriting [18]. Reduction of interaction nets is *strongly confluent*: pairs of interacting agents can not only be contracted locally but also independently, and therefore any peak can be joined immediately. This gives rise to an elegant formalism to express parallel computations. These and other merits make interaction nets a promising programming paradigm, either as an execution platform for functional programs, or as a conceptional device for the (optimal) implementation of the λ-calculus [6, 13, 7]. However, the elegance

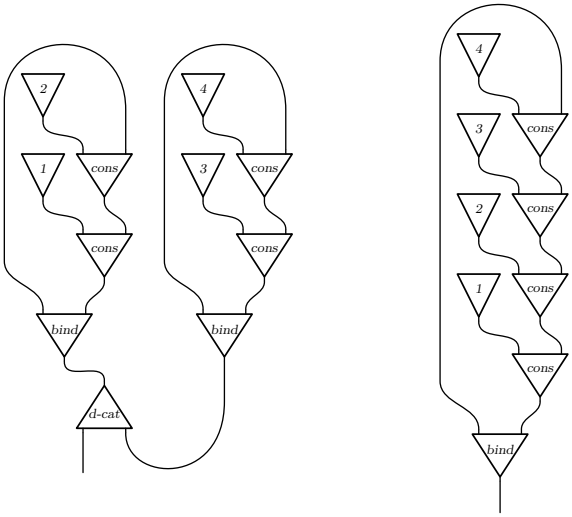
* This work is partially supported by FWF (Austrian Science Fund) project I-603-N18.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 316–331

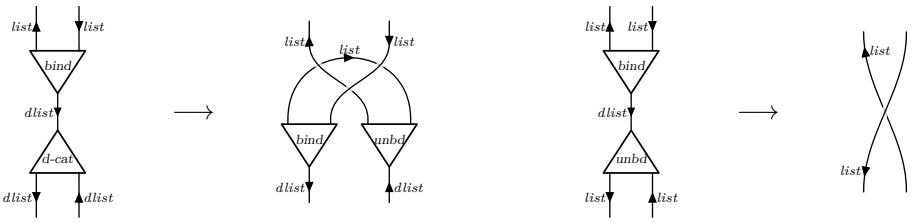
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of nets has always been undermined by the lack of a versatile framework to guarantee basic correctness assumptions.

▶ **Example 1.1.** As an illustration, we reuse the difference lists example from [11], an implementation of lists that allows constant time concatenation. The two nets pictured below describe the concatenation of integer lists $[1, 2]$ and $[3, 4]$, before and after evaluation. Node *cons* is used as a standard list constructor, and *bind* creates a difference list from links to the tail and to the head of a standard list.



The computation is performed by means of the following rewriting rules:



After two reduction steps, reduction of the initial net yields the given normal form.

Such simple examples suffice to understand that interaction nets crucially miss structure, making reasoning about them difficult. Individual portions of nets found in left and right hand side of reduction rules are not standardly considered valid, even though they are subnets of valid nets. The reason being that usual net-construction rules available for this system, which are inherited from sequent-calculus inference rules, are too coarse-grained. Without preserving additional typing information, the inherent well-foundness of such rewriting rules could not be formulated directly [11]; a proof of a correctness preservation property would have to be conducted globally for the full reduction relation. The correspondence proposed in this paper allows for a finer analysis of nets, keeping track of their substantial structure, and is able to do so even in the most delicate case of sharing graphs.

Deep-inference deduction systems allow inference rules to modify formulas inside an arbitrary context. For most logics deep-inference presentations have been established [16, 9, 17, 8]. Their flexibility enables us to type and to ensure proper combination of individual cell-components of a net. We can fuse through this type system the essential properties of interaction nets and MELL. From nets we inherit strong confluence and from MELL

we inherit weak normalisation. Technically we adapt Girard's *reducibility method* to the context of deep-inference formalism to obtain weak normalisation. The proof is modular and can easily be extended, for instance, to additive connectives (which are used to type conditionals) or to second-order (polymorphism).

This work is related to very recent work on computational interpretations of deep-inference systems. In [10] Gundersen, Heijltjes and Parigot introduce the *atomic lambda calculus* as a typed λ-calculus that admits some particular form of sharing and preserves strong normalisation. The atomic lambda calculus provides (among others) a computational interpretation of the medial rule of calculi of structures [17]. Furthermore, recently, Pagani and Tortora de Falco established in [14] the very tedious and technical standardisation theorem required to prove strong normalisation of second-order linear logic. Its "delicate but boring" proof had been postponed but Girard's reducibility method however suffices in showing weak normalisation. When adapted to our framework, thanks to strong confluence, weak normalisation automatically promotes to a strong normalisation theorem.

This paper is structured as follows. In Section 2 we provide the proposed deep-inference presentation of MELL together with an embedding of its standard sequent calculus presentation. Section 3 defines the system of interaction nets which is of interest, picturing the direct correspondance with this presentation of MELL. In Section 4, we define reduction steps on the logic side, in a way that preserves strong confluence from nets. Section 5 adapts Girard's reducibility method to this setting and establishes weak normalisation (and thus strong normalisation). Finally, we conclude in Section 6 and mention potential future work.

2 A Convenient Presentation of Linear Logic with Structures

Structures. We define structures as follows:

$$\sigma \quad ::= \quad \star \;\mid\; \sigma \,;\, \sigma \;\mid\; \circ \;\mid\; \sigma \,,\, \sigma \;\mid\; \Box\sigma \;\mid\; \Diamond\sigma \;\mid\; A$$

This syntax is then quotiented so that binary connectives $\langle\ ;\ \rangle$ and $\langle\ ,\ \rangle$ are associative, commutative, respectively admit $\langle\ \star\ \rangle$ and $\langle\ \circ\ \rangle$ as neutral elements, and so that $\star = \Box\star$ and $\circ = \Diamond\circ$. Connectives $\star$ and $\circ$ can conveniently be thought of and referred to as "true" and "false" respectively. Last, A ranges over linear logic formulas:

$$A \quad ::= \quad 1 \;\mid\; A \otimes A \;\mid\; \bot \;\mid\; A \parr A \;\mid\; !A \;\mid\; ?A \;\mid\; \alpha \;\mid\; \bar{\alpha}$$

where α ranges over some arbitrary set of base types.

Negation $\sigma^\perp$ of a structure σ is an involutive operation defined on linear logic formulas according to the usual de Morgan laws, and naturally extended to structures with:

$$\star^\perp = \circ \qquad (\sigma \,;\, \tau)^\perp = \sigma^\perp \,,\, \tau^\perp \qquad (\Box\sigma)^\perp = \Diamond(\sigma^\perp)$$

Our approach is similar in concept to other calculus of structures presentations [16, 1, 9, 10]. Among modifications introduced, a "computational layer" now appears underneath the "structural layer". This will allow for a direct correspondence with interaction nets systems.

Derivations. A derivation π from σ to τ, written $\pi : \sigma \to \tau$, is a sequence of structures whose first element is σ and last element is τ, such that every succession of two structures in this sequence is associated with a derivation rule. Basic derivation rules are split into three main categories.

- *Core structural rules*

$$\frac{\star}{\sigma^\perp \ , \ \sigma}\ \mathit{axiom} \qquad \frac{\sigma \ ; \ \sigma^\perp}{\circ}\ \mathit{cut} \qquad \frac{\omega \ ; \ (\sigma \ , \ \tau)}{(\omega \ ; \ \sigma) \ , \ \tau}\ \mathit{switch} \qquad \frac{\Box\sigma \ ; \ \Box\tau}{\Box(\sigma \ ; \ \tau)}\ \mathit{merge}$$

- *Administrative rules*

$$\frac{\Box\sigma}{\star}\ \mathit{erase}\uparrow \qquad \frac{\Box\sigma}{\Box\sigma \ ; \ \Box\sigma}\ \mathit{duplicate}\uparrow \qquad \frac{\Box\sigma}{\sigma}\ \mathit{open}\uparrow \qquad \frac{\Box\sigma}{\Box\Box\sigma}\ \mathit{nest}\uparrow$$

- *Computational rules* (we restrict here ourselves to the following set of rules which is sufficient to externalise linear logic)

$$\frac{\star}{1}\ \mathit{one}\downarrow \qquad \frac{\circ}{\bot}\ \mathit{bottom}\downarrow \qquad \frac{A \ ; \ B}{A \otimes B}\ \mathit{tensor}\downarrow \qquad \frac{A \ , \ B}{A \parr B}\ \mathit{par}\downarrow$$

$$\frac{\Box A}{!A}\ \mathit{of\text{-}course}\downarrow \qquad \frac{\Diamond A}{?A}\ \mathit{why\text{-}not}\downarrow$$

For reasons that follow, the following rule is admissible:

$$\frac{\Box(\sigma \ , \ \tau)}{\Box\sigma \ , \ \Diamond\tau}\ \mathit{select} \ := \ \cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\Box(\sigma \ , \ \tau)}{\Box(\sigma \ , \ \tau) \ ; \ (\Box\tau^\perp \ , \ \Diamond\tau)}\ \mathit{switch}}{(\Box(\sigma \ , \ \tau) \ ; \ \Box\tau^\perp) \ , \ \Diamond\tau}\ \mathit{merge}, -}{\Box((\sigma \ , \ \tau) \ ; \ \tau^\perp) \ , \ \Diamond\tau}\ \Box\mathit{switch}, -}{\Box(\sigma \ , \ (\tau \ ; \ \tau^\perp)) \ , \ \Diamond\tau}\ \Box(-, \mathit{cut}), -}{\Box\sigma \ , \ \Diamond\tau}$$

Deep inference. All basic deduction rules can be applied inside a structural context. If $\rho : \sigma \to \tau$ is a basic deduction rule and ν is a structural context (a structure with one hole), then $\nu[\rho] : \nu[\sigma] \to \nu[\tau]$ is also accepted as a deduction rule. For example, *tensor*$\downarrow$ can be applied inside disjunctive, conjunctive or exponential contexts, as well as any combination thereof, as follows:

$$\frac{\sigma \ , \ (A \ ; \ B)}{\sigma \ , \ A \otimes B}\ -, \mathit{tensor}\downarrow \qquad \frac{\sigma \ ; \ A \ ; \ B}{\sigma \ ; \ A \otimes B}\ -; \mathit{tensor}\downarrow \qquad \frac{\Box(A \ ; \ B)}{\Box(A \otimes B)}\ \Box\mathit{tensor}\downarrow$$

Symmetry. Besides, all rules can be turned upside-down: each rule $\rho : \sigma \to \tau$ is to be paired with a matching $\rho^\perp : \tau^\perp \to \sigma^\perp$ rule (calculus of structures implements contraposition natively). Core structural rules are paired with core structural rules, in particular *axiom* and *cut* are paired together and *switch* happens to be self-symmetric. Introduction rules (labeled with an arrow oriented downwards) are turned into elimination rules (labeled with an arrow oriented upwards) and vice versa. For example, symmetric variants of *tensor*$\downarrow$, *duplicate*$\uparrow$ and *merge* write as follows:

$$\frac{A \parr B}{A \ , \ B}\ \mathit{tensor}\uparrow \qquad \frac{\Diamond\sigma \ , \ \Diamond\sigma}{\Diamond\sigma}\ \mathit{duplicate}\downarrow \qquad \frac{\Diamond(\sigma \ , \ \tau)}{\Diamond\sigma \ , \ \Diamond\tau}\ \mathit{merge}$$

Given any structure ω, an empty sequence derives ω to ω itself. Such derivations are denoted by id_ω. Composition of two derivations $\pi_1 : \sigma \to \omega$ and $\pi_2 : \omega \to \tau$ is written $\pi_1 \cdot \pi_2 : \sigma \to \tau$. This defines a category whose objects are structures and whose morphisms are derivations.

Relaxed derivations. We quotient previously defined derivations by a few trivialities (and their symmetric variants, which are not listed), which are natural consequences of the quotient on structures.

$$\frac{\star}{\star \,,\, \circ}\,\mathit{axiom} \;\equiv\; \frac{\star}{\star}\,\mathit{id} \qquad \frac{\star \,;\, (\sigma \,,\, \tau)}{(\star \,;\, \sigma) \,,\, \tau}\,\mathit{switch} \;\equiv\; \frac{\sigma \,,\, \tau}{\sigma \,,\, \tau}\,\mathit{id} \qquad \frac{\Box\star \,;\, \Box\omega}{\Box(\star \,;\, \omega)}\,\mathit{merge} \;\equiv\; \frac{\Box\omega}{\Box\omega}\,\mathit{id}$$

$$\frac{\Box\star}{\star}\,\mathit{erase}\uparrow \;\equiv\; \frac{\Box\star}{\Box\star \,;\, \Box\star}\,\mathit{duplicate}\uparrow \;\equiv\; \frac{\Box\star}{\star}\,\mathit{open}\uparrow \;\equiv\; \frac{\Box\star}{\Box\Box\star}\,\mathit{nest}\uparrow \;\equiv\; \frac{\star}{\star}\,\mathit{id}$$

We moreover allow rules with disjoint scopes to freely pass each other. For example, given $\rho_1 : \sigma_1 \to \tau_1$ and $\rho_2 : \sigma_2 \to \tau_2$:

$$\frac{\dfrac{\sigma_1 \,;\, \sigma_2}{\tau_1 \,;\, \sigma_2}\,\rho_1 ; -}{\tau_1 \,;\, \tau_2}\,-\,;\rho_2 \;\equiv\; \frac{\dfrac{\sigma_1 \,;\, \sigma_2}{\sigma_1 \,;\, \tau_2}\,-\,;\rho_2}{\tau_1 \,;\, \tau_2}\,\rho_1 ; -$$

This, extended to arbitrary derivations $\pi_1 : \sigma_1 \to \tau_1$ and $\pi_2 : \sigma_2 \to \tau_2$, enables us to define contextual conjunction of derivations $\pi_1 \,;\, \pi_2 : \sigma_1 \,;\, \sigma_2 \to \tau_1 \,;\, \tau_2$, as well as disjunction $\pi_1 \,,\, \pi_2 : \sigma_1 \,,\, \sigma_2 \to \tau_1 \,,\, \tau_2$ unambiguously. Given $\pi : \sigma \to \tau$, we use similarly $\Box\pi : \Box\sigma \to \Box\tau$ and $\Diamond\pi : \Diamond\sigma \to \Diamond\tau$ to denote the use of a derivation π inside an exponential context.

Other equivalences occur when the scope of one rule is captured within one structural variable of an adjacent core structural rule. We provide the following, given an arbitrary rule $\rho : \sigma \to \tau$, as an illustration:

$$\frac{\dfrac{\omega \,;\, (\sigma \,,\, \omega')}{\omega \,;\, (\tau \,,\, \omega')}\,-\,;(\rho\,,-)}{(\omega \,;\, \tau) \,,\, \omega'}\,\mathit{switch} \;\equiv\; \frac{\dfrac{\omega \,;\, (\sigma \,,\, \omega')}{(\omega \,;\, \sigma) \,,\, \omega'}\,\mathit{switch}}{(\omega \,;\, \tau) \,,\, \omega'}\,(-\,;\rho)\,,-$$

Given that the use of structural variables in rules *axiom* and *cut* is also linear in some sense, similar commutations are considered. Those are however slightly less straightforward as they reverse orientation of rules.

$$\frac{\dfrac{\star}{\sigma \,,\, \sigma^\perp}\,\mathit{axiom}}{\tau \,,\, \sigma^\perp}\,\rho\,,-} \;\equiv\; \frac{\dfrac{\star}{\tau \,,\, \tau^\perp}\,\mathit{axiom}}{\tau \,,\, \sigma^\perp}\,-\,,\rho^\perp} \qquad \frac{\dfrac{\sigma \,;\, \tau^\perp}{\tau \,;\, \tau^\perp}\,\rho ; -}{\circ}\,\mathit{cut} \;\equiv\; \frac{\dfrac{\sigma \,;\, \tau^\perp}{\sigma \,;\, \sigma^\perp}\,-\,;\rho^\perp}{\circ}\,\mathit{cut}$$

In fact, all definitions and properties about derivations mentioned in the sequel are compatible with these equivalences. From now on, the term *derivation* will be used to refer to equivalence classes, which abstract away the irrelevant sequentiality found in concrete syntactic derivations.

An embedding of the sequent calculus. Any sequent calculus proof Π of formula A in MELL can be translated to a derivation $\pi : \star \to A$, which we consider a proof of formula A in calculus of structures. The basic idea is to combine all formulas found inside hypothesis or conclusion sequents using $\langle \;,\; \rangle$ connectives and combine hypothesis sequents themselves with $\langle \;;\; \rangle$ connectives. Inference rules from the one-sided sequent calculus are of shape (a):

$$(a)\quad \frac{\vdash A_1, \ldots, A_n \quad \cdots \quad \vdash B_1, \ldots, B_m}{\vdash C_1, \ldots, C_k} \qquad (b)\quad \frac{(A_1 \,,\, \cdots \,,\, A_n) \,;\, \cdots \,;\, (B_1 \,,\, \cdots \,,\, B_m)}{\begin{array}{c}\vdots\\ C_1 \,,\, \cdots \,,\, C_k\end{array}}$$

Every specific inference rule will be encoded as a derivation of shape (b). Since derivations associated to branches of an arborescent proof may be combined by means of structural contexts, the tree structure of sequent-calculus proofs is then easily embedded inside a sequential derivation.

The encoding of individual rules goes as follows (whenever Γ or Δ is used to denote a disjunction of formulas in sequents, γ or δ denotes the same disjunction in structures):

- *Multiplicatives*

$$\frac{\vdash \Gamma, A \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}\ tensor \quad \mapsto \quad \frac{\dfrac{(\gamma\ ,\ A)\ ;\ (B\ ,\ \delta)}{\gamma\ ,\ (A\ ;\ B)\ ,\ \delta}\ switch(\times 2)}{\gamma\ ,\ A \otimes B\ ,\ \delta}\ -, tensor\downarrow, - \qquad \frac{}{\vdash 1}\ one \quad \mapsto \quad \frac{\star}{1}\ one\downarrow$$

$$\frac{\vdash A, B, \Gamma}{\vdash A \parr B, \Gamma}\ par \quad \mapsto \quad \frac{A\ ,\ B\ ,\ \gamma}{A \parr B\ ,\ \gamma}\ par\downarrow, - \qquad \frac{\vdash \Gamma}{\vdash \bot, \Gamma}\ bottom \quad \mapsto \quad \frac{\gamma}{\bot\ ,\ \gamma}\ bottom\downarrow, -$$

- *Identities*

$$\frac{}{\vdash A, A^\perp}\ axiom \qquad \mapsto \qquad \frac{\star}{A\ ,\ A^\perp}\ axiom$$

$$\frac{\vdash A, \Gamma \quad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}\ cut \qquad \mapsto \qquad \frac{\dfrac{(\gamma\ ,\ A)\ ;\ (A^\perp\ ,\ \delta)}{\gamma\ ,\ (A\ ;\ A^\perp)\ ,\ \delta}\ switch(\times 2)}{\gamma\ ,\ \delta}\ -, cut, -$$

- *Exponentials*

$$\frac{\vdash \Gamma}{\vdash ?A, \Gamma}\ weakening \qquad \mapsto \qquad \frac{\dfrac{\gamma}{\Diamond A\ ,\ \gamma}\ erase\downarrow, -}{?A\ ,\ \gamma}\ why\text{-}not\downarrow, -$$

$$\frac{\vdash ?A, ?A, \Gamma}{\vdash ?A, \Gamma}\ contraction \qquad \mapsto \qquad \frac{\dfrac{\dfrac{?A\ ,\ ?A\ ,\ \gamma}{\Diamond A\ ,\ \Diamond A\ ,\ \gamma}\ of\text{-}course\uparrow, of\text{-}course\uparrow, -}{\Diamond A\ ,\ \gamma}\ duplicate\downarrow, -}{?A\ ,\ \gamma}\ why\text{-}not\downarrow, -$$

$$\frac{\vdash A, \Gamma}{\vdash ?A, \Gamma}\ dereliction \quad \mapsto \quad \frac{\dfrac{A\ ,\ \gamma}{\Diamond A\ ,\ \gamma}\ open\downarrow, -}{?A\ ,\ \gamma}\ why\text{-}not\downarrow, - \qquad \frac{\vdash ??A, \Gamma}{\vdash ?A, \Gamma}\ digging \quad \mapsto \quad \frac{\dfrac{\dfrac{\dfrac{??A\ ,\ \gamma}{\Diamond ?A\ ,\ \gamma}\ of\text{-}course\uparrow, -}{\Diamond\Diamond A\ ,\ \gamma}\ \Diamond of\text{-}course\uparrow, -}{\Diamond A\ ,\ \gamma}\ nest\downarrow, -}{?A\ ,\ \gamma}\ why\text{-}not\downarrow, -$$

Last, we give an encoding of functorial promotion; any non-functorial promotion can as usual be encoded as a composition of one functorial promotion and diggings. A functorial promotion is encoded in one step together with the whole proof of its premise Π. Assuming (inductively) that $\Pi \mapsto \pi$, the promoted branch is encoded as follows:

$$\dfrac{\dfrac{}{\vdash A, B_1, \ldots, B_n}\,\Pi}{\vdash\, !A, ?B_1, \ldots, ?B_n}\,\textit{functorial-promotion} \quad\mapsto\quad \dfrac{\dfrac{\dfrac{\dfrac{\star}{\Box(A\ ,\ B_1\ ,\ \cdots\ ,\ B_n)}\,\Box\pi}{\Box A\ ,\ \Diamond(B_1\ ,\ \cdots\ ,\ B_n)}\,\textit{select}}{\Box A\ ,\ \Diamond B_1\ ,\ \cdots\ ,\ \Diamond B_n}\,-\,,\textit{merge}(\times n)}{!A\ ,\ ?B_1\ ,\ \cdots\ ,\ ?B_n}\,\textit{of-course}\downarrow, \textit{why-not}\downarrow, \cdots, \textit{why-not}\downarrow$$

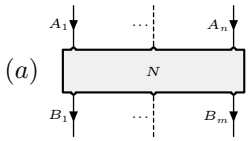
▶ **Lemma 2.1.** *A linear-logic formula A admits a proof Π in the standard sequent-calculus-style presentation of MELL if and only if there exists a derivation $\pi : \star \to A$ in the present calculus of structures.*

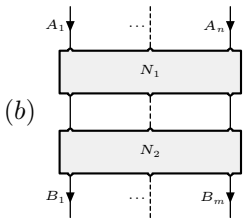
Proof. One direction follows directly from the provided encoding. For the other direction, one can reuse the (simple) inductive argument in [16]. ◀

Provability-wise this system is also equivalent to previous calculus of structure systems for linear logic which were studied by Straßburger [16].

3 The Underlying Computational Model

Translation from structures to interaction nets. Any derivation $\pi : \sigma \to \tau$ projects into a net, as shown in (a):

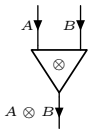




Input wires are typed with formulas $A_1, \ldots, A_n$ found in σ and output wires are typed with formulas $B_1, \ldots, B_m$ found in τ. The composition of any two derivations $\pi_1 : \sigma \to \omega$ and $\pi_2 : \omega \to \tau$, whose respective projections are assumed to be N_1 and N_2, projects as (b).

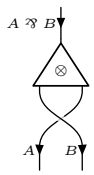
Computational rules are translated by single-cell nets, as described below. These rules may be used inside structural contexts, in which case one wire has to be added to their translations for every formula that appears in the context.

$$\dfrac{A\ ;\ B}{A \otimes B}\,\textit{tensor}\downarrow \quad\mapsto\quad$$



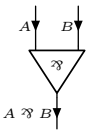
$$\dfrac{\star}{1}\,\textit{one}\downarrow \quad\mapsto\quad$$

$$\dfrac{A \parr B}{A\ ,\ B}\,\textit{tensor}\uparrow \quad\mapsto\quad$$



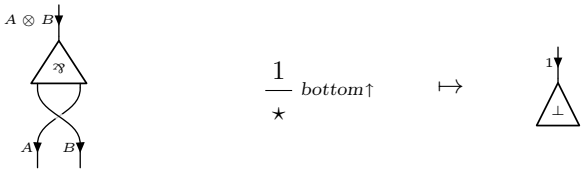
$$\dfrac{\bot}{\circ}\,\textit{one}\uparrow \quad\mapsto\quad$$

$$\dfrac{A\ ,\ B}{A \parr B}\,\textit{par}\downarrow \quad\mapsto\quad$$



$$\dfrac{\circ}{\bot}\,\textit{bottom}\downarrow \quad\mapsto\quad$$

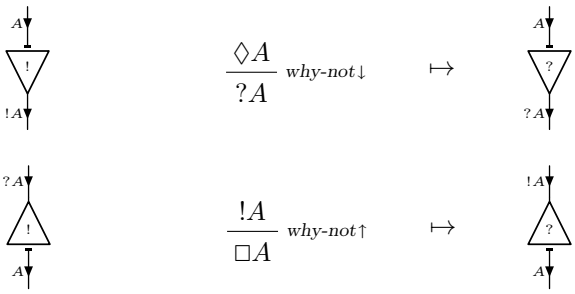
$$\frac{A \otimes B}{A \ ; \ B}\ par\uparrow \quad \mapsto \quad \frac{1}{\star}\ bottom\uparrow \quad \mapsto$$



We use a special notation for exponential cells, to distinguish them from standard dereliction (and co-dereliction) notation. A special marker is added to ports which were originally typed with a structural exponential connective.

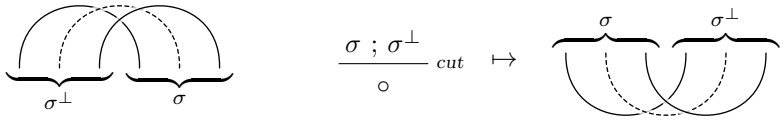
$$\frac{\Box A}{!A}\ of\text{-}course\downarrow \quad \mapsto \quad \frac{\Diamond A}{?A}\ why\text{-}not\downarrow \quad \mapsto$$

$$\frac{?A}{\Diamond A}\ of\text{-}course\uparrow \quad \mapsto \quad \frac{!A}{\Box A}\ why\text{-}not\uparrow \quad \mapsto$$



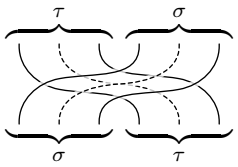
Rules from the identity fragment allow reversing the direction of wires:

$$\frac{\star}{\sigma^\perp \ , \ \sigma}\ axiom \quad \mapsto \quad \frac{\sigma \ ; \ \sigma^\perp}{\circ}\ cut \quad \mapsto$$

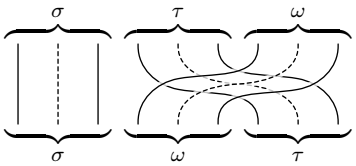


Other core structural rules are simple wirings that reflect implicit permutations of formulas. For example:

$$\frac{\tau \ ; \ \sigma}{\sigma \ ; \ \tau}\ id \quad \mapsto$$

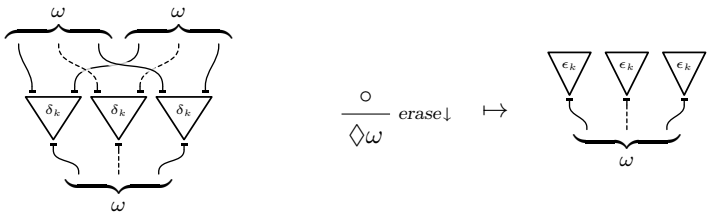


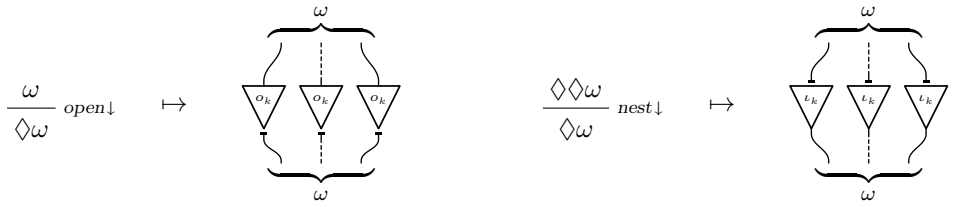
$$\frac{(\sigma \ , \ \tau) \ ; \ \omega}{(\sigma \ ; \ \omega) \ , \ \tau}\ switch \quad \mapsto$$



Only the translation of administrative rules *duplicate*, *erase*, *open* and *nest* requires particular cells, which are indexed by the number k of structural exponential connectives these derivation rules are applied beneath:

$$\frac{\Diamond\omega \ , \ \Diamond\omega}{\Diamond\omega}\ duplicate\downarrow \quad \mapsto \quad \frac{\circ}{\Diamond\omega}\ erase\downarrow \quad \mapsto$$





All structural connectives are lost during the projection. Formulas that appear inside structures can however be used to label wires in accordance with traditional "shallow" typing practices in interaction nets. Derivations appear as an "enriched" form of typing for nets and expose additional information about their structure.

A reduction for the interaction nets system we thus obtained could be defined directly and would be very similar to sharing graphs' reduction. The choice of an appropriate reduction rule when two δ-cells (which correspond to "fan" cells in sharing-graphs terminology) interact would rely on their assigned indexes, which are in fact the only piece of data extracted from the structure of nets that is really needed for computation purposes. But, more conveniently, all the necessary reduction rules can be enriched with structure and written within the deep-inference framework itself, as presented in Section 4. In particular, this automatically ensures the preservation of "enriched" type information after every reduction step.

Type systems for specialised interaction nets. Interaction nets from Example 1.1 can similarly be assigned a type system. This system is simple in the sense that concatenation of difference lists does not require duplication of data. Typing will therefore in this case not rely on structural exponential connectives. Adapting [11], it can be enriched with the following structure (where *dlist*, *list* and *int* are used as base types):

$$\frac{dlist}{\bar{list}\ ,\ list}\ unbd\uparrow \qquad \frac{dlist}{dlist\ ,\ d\bar{list}}\ d\text{-}cat\uparrow$$

$$\frac{\bar{list}\ ,\ list}{dlist}\ bind\downarrow \qquad \frac{int\ ;\ list}{list}\ cons\downarrow \qquad \frac{\star}{int}\ n\downarrow$$

The initial net from our example types as follows:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\star}{(\bar{list}\ ,\ list)\ ;\ (\bar{list}\ ,\ list)}\ axiom;axiom}{(\bar{list}\ ,\ list)\ ;\ (\bar{list}\ ,\ list)}\ (-,cons\_2\downarrow);(-,cons\_4\downarrow)}{(\bar{list}\ ,\ list)\ ;\ (\bar{list}\ ,\ list)}\ (-,cons\_1\downarrow);(-,cons\_3\downarrow)}{dlist\ ;\ dlist}\ bind\downarrow;bind\downarrow}{(dlist\ ,\ d\bar{list})\ ;\ dlist}\ d\text{-}cat\uparrow;-}{dlist\ ,\ (d\bar{list}\ ;\ dlist)}\ switch}{dlist}\ -;cut$$

where:

$$\frac{list}{list}\ cons\_n\downarrow\ :=\ \cfrac{\cfrac{list}{int\ ;\ list}\ n\downarrow;-}{list}\ cons\downarrow$$

Projections of derivations built with the multiplicative structural fragment and the above-provided five rules are correctly built, meaning they are subnets of nets standardly considered as valid. Moreover, it can be shown that any valid net accepts such a derivation.

The fact that reduction preserves correctness of nets is automatically deduced from the fact that known reduction rules for this system of interaction nets can be enriched with

structure as well. For example, the first reduction rule types as follows:

$$\dfrac{\dfrac{\bar{list}\ ,\ list}{dlist}\ bind{\downarrow}}{dlist\ ,\ d\bar{list}}\ d\text{-}cat{\uparrow} \quad\longrightarrow\quad \dfrac{\dfrac{\dfrac{\bar{list}\ ,\ list}{(\bar{list}\ ,\ list)\ ;\ (\bar{list}\ ,\ list)}\ -;axiom}{\bar{list}\ ,\ list\ ,\ (list\ ;\ \bar{list})}\ switch(\times 2)}{dlist\ ,\ d\bar{list}}\ bind{\downarrow}, unbd{\downarrow}$$

4 Reduction within the Deep-Inference Formalism

Considering Curry–Howard correspondences between natural-deduction proof formalisms and several λ-calculus variants, it is natural to understand introduction rules as data constructions and elimination rules as data deconstructions; also to notice that computation arises from the interaction of the former with the latter.

Computation steps. Reduction can be defined in a similar fashion within our calculus by solving every potential interaction between an introduction (a downward-oriented rule) and a matching elimination (an upward-oriented rule) according to rewriting rules of this shape:

$$\dfrac{\dfrac{\sigma}{\omega}\downarrow}{\tau}\uparrow \quad\longrightarrow\quad \dfrac{\sigma}{\vdots}{\tau}$$

In particular, interfaces (hypothesis and conclusion) of derivations are preserved during reduction, which means that computation can be performed deep inside a structural context as well as inside a derivation context.

Our system for linear logic includes the following simple computation steps (symmetric variants are, and will always be, omitted):

$$\dfrac{\dfrac{\star}{1}\ one{\downarrow}}{\star}\ bottom{\uparrow} \quad\longrightarrow\quad \dfrac{\star}{\star}\ id \qquad \dfrac{\dfrac{A\ ;\ B}{A\otimes B}\ tensor{\downarrow}}{A\ ;\ B}\ par{\uparrow} \quad\longrightarrow\quad \dfrac{A\ ;\ B}{A\ ;\ B}\ id$$

$$\dfrac{\dfrac{\Box A}{!A}\ of\text{-}course{\downarrow}}{\Box A}\ why\text{-}not{\uparrow} \quad\longrightarrow\quad \dfrac{\Box A}{\Box A}\ id$$

Administrative reduction steps. In order to concretise any potential interaction between two computational rules, we will force introduction rules downwards and elimination rules upwards, relying on a few reduction rules which are described hereafter and which are to be considered additionally to commutations already assumed by the equivalence on derivations.

First, we explicit direct interactions of administrative deduction rules with *merge*:

$$\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box(\sigma\ ;\ \tau)}\ merge}{\star}\ erase{\uparrow} \quad\longrightarrow\quad \dfrac{\Box\sigma\ ;\ \Box\tau}{\star}\ erase{\uparrow};erase{\uparrow}$$

$$\dfrac{\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box(\sigma\ ;\ \tau)}\,merge}{\Box(\sigma\ ;\ \tau)\ ;\ \Box(\sigma\ ;\ \tau)}\,duplicate\uparrow}\quad\longrightarrow\quad\dfrac{\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box\sigma\ ;\ \Box\sigma\ ;\ \Box\tau\ ;\ \Box\tau}\,id}{\Box\sigma\ ;\ \Box\tau\ ;\ \Box\sigma\ ;\ \Box\tau}\,duplicate\uparrow;duplicate\uparrow}{\Box(\sigma\ ;\ \tau)\ ;\ \Box(\sigma\ ;\ \tau)}\,merge;merge$$

$$\dfrac{\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box(\sigma\ ;\ \tau)}\,merge}{\sigma\ ;\ \tau}\,open\uparrow}\quad\longrightarrow\quad\dfrac{\Box\sigma\ ;\ \Box\tau}{\sigma\ ;\ \tau}\,open\uparrow;open\uparrow$$

$$\dfrac{\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box(\sigma\ ;\ \tau)}\,merge}{\Box\Box(\sigma\ ;\ \tau)}\,nest\uparrow}\quad\longrightarrow\quad\dfrac{\dfrac{\dfrac{\Box\sigma\ ;\ \Box\tau}{\Box\Box\sigma\ ;\ \Box\Box\tau}\,nest\uparrow;nest\uparrow}{\Box(\Box\sigma\ ;\ \Box\tau)}\,merge}{\Box\Box(\sigma\ ;\ \tau)}\,\Box merge$$

Remaining administrative steps take care of promoted content found under exponential contexts. We use the symbol $\check{\rho} : \sigma \to \tau$ to range over blocks of several structural rules whose projections as nets do not link together two inputs with a wire (for instance a single *cut* rule is excluded) or to range over a single introduction rule. Labels $\Box\check{\rho}$ used in left-hand sides of the following reduction steps denote any such deduction pattern used in a context that admits a box connective as top-level connective:

$$\dfrac{\dfrac{\Box\sigma}{\Box\tau}\,\Box\check{\rho}}{\star}\,erase\uparrow\quad\longrightarrow\quad\dfrac{\Box\sigma}{\star}\,erase\uparrow\qquad\dfrac{\dfrac{\Box\sigma}{\Box\tau}\,\Box\check{\rho}}{\Box\tau\ ;\ \Box\tau}\,duplicate\uparrow\quad\longrightarrow\quad\dfrac{\Box\sigma}{\Box\tau\ ;\ \Box\tau}\,\Box\check{\rho};\Box\check{\rho}\,duplicate\uparrow$$

$$\dfrac{\dfrac{\Box\sigma}{\Box\tau}\,\Box\check{\rho}}{\tau}\,open\uparrow\quad\longrightarrow\quad\dfrac{\dfrac{\Box\sigma}{\sigma}\,open\uparrow}{\tau}\,\check{\rho}\qquad\dfrac{\dfrac{\Box\sigma}{\Box\tau}\,\Box\check{\rho}}{\Box\Box\tau}\,nest\uparrow\quad\longrightarrow\quad\dfrac{\Box\sigma}{\Box\Box\sigma}\,nest\uparrow\,\Box\Box\check{\rho}$$

When $\sigma = \star$, the above reduction steps applied to blocks of core structural rules correspond, in the interaction-net formalism, to final steps of administrative reductions performed on nets that are purely made of wires.

Normal forms. A derivation in normal form is a derivation which can be written $\hat{\pi} \cdot \bar{\pi} \cdot \check{\pi}$, where $\hat{\pi}$, $\bar{\pi}$ and $\check{\pi}$ respectively contain eliminations, core structural rules and introductions only (all other configurations can be reduced).

▶ Remark. Although potential interaction between rules *axiom* and *cut* may be considered an artefact (because it leaves the underlying computation model unaffected, wirings are normal forms in interaction nets), we have reasons to believe they could be substituted with oriented variants *axiom*↓ and *cut*↑ to induce a call for further-simplified derivations. Those two rules were however left in the core structural fragment as we unfortunately do not know yet how to properly resolve those interactions when other structural rules interpose. For example, in the following simple case, reduction is possible given that both hand sides project to the same net:

$$\dfrac{\dfrac{\dfrac{\omega}{\omega\ ;\ (\omega^{\perp}\ ,\ \omega)}\,-;axiom\downarrow}{(\omega\ ;\ \omega^{\perp})\ ,\ \omega}\,switch}{\omega}\,cut\uparrow,-\quad\longrightarrow\quad\dfrac{\omega}{\omega}\,id$$

But generalisation seems difficult and it would require introduction of additional core structural rules, would we ever want to perform the reduction through local steps. Specifically, the following pairs (and their symmetric variants) cannot be commuted directly:

$$\dfrac{\dfrac{\omega \;;\; (\omega^{\perp} \;,\; \tau)}{(\omega \;;\; \omega^{\perp}) \;,\; \tau}\, switch}{\tau}\, cut\uparrow, - \qquad\qquad \dfrac{\dfrac{\Box\omega \;;\; \Box\omega^{\perp}}{\Box(\omega \;;\; \omega^{\perp})}\, merge}{\circ}\, \Box cut\uparrow$$

Daimons. For technical purposes, we introduce a new pair of basic deduction rules used to wrap up computations, together with an associated computation residual. Those may be understood computationally as an input/output mechanism.

$$\dfrac{\star}{\tau}\, daimon\downarrow \qquad\qquad \dfrac{\sigma}{\circ}\, daimon\uparrow \qquad\qquad \dfrac{\star}{\circ}\, done$$

They obviously break the consistency of the logic and are therefore not expected to be found in any acceptable proof. They are only used as a tool to prove normalisation of our system. Reduction-wise, daimons "collect" interacting rules as follows:

$$\dfrac{\dfrac{\sigma}{\tau}\,\check{\rho}}{\circ}\, daimon\uparrow \longrightarrow \dfrac{\sigma}{\circ}\, daimon\uparrow \qquad\qquad \dfrac{\dfrac{\star}{\omega}\, daimon\downarrow}{\circ}\, daimon\uparrow \longrightarrow \dfrac{\star}{\circ}\, done$$

Ultimately, given any derivation $\pi : \sigma \to \tau$, if the reduction process terminates, $daimon\downarrow \cdot \pi \cdot daimon\uparrow$ is expected to reduce to *done*, a normal form which marks the completion of the evaluation.

Strong confluence. Notice that in this system, redexes for administrative reductions may overlap with one another (for example, a single *duplication* or a *daimon* rule may have distinct interactions with each of several rules that commute), however, just like in the interaction-net formalism, peaks can be joined immediately and this system therefore enjoys the diamond property as well. In such a strong confluence setting, existence of a reduction path from a given derivation to a normal form (weak normalisation property) is equivalent to all rewriting paths from this derivation leading to a normal form (strong normalisation property). The normal form is moreover unique.

▶ **Lemma 4.1.** *Assuming $\pi : \star \to \omega$ admits π' as normal form:*

- $\Box\pi \cdot duplicate\uparrow$ *reduces to normal form* $\Box\pi' \;;\; \Box\pi'$
- $\Box\pi \cdot erase\uparrow$ *reduces to normal form* $id_{\star}$
- $\Box\pi \cdot open\uparrow$ *reduces to normal form* π'
- $\Box\pi \cdot nest\uparrow$ *reduces to normal form* $\Box\Box\pi'$

Proof. Inner reduction of π is followed by successive reduction steps on π' which, as a normal form, can be written $\bar{\pi} \cdot \check{\pi}$, where $\bar{\pi} : \star \to \sigma$ contains core structural rules only and $\check{\pi} : \sigma \to \omega$ contains introduction rules only. Introduction rules are handled inductively. The final reduction against $\bar{\pi}$ is an instance of an administrative reduction step applied to a block of core structural rules without hypothesis formulas. ◀

5 An Extensible Normalisation Proof via the Reducibility Technique

Let $\mathcal{O}$ be the set of normalisable derivations. A set of derivations from $\star$ to ω is called an *ω-initialiser set* and a set of derivations from ω to $\circ$ is called an *ω-finaliser set*. The *orthogonal* of a given ω-initialiser set $\mathcal{I}$ is the ω-finaliser set defined as follows:

$$\mathcal{I}^{\vee} \;:=\; \{\, \psi : \omega \to \circ \mid \forall \pi \in \mathcal{I},\, \pi \cdot \psi \in \mathcal{O} \,\}$$

Symmetrically, to any *ω-finaliser set* $\mathcal{F}$, we associate an orthogonal *ω-initialiser set* $\mathcal{F}^{\wedge}$:

$$\mathcal{F}^{\wedge} \;:=\; \{\, \phi : \star \to \omega \mid \forall \pi \in \mathcal{F},\, \phi \cdot \pi \in \mathcal{O} \,\}$$

These definitions come with the following properties:

$$\mathcal{I}^{\vee\wedge\vee} \;=\; \mathcal{I}^{\vee} \qquad\qquad \mathcal{F}^{\wedge\vee\wedge} \;=\; \mathcal{F}^{\wedge}$$
$$\mathcal{I} \;\subseteq\; \mathcal{I}^{\vee\wedge} \qquad\qquad \mathcal{F} \;\subseteq\; \mathcal{F}^{\wedge\vee}$$

By definition, an *ω-initialiser behaviour* is an ω-initialiser set $\mathcal{I}$ such that $\mathcal{I} = \mathcal{I}^{\vee\wedge}$, and an *$\omega$-finaliser behaviour* is an ω-finaliser set $\mathcal{F}$ such that $\mathcal{F} = \mathcal{F}^{\wedge\vee}$. *Initialiser candidates* $\lfloor \omega \rfloor$ and *finaliser candidates* $\lceil \omega \rceil$ are respectively ω-initialiser and ω-finaliser behaviours which are defined inductively, altogether, over ω. Hereafter, we provide inductive definition bodies for initialiser candidates in the case of top-level positive connectives.

- *Structural layers*

$$\lfloor \star \rfloor \;:=\; \{\, \mathit{daimon}{\downarrow} \,\}^{\vee\wedge} \qquad \lfloor \sigma_1 \,;\, \sigma_2 \rfloor \;:=\; \{\, \pi_1 \,;\, \pi_2 \mid \pi_1 \in \lfloor \sigma_1 \rfloor, \pi_2 \in \lfloor \sigma_2 \rfloor \,\}^{\vee\wedge}$$
$$\lfloor \Box\sigma \rfloor \;:=\; \{\, \Box\pi \mid \pi \in \lfloor \sigma \rfloor \,\}^{\vee\wedge}$$

- *Computational layers*

$$\lfloor 1 \rfloor \;:=\; \{\, \pi \cdot \mathit{one}{\downarrow} \mid \pi \in \lfloor \star \rfloor \,\}^{\vee\wedge} \qquad \lfloor A \otimes B \rfloor \;:=\; \{\, \pi \cdot \mathit{tensor}{\downarrow} \mid \pi \in \lfloor A \,;\, B \rfloor \,\}^{\vee\wedge}$$
$$\lfloor !A \rfloor \;:=\; \{\, \pi \cdot \mathit{of\text{-}course}{\downarrow} \mid \pi \in \lfloor A \rfloor \,\}^{\vee\wedge}$$

- *Base types*

$$\lfloor \alpha \rfloor \;:=\; \{\, \mathit{daimon}{\downarrow} \,\}^{\vee\wedge}$$

Finaliser candidates definitions for negative connectives (and base types) are handled symmetrically. Initialiser candidates for negative connectives and finaliser candidates for positive connectives are then defined as orthogonals to the former, so that $\lfloor \omega \rfloor = \lceil \omega \rceil^{\wedge}$ and $\lceil \omega \rceil = \lfloor \omega \rfloor^{\vee}$ globally holds. This inductive definition is well founded.

▶ **Definition 5.1.** A derivation $\pi : \sigma \to \tau$ is said to be *reducible* when:

$$\forall \phi \in \lfloor \sigma \rfloor,\, \forall \psi \in \lceil \tau \rceil,\, \phi \cdot \pi \cdot \psi \in \mathcal{O}$$

Notice that equivalently, $\pi : \sigma \to \tau$ is reducible iff $\forall \phi \in \lfloor \sigma \rfloor$, $\phi \cdot \pi \in \lfloor \tau \rfloor$, or, symmetrically, iff $\forall \psi \in \lceil \tau \rceil$, $\pi \cdot \psi \in \lceil \sigma \rceil$.

▶ **Lemma 5.2.** *For any structure ω, candidates $\lceil \omega \rceil$ and $\lfloor \omega \rfloor$ contain normalisable nets only, and moreover $\mathit{daimon}{\uparrow} \in \lceil \omega \rceil$ and $\mathit{daimon}{\downarrow} \in \lfloor \omega \rfloor$.*

Proof. Valid at base types, this combination of statements propagates to all structures by induction. ◀

▶ **Lemma 5.3.** *Every derivation is reducible.*

Proof. By induction on the derivation; we address all derivation constructions separately. The definition of reducibility being symmetric, the number of cases to consider is reduced.

Identity. Given $\phi \in \lfloor \sigma \rfloor$ and $\psi \in \lceil \sigma \rceil$, the derivation $\phi \cdot id_\sigma \cdot \psi = \phi \cdot \psi$ normalises as a composition of two derivations which belong to dual candidates.

Composition. Given two reducible derivations $\pi_1 : \sigma \to \omega$ and $\pi_2 : \omega \to \tau$, using properties $\forall \phi \in \lfloor \sigma \rfloor, \phi \cdot \pi_1 \in \lfloor \omega \rfloor$ and $\forall \psi \in \lceil \tau \rceil, \pi_2 \cdot \psi \in \lceil \omega \rceil$, we obtain that any derivation $\phi \cdot \pi_1 \cdot \pi_2 \cdot \psi$ such that $\phi \in \lfloor \sigma \rfloor$ and $\psi \in \lceil \tau \rceil$ normalises. We have thus shown reducibility of $\pi_1 \cdot \pi_2$.

Contexts

- *Conjunctions.* Given reducible derivations $\pi_1 : \sigma_1 \to \tau_1$ and $\pi_2 : \sigma_2 \to \tau_2$, we show that $\pi_1 \,;\, \pi_2 : \sigma_1 \,;\, \sigma_2 \to \tau_1 \,;\, \tau_2$ is reducible, written as follows: for all $\psi \in \lceil \tau_1 \,;\, \tau_2 \rceil$ we have $(\pi_1 \,;\, \pi_2) \cdot \psi \in \lceil \sigma_1 \,;\, \sigma_2 \rceil$. Given that $\lceil \sigma_1 \,;\, \sigma_2 \rceil = \{\, \pi_1 \,;\, \pi_2 \mid \pi_1 \in \lfloor \sigma_1 \rfloor, \pi_2 \in \lfloor \sigma_2 \rfloor \,\}^\vee$, according to orthogonality's definition, this is equivalent to showing that for all $\phi_1 \in \lfloor \sigma_1 \rfloor$ and $\phi_2 \in \lfloor \sigma_2 \rfloor$ we have $(\phi_1 \,;\, \phi_2) \cdot (\pi_1 \,;\, \pi_2) = (\phi_1 \cdot \pi_1) \,;\, (\phi_2 \cdot \pi_2) \in \lfloor \tau_1 \,;\, \tau_2 \rfloor$. This holds by definition since by reducibility of π_1 and π_2 we have $\phi_1 \cdot \pi_1 \in \lfloor \tau_1 \rfloor$ and $\phi_2 \cdot \pi_2 \in \lfloor \tau_2 \rfloor$.
- *Box.* Given a reducible derivation $\pi : \sigma \to \tau$, we show that $\Box\pi : \Box\sigma \to \Box\tau$ is reducible, written as follows: for all $\psi \in \lceil \Box\tau \rceil$ we have $\Box\pi \cdot \psi \in \lceil \Box\sigma \rceil$. Given that $\lceil \Box\sigma \rceil = \{\, \Box\pi \mid \pi \in \lfloor \sigma \rfloor \,\}^\vee$, this is equivalent to showing that for all $\phi_0 \in \lfloor \sigma \rfloor$, we have $\Box\phi_0 \cdot \Box\pi = \Box(\phi_0 \cdot \pi) \in \lfloor \Box\tau \rfloor$. This holds by definition since $\phi_0 \cdot \pi \in \lfloor \tau \rfloor$, by reducibility of π.

Core structural rules

- *Axiom.* We show for all $\phi \in \lfloor \star \rfloor$ that $\phi \cdot axiom \in \lfloor \sigma^\perp \,,\, \sigma \rfloor$, or equivalently that derivation $axiom \cdot (\psi_1^\perp \,,\, \psi_2)$ normalises for all $\psi_1 \in \lfloor \sigma \rfloor$ and $\psi_2 \in \lceil \sigma \rceil$. This derivation equivalently writes $\psi_1 \cdot \psi_2$, which normalises.
- *Switch.* We show for all $\phi \in \lfloor \omega \,;\, (\sigma \,,\, \tau) \rfloor$ that $\phi \cdot switch \in \lfloor (\omega \,;\, \sigma) \,,\, \tau \rfloor$, or equivalently that $switch \cdot (\psi_1 \,,\, \psi_2) \in \lceil \omega \,;\, (\sigma \,,\, \tau) \rceil$ for all $\psi_1 \in \lceil \omega \,;\, \sigma \rceil$ and $\psi_2 \in \lceil \tau \rceil$, which in turn is equivalent to showing that $(\phi_1 \,;\, \phi_2) \cdot switch \cdot (\psi_1 \,,\, \psi_2)$ normalises for all $\phi_1 \in \lfloor \omega \rfloor$, $\phi_2 \in \lfloor \sigma \,,\, \tau \rfloor$, $\psi_1 \in \lceil \omega \,;\, \sigma \rceil$ and $\psi_2 \in \lceil \tau \rceil$. Given that $\phi_1 : \star \to \omega$ and $\psi_2 : \tau \to \circ$, this last derivation equivalently writes $\pi_1 \cdot \pi_2$ where $\pi_1 = \phi_2 \cdot (id_\sigma \,,\, \psi_2)$ and $\pi_2 = (\phi_1 \,;\, id_\sigma) \cdot \psi_1$; it normalises because $\pi_1 \in \lfloor \sigma \rfloor$ and $\pi_2 \in \lceil \sigma \rceil$.
- *Merge.* We show for all $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$ that $merge \cdot \psi \in \lceil \Box\sigma \,;\, \Box\tau \rceil$, or equivalently that:
  - $(\phi_1 \,;\, \phi_2) \cdot merge \cdot \psi$ normalises for all $\phi_1 \in \lfloor \Box\sigma \rfloor$, $\phi_2 \in \lfloor \Box\tau \rfloor$, $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$
  - $(\phi_1 \,;\, id_{\Box\tau}) \cdot merge \cdot \psi \in \lceil \Box\tau \rceil$ for all $\phi_1 \in \lfloor \Box\sigma \rfloor$, $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$
  - $(\phi_1 \,;\, \Box\phi_2') \cdot merge \cdot \psi$ normalises for all $\phi_1 \in \lfloor \Box\sigma \rfloor$, $\phi_2' \in \lfloor \tau \rfloor$, $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$
  - $(id_{\Box\sigma} \,;\, \Box\phi_2') \cdot merge \cdot \psi \in \lceil \Box\sigma \rceil$ for all $\phi_2' \in \lfloor \tau \rfloor$, $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$
  - $(\Box\phi_1' \,;\, \Box\phi_2') \cdot merge \cdot \psi$ normalises for all $\phi_1' \in \lfloor \sigma \rfloor$, $\phi_2' \in \lfloor \tau \rfloor$, $\psi \in \lceil \Box(\sigma \,;\, \tau) \rceil$.

  The last derivation writes $\Box(\phi_1' \,;\, \phi_2') \cdot \psi$ and normalises since $\Box(\phi_1' \,;\, \phi_2') \in \lfloor \Box(\sigma \,;\, \tau) \rfloor$.

Administrative rules

- *Duplicate.* We show for all $\psi \in \lceil \Box\sigma \,;\, \Box\sigma \rceil$ that $duplicate{\uparrow} \cdot \psi \in \lceil \Box\sigma \rceil$, or equivalently that $\Box\phi_0 \cdot duplicate{\uparrow} \cdot \psi$ normalises for all $\phi_0 \in \lfloor \sigma \rfloor$. By Lemma 4.1, this derivation reduces to $(\Box\phi_0 \,;\, \Box\phi_0) \cdot \psi$, which normalises since $\Box\phi_0 \,;\, \Box\phi_0 \in \lfloor \Box\sigma \,;\, \Box\sigma \rfloor$.
- *Erase, Open, Nest.* Similar to the *Duplicate* case.

Computational rules

- *Tensor.* We obtain $\phi \cdot tensor{\downarrow} \in \lfloor A \otimes B \rfloor$ for all $\phi \in \lfloor A \; ; \; B \rfloor$ directly from the candidate definition. This shows reducibility of $tensor{\downarrow}$ derivations.
- *One*, *Of-course.* Similar to the *Tensor* case.
- *Par.* Reducibility of $par{\downarrow}$ holds iff for all $\phi \in \lfloor A \; , \; B \rfloor$, we have $\phi \cdot par{\downarrow} \in \lfloor A \parr B \rfloor$. Unfolding the candidate definition, it suffices to show that $\phi \cdot par{\downarrow} \cdot tensor{\uparrow} \cdot \psi$ normalises for all $\psi \in \lceil A \; , \; B \rceil$. This holds because reduced derivations $\phi \cdot \psi$ normalise.
- *Bottom*, *Why-not.* Similar to the *Par* case. ◀

▶ **Theorem 5.4.** *All derivations normalise.*

Proof. By the previous lemma, any derivation $\pi : \sigma \to \tau$ is reducible, hence $daimon{\downarrow} \cdot \pi \cdot daimon{\uparrow}$ normalises and so does π. ◀

Discussion about modularity and possible extensions. In the present development, we handled base types as purely abstract types which cannot be introduced or eliminated, these could only be passed around. In practice, we may want to use base types to represent primitive types which come together with associated primitive operations. Our developments could be extended to take those into account by adding related data constructions to respective candidate definitions. Normalisation remains of course ultimately dependent on primitives' behaviour, since associated inductive cases appear in the proof. Base types can also implement type variables used in second-order quantifiers (the reader is referred to the original proof by Girard [4] as for how to handle these).

6 Conclusion

In this paper we have introduced a Curry–Howard correspondence between well-formed interaction nets and a deep-inference deduction system based on multiplicative-exponential linear logic. Linear logic itself can be expressed in the system and its computational aspects materialise though the correspondence as a system of sharing graphs. The enriched type system for nets that stems from this correspondence not only sheds additional light upon the structure of multiplicatives [2], but moreover encompasses the exponential layer and could easily be extended further.

Our approach fuses the essential properties of strong confluence (interaction nets) and weak normalisation (via Girard's reducibility method) to obtain a consice, modular and extensible proof of strong normalisation. However, it currently relies on a somewhat unorthodox notion of normal form, which does not consider interactions between identity rules. As the sought computational interpretation is unaffected this seems to be negligible, but we will study this peculiarity in more detail in future work.

Furthermore, we are interested in extending the method to stronger logics: additive, second-order, inductive constructions, etc. Extensions towards differential linear logic, which was introduced by Ehrhard and Regnier and features further symmetry [3], will also be investigated in these studies.

References

1 Kai Brünnler. Deep inference and its normal form of derivations. In *CiE*, volume 3988 of *LNCS*, pages 65–74, 2006.

2 Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Arch. Math. Logic*, 28:181–203, 1989.

3 Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theor. Comput. Sci.*, 364(2):166–195, 2006.

4 Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

5 G. Gonthier, M. Abadi, and J.-J. Lévy. The geometry of optimal lambda reduction. In *Proc. 19th POPL*, pages 15–26. ACM Press, 1992.

6 Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. Linear logic without boxes. In *Proc. 7th LICS*, pages 223–34, 1992.

7 S. Guerrini, S. Martini, and A. Masini. Coherence for sharing proof-nets. *Theor. Comput. Sci.*, 294(3):379–409, 2003.

8 A. Guglielmi. A system of interaction and structure. *ACM Trans. Comput. Log.*, 8(1), 2007.

9 A. Guglielmi and L. Straßburger. Non-commutativity and MELL in the calculus of structures. In *Proc. 10th CSL*, volume 2142 of *LNCS*, pages 54–68, 2001.

10 T. Gundersen, W. Heijltjes, and M. Parigo. Atomic lambda calculus – a typed lambda calculus with explicit sharing. In *Proc. 28th LICS*, 2013. To appear.

11 Yves Lafont. Interaction nets. *Proc. 17th POPL*, pages 95–108, 1990.

12 J. Lamping. An algorithm for optimal lambda calculus reduction. In *Proc. 7th POPL*, pages 16–30. ACM Press, 1990.

13 I. Mackie. Interaction nets for linear logic. *Theor. Comput. Sci.*, 247(1-2):83–140, 2000.

14 Michele Pagani and Lorenzo Tortora de Falco. Strong normalization property for second order linear logic. *Theor. Comput. Sci.*, 411(2):410–444, 2010.

15 Lutz Straßburger and Alessio Guglielmi. A system of interaction and structure IV: The exponentials and decomposition. *ACM Trans. Comput. Log.*, 12(4):23, 2011.

16 Lutz Straßburger. MELL in the calculus of structures. *Theor. Comput. Sci.*, 309:213–285, 2003.

17 A. Tiu. A local system for intuitionistic logic. In *Proc. 13th LPAR*, volume 4246 of *LNCS*, pages 242–256, 2006.

18 C.P. Wadsworth. *Semantics and Pragmatics of the Lambda-Calculus*. PhD thesis, University of Oxford, 1971.

The Fixed-Parameter Tractability of Model Checking Concurrent Systems *

Stefan Göller

LIAFA/CNRS, Paris 7, France
Universität Bremen, Fachbereich Mathematik und Informatik, Germany
goeller@informatik.uni-bremen.de

Abstract

We study the fixed-parameter complexity of model checking temporal logics on concurrent systems that are modeled as the product of finite systems and where the size of the formula is the parameter. We distinguish between asynchronous product and synchronous product. Sometimes it is possible to show that there is an algorithm for this with running time $(\sum_i |\mathcal{T}_i|)^{O(1)} \cdot f(|\varphi|)$, where the $\mathcal{T}_i$ are the component systems and φ is the formula and f is computable function, thus model checking is fixed-parameter tractable when the size of the formula is the parameter.

In this paper we concern ourselves with the question, provided fixed-parameter tractability is known, whether it holds for an elementary function f. Negative answers to this question are provided for modal logic and EF logic: Depending on the mode of synchronization we show the non-existence of such an elementary function f under different assumptions from (parameterized) complexity theory.

1998 ACM Subject Classification F.4.1 Mathematical Logic, F.2.0 Analysis of Algorithms and Problem Complexity: General

Keywords and phrases Model Checking, Concurrent Systems, Parameterized Complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.332

1 Introduction

Model checking is one of the most successful approaches in formal verification; it asks to verify if a given specification is satisfied in a given system [5, 2]. The computational complexity of model checking finite systems is very well understood (beginning with the pioneer work [15]), but model checking the modal μ-calculus is an exception: The best-known upper bound in $\mathsf{UP} \cap \mathsf{co\text{-}UP}$ and the lower bound is P. In general it turns out that typically the source of hardness of model checking lies in the size of the formula and not in the size of the input system. Indeed, the complexity of model checking the modal μ-calculus lies in P for every fixed formula. Another such famous example is model checking of linear temporal logic LTL that is generally PSPACE-complete but which can be solved in time $|\mathcal{T}|^{O(1)} \cdot 2^{|\varphi|}$, where $\mathcal{T}$ is the (transition) system and φ is the formula [12]. Measuring the complexity only in the size of the system and not in the size of the specification is indeed justified since typically the specification is small and the system is big.

Yet, model checking tools have to deal with a combinatorial blowup of the state space of the input system, commonly known as the *state explosion problem*, that can be seen as one of the biggest challenges in real-world problems. Different sources of explosion arise, for

* The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 259454.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 332–347

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

instance the number of program variables or clocks, the number of concurrently running components, or the number of different subroutines, just to mention few of them. Technically speaking, this can be seen that the input consists of systems $\mathcal{T}_1, \ldots, \mathcal{T}_d$ (the *components*), but the actual system of interest is the product of these systems; independent on the underlying synchronization mechanism, we refer to such systems as *concurrent systems* in the following.

Parameterized complexity theory allows for a fine-grained complexity investigation of problems where certain sizes of the input are declared as a parameter. A central class in this theory is the class FPT consisting of all problems that can be solved in time $n^{O(1)} \cdot f(k)$ for some computable function f, where n is the input size and k is the parameter. For instance, the above-mentioned model checking problem of LTL is in FPT when the size of the formula is the parameter. Although their exact definitions are not relevant here, we list the parameterized complexity classes that are subject of this paper

$$\mathsf{FPT} \subseteq \mathsf{AW}[*] \subseteq \mathsf{AW}[\mathsf{SAT}],$$

where none of the inclusions is known but believed to be strict. The class AW[$*$] can be seen as fixed parameter tractability plus parameter bounded alternating nondeterminism [4] and AW[SAT] is the set of all problems that are (FPT)-reducible to a variant of QBF, where quantification is restricted to a partition of the variables and in which the number of variables that are assigned to true in each quantifier block plus the number of partitions of the variables is the parameter. We refer to [8] for more details on parameterized complexity theory.

The parameterized complexity of model checking synchronous concurrent systems has intensively been studied by Demri, Laroussinie and Schnoebelen [7]. Among their results it is shown that already very simple questions like reachability or model checking modal logic are not fixed-parameter tractable when the number of components and the size of the formula is the parameter, respectively. More precisely in [7] it is shown that (i) already when the number of components d is the parameter it is AWT[SAT]-hard to decide reachability of a synchronous product of d transition systems, already when transitions can only synchronize over a binary alphabet and (ii) when the size of the formula is the parameter, model checking the synchronous product of systems with respect to modal logic is AW[$*$]-complete. Since the above-mentioned parameterized complexity classes are believed to be strict, we have that (i) and (ii) show that already these two basic model checking problems are *not* fixed-parameter tractable unless FPT coincides with them.

It is important to mention that in [7] the technical reason for AW[$*$]-hardness for model checking modal logic on the synchronous product lies in the fact the transitions can synchronize over a transition alphabet that is independent of the size of the formula. Thus, the question arises whether fixed-parameter tractability can be gained when the transition alphabet is bounded by the size of the formula (which is natural if one considers multi-modal logic, for instance). Too, the question arises whether fixed-parameter tractability can be gained when we restrict the synchronization mechanism to be the *asynchronous* product.

Unfortunately still, in both cases under the assumption $\mathsf{FPT} \neq \mathsf{AWT}[\mathsf{SAT}]$, already for powerful branching-time logics like CTL, the answer to both questions is negative since reachability of synchronous product [7] can easily be encoded. However, the compositional method à la Feferman and Vaught, which has recently been developed for the fragments modal logic and EF logic by Rabinovich [14], gives positive answers to the above questions: In certain cases, one can show that for model checking the product of given systems $\mathcal{T}_1, \ldots, \mathcal{T}_d$ against a given formula φ there is an algorithm with running time $(\sum_i |\mathcal{T}_i|)^{O(1)} \cdot f(|\varphi|)$ for a primitive recursive function f, thus being fixed-parameter tractable. Unfortunately, the

compositional method yields an algorithm whose running time is bounded by a nonelementary function f; moreover this is not avoidable as recently proven [11, 10].

In this paper we concern ourselves with the question, provided fixed-parameter tractability for model checking concurrent systems is known, whether one can hope for *any* algorithm witnessing fixed-parameter tractability with an elementarily growing function f. The main results of this paper answer this question negatively under different assumptions from (parameterized) complexity theory depending on the logic and the synchronization mode under consideration.

Our contribution. We revisit briefly that the compositional method allows to model check given systems $\mathcal{T}_1, \ldots, \mathcal{T}_d$ and a formula φ in time $(\sum_i |\mathcal{T}_i|)^{O(1)} \cdot f(|\varphi|)$ in case either (i) φ is a formula of modal logic and the asynchronous product of the $\mathcal{T}_1, \ldots, \mathcal{T}_d$ is considered, (ii) φ is a formula of EF logic and the asynchronous product of the $\mathcal{T}_1, \ldots, \mathcal{T}_d$ is considered, or (iii) φ is a formula of modal logic and the synchronous product of the $\mathcal{T}_1, \ldots, \mathcal{T}_d$ is considered. For (i) we show that there is no algorithm for this that runs in time $(\sum_i |\mathcal{T}_i|)^{O(1)} \cdot f(|\varphi|)$ for any elementary function f unless $\mathsf{FPT} = \mathsf{AW}[*]$. For (ii) and (iii) we prove that there is no algorithm for this that runs in time $(\sum_i |\mathcal{T}_i|)^{O(1)} \cdot f(|\varphi|)$ for any elementary function f unless $\mathsf{P} = \mathsf{NP}$. We remark that the assumption $\mathsf{FPT} \neq \mathsf{AW}[*]$ is a stronger assumption than $\mathsf{P} \neq \mathsf{NP}$. The overall picture of the fixed-parameter tractability of model checking modal logic and EF logic on concurrent systems is summarized in Table 1.

Related work. The parameterized complexity of various problems in formal verification has been investigated in [7, 13] and we refer to the reference therein and to [8] for more information on parameterized complexity. The parameterized complexity of model checking first-order logic and monadic second-order logic over words has been studied in [9] and in fact we give a reduction from model checking first-order logic over words to model checking modal logic on the asynchronous product of systems in this paper. The parameterized complexity of satisfiability of modal logic under various parameters of the input formulas has been investigated in [1].

Organization of this paper. Preliminaries, the compositional method and upper bounds are content of Section 2. Section 3 provides technical tools that allow us to compare trees that encode large numbers with small formulas. In Section 4 we discuss the proof strategies of our main results. In Section 5 we show that fixed-parameter tractability is not possible with an elementary running time in the size of the formula for model checking modal logic on the asynchronous product unless $\mathsf{FPT} = \mathsf{AW}[*]$. We show in Section 6 that for model checking modal logic on the synchronous product and for model checking EF on the asynchronous product fixed-parameter tractability is not possible with an elementary running time in the size of the formula unless $\mathsf{P} = \mathsf{NP}$. We conclude in Section 7. Missing proofs due to space restrictions can be found in the appendix.

2 Preliminaries

The exact definitions of the parameterized complexity classes that appear in this paper are not important, we refer the reader to [8] for more details. For two integers i and j, we define the interval $[i, j] \stackrel{\text{def}}{=} \{i, i+1, \ldots, j\}$. By $\mathbb{N} \stackrel{\text{def}}{=} \{0, 1, 2, \ldots, \}$ we denote the set of *non-negative integers*. The *tower function* $\mathsf{Tower} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is defined as $\mathsf{Tower}(0, n) \stackrel{\text{def}}{=} n$ and $\mathsf{Tower}(\ell + 1, n) \stackrel{\text{def}}{=} 2^{\mathsf{Tower}(\ell, n)}$ for each $\ell \in \mathbb{N}$ and each $n \in \mathbb{N}$. We also introduce the tower function in one parameter as $\mathsf{Tower}(\ell) \stackrel{\text{def}}{=} \mathsf{Tower}(\ell, 2)$ for each $\ell \in \mathbb{N}$. Define the

Table 1 The fixed-parameter tractability of model checking the product of finite systems $\mathcal{T}_1, \ldots, \mathcal{T}_d$ against a formula φ of modal logic or of EF logic, where $|\varphi|$ is the parameter.

		Asynchronous product ($\bigotimes$)	**Synchronous Product ($\prod$)**
ML	**upper**	$(\sum_{i\in[1,d]} \lvert\mathcal{T}_i\rvert)^{O(1)} \cdot f(\lvert\varphi\rvert)$ for some primitive recursive f by Theorem 2 ([14])	$(\sum_{i\in[1,d]} \lvert\mathcal{T}_i\rvert)^{O(1)} \cdot f(\lvert\varphi\rvert)$ for some primitive recursive f by Theorem 2 ([14])
	lower	**not in** $(\sum_{i\in[1,d]} \lvert\mathcal{T}_i\rvert)^{O(1)} \cdot f(\lvert\varphi\rvert)$ for any elementary f unless $\mathsf{FPT} = \mathsf{AW}[*]$ by Theorem 8	**not in** $(\sum_{i\in[1,d]} \lvert\mathcal{T}_i\rvert)^{O(1)} \cdot f(\lvert\varphi\rvert)$ for any elementary f unless $\mathsf{P} = \mathsf{NP}$ by Theorem 9
EF	**upper**	$(\sum_{i\in[1,d]} \lvert\mathcal{T}_i\rvert)^{O(1)} \cdot f(\lvert\varphi\rvert)$ for some primitive recursive f by Theorem 2 ([14])	**not in** FPT unless $\mathsf{FPT} = \mathsf{AW}[\mathsf{SAT}]$ by Theorem 3 (Theorem 5.1 in [7])
	lower	**not in** $(\sum_{i\in[1,d]} \lvert\mathcal{T}_i\rvert)^{O(1)} \cdot f(\lvert\varphi\rvert)$ for any elementary f unless $\mathsf{P} = \mathsf{NP}$ by Theorem 10	

inverse function $\log^*$ as $\log^*(n) \stackrel{\text{def}}{=} \min\{\ell \in \mathbb{N} \mid \mathsf{Tower}(\ell) \geq n\}$. Let $f, g : \mathbb{N} \to \mathbb{N}$ be functions. We say f *is bounded by* g if $f(n) \leq g(n)$ for all but finitely many $n \in \mathbb{N}$. A function $f : \mathbb{N} \to \mathbb{N}$ is *elementary* if it can be formed from the successor function, addition, subtraction, and multiplication using compositions, projections, bounded additions and bounded multiplications (of the form $\sum_{z\leq y} g(\overline{x}, z)$ and $\prod_{z\leq y} g(\overline{x}, z)$). For our purposes it will only be important that a function $f : \mathbb{N} \to \mathbb{N}$ is bounded by an elementary function if and only if there is some $\ell \in \mathbb{N}$ such that f is bounded by the function $n \mapsto \mathsf{Tower}(\ell, n)$.

Throughout the paper, let us fix a countable set of *atomic propositions* $\mathbb{P}$ and a countable set of *atomic actions* $\mathbb{A}$. A *signature* is a pair (P, A), where $\mathsf{P} \subseteq \mathbb{P}$ is a finite set of atomic propositions and where $\mathsf{A} \subseteq \mathbb{A}$ is a finite set of atomic actions. A *transition system* is a tuple $\mathcal{T} = (S, \{S_p \mid p \in \mathsf{P}\}, \{\xrightarrow{a} \mid a \in \mathsf{A}\})$, where (P, A) is some signature, S is a set of *states*, $S_p \subseteq S$ is a *valuation* of the atomic propositions for each $p \in \mathsf{P}$, and $\xrightarrow{a} \subseteq S \times S$ is a binary *transition relation* for each $a \in \mathsf{A}$. All transition systems that appear in this paper have finite state sets, thus we denote by $|\mathcal{T}| \stackrel{\text{def}}{=} |S| + |\mathsf{P}| + |\mathsf{A}|$ the *size* of $\mathcal{T}$. A *pointed transition system* is a pair $(s, \mathcal{T})$, where $\mathcal{T}$ is a transition system and s is a state of $\mathcal{T}$ that we also denote by the *point* of $\mathcal{T}$. We sometimes write $\mathcal{T}$ to denote $(s, \mathcal{T})$ whenever s has been fixed from the context.

Formulas φ of the logic EF are given by the following grammar, where $p \in \mathbb{P}$ and $a \in \mathbb{A}$:

$$\varphi \quad ::= \quad p \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \varphi \quad | \quad \langle a \rangle \varphi \quad | \quad \mathsf{EF}\varphi$$

The *size* $|\varphi|$ of a formula φ is inductively defined as $|p| \stackrel{\text{def}}{=} 1$ for each $p \in \mathbb{P}$, $|\neg\varphi| \stackrel{\text{def}}{=} |\mathsf{EF}\varphi| \stackrel{\text{def}}{=} |\langle a \rangle \varphi| \stackrel{\text{def}}{=} |\varphi| + 1$, and $|\varphi_1 \wedge \varphi_2| \stackrel{\text{def}}{=} |\varphi_1| + |\varphi_2| + 1$. We introduce the usual abbreviations $\varphi_1 \vee \varphi_2 \stackrel{\text{def}}{=} \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \to \varphi_2 \stackrel{\text{def}}{=} \neg\varphi_1 \vee \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2 \stackrel{\text{def}}{=} (\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$, $[a]\varphi \stackrel{\text{def}}{=} \neg\langle a \rangle \neg\varphi$ for each $a \in \mathsf{A}$ and finally $\mathsf{AG}\varphi \stackrel{\text{def}}{=} \neg\mathsf{EF}\neg\varphi$. *(Multi-) Modal logic (*ML*)* is the fragment of EF, where the EF-operator does not occur. Given a signature (P, A) and an EF-formula φ, we say that φ is *defined over* (P, A) if P (resp. A) contains the set of atomic propositions (resp. atomic actions) that appear in φ.

Given a transition system $\mathcal{T} = (S, \{S_p \mid p \in \mathsf{P}\}, \{\xrightarrow{a} \mid a \in \mathsf{A}\})$, a state $s \in S$, and an EF-formula φ over (P, A), we define $(s, \mathcal{T}) \models \varphi$ by structural induction on the formula φ as follows (1) $(s, \mathcal{T}) \models p$ if and only if $s \in S_p$, (2) $(s, \mathcal{T}) \models \neg\varphi$ if and only if $(s, \mathcal{T}) \not\models \varphi$, (3) $(s, \mathcal{T}) \models \varphi_1 \wedge \varphi_2$ if and only if $(s, \mathcal{T}) \models \varphi_1$ and $(s, \mathcal{T}) \models \varphi_2$, (4) $(s, \mathcal{T}) \models \langle a \rangle \varphi$ if and

only if $(s', \mathcal{T}) \models \varphi$ for some $s' \in S$ with $s \xrightarrow{a} s'$ and finally (5) $(s, \mathcal{T}) \models \mathsf{EF}\varphi$ if and only if $(s', \mathcal{T}) \models \varphi$ for some $s' \in S$ with $s \to^* s'$, where $\to = \bigcup_{a \in \mathsf{A}} \xrightarrow{a}$.

Products and the compositional method. Let $d \geq 1$ and let us assume a transition system $\mathcal{T}_i = (S_i, \{S_{p,i} \mid p \in \mathsf{P}_i\}, \{\xrightarrow{a}_i \mid a \in \mathsf{A}_i\})$ over the signature $(\mathsf{P}_i, \mathsf{A}_i)$ for each $i \in [1, d]$, where P_i and P_j (resp. A_i and A_j) could possibly have non-empty intersection for different $i, j \in [1, d]$. Assume P to be the union of all the P_i and assume A to be the union of all the A_i.

We define their *asynchronous product* as $\bigotimes_{i \in [1,d]} \mathcal{T}_i \stackrel{\text{def}}{=} (\overline{S}, \{\overline{S}_p \mid p \in \mathsf{P}\}, \{\xrightarrow{a} \mid a \in \mathsf{A}\})$, where $\overline{S} \stackrel{\text{def}}{=} \prod_{i \in [1,d]} S_i$, for each $(s_1, \ldots, s_d) \in \overline{S}$ and each $p \in \mathsf{P}$ we have $(s_1, \ldots, s_d) \in \overline{S}_p$ if and only if $s_i \in S_{p,i}$ for some $i \in [1, d]$, for each $(s_1, \ldots, s_d)$, $(s'_1, \ldots, s'_d) \in \overline{S}$ and each $a \in \mathsf{A}$ we have $(s_1, \ldots, s_d) \xrightarrow{a} (s'_1, \ldots, s'_d)$ if and only if there is some $i \in [1, d]$ such that $s_i \xrightarrow{a}_i s'_i$ in $\mathcal{T}_i$ and $s_j = s'_j$ for each $j \in [1, d]$ with $j \neq i$.

We define their *synchronous product*[1] as $\prod_{i \in [1,d]} \mathcal{T}_i \stackrel{\text{def}}{=} (\overline{S}, \{\overline{S}_p \mid p \in \mathsf{P}\}, \{\xrightarrow{a} \mid a \in \mathsf{A}\})$, where $\overline{S}$ and the $\overline{S}_p$ are defined as for the asynchronous product, but where the transition relation is defined as follows: For each $(s_1, \ldots, s_d), (s'_1, \ldots, s'_d) \in \overline{S}$ and each $a \in \mathsf{A}$ we have $(s_1, \ldots, s_d) \xrightarrow{a} (s'_1, \ldots, s'_d)$ if and only if for all $i \in [1, d]$ we have $s_i \xrightarrow{a}_i s'_i$ in $\mathcal{T}_i$.

Let us state the compositional method for ML and EF as proven in [14]. Here, we state it for EF logic and the asynchronous product. Also note that in [14] it has been proven for more general interpretations of the atomic propositions and more general products.

▶ **Theorem 1** ([14],Theorem 21). *The following is primitive recursive:*

INPUT: An EF *formula* φ *over* (P, A)*, where* $\mathsf{P} = \bigcup_{i=1}^{d} \mathsf{P}_i$ *and* $\mathsf{A} = \bigcup_{i=1}^{d} \mathsf{A}_i$ *for a* $d \geq 1$*.*

OUTPUT: A tuple $(\Psi_1, \ldots, \Psi_d, \beta)$*, where* $\Psi_i = \{\psi_i^j \mid j \in J_i\}$ *is a finite set of* EF *formulas over* $(\mathsf{P}_i, \mathsf{A}_i)$*, and a boolean formula* β *with variables from* $X \stackrel{def}{=} \{x_i^j \mid i \in [1, d], j \in J_i\}$ *such that for every transition system* $\mathcal{T}_i = (S_i, \{S_{p,i} \mid p \in \mathsf{P}_i\}, \{\xrightarrow{a}_i \mid a \in \mathsf{A}_i\})$ *over* $(\mathsf{P}_i, \mathsf{A}_i)$ *and every state* s_i *of* $\mathcal{T}_i$ *(*$i \in [1, d]$*) it holds:*

$$\left(\langle s_1, \ldots, s_d \rangle, \bigotimes_{i=1}^{d} \mathcal{T}_i\right) \models \varphi \qquad \Longleftrightarrow \qquad \mu \models \beta$$

Here, $\mu : X \to \{0, 1\}$ *is defined by* $\mu(x_i^j) = 1$ *if and only if* $(s_i, \mathcal{T}_i) \models \psi_i^j$*. Moreover, we have* $|\mathcal{D}(\varphi, d)| \leq g(d + |\varphi|)$ *for some primitive recursive function* g*, where* $\mathcal{D}(\varphi, d) \stackrel{def}{=} (\Psi_1, \ldots, \Psi_d, \beta)$ *is the* decomposition *of* φ *and its* size *is* $|\mathcal{D}(\varphi, d)| \stackrel{def}{=} |\beta| + \sum_{i \in [1,d]} \sum_{j \in J_i} |\psi_i^j|$*.*

▶ Remark. In [14] an analog of Theorem 1 has also been shown for the following cases: (i) φ and each formula in $\bigcup_{i \in [1,d]} \Psi_i$ is an ML formula, or (ii) φ and each formula in $\bigcup_{i \in [1,d]} \Psi_i$ is an ML formula and the asynchronous product $\bigotimes$ is replaced by the synchronous product $\prod$. Moreover, in [14] it is shown that for EF and the synchronous product such desirable decompositions as stated in Theorem 1 do *not* exist in general (even when the computability requirement is dropped).

Since model checking of EF and ML is decidable in polynomial time, Theorem 1 (whenever applicable) delivers a running time for model checking the product of finite systems in time

$$\Big(\sum_{i \in [1,d]} |\mathcal{T}_i|\Big)^{O(1)} \cdot f(d + |\varphi|)$$

[1] also known as *strong synchronization*, e.g. [7]

for a primitive recursive function f and is thus fixed-parameter tractable when $d + |\varphi|$ is the parameter. The following theorem states that it is even fixed-parameter tractable when only $|\varphi|$ is the parameter. Although not explictly stated in [14] (Corollary 22 of [14] requires $d + |\varphi|$ to be the parameter), its proof can be deduced from a refined analysis of Theorem 21 in [14] by using the notions of generalized product that were defined in [14].

▶ **Theorem 2** (A consequence from [14]). *Let* $\mathsf{op} \in \{\bigotimes, \prod\}$ *either stand for the asynchronous or synchronous product. Given a formula* φ *over* (P, A) *(where* $\mathsf{P} = \bigcup_{i=1}^{d} \mathsf{P}_i$ *and* $\mathsf{A} = \bigcup_{i=1}^{d} \mathsf{A}_i$*) and transition systems* $(\mathcal{T}_i)_{i\in[1,d]}$ *and a state* $\overline{s}$ *of* $\mathsf{op}_{i=1}^{d}\mathcal{T}_i$*, one can decide* $\left(\overline{s}, \mathsf{op}_{i=1}^{d}\mathcal{T}_i\right) \models \varphi$ *in time* $\left(\sum_{i\in[1,d]} |\mathcal{T}_i|\right)^{O(1)} \cdot f(|\varphi|)$ *for some primitive recursive function* f *in either of the following cases: (i)* φ *is some* ML *formula and* $\mathsf{op} = \prod$*, or (ii)* φ *is some* ML *formula and* $\mathsf{op} = \bigotimes$*, or (iii)* φ *is some* EF *formula and* $\mathsf{op} = \bigotimes$.

The question of fixed-parameter tractability of model checking EF for synchronous product has already been answered negatively for reachability in [7] (unless FPT = AWT[SAT]) and moreover reflects the non-decomposability of EF for synchronous product as mentioned in the end of Remark 2.

▶ **Theorem 3** (Theorem 5.1 in [7]). *Given transition systems* $\mathcal{T}_1, \ldots, \mathcal{T}_d$ *over a common signature* (A, P) *and two states* $\overline{s}, \overline{t}$ *in* $\prod_{i=1}^{d} \mathcal{T}_i$*, there is no algorithm that decides* $\overline{s} \to^* \overline{t}$ *in* $\prod_{i=1}^{d} \mathcal{T}_i$ *in time* $\left(\sum_{i=1}^{d} |\mathcal{T}_i|\right)^{O(1)} \cdot f(d)$ *for any computable function* f *unless* FPT = AWT[SAT].

3 Encoding huge numbers via sibling-ordered trees

In this section we show how one can represent numbers of size $O(n)$ by sibling-ordered trees of size $O(n)$ of depth $O(\log^*(n))$.

The idea of using wide trees of small height for proving nonelementary lower bounds has already been considered before in the literature: such wide trees, as discussed in [8], have been used in [1, 6, 9] to prove nonelementary lower bounds for parameterized complexity of satisfiability for modal logic, nonelementary lower bounds on the sizes of several normal forms of first-order logic formulas, and for the parameterized complexity of model checking first-order logic. More discussion on the particular choice of our sibling-ordered trees follows in Section 4.

In the following, let $\mathsf{P}_\ell \stackrel{\text{def}}{=} \{b_i \mid i \in [0, \ell+1]\}$ and $\mathsf{A}_\ell \stackrel{\text{def}}{=} \{a_i \mid i \in [0, \ell]\} \cup \{\Rightarrow, \Leftarrow\}$ for each $\ell \in \mathbb{N}$. For each $\ell \in \mathbb{N}$ a *pointed ℓ-sotree (for sibling-ordered tree)* that is either 0-pointed (i.e. the point does *not* satisfy proposition $b_{\ell+1}$) or 1-pointed (i.e. the point satisfies proposition $b_{\ell+1}$) and that has a *value* from $[0, \mathsf{Tower}(\ell) - 1]$ is a pointed transition system over $(\mathsf{P}_\ell, \mathsf{A}_\ell)$ that is defined inductively on ℓ:

- **Base case when $\ell = 0$:** A *pointed 0-sotree* is one of the following four pointed transition systems each with point ● over $(\mathsf{P}_0, \mathsf{A}_0)$:

The 0-pointed 0-sotree of value 0. The 0-pointed 0-sotree of value 1. The 1-pointed 0-sotree of value 0. The 1-pointed 0-sotree of value 1.

- **Inductive step for $\ell+1$:** A *pointed* $(\ell+1)$*-sotree* is a pointed transition system $(r_{\ell+1}, \mathcal{T}_{\ell+1})$ over $(\mathsf{P}_{\ell+1}, \mathsf{A}_{\ell+1})$ that can be obtained as follows:
  (1) The point $r_{\ell+1}$ does not satisfy any of the propositions $b_0, \ldots, b_\ell$.
  (2) The states of $\mathcal{T}_{\ell+1}$ are obtained by the union of $\{r_{\ell+1}\}$ and for each $j \in [0, \mathsf{Tower}(\ell)-1]$ *exactly one* of the possible two ℓ-sotrees of value j (either 0-pointed or 1-pointed), let us denote it by $(r_\ell(j), \mathcal{T}_\ell(j))$.
  (3) Add the a_ℓ-labeled transitions $\{(r_{\ell+1}, r_\ell(j)) \mid j \in [0, \mathsf{Tower}(\ell)-1]\}$ to $\mathcal{T}_{\ell+1}$.
  (4) Add the $\Leftarrow$-labeled transitions $\{(r_\ell(j), r_\ell(j')) \mid j, j' \in [0, \mathsf{Tower}(\ell)-1], j > j'\}$ and the $\Rightarrow$-labeled transitions $\{(r_\ell(j), r_\ell(j')) \mid j, j' \in [0, \mathsf{Tower}(\ell)-1], j < j'\}$ between siblings.
  (5) Define the *value* of $(r_{\ell+1}, \mathcal{T}_{\ell+1})$ as
  $$\mathsf{val}(r_{\ell+1}, \mathcal{T}_{\ell+1}) \stackrel{\text{def}}{=} \sum \left\{2^j \mid j \in [0, \mathsf{Tower}(\ell)-1] : (r_\ell(j), \mathcal{T}_\ell(j)) \text{ is 1-pointed}\right\}.$$
  Note that $\mathsf{val}(r_{\ell+1}, \mathcal{T}_{\ell+1}) \in [0, \mathsf{Tower}(\ell+1)-1]$.
  (6) In case $r_{\ell+1}$ satisfies $b_{\ell+1}$ we say $(r_{\ell+1}, \mathcal{T}_{\ell+1})$ is 1-*pointed*, otherwise we say $(r_{\ell+1}, \mathcal{T}_{\ell+1})$ is 0-*pointed*.

  Again recall that, up to isomorphism, for each $j \in [0, \mathsf{Tower}(\ell+1)-1]$ there are exactly two $\ell+1$-sotrees of value j, one being 0-pointed and one being 1-pointed.

Figure 1 shows an example 2-sotree. Let $\mathsf{size}(\ell)$ denote the *number of nodes of each ℓ-sotree*. Note $\mathsf{size}(\ell)$ can be expressed by the recurrence

$$\mathsf{size}(0) = 2 \qquad \text{and} \qquad \mathsf{size}(\ell+1) = \mathsf{Tower}(\ell) \cdot \mathsf{size}(\ell) + 1.$$

▶ **Lemma 4.** *For each $\ell \geq 3$ we have $\mathsf{size}(\ell+1) \leq \mathsf{Tower}(\ell)^2$.*

Recall that the *value* of each ℓ-sotree lies in the interval $[0, \mathsf{Tower}(\ell)-1]$ and for every value in this interval there is an ℓ-sotree with this value. In the following, for each $j, \ell \in \mathbb{N}$ with $j \in [0, \mathsf{Tower}(\ell)-1]$ we define $\Upsilon_\ell(j) \stackrel{\text{def}}{=} (r_\ell(j), \mathcal{T}_\ell(j))$ to be the (unique) 0-pointed ℓ-sotree of value j.

Next, we aim at defining formulas that allow us to compare the values of ℓ-sotrees with small formulas.

Let $(s, \mathcal{T})$ be a pointed transition system with $\mathcal{T} = (S, \{S_p \mid p \in \mathsf{P}\}, \{\xrightarrow{a} \mid a \in \mathsf{A}\})$ and let α be some label. We define the *asynchronous α-extension of $\mathcal{T}$* to be the pointed transition system with point s over the signature $(\{\alpha\} \times \mathsf{P}, \{\alpha\} \times \mathsf{A})$ that can be obtained from $\mathcal{T}$ by setting $S_{(\alpha,p)} \stackrel{\text{def}}{=} S_p$ and by setting $\xrightarrow{(\alpha,a)} \stackrel{\text{def}}{=} \xrightarrow{a}$. Likewise, we define the *synchronous α-extension of $\mathcal{T}$* to be the pointed transition system with point s over the signature $(\{\alpha\} \times \mathsf{P}, \mathsf{A})$ that can be obtained from $\mathcal{T}$ by setting $S_{(\alpha,p)} \stackrel{\text{def}}{=} S_p$ and keeping $\xrightarrow{a}$ as it is. The *value* val of each asynchronous/synchronous α-extension of a pointed ℓ-sotree $(r, \mathcal{T})$ is inherited from $\mathsf{val}(r, \mathcal{T})$.

The following lemma states that one can construct ML formulas of size $O(\log^*(n))$ that allow us to compare the value of the asynchronous product of an asynchronous α-extension and an asynchronous β-extension of ℓ-sotrees of value $O(n)$. The formulas are inspired from [10], but, as will be discussed in Section 4, we require the additional transitions $\Leftarrow$ and $\Rightarrow$ for expressing order.

▶ **Lemma 5** (Formulas for the asynchronous product of ℓ-sotrees)**.** *For each $\ell \in \mathbb{N}$ and labels α, β there are formulas $(\mathsf{eq}_\ell^\otimes(\alpha,\beta))_{\ell\in\mathbb{N}} = (\mathsf{eq}_\ell^\otimes)_{\ell\in\mathbb{N}}$ and $(\mathsf{less}_\ell^\otimes(\alpha,\beta))_{\ell\in\mathbb{N}} = (\mathsf{less}_\ell^\otimes)_{\ell\in\mathbb{N}}$, each over the signature $(\{\alpha,\beta\} \times \mathsf{P}_\ell, \{\alpha,\beta\} \times \mathsf{A}_\ell)$ and each of size $2^{O(\ell)}$ such that the asynchronous α-extension $(r, \mathcal{T})$ of each pointed ℓ-sotree the asynchronous β-extension $(r', \mathcal{T}')$ of each pointed ℓ-sotree the following holds:*

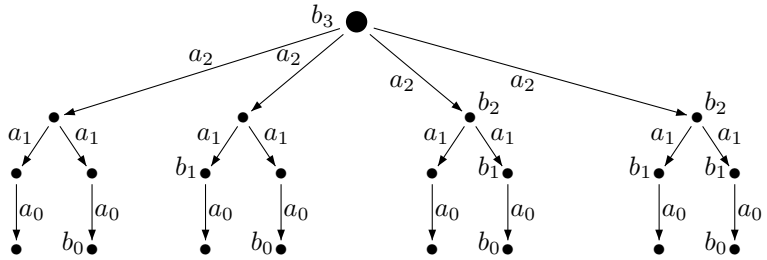


Figure 1 The 1-pointed 2-sotree of value 12, where the horizontal transitions $\Rightarrow$ and $\Leftarrow$ between siblings are omitted.

(1) $(r,\mathcal{T}) \otimes (r',\mathcal{T}') \models \mathsf{eq}_\ell^\otimes$ *if and only if* $\mathsf{val}(r,\mathcal{T}) = \mathsf{val}(r',\mathcal{T}')$,
(2) $(r,\mathcal{T}) \otimes (r',\mathcal{T}') \models \mathsf{less}_\ell^\otimes$ *if and only if* $\mathsf{val}(r,\mathcal{T}) < \mathsf{val}(r',\mathcal{T}')$, *and*
(3) $(r,\mathcal{T}) \otimes (r',\mathcal{T}') \models \mathsf{succ}_\ell^\otimes$ *if and only if* $\mathsf{val}(r,\mathcal{T}) + 1 = \mathsf{val}(r',\mathcal{T}')$.

Proof. We define the formulas by induction on ℓ. For the induction base $\ell = 0$ we put:
(1) $\mathsf{eq}_0^\otimes \stackrel{\text{def}}{=} \langle\alpha,a_0\rangle\langle\beta,a_0\rangle\,((\alpha,b_0) \leftrightarrow (\beta,b_0))$.
(2,3) $\mathsf{less}_0^\otimes \stackrel{\text{def}}{=} \mathsf{succ}_0^\otimes \stackrel{\text{def}}{=} \langle\alpha,a_0\rangle\langle\beta,a_0\rangle\,(\neg(\alpha,b_0) \wedge (\beta,b_0))$.
For the induction step, we define:

(1) $\mathsf{eq}_{\ell+1}^\otimes \stackrel{\text{def}}{=} [\alpha,a_{\ell+1}][\beta,a_{\ell+1}]\left(\mathsf{eq}_\ell^\otimes \longrightarrow ((\alpha,b_\ell) \leftrightarrow (\beta,b_\ell))\right)$.
(2) $\mathsf{less}_{\ell+1}^\otimes \stackrel{\text{def}}{=} \langle\alpha,a_{\ell+1}\rangle\langle\beta,a_{\ell+1}\rangle\varphi_{\ell+1}^\otimes$, where
$$\varphi_{\ell+1}^\otimes \stackrel{\text{def}}{=} \left(\mathsf{eq}_\ell^\otimes \wedge \neg(\alpha,b_\ell) \wedge (\beta,b_\ell) \wedge [\alpha,\Rightarrow][\beta,\Rightarrow]\left(\mathsf{eq}_\ell^\otimes \rightarrow ((\alpha,b_\ell) \leftrightarrow (\beta,b_\ell))\right)\right),$$
thus expressing that there is an $i \in [0,\mathsf{Tower}(\ell)-1]$ such that the i^{th} bit of $\mathsf{val}(r,\mathcal{T})$ is *not* set, the i^{th} bit of $\mathsf{val}(r',\mathcal{T}')$ is set and moreover $\mathsf{val}(r,\mathcal{T})$ and $\mathsf{val}(r',\mathcal{T}')$ agree on the j^{th} bit for all $i < j \leq \mathsf{Tower}(\ell)-1$.
(3) $\mathsf{succ}_{\ell+1}^\otimes \stackrel{\text{def}}{=} \langle\alpha,a_{\ell+1}\rangle\langle\beta,a_{\ell+1}\rangle(\varphi_{\ell+1}^\otimes \wedge [\alpha,\Leftarrow](\alpha,b_\ell) \wedge [\beta,\Leftarrow]\neg(\beta,b_\ell))$, thus expressing that there is an $i \in [0,\mathsf{Tower}(\ell)-1]$ such that the i^{th} bit of $\mathsf{val}(r,\mathcal{T})$ is *not* set, the i^{th} bit of $\mathsf{val}(r',\mathcal{T}')$ is set, $\mathsf{val}(r,\mathcal{T})$ and $\mathsf{val}(r',\mathcal{T}')$ agree on the j^{th} bit for all $i < j \leq \mathsf{Tower}(\ell)-1$ and finally the j^{th} bit of $\mathsf{val}(r,\mathcal{T})$ is set whereas the j^{th} bit of $\mathsf{val}(r',\mathcal{T}')$ is *not* set for each $0 \leq j < i$. ◀

Similar formulas as in Lemma 5 can be shown for the synchronous product.

▶ **Lemma 6** (Formulas for the synchronous product of ℓ-sotrees)**.** *For each $\ell \in \mathbb{N}$ and labels α,β there are formulas $(\mathsf{eq}_\ell^\times(\alpha,\beta))_{\ell\in\mathbb{N}} = (\mathsf{eq}_\ell^\times)_{\ell\in\mathbb{N}}$, $(\mathsf{succ}_\ell^\times(\alpha,\beta))_{\ell\in\mathbb{N}} = (\mathsf{succ}_\ell^\times)_{\ell\in\mathbb{N}}$, and $(\mathsf{less}_\ell^\times(\alpha,\beta))_{\ell\in\mathbb{N}} = (\mathsf{less}_\ell^\times)_{\ell\in\mathbb{N}}$, each over the signature $(\{\alpha,\beta\} \times \mathsf{P}_\ell, \mathsf{A}_\ell)$ and each of size $2^{O(\ell)}$ such that for the synchronous α-extension $(r,\mathcal{T})$ of each pointed ℓ-sotree and the synchronous β-extension $(r',\mathcal{T}')$ of each pointed ℓ-sotree the following holds:*
(1) $(r,\mathcal{T}) \times (r',\mathcal{T}') \models \mathsf{eq}_\ell^\times$ *if and only if* $\mathsf{val}(r,\mathcal{T}) = \mathsf{val}(r',\mathcal{T}')$,
(2) $(r,\mathcal{T}) \times (r',\mathcal{T}') \models \mathsf{less}_\ell^\times$ *if and only if* $\mathsf{val}(r,\mathcal{T}) < \mathsf{val}(r',\mathcal{T}')$, *and*
(3) $(r,\mathcal{T}) \times (r',\mathcal{T}') \models \mathsf{succ}_\ell^\times$ *if and only if* $\mathsf{val}(r,\mathcal{T}) + 1 = \mathsf{val}(r',\mathcal{T}')$.

4 Overview of the proofs

The reason for choosing our particular encoding via sibling-ordered trees from Section 3 is that we would like to provide possibly short proofs for our main results. Let us discuss this in more detail.

For showing, unless FPT = AW[$*$], that for given systems $\mathcal{T}_1, \ldots, \mathcal{T}_d$ and an ML formula φ there is no algorithm that decides whether φ holds in the asynchronous product of $\mathcal{T}_1, \ldots, \mathcal{T}_d$ and runs in time $(\sum_i |\mathcal{T}_i|)^{O(1)} \cdot f(|\varphi|)$ for any elementary function f, we reduce from model checking first-order logic on finite words with order from [9] (which has been shown not be decidable in time $|\mathcal{W}|^{O(1)} \cdot f(|\varphi|)$ for any elementary function unless FPT = AW[$*$] in [9]). In fact, it might be possible to prove our result directly by working with the asynchronous product of trees from [8] but this would have involved a complicated encoding of 3-CNF formulas as in [9] and hence result in a proof that is technically much more involved than our current proof. Moreover it would not shed new light into the problem.

In our reduction from model checking first-order logic over words we encode each position of the input word by one of our sibling-ordered trees (from Section 3) and present the whole word by an asynchronous product of these sibling-ordered trees such that ML formulas of size $O(\log^*(n))$ can test whether two such marked trees are related by the order. For encoding this we cannot use trees that were used in [10] for proving nonelementary lower bounds for satisfiability checking two-dimensional modal logic since it is not at all clear how to simulate the above-mentioned order relation between an asynchronous product between two of them (in [10] only the successor relation was simulated with involved technical machinery).

We are able to prove under the weaker assumption P $\neq$ NP that there is neither such an algorithm that runs in time $(\sum_i |\mathcal{T}_i|)^{O(1)} \cdot f(|\varphi|)$ for elementary function f for model checking ML on the synchronous product of systems nor one for model checking EF on the asynchronous product of systems. To obtain this, one possibility could have been to reduce from the model checking problem of monadic second-order logic over finite words (which has been shown not to be decidable in time $|\mathcal{W}|^{O(1)} \cdot f(|\varphi|)$ for any elementary function f unless P = NP in [9]) but for this result it turned out that a direct proof is simpler than to encode monadic second-order quantification. We reduce from the NP-complete $n \times n$-tiling problem.

Let us point out the crucial differences to model checking the asynchronous product and why we are able to weaken our complexity-theoretic assumption from FPT $\neq$ AW[$*$] to P $\neq$ NP when considering the synchronous product. The asynchronous product of systems $\mathcal{T}_1, \ldots, \mathcal{T}_d$ is generally also exponentially big in the input size, but the out-degree in $\otimes_i \mathcal{T}_i$ is only polynomially bounded. When encoding the NP-hard $n \times n$-tiling problem as instances of model checking modal logic on the synchronous product (and thus assuming the weaker P $\neq$ NP assumption), the idea is to use for each of the n^2 coordinates (x, y) one sibling-ordered tree $\mathcal{T}_{x,y}$ for representing this coordinate to be colored with a tile type. Guessing one coloring (of the exponentially many) to the tiling problem can be achieved in *one* step by an ML formula in the synchronous product, whereas it would take exponentially many steps in the asynchronous product (thus, the lower bound cannot be applied to the asynchronous product). Hence, as mentioned above, for proving lower bounds for the asynchronous product, we had to make a stronger complexity theoretic assumption, namely FPT $\neq$ AW[$*$].

5 Model Checking ML on the asynchronous product

In this section we prove that model checking ML for asynchronous product is not fixed-parameter tractable with an elementary function in the size of the formula unless FPT = AW[$*$]. We reduce from an orthogonal hardness result for model checking first-order logic on words from [9].

Given a finite alphabet Σ the signature of first-order (FO) formulas for words over Σ is $(\Sigma, <)$, where each $a \in \Sigma$ is a unary symbol (the letter predicate) and $<$ (the order on positions). We use the following lower bound result from model checking first-order logic on words.

▶ **Theorem 7** ([9]). *Let f be an elementary function. Then there is no algorithm that decides*
INPUT: A word $\mathcal{W}$ and some FO *sentence φ each over some alphabet Σ.*
QUESTION: $\mathcal{W} \models \varphi$?
in time $|\mathcal{W}|^{O(1)} \cdot f(|\varphi|)$ unless FPT $=$ AW$[*]$.

We can now present the main result of this section.

▶ **Theorem 8.** *Let f be an elementary function. Then there is no algorithm that decides*
INPUT: Transition systems $\mathcal{T}_1, \dots, \mathcal{T}_d$, a state $\overline{s}$ of $\bigotimes_{i=1}^{d} \mathcal{T}_i$ and an ML *formula ψ.*
QUESTION: $(\overline{s}, \bigotimes_{i\in[1,d]} \mathcal{T}_i) \models \psi$?
in time $(\sum_{i=1}^{d} |\mathcal{T}_i|)^{O(1)} \cdot f(|\psi|)$ unless FPT $=$ AW$[*]$.

Proof. The idea is to show that the existence of such an elementary function f contradicts Theorem 7. So for the sake of contradiction, let us assume that there were an elementary function f and an algorithm that decides the model checking problem for a given ML formula ψ and a state $\overline{s}$ in the asynchronous product of d finite systems $\mathcal{T}_1, \dots, \mathcal{T}_d$ in time $(\sum_{i\in[1,d]} |\mathcal{T}_i|)^{O(1)} \cdot f(|\psi|)$. Let us show that this algorithm can be used for model checking FO over words. For this, let $\mathcal{W} = a_0 \cdots a_{n-1}$, where $a_i \in \Sigma$ for each $i \in [0, n-1]$ and let $\varphi = \exists x_1 \forall x_2 \cdots \exists x_{2k-1} \forall x_{2k} \psi(x_1, \dots, x_{2k})$ be an FO sentence over Σ in prenex normal form with alternating quantifiers without loss of generality. We will compute some $d \geq 1$, transition systems $\mathcal{T}_1, \dots, \mathcal{T}_d$, some state $\overline{s}$ of $\mathcal{T} \stackrel{\text{def}}{=} \bigotimes_{i=1}^{d} \mathcal{T}_i$ and some ML formula ψ such that

(i) $\mathcal{W} \models \varphi$ if and only if $(\overline{s}, \mathcal{T}) \models \psi$,
(ii) $d \leq O(|\varphi|)$,
(iii) $|\mathcal{T}_i| \leq |\mathcal{W}|^{O(1)}$ for each $i \in [1, d]$,
(iv) $|\psi| \leq O(|\varphi|) \cdot 2^{O(\log^*(|\mathcal{W}|))}$, and
(v) each $\mathcal{T}_i$, $\overline{s}$ and ψ is computable in time $|\mathcal{W}|^{O(1)} \cdot g(|\varphi|)$ for an elementary g.

Before we show how to compute the $\mathcal{T}_i$, $\overline{s}$ and ψ, let us start with the desired contradiction. Note that an elementary computation of such $\mathcal{T}_i$, $\overline{s}$ and ψ as stated above would lead to an algorithm that decides $\mathcal{W} \models \varphi$ in time $|\mathcal{W}|^{O(1)} \cdot g(|\varphi|)$ (for the reduction, Point (v) from above) plus

$$
\begin{aligned}
& (\sum_{i\in[1,d]} |\mathcal{T}_i|)^{O(1)} \cdot f(|\psi|) \\
\overset{\text{(iii),(iv)}}{\leq}\ & (d \cdot |\mathcal{W}|^{O(1)})^{O(1)} \cdot f(O(|\varphi|) \cdot 2^{O(\log^*(|\mathcal{W}|))}) \\
\leq\ & |\mathcal{W}|^{O(1)} \cdot d^{O(1)} \cdot f(O(|\varphi|) \cdot 2^{O(\log^*(|\mathcal{W}|))}) \\
\overset{\text{(ii)}}{\leq}\ & |\mathcal{W}|^{O(1)} \cdot |\varphi|^{O(1)} \cdot f(O(|\varphi|) \cdot 2^{O(\log^*(|\mathcal{W}|))})
\end{aligned}
$$

by the binomial theorem. Furthermore, it is easy to see that there exist elementary functions f_1, f_2 such that the latter is bounded by

$$
|\mathcal{W}|^{O(1)} \cdot f_1(|\varphi|) \cdot f_2(2^{O(|\log^*(|\mathcal{W}|))}) \quad \leq \quad |\mathcal{W}|^{O(1)} \cdot f_1(|\varphi|)
$$

since $f_2(2^{O(\log^*(|\mathcal{W}|))})$ is bounded by a sublinearly growing function in $|\mathcal{W}|$ (and is thus in particular bounded by $|\mathcal{W}|^{O(1)}$). The latter contradicts Theorem 7. This concludes the analysis of the overall running time of our reduction.

Let us conclude to show that we can compute transition systems $\mathcal{T}_1, \dots, \mathcal{T}_d$, state $\overline{s}$ of $\mathcal{T} \stackrel{\text{def}}{=} \bigotimes_{i=1}^{d} \mathcal{T}_i$ and some ML formula ψ such that moreover Points (i) to (v) from above

hold. Recall that $\mathcal{W} = a_0 \cdots a_{n-1}$. Let us first compute the smallest $\ell \geq 1$ such that $\mathsf{Tower}(\ell-1) < n \leq \mathsf{Tower}(\ell)$. Note that $\ell \leq O(\log^*(n)) \leq O(\log^*(|\mathcal{W}|))$. Let us compute the 0-pointed ℓ-sotrees $\Upsilon_\ell(0), \ldots, \Upsilon_\ell(n-1)$, with their points $r_\ell(0), \ldots, r_\ell(n-1)$, respectively.

Recall that each pointed ℓ-sotree was defined over the signature $(\mathsf{P}_\ell, \mathsf{A}_\ell)$. Without loss of generality we assume that $\Sigma \cap \mathsf{P}_\ell = \emptyset$.

Let $\mathcal{U}$ be the pointed transition system over $(\mathsf{P}_\ell \cup \Sigma, \mathsf{A}_\ell \cup \{\gamma\})$ that one obtains from the disjoint union of $\Upsilon_\ell(0), \ldots, \Upsilon_\ell(n-1)$ and some fresh state s (which will be the point of $\mathcal{U}$) and by adding the atomic proposition $a_j \in \Sigma$ to $\mathcal{U}$'s state $r_\ell(j)$ and connecting s with $r_\ell(j)$ with a γ-labeled transition for each $j \in [0, n-1]$. Recall that $\varphi = \exists x_1 \forall x_2 \cdots \exists x_{2k-1} \forall x_{2k}$ $\psi(x_1, \ldots, x_{2k})$ is our input FO formula. We set $d \stackrel{\text{def}}{=} 2k$, hence $d \leq O(|\varphi|)$ and thus Point (ii) is shown. We define $\mathcal{T}_i$ to be the asynchronous i-extension of $\mathcal{U}$ for each $i \in [1, 2k]$. Recall that $\mathsf{size}(\ell)$ denotes the size of each $\Upsilon_\ell(j)$. Hence

$$|\mathcal{T}_i| \;\leq\; O(n \cdot \mathsf{size}(\ell)) \;\stackrel{\text{Lemma 4}}{\leq}\; O(n \cdot \mathsf{Tower}(\ell-1)^2) \;\leq\; O(n^3) \;\leq\; |\mathcal{W}|^{O(1)}$$

for each $i \in [1, d]$ and thus (iii) is shown. We define the pointed transition system $\mathcal{T} \stackrel{\text{def}}{=} \bigotimes_{i \in [1,2k]} \mathcal{T}_i$ with point $\overline{s} \stackrel{\text{def}}{=} (s, \ldots, s)$. The intuition is that each position $j \in [0, n-1]$ of $\mathcal{W}$ will be presented by $\Upsilon_\ell(j)$ and $\mathcal{U}$ will correspond to $\mathcal{W}$. The transition system $\mathcal{T}$ consists of the asynchronous product of $d = 2k$ copies of $\mathcal{U}$ since φ consists of $2k$ quantifiers. Thus, the pointed transition system $\mathcal{T}_i$ will handle the position where variable x_i will be bound to, for each $i \in [1, 2k]$.

Finally, we define the ML-formula ψ as follows

$$\psi \;\stackrel{\text{def}}{=}\; \langle 1, \gamma \rangle [2, \gamma] \cdots \langle 2k-1, \gamma \rangle [2k, \gamma]\; \widehat{\psi},$$

where the ML formula $\widehat{\psi}$ is obtained from φ by replacing each occurrence of $a(x_i)$ by (i, a) and by replacing each occurrence of $x_i < x_{i'}$ by $\mathsf{less}^{\otimes}_\ell(i, i')$, where each formula $\mathsf{less}^{\otimes}_\ell(i, i')$ is taken from Lemma 5 for $\alpha = i$ and $\beta = i'$. It is straightforward to verify that we have $\mathcal{W} \models \varphi$ if and only if $\left(\overline{s}, \bigotimes_{i=1}^{d} \mathcal{T}_i\right) \models \psi$, which delivers Point (i). Recall that $\ell \leq O(\log^*(|\mathcal{W}|))$ and the size of each formula $\mathsf{less}^{\otimes}_\ell(i, i')$ is bounded by $2^{O(\ell)}$ by Lemma 5, thus Point (iv) holds. Point (v) is easy to see by the fact that each $\mathcal{T}_i$ is computable in time $|\mathcal{T}_i|^{O(1)}$ and the formula ψ is computable in time $|\psi|^{O(1)}$ and by using Points (ii), (iii) and (iv) and analogous arguments as for the above running time analysis. ◀

An adaption of the latter proof can be carried out for model checking ML on the synchronous product. Remarkably, for model checking the synchronous product the complexity-theoretic assumption can be indeed be weakened to $\mathsf{P} \neq \mathsf{NP}$ as will be shown in the next section.

6 Model checking ML on the synchronous product and model checking EF on the asynchronous product

In this section we prove that the fixed-parameter tractability of model checking ML for synchronous product and EF for asynchronous product is cannot be witnessed by an elementary running time in the size of the formula unless $\mathsf{P} = \mathsf{NP}$.

We recall tiling systems for this. A *tiling system* is a tuple $\mathcal{S} = (\Theta, \mathbb{H}, \mathbb{V})$, where Θ is a finite set of *tile types*, $\mathbb{H} \subseteq \Theta \times \Theta$ is a *horizontal matching relation*, and $\mathbb{V} \subseteq \Theta \times \Theta$ is a *vertical matching relation*. A mapping $\sigma : [0, n-1]^2 \to \Theta$ (where $n \geq 0$) is an $n \times n$*-solution for* $\mathcal{S}$ if for all $x, y \in [0, n-1]$ the following holds: (i) if $x < n-1$, $\sigma(x, y) = \theta$, and $\sigma(x+1, y) = \theta'$,

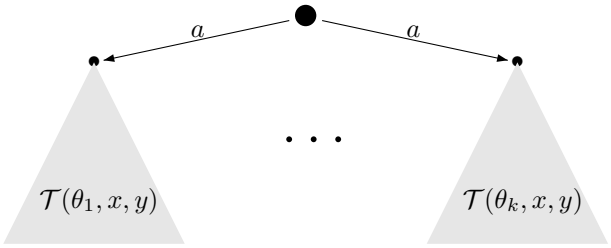


Figure 2 The pointed transition system $\mathcal{T}_{x,y}$ for each $x \in [0, n-1]$ and each $y \in [1, n-1]$.

then $(\theta, \theta') \in \mathbb{H}$, and (ii) if $y < n-1$, $\sigma(x,y) = \theta$, and $\sigma(x, y+1) = \theta'$, then $(\theta, \theta') \in \mathbb{V}$. Let $\mathcal{W} = \theta_0 \cdots \theta_{n-1} \in \Theta^n$ be a word. By $\mathsf{Sol}_n(\mathcal{S}, \mathcal{W})$ we denote the set of all $n \times n$-solutions σ for $\mathcal{S}$ such that $\sigma(x,0) = \theta_x$ for all $x \in [0, n-1]$. For a fixed tiling system $\mathcal{S}$, its $n \times n$-*tiling problem* asks for a given word $\mathcal{W} \in \Theta^n$, whether $\mathsf{Sol}_n(\mathcal{S}, \mathcal{W}) \neq \emptyset$ holds. It is folklore that there exists a *fixed* tiling system $\mathcal{S}_0$ whose $n \times n$-tiling problem is NP-hard; see also [3]. Let us fix such a tiling system $\mathcal{S}_0 = (\Theta_0, \mathbb{H}_0, \mathbb{V}_0)$ for the rest of this section.

▶ **Theorem 9.** *Let f be an elementary function. Then there is no algorithm that decides*
INPUT: Transition systems $\mathcal{T}_1, \ldots, \mathcal{T}_d$, a state $\overline{s}$ of $\prod_{i=1}^d \mathcal{T}_i$ and an ML *formula ψ.*
QUESTION: $(\overline{s}, \prod_{i=1}^d \mathcal{T}_i) \models \psi$?
in time $(\sum_{i=1}^d |\mathcal{T}_i|)^{O(1)} \cdot f(|\psi|)$ unless P = NP.

Proof. The idea is to show that the existence of such an elementary function f implies that the $n \times n$-tiling problem for $\mathcal{S}_0$ is in P and thus P = NP. So for the sake of contradiction, let us assume that there were an elementary function f and an algorithm that decides the model checking problem for a given ML formula ψ and a state $\overline{s}$ in the synchronous product of d finite systems $\mathcal{T}_1, \ldots, \mathcal{T}_d$ in time $(\sum_{i=1}^d |\mathcal{T}_i|)^{O(1)} \cdot f(|\psi|)$. Let us show that this algorithm can be used for deciding the $n \times n$-tiling problem for $\mathcal{S}_0$ in polynomial time. Recall $\mathcal{S}_0 = (\Theta_0, \mathbb{H}_0, \mathbb{V}_0)$. Let us assume $\Theta_0 = \{\theta_1, \ldots, \theta_k\}$. Let $\mathcal{W} = \mathcal{W}_0 \cdots \mathcal{W}_{n-1} \in \Theta_0^n$ be an input word to the $n \times n$-tiling problem for $\mathcal{S}_0$. We will compute transition systems $\{\mathcal{T}_{x,y} \mid x, y \in [0, n-1]\}$, some state $\overline{s}$ of $\prod_{x,y \in [0,n-1]} \mathcal{T}_{x,y}$ and some ML formula ψ such that

(i) $\mathsf{Sol}_n(\mathcal{S}_0, \mathcal{W}) \neq \emptyset$ if and only if $(\overline{s}, \prod_{x,y \in [0,n-1]} \mathcal{T}_{x,y}) \models \psi$,
(ii) $|\mathcal{T}_{x,y}| \leq |\mathcal{W}|^{O(1)}$ for each $x, y \in [0, n-1]$,
(iii) $|\psi| \leq 2^{O(\log^*(|\mathcal{W}|))}$, and
(iv) each $\mathcal{T}_{x,y}$, $\overline{s}$ and ψ is computable in time $|\mathcal{W}|^{O(1)}$.

The proof that Points (i) to (iv) lead to an overall algorithm that decides the non-emptiness of $\mathsf{Sol}_n(\mathcal{S}_0, \mathcal{W})$ in time $|\mathcal{W}|^{O(1)}$ works analogously as the proof of Theorem 8 and is therefore omitted. Recall that $\mathcal{W} = \mathcal{W}_0 \cdots \mathcal{W}_{n-1} \in \Theta^n$ and let us assume without loss of generality that $n \geq 3$.

Let us first compute the smallest $\ell \geq 1$ such that $\mathsf{Tower}(\ell - 1) < n \leq \mathsf{Tower}(\ell)$. Note that $\ell \leq O(\log^*(n)) \leq O(\log^*(|\mathcal{W}|))$. Recall that $\Upsilon_\ell(j)$ denotes the 0-pointed ℓ-sotree of value j for each $j \in [0, \mathsf{Tower}(\ell) - 1]$. Let $\Upsilon_\ell^\alpha(i)$ (resp. $\Upsilon_\ell^\beta(i)$) denote the synchronous α-extension (resp. synchronous β-extension) of $\Upsilon_\ell(j)$ for each $j \in [0, \mathsf{Tower}(\ell) - 1]$.

We will call each pointed transition system $\mathcal{T}_{x,y}$ a *component* of the product transition system $\mathcal{T} \stackrel{\text{def}}{=} \prod_{x,y \in [0,n-1]} \mathcal{T}_{x,y}$. Let us mention the purpose of the pointed transition systems $\mathcal{T}_{x,y}$. With points denoted by ●, Figure 2 shows the pointed transition system $\mathcal{T}_{x,y}$ whenever $1 \leq y \leq n-1$ and Figure 3 shows them whenever $y = 0$, where the pointed transition systems

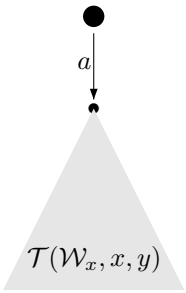


Figure 3 The pointed transition system $\mathcal{T}_{x,y}$ for each $x \in [0, n-1]$ and $y = 0$.

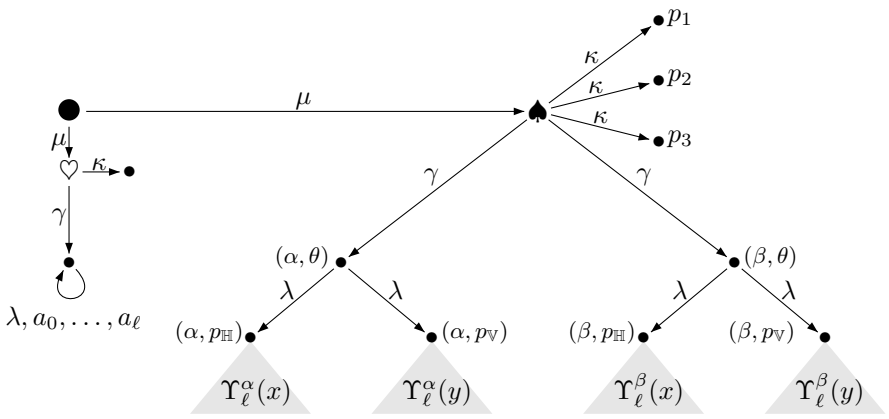


Figure 4 The pointed transition system $\mathcal{T}(\theta, x, y)$ for each $\theta \in \Theta_0$ and each $x, y \in [0, n-1]$.

$\mathcal{T}(\theta, x, y)$ for each $\theta \in \Theta_0$ and each $x, y \in [0, n-1]$ will be described in more detail below. Note that we have hereby specified the point $\overline{s}$ of $\mathcal{T} = \prod_{x,y \in [0,n-1]} \mathcal{T}_{x,y}$. Each a-successor of $\overline{s}$ in $\mathcal{T}$ hence corresponds to a choice function from $[0, n-1] \times [0, n-1]$ to Θ_0, thus selecting for each $(x, y) \in [0, n-1] \times [0, n-1]$, some tile type $\theta_{x,y}$ from Θ_0 (by taking in the component $\mathcal{T}_{x,y}$ the a-successor to the point of $\mathcal{T}(\theta_{x,y}, x, y)$); however for each (x, y) with $y = 0$ we only allow to choose $\theta_{x,y} = \mathcal{W}_x$, since we would like to restrict ourselves to choice functions ρ with $\rho(x, 0) = \mathcal{W}_x$ for each $x \in [0, n-1]$.

Having selected a choice function that respects our input word $\mathcal{W}$, let us now comment on the pointed transition systems $\mathcal{T}(\theta, x, y)$ with point $\bullet$ as depicted in Figure 4. Let assume, for the moment, that for each $\mathcal{T}(\theta, x, y)$ we are either in state $\heartsuit$ or in state $\spadesuit$. Then in the corresponding pointed synchronous product transition system the formula $\varphi_{two} \stackrel{\text{def}}{=} \langle\kappa\rangle(p_1 \wedge p_2) \wedge \neg\langle\kappa\rangle(p_1 \wedge p_2 \wedge p_3)$ holds if and only if exactly two components are in state $\spadesuit$. Note that since $k = |\Theta_0|$ is a fixed constant we have $|\mathcal{T}_{x,y}| \leq O(\mathsf{size}(\ell)) \stackrel{\text{Lemma 4}}{\leq} O(\mathsf{Tower}(\ell - 1)^2) \leq O(n^2) \leq |\mathcal{W}|^{O(1)}$ for each $x, y \in [0, n-1]$ and thus (ii) is shown.

Let us define the auxiliary formulas $\varphi_\alpha \stackrel{\text{def}}{=} \bigvee_{\theta \in \Theta_0} (\alpha, \theta)$ and $\varphi_\beta \stackrel{\text{def}}{=} \bigvee_{\theta \in \Theta_0} (\beta, \theta)$. Let us list our final formula ψ before we comment on it below.

$$\langle a\rangle[\mu]\Big[\varphi_{two} \to [\gamma]\Big(\varphi_\alpha \wedge \varphi_\beta \wedge \langle\lambda\rangle((\alpha,p_{\mathbb{H}}) \wedge (\beta,p_{\mathbb{H}}) \wedge \mathsf{eq}_\ell^\times(\alpha,\beta)) \wedge \langle\lambda\rangle((\alpha,p_{\mathbb{V}}) \wedge (\beta,p_{\mathbb{V}}) \wedge \mathsf{succ}_\ell^\times(\alpha,\beta))\Big) \to \varphi_{\mathbb{V}}\Big]$$

$$\wedge \Big[\varphi_{two} \to [\gamma]\Big(\varphi_\alpha \wedge \varphi_\beta \wedge \langle\lambda\rangle((\alpha,p_{\mathbb{H}}) \wedge (\beta,p_{\mathbb{H}}) \wedge \mathsf{succ}_\ell^\times(\alpha,\beta)) \wedge \langle\lambda\rangle((\alpha,p_{\mathbb{V}}) \wedge (\beta,p_{\mathbb{V}}) \wedge \mathsf{eq}_\ell^\times(\alpha,\beta))\Big) \to \varphi_{\mathbb{H}}\Big]$$

where

$$\varphi_{\mathbb{V}} \stackrel{\text{def}}{=} \bigvee_{(\theta,\theta')\in\mathbb{V}_0} (\alpha,\theta) \wedge (\beta,\theta') \quad \text{and} \quad \varphi_{\mathbb{H}} \stackrel{\text{def}}{=} \bigvee_{(\theta,\theta')\in\mathbb{H}_0} (\alpha,\theta) \wedge (\beta,\theta').$$

When evaluating it from $(\overline{s}, \prod_{x,y\in[0,n-1]} \mathcal{T}_{x,y})$, the formula ψ can be read as follows: There is a choice function $\rho \in \Theta_0^{[0,n-1]\times[0,n-1]}$ that respects our input word $\mathcal{W}$ (this corresponds to the part $\langle a\rangle$) such that whenever we choose (this corresponds to $[\mu]$) precisely two (this is realized by going to $\heartsuit$ and $\spadesuit$ and is is controlled by the formula φ_{two}) different elements (x,y) and (x',y') from $[0,n-1]\times[0,n-1]$ and exactly one of these two is "colored" with α and the other with β (by going along the transition γ and checking the formula $\varphi_\alpha \wedge \varphi_\beta$) we have that, firstly, $x = x'$ and $y' = y+1$ implies that $(\rho(x,y),\rho(x',y')) \in \mathbb{V}_0$ and, secondly, $x' = x+1$ and $y = y'$ implies that $(\rho(x,y),\rho(x',y')) \in \mathbb{H}_0$. Thus $\mathsf{Sol}(\mathcal{S}_0,\mathcal{W}) \neq \emptyset$ if and only if $(\overline{s}, \prod_{x,y\in[0,n-1]} \mathcal{T}_{x,y}) \models \psi$, thus Point (i) holds. The definition of ψ shows that $|\psi| \leq 2^{O(\ell)} \leq 2^{O(\log^*(|\mathcal{W}|))}$ by Lemma 6 and thus Point (iii) follows. The reader easily verifies that the transition systems $\mathcal{T}_{x,y}$, the state $\overline{s}$ of $\mathcal{T}$ and the formula ψ can in total be computed in time $|\mathcal{W}|^{O(1)}$, which delivers Point (iv) and completes the proof of the theorem. ◀

The following theorem states an analogous result for model checking EF on the asynchronous product. One can recycle the proof of Theorem 9, replace the synchronous α/β-extension by asynchronous α/β-extension, respectively, replacing the $\mathsf{succ}_\ell^\times(\alpha,\beta)$ by $\mathsf{succ}_\ell^\otimes(\alpha,\beta)$, replacing $\langle e\rangle$ by EF and $[e]$ by AG in a relativized fashion by introducing fresh atomic propositions that allow us to correctly mimick one transition in the synchronous product by a sequence of transitions in the asynchronous product. Recall that each transition system $\mathcal{T}_{x,y}$ is essentially a tree (almost) except for the substructures of the form $\Upsilon_\ell^{\alpha/\beta}(x/y)$. Simulating each modality $\langle z\rangle$, where $z \in \{a,\mu,\kappa,\gamma,\delta\}$, in the synchronous product can be simulated by the modality EF in the asynchronous product in addition to some formula that expresses (1) each asynchronous component can no longer execute z and (2) we have not moved too far down in the tree than just along one z-labeled transition.

▶ **Theorem 10.** *Let f be an elementary function. Then there is no algorithm that decides*
INPUT: Transition systems $\mathcal{T}_1,\ldots,\mathcal{T}_d$, a state $\overline{s}$ of $\bigotimes_{i=1}^d \mathcal{T}_i$ and an EF formula φ.
QUESTION: $(\overline{s}, \bigotimes_{i=1}^d \mathcal{T}_i) \models \varphi$?
in time $(\sum_{i=1}^d |\mathcal{T}_i|)^{O(1)} \cdot f(|\varphi|)$ unless $\mathsf{P} = \mathsf{NP}$.

7 Conclusion

In this paper we considered the fixed-parameter tractability of model checking modal logic and EF logic on concurrent systems that are modeled as the asynchronous or synchronous product of finite systems when the size of the input formula is the parameter. We showed that although these model checking problems are often fixed-parameter tractable one cannot hope for any FPT algorithm that runs elementary in the size of formula. It turned out that for model checking modal logic the mode of synchronization plays a role: for the asynchronous product we had to assume $\mathsf{FPT} \neq \mathsf{AWT}[*]$, whereas we were able to weaken our assumption for the synchronous product to $\mathsf{P} \neq \mathsf{NP}$. Let us conclude with some questions that we would like to answer in the full version of this paper. In analogy to [9] it would be interesting to study the question if even weaker complexity theoretic assumptions such as $\mathsf{P} \neq \mathsf{PSPACE}$ can be assumed. Moreover, we remark that some of our lower bound proofs also hold even when $d + |\varphi|$ (instead of $|\varphi|$) is the parameter; it seems worth investigating when this strengthening is possible. Too, the question arises what (elementary) bounds one can prove for transition systems of bounded degree.

References

1 Antonis Achilleos, Michael Lampis, and Valia Mitsou. Parameterized Modal Satisfiability. In *ICALP (2)*, volume 6199 of *Lecture Notes in Computer Science*, pages 369–380. Springer, 2010.

2 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

3 Egon Börger, Erich Grädel, and Yuri Gurevich. *The classical decision problem*. Universitext. Springer-Verlag, Berlin, 2001.

4 Yijia Chen, Jörg Flum, and Martin Grohe. Bounded nondeterminism and alternation in parameterized complexity theory. In *IEEE Conference on Computational Complexity*, pages 13–29. IEEE Computer Society, 2003.

5 E. M. Clarke and E. A. Emerson. *Model Checking*. MIT Press, 1999.

6 Anuj Dawar, Martin Grohe, Stephan Kreutzer, and Nicole Schweikardt. Model Theory Makes Formulas Large. In *Proc. of ICALP*, volume 4596 of *Lecture Notes in Computer Science*. Springer, 2007.

7 Stéphane Demri, François Laroussinie, and Ph. Schnoebelen. A parametric analysis of the state-explosion problem in model checking. *J. Comput. Syst. Sci.*, 72(4):547–575, 2006.

8 Jörg Flum and Martin Grohe. *Parametrized Complexity Theory*. Springer, 2006.

9 Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130(1-3):3–31, 2004.

10 Stefan Göller, Jean Christoph Jung, and Markus Lohrey. The Complexity of Decomposing Modal and First-Order Theories. In *LICS*, pages 325–334. IEEE, 2012.

11 Stefan Göller and Anthony Widjaja Lin. Concurrency Makes Simple Theories Hard. In *STACS*, volume 14 of *LIPIcs*, pages 148–159. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

12 Orna Lichtenstein and Amir Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *POPL*, pages 97–107. ACM Press, 1985.

13 M. Praveen. *Parametrized Complexity of some Problems in Concurrency and Verification*. PhD thesis, Homi Bhabha National Institute, 2011.

14 Alexander Rabinovich. On compositionality and its limitations. *ACM Trans. Comput. Log.*, 8(1), 2007.

15 A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3), 1985.

A Proof of Lemma 4

Proof. One easily proves via induction on n that $n^2 + 1 \leq 2^n$ for each $n \geq 5$. We prove the lemma by induction on $\ell \geq 3$. For the induction base $\ell = 3$ we have

$$\mathsf{size}(4) = \mathsf{Tower}(3) \cdot \mathsf{size}(3) + 1 = \mathsf{Tower}(3) \cdot (\mathsf{Tower}(2) \cdot \mathsf{size}(2) + 1) + 1$$

$$\overset{\text{Figure 1}}{=} 2^{16} \cdot (16 \cdot 21 + 1) + 1 = 2^{16} \cdot 337 < (2^{16})^2 = \mathsf{Tower}(3)^2.$$

For the induction step, we have for each $\ell \geq 3$

$$\begin{aligned}
\mathsf{size}(\ell+1) &= \mathsf{Tower}(\ell) \cdot \mathsf{size}(\ell) + 1 \\
&\overset{\text{IH}}{\leq} \mathsf{Tower}(\ell) \cdot \mathsf{Tower}(\ell-1)^2 + 1 \\
&\leq \mathsf{Tower}(\ell) \cdot (\mathsf{Tower}(\ell-1)^2 + 1) \\
&\overset{\mathsf{Tower}(\ell-1) \geq 5}{\leq} \mathsf{Tower}(\ell) \cdot 2^{\mathsf{Tower}(\ell-1)} \\
&= \mathsf{Tower}(\ell)^2.
\end{aligned}$$

◀

B Proof of Lemma 6

Proof. We define the formulas by induction on ℓ. For the induction base $\ell = 0$ we put:

(1) $\mathsf{eq}_0^\times \overset{\text{def}}{=} \langle a_0 \rangle ((\alpha, b_0) \leftrightarrow (\beta, b_0))$.

(2) $\mathsf{less}_0^\times \overset{\text{def}}{=} \langle a_0 \rangle (\neg(\alpha, b_0) \wedge (\beta, b_0))$.

(3) $\mathsf{succ}_0^\times \overset{\text{def}}{=} \mathsf{less}_0^\times$.

For the induction step, we define:

(1) $\mathsf{eq}_{\ell+1}^\times \overset{\text{def}}{=} [a_{\ell+1}] \left(\mathsf{eq}_\ell^\times \longrightarrow ((\alpha, b_\ell) \leftrightarrow (\beta, b_\ell))\right)$.

(2) $\mathsf{less}_{\ell+1}^\times \overset{\text{def}}{=} \langle a_{\ell+1} \rangle \varphi_{\ell+1}^\times$, where

$$\varphi_{\ell+1}^\times \overset{\text{def}}{=} (\mathsf{eq}_\ell^\times \wedge \neg(\alpha, b_\ell) \wedge (\beta, b_\ell) \wedge [\Rightarrow](\mathsf{eq}_\ell^\times \rightarrow ((\alpha, b_\ell) \leftrightarrow (\beta, b_\ell))).$$

(3) $\mathsf{succ}_{\ell+1}^\times \overset{\text{def}}{=} \langle a_{\ell+1} \rangle (\varphi_{\ell+1}^\times \wedge [\Leftarrow](\alpha, b_\ell) \wedge [\Leftarrow] \neg(\beta, b_\ell))$.

◀

One-variable first-order linear temporal logics with counting

Christopher Hampson and Agi Kurucz

Department of Informatics
King's College London

Abstract

First-order temporal logics are notorious for their bad computational behaviour. It is known that even the two-variable monadic fragment is highly undecidable over various timelines. However, following the introduction of the monodic formulas (where temporal operators can be applied only to subformulas with at most one free variable), there has been a renewed interest in understanding extensions of the one-variable fragment and identifying those that are decidable. Here we analyse the one-variable fragment of temporal logic extended with counting (to two), interpreted in models with constant, decreasing, and expanding first-order domains. We show that over most classes of linear orders these logics are (sometimes highly) undecidable, even without constant and function symbols, and with the sole temporal operator 'eventually'. A more general result says that the bimodal logic of commuting linear and pseudo-equivalence relations is undecidable. The proofs are by reductions of various counter machine problems.

1998 ACM Subject Classification F.4.1 Mathematical Logic: Temporal logic, modal logic

Keywords and phrases modal and temporal logic, counting, decision procedures

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.348

1 Introduction and results

Though first-order temporal logics are natural and expressive languages for querying temporal databases [3, 4] and reasoning about knowledge that changes in time [15], their practical use has been discouraged by their high computational complexity. It is well-known that even the two-variable monadic fragment is undecidable over various timelines, and Π^1_1-hard over the natural numbers [27, 28, 19, 6, 7]. In contrast to classical first-order logic where the decision problems of its fragments were studied in great detail [2], there have been only a few attempts on finding well-behaved decidable fragments of first-order temporal logics, mostly by restricting the available quantifier patterns [4, 14, 15].

In this paper we contribute to this research line by studying the one-variable 'future' fragment of linear temporal logic with counting (to two), interpreted in models over various timelines, and having constant, decreasing, or expanding first-order domains. Our language FOLTL$^{\neq}$ is strong enough to express uniqueness of a property of domain elements ($\exists^{=1}x$), and the 'elsewhere' quantifier ($\forall^{\neq}x$). However, FOLTL$^{\neq}$ has no equality, no constant or function symbols, and its only temporal operators are 'eventually' and 'always in the future'. FOLTL$^{\neq}$ is weaker than the two-variable monadic *monodic* fragment with equality, where temporal operators can be applied only to subformulas with at most one free variable. (This fragment with the 'next time' operator is known to be Π^1_1-hard over the natural numbers [31, 5].) FOLTL$^{\neq}$ is connected to bimodal product logics [8, 7], and to the temporalisation of the expressive description logic $\mathcal{CQ}$ with one global universal role [30]. Here are some examples of FOLTL$^{\neq}$-formulas:

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 348–362

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- [4] "*An order can only be submitted once:*" $\forall x \Box_F \big(\mathsf{Subm}(x) \to \Box_F \neg \mathsf{Subm}(x)\big)$.
- The Barcan formula: $\exists x \Diamond_F \mathsf{P}(x) \leftrightarrow \Diamond_F \exists x \, \mathsf{P}(x)$.
- "*Every day has its unique dog:*" $\Box_F \exists^{=1} x \, \big(\mathsf{Dog}(x) \wedge \Box_F \neg \mathsf{Dog}(x)\big)$.
- "*It's only me who is always unlucky:*" $\forall^{\neq} x \, \Diamond_F \mathsf{Lucky}(x)$.

While the addition of 'elsewhere' quantifiers to the two-variable fragment of classical first-order logic does not increase the coNEXPTIME complexity of its validity problem [12, 21], we show that adding the same feature to the (decidable) one-variable fragment of first-order temporal logic results in (sometimes highly) undecidable logics over most timelines, not only in models with constant domains, but even those with decreasing and expanding first-order domains. Our main results on the FOLTL$^{\neq}$-validity problem are summarised in the following table:

	$\langle \omega, < \rangle$	all finite linear orders	all linear orders	all modally discrete linear orders
constant domains	Π^1_1-hard Theorem 1	Π^0_1-hard Theorem 2	undecidable, r.e. Theorems 5, 10	Π^1_1-hard Theorem 8
decreasing domains	Π^0_1-hard Theorem 2	Π^0_1-hard Theorem 2	undecidable, r.e. Theorems 7, 10	Π^1_1-hard Theorem 8
expanding domains	undecidable r.e.? Theorem 4	decidable Ackermann-hard Theorems 11, 3	decidable? r.e. Theorem 10	decidable? r.e.?

The structure of the paper is as follows. In Section 2 we provide the definitions of the studied logics. In Section 3 we introduce counter machines, and show several ways of simulating their behaviour in FOLTL$^{\neq}$. In Sections 4–6, with the help of the techniques developed in Section 3, we reduce various counter machine problems to FOLTL$^{\neq}$-satisfiability over different timelines and first-order domain settings. In Section 7 we discuss the connection between FOLTL$^{\neq}$ and propositional bimodal logics, and give a general undecidability result (Theorem 12). Finally, in Section 8 we formulate some open problems. Complete proofs will appear in the full paper.

2 Definitions and basic properties

We define FOLTL$^{\neq}$-*formulas* by the following grammar:

$$\phi :: \quad \mathsf{P}(x) \mid \neg\phi \mid \phi \wedge \psi \mid \exists x \, \phi \mid \exists^{\geq 2} x \, \phi \mid \Diamond_F \phi$$

where P ranges over an infinite set $\mathcal{P}$ of *monadic* predicate symbols. We use the usual abbreviations $\vee$, $\to$, $\leftrightarrow$, $\forall x$, $\Box_F$, and also $\Box_F^+ \phi := \phi \wedge \Box_F \phi$.

A FOLTL-*model* is a tuple $\mathfrak{M} = \big\langle \langle T, < \rangle, D_t, I \big\rangle_{t \in T}$, where $\langle T, < \rangle$ is a linear order[1], representing the *timeline*, D_t is a non-empty set, the *domain at moment* t, for each $t \in T$, and I is a function associating with every $t \in T$ a first-order structure $I(t) = \langle D_t, \mathsf{P}^{I(t)} \rangle_{\mathsf{P} \in \mathcal{P}}$. We say that $\mathfrak{M}$ is *based on* the linear order $\langle T, < \rangle$. $\mathfrak{M}$ is a *constant* (resp. *decreasing*, *expanding*) *domain model*, if $D_t = D_{t'}$, (resp. $D_t \supseteq D_{t'}$, $D_t \subseteq D_{t'}$) whenever $t, t' \in T$ and

[1] By a linear order we mean a *strict* one. This is for simplifying the presentation only. For reflexive orders, see the 'interval trick' in Section 5.

$t < t'$. A constant domain model is clearly both a decreasing and expanding domain model as well, and can be represented as a triple $\langle\langle T, <\rangle, D, I\rangle$.

The *truth-relation* $(\mathfrak{M}, t) \models^a \phi$ (or simply $t \models^a \phi$ if $\mathfrak{M}$ is understood) is defined, for all $t \in T$ and $a \in D_t$, by induction on ϕ as follows:

- $t \models^a \mathsf{P}(x)$ iff $a \in \mathsf{P}^{I(t)}$, $t \models^a \neg\phi$ iff $t \not\models^a \phi$, $t \models^a \phi \wedge \psi$ iff $t \models^a \phi$ and $t \models^a \psi$,
- $t \models^a \exists x\, \phi$ iff there exists $b \in D_t$ with $t \models^b \phi$,
- $t \models^a \exists^{\geq 2} x\, \phi$ iff there exist $b, b' \in D_t$ with $b \neq b'$, $t \models^b \phi$ and $t \models^{b'} \phi$,
- $t \models^a \Diamond_F \phi$ iff there is $t' \in T$ such that $t' > t$, $a \in D_{t'}$ and $t' \models^a \phi$.

We say that ϕ is *true* (*satisfiable*) *in* $\mathfrak{M}$, if $t \models^a \phi$ holds for all (some) $t \in T$ and $a \in D_t$. Given a class $\mathcal{C}$ of linear orders, we say that ϕ is FOLTL$^{\neq}$*-valid in constant* (*decreasing*, *expanding*) *domain models over* $\mathcal{C}$, if ϕ is true in every constant (decreasing, expanding) domain FOLTL-model based on some linear order from $\mathcal{C}$. It is not hard to see that for every class $\mathcal{C}$ of linear orders, FOLTL$^{\neq}$-validity in decreasing (expanding) domain models over $\mathcal{C}$ is reducible to FOLTL$^{\neq}$-validity in constant domain models over $\mathcal{C}$.

We introduce the following abbreviations:

$$\exists^{=1} x\, \phi :: \quad \exists x\, \phi \wedge \neg\exists^{\geq 2} x\, \phi, \qquad \exists^{\neq} x\, \phi :: \quad (\neg\phi \wedge \exists x\, \phi) \vee \exists^{\geq 2} x\, \phi, \qquad \forall^{\neq} x\, \phi :: \quad \neg\exists^{\neq} x\, \neg\phi.$$

It is straightforward to see that they have the intended semantics:

- $t \models^a \exists^{=1} x\, \phi$ iff there exists a unique $b \in D_t$ with $t \models^b \phi$,
- $t \models^a \forall^{\neq} x\, \phi$ iff $t \models^b \phi$, for every $b \in D_t$ with $b \neq a$.

Further, we could have chosen $\exists^{\neq} x$ as our primary connective instead of $\exists x$ and $\exists^{\geq 2} x$, as

$$\exists x\, \phi \leftrightarrow \phi \vee \exists^{\neq} x\, \phi \qquad \text{and} \qquad \exists^{\geq 2} x\, \phi \leftrightarrow \exists x\, (\phi \wedge \exists^{\neq} x\, \phi). \tag{1}$$

3 Encoding counter machines in FOLTL-models

A *Minsky* or *counter machine* M is described by a finite set Q of states, a set $H \subseteq Q$ of terminal states, a finite set $C = \{c_0, \ldots, c_{N-1}\}$ of counters, a finite nonempty set $I_q \subseteq Op_C \times Q$ of instructions, for each $q \in Q - H$, where each operation in Op_C is one of the following forms, for some $i < N$:

- c_i^{++} —*increment counter* c_i *by one*,
- c_i^{--} —*decrement counter* c_i *by one*,
- $c_i^{??}$ —*test whether counter* c_i *is empty*.

A *configuration* of M is a tuple $\langle q, \mathbf{c}\rangle$ with $q \in Q$ representing the current state, and an N-tuple $\mathbf{c} = \langle c_0, \ldots, c_{N-1}\rangle$ of natural numbers representing the current contents of the counters. We say that there is a (*reliable*) *step* between configurations $\sigma = \langle q, \mathbf{c}\rangle$ and $\sigma' = \langle q', \mathbf{c}'\rangle$ (written $\sigma \to \sigma'$) iff there is $i < N$ such that

- either $c_i' = c_i + 1$, $c_j' = c_j$ for $j \neq i$, $j < N$, and $\langle c_i^{++}, q'\rangle \in I_q$,
- or $c_i' = c_i - 1$, $c_j' = c_j$ for $j \neq i$, $j < N$, and $\langle c_i^{--}, q'\rangle \in I_q$,
- or $c_i' = c_i = 0$, $c_j' = c_j$ for $j < N$, and $\langle c_i^{??}, q'\rangle \in I_q$.

We write $\sigma \to_{lossy} \sigma'$ if there are configurations $\sigma^1 = \langle q, \mathbf{c}^1\rangle$ and $\sigma^2 = \langle q', \mathbf{c}^2\rangle$ such that $\sigma^1 \to \sigma^2$, $c_i \geq c_i^1$ and $c_i^2 \geq c_i'$ for every $i < N$. A sequence $\langle \sigma_n : n < B\rangle$ of configurations, with $0 < B \leq \omega$, is called a *run* (resp. *lossy run*), if $\sigma_{n-1} \to \sigma_n$ (resp. $\sigma_{n-1} \to_{lossy} \sigma_n$) holds for every $0 < n < B$.

FOLTL$^{\neq}$ does not have the 'next time' temporal operator. So in order to simulate the behaviour of counter machines in FOLTL-models, first we generate an infinite 'diagonal', using two monadic predicate symbols, N (for 'next') and S (for 'state'). The limited counting

capabilities of FOLTL$^{\neq}$ will be used in forcing the *uniqueness* of the diagonal. To this end, let $\mathsf{diag}^{dec}_{\infty}$ be the conjunction of the following formulas:

$$\mathsf{S}(x) \wedge \forall x\, \Box^+_F\big(\mathsf{S}(x) \to \exists^{\neq} x\, \mathsf{N}(x)\big),$$
$$\forall x\, \Box^+_F\big[\mathsf{N}(x) \to \big(\forall^{\neq} x\, \neg\mathsf{N}(x) \wedge \Diamond_F \mathsf{S}(x) \wedge \Box_F \Box_F \neg \mathsf{S}(x)\big)\big], \tag{2}$$
$$\forall x\, \Box^+_F\big[\mathsf{S}(x) \to \big(\forall^{\neq} x\, \neg\mathsf{S}(x) \wedge \Box_F \neg \mathsf{S}(x)\big)\big]. \tag{3}$$

The formula $\mathsf{diag}^{dec}_{\infty}$ forces a unique infinite diagonal not only in constant but also in decreasing domain models. A straightforward induction proves the following:

▶ **Lemma 1.** Suppose that $t_0 \models^{a_0} \mathsf{diag}^{dec}_{\infty}$ in some decreasing domain FOLTL-model $\big\langle \langle T, <\rangle, D_t, I\big\rangle_{t\in T}$. Then there are sequences $\langle t_n \in T : n < \omega\rangle$ and $\langle a_n \in D_{t_n} : n < \omega\rangle$ such that the following hold, for all $n < \omega$ and $a \in D_{t_n}$: if $n > 0$ then t_n is the immediate $<$-successor of t_{n-1}, $t_n \models^a \mathsf{S}(x)$ iff $a = a_n$, and $t_n \models^a \mathsf{N}(x)$ iff $a = a_{n+1}$.

Observe that in Lemma 1, if $\langle T, <\rangle$ is $\langle \omega, <\rangle$ and $t_0 = 0$, then $t_n = n$ for all $n < \omega$.

Constant domain models. We begin by showing how to encode runs that start with all-0 counters by going *forward* along the created diagonal. For each counter $i < N$, we take two fresh predicate symbols C^+_i and C^-_i that will be used to mark those domain points where M increments and decrements counter c_i at each moment of time. The actual content of counter c_i is represented by those domain points where $\mathsf{C}^+_i(x) \wedge \neg\mathsf{C}^-_i(x)$ holds. The following formula ensures that each point of our constant domain is used only once, and only previously incremented points can be decremented:

$$\bigwedge_{i<N} \forall x\, \Box^+_F\big[\big(\mathsf{C}^+_i(x) \to \Box_F \mathsf{C}^+_i(x)\big) \wedge \big(\mathsf{C}^-_i(x) \to \Box_F \mathsf{C}^-_i(x)\big) \wedge \big(\mathsf{C}^-_i(x) \to \mathsf{C}^+_i(x)\big)\big]. \tag{4}$$

For each $i < N$, the following formulas simulate the possible changes in the counters:

$$\mathsf{Fix}_i :: \ \forall x\, \big(\Box_F \mathsf{C}^+_i(x) \to \mathsf{C}^+_i(x)\big) \wedge \forall x\, \big(\Box_F \mathsf{C}^-_i(x) \to \mathsf{C}^-_i(x)\big),$$
$$\mathsf{Inc}_i :: \ \exists^{=1} x\, \big(\neg\mathsf{C}^+_i(x) \wedge \Box_F \mathsf{C}^+_i(x)\big) \wedge \forall x\, \big(\Box_F \mathsf{C}^-_i(x) \to \mathsf{C}^-_i(x)\big),$$
$$\mathsf{Dec}_i :: \ \exists^{=1} x\, \big(\mathsf{C}^+_i(x) \wedge \neg\mathsf{C}^-_i(x) \wedge \Box_F \mathsf{C}^-_i(x)\big) \wedge \forall x\, \big(\Box_F \mathsf{C}^+_i(x) \to \mathsf{C}^+_i(x)\big).$$

Using these formulas, we can encode the steps of M. For each $\iota \in Op_C$, we define the formula do_ι by taking

$$\mathsf{do}_\iota :: \begin{cases} \mathsf{Inc}_i \wedge \bigwedge\limits_{i\neq j<N} \mathsf{Fix}_j, & \text{if } \iota = c_i^{++}, \\ \mathsf{Dec}_i \wedge \bigwedge\limits_{i\neq j<N} \mathsf{Fix}_j, & \text{if } \iota = c_i^{--}, \\ \forall x\, \big(\mathsf{C}^+_i(x) \to \mathsf{C}^-_i(x)\big) \wedge \bigwedge\limits_{j<N} \mathsf{Fix}_j, & \text{if } \iota = c_i^{??}. \end{cases}$$

Now we can encode runs that start with all-0 counters. For each $q \in Q$, we take a fresh predicate symbol S_q, and define φ_M to be the conjunction of (4) and the following formulas:

$$\bigwedge_{i<N} \forall x\, \big(\neg\mathsf{C}^+_i(x) \wedge \neg\mathsf{C}^-_i(x)\big),$$
$$\forall x\, \Box^+_F\big[\mathsf{S}(x) \leftrightarrow \bigvee_{q\in Q} \big(\mathsf{S}_q(x) \wedge \bigwedge_{q\neq q'\in Q} \neg\mathsf{S}_{q'}(x)\big)\big], \tag{5}$$
$$\forall x\, \Box^+_F \bigwedge_{q\in Q-H} \Big[\mathsf{S}_q(x) \to \bigvee_{\langle \iota, q'\rangle \in I_q} \Big(\mathsf{do}_\iota \wedge \forall x\, \big[\mathsf{N}(x) \to \Box_F\big(\mathsf{S}(x) \to \mathsf{S}_{q'}(x)\big)\big]\Big)\Big].$$

The following lemma says that in constant domain models, going forward along the diagonal points generated in Lemma 1, we can force (finite or infinite) runs of M:

▶ **Lemma 2.** Suppose $t_0 \models^{a_0} \mathsf{diag}^{dec}_{\infty} \wedge \varphi_M$ in some constant domain FOLTL-model $\langle\langle T, <\rangle, D, I\rangle$. For all $n < \omega$ and $i < N$, let

$$q_n := q, \text{ if } t_n \models^{a_n} \mathsf{S}_q(x), \qquad c_i(n) := |\{a \in D : t_n \models^{a} \mathsf{C}^+_i(x) \wedge \neg\mathsf{C}^-_i(x)\}|.$$

Then $\langle\langle q_n, \mathbf{c}(n)\rangle : n < B\rangle$ is a well-defined run of M starting with all-0 counters, whenever $0 < B \leq \omega$ is such that $t_n \models^{a_n} \bigwedge_{h\in H} \neg\mathsf{S}_h(x)$, for every $n < B$.

Decreasing domain models. We can also encode runs that start with all-0 counters by going *backward* along the diagonal. Moreover, this way we have more control over the points representing the content of the counters, and so we can simulate runs not only in constant but also in decreasing domain models.

We take a fresh predicate symbol start, intended to mark the start of runs and being constant along each first-order domain. In decreasing domain models we can say this by

$$\forall x\, \Box^+_F\big(\mathsf{start}(x) \to \forall x\, \mathsf{start}(x)\big). \tag{6}$$

For each counter $i < N$, we take a fresh predicate symbol C_i. The actual content of counter c_i will be represented by those points where $\mathsf{C}_i(x)$ holds. We want to force these points only to be among the domain points a_n generated in Lemma 1. We can achieve this by the following formulas, simulating the possible changes in the counters:

$$\mathsf{AllC}_i(x) :: \Diamond_F\mathsf{N}(x) \wedge \Box_F\big(\mathsf{N}(x) \vee \Diamond_F\mathsf{N}(x) \to \mathsf{C}_i(x)\big),$$

$$\mathsf{Fix}^{bw}_i :: \forall x\, \big(\mathsf{C}_i(x) \leftrightarrow \mathsf{AllC}_i(x)\big), \tag{7}$$

$$\mathsf{Inc}^{bw}_i :: \forall x\, \big[\mathsf{C}_i(x) \leftrightarrow \big(\mathsf{N}(x) \vee \mathsf{AllC}_i(x)\big)\big], \tag{8}$$

$$\mathsf{Dec}^{bw}_i :: \forall x\, \big(\mathsf{C}_i(x) \to \mathsf{AllC}_i(x)\big) \wedge \exists^{=1} x\, \big(\neg\mathsf{C}_i(x) \wedge \mathsf{AllC}_i(x)\big). \tag{9}$$

Next, we encode the steps of M, going backward along the diagonal. For every $\iota \in Op_C$, we define the formula do^{bw}_ι by taking

$$\mathsf{do}^{bw}_\iota :: \begin{cases} \mathsf{Inc}^{bw}_i \wedge \bigwedge\limits_{i\neq j<N} \mathsf{Fix}^{bw}_j, & \text{if } \iota = c_i^{++}, \\ \mathsf{Dec}^{bw}_i \wedge \bigwedge\limits_{i\neq j<N} \mathsf{Fix}^{bw}_j, & \text{if } \iota = c_i^{--}, \\ \forall x\, \neg\mathsf{C}_i(x) \wedge \bigwedge\limits_{j<N} \mathsf{Fix}^{bw}_j, & \text{if } \iota = c_i^{??}. \end{cases}$$

Finally, it remains to encode runs that start with all-0 counters, by going backward along the diagonal. Given a counter machine M, we define $\varphi^{bw\text{-}dec}_M$ to be the conjunction of (5), (6), and the following formulas:

$$\forall x\, \Box^+_F\big(\mathsf{S}(x) \wedge \mathsf{start}(x) \to \bigwedge_{i<N} \forall x\, \neg\mathsf{C}_i(x)\big),$$

$$\forall x\, \Box^+_F \bigwedge_{q\in Q-H} \Big[\Big(\mathsf{S}(x) \wedge \neg\mathsf{start}(x) \wedge \exists x\, \big(\mathsf{N}(x) \wedge \Diamond_F\mathsf{S}_q(x)\big)\Big) \to \bigvee_{\langle\iota,q'\rangle\in I_q} \big(\mathsf{do}^{bw}_\iota \wedge \mathsf{S}_{q'}(x)\big)\Big].$$

The next lemma says that in decreasing domain models, starting at a 'start' point and going backward along the diagonal generated in Lemma 1, we can force *finite* runs of M:

▶ **Lemma 3.** Suppose $t_0 \models^{a_0} \mathsf{diag}^{dec}_{\infty} \wedge \varphi^{bw\text{-}dec}_M$ in some decreasing domain FOLTL-model $\langle\langle T, <\rangle, D_t, I\rangle_{t\in T}$. For all $n < \omega$ and $i < N$, let

$$q_n := q, \text{ if } t_n \models^{a_n} \mathsf{S}_q(x), \qquad c_i(n) := |\{a \in D_{t_n} : t_n \models^{a} \mathsf{C}_i(x)\}|, \qquad \sigma_n := \langle q_n, \mathbf{c}(n)\rangle. \tag{10}$$

Then $\langle \sigma_{n_2}, \sigma_{n_2-1}, \dots \sigma_{n_1} \rangle$ is a well-defined run of M starting with all-0 counters, whenever $n_1 \leq n_2 < \omega$ is such that $t_{n_2} \models^{a_{n_2}} \mathsf{start}(x)$ and $t_n \models^{a_n} \neg\mathsf{start}(x) \wedge \bigwedge_{h \in H} \neg \mathsf{S}_h(x)$, for every n with $n_1 \leq n < n_2$.

Expanding domain models. We can still say something about counter machine runs by going backward along the diagonal in expanding domain models. However, in this case some of the content of the counters might get lost as the runs progress, so we can force only *lossy* runs. To this end, let $\mathsf{diag}_\infty^{exp}$ and $\varphi_M^{bw\text{-}exp}$ be obtained from $\mathsf{diag}_\infty^{dec}$ and $\varphi_M^{bw\text{-}dec}$, respectively, by simultaneously replacing all occurrences of the prefix $\forall x \, \Box_F^+$ with $\Box_F^+ \forall x$. Then let φ_M^{lossy} be obtained from $\varphi_M^{bw\text{-}exp}$ by replacing all occurrences of the formulas (7)–(9) by their 'lossy versions':

$$\mathsf{Fix}_i^{bw\text{-}lossy} :: \forall x \, \big(\mathsf{C}_i(x) \to \mathsf{AllC}_i(x)\big),$$
$$\mathsf{Inc}_i^{bw\text{-}lossy} :: \forall x \, \big[\mathsf{C}_i(x) \to \big(\mathsf{N}(x) \vee \mathsf{AllC}_i(x)\big)\big],$$
$$\mathsf{Dec}_i^{bw\text{-}lossy} :: \forall x \, \big(\mathsf{C}_i(x) \to \mathsf{AllC}_i(x)\big) \wedge \exists x \, \big(\neg \mathsf{C}_i(x) \wedge \mathsf{AllC}_i(x)\big).$$

Then we have the expanding domain version of Lemma 1 for $\mathsf{diag}_\infty^{exp}$, and the following 'lossy analogue' of Lemma 3:

▶ **Lemma 4.** Suppose $t_0 \models^{a_0} \mathsf{diag}_\infty^{exp} \wedge \varphi_M^{lossy}$ in an expanding domain FOLTL-model $\big\langle \langle T, < \rangle, D_t, I \big\rangle_{t \in T}$. Then $\langle \sigma_{n_2}, \sigma_{n_2-1}, \dots \sigma_{n_1} \rangle$, as defined in (10), is a well-defined lossy run of M starting with all-0 counters, whenever $n_1 \leq n_2 < \omega$ is such that $t_{n_2} \models^{a_{n_2}} \mathsf{start}(x)$ and $t_n \models^{a_n} \neg\mathsf{start}(x) \wedge \bigwedge_{h \in H} \neg \mathsf{S}_h(x)$, for every n with $n_1 \leq n < n_2$.

Observe that in this case the counting capabilities of FOLTL$^{\neq}$ are only used in forcing the uniqueness of the diagonal in the expanding domain version of Lemma 1.

4 FOLTL$^{\neq}$ over $\langle \omega, < \rangle$ and finite linear orders

▶ **Theorem 1.** FOLTL$^{\neq}$*-validity is* Π_1^1*-hard in constant domain models over* $\langle \omega, < \rangle$.

We prove this theorem by reducing the following Σ_1^1-complete [1] problem to FOLTL$^{\neq}$-satisfiability in constant domain models over $\langle \omega, < \rangle$:

CM recurrence:

Given a counter machine M and two states q_0, q_r, is there a run starting from $\langle q_0, \mathbf{0} \rangle$ and visiting q_r infinitely often?

The following claim is a straightforward consequence of Lemma 2:

▶ **Claim 1.1.** Suppose $\mathsf{diag}_\infty^{dec} \wedge \varphi_M \wedge \mathsf{S}_{q_0}(x) \wedge \forall x \, \Box_F^+ \bigwedge_{h \in H} \neg \mathsf{S}_h(x)$ is satisfiable in some constant domain FOLTL-model. Then M has an infinite run starting from $\langle q_0, \mathbf{0} \rangle$.

Now it clearly follows from Claim 1.1 that if

$$\mathsf{diag}_\infty^{dec} \wedge \varphi_M \wedge \mathsf{S}_{q_0}(x) \wedge \forall x \, \Box_F^+ \bigwedge_{h \in H} \neg \mathsf{S}_h(x) \wedge \Box_F \Diamond_F \exists x \, \mathsf{S}_{q_r}(x) \tag{11}$$

is satisfiable in some constant domain FOLTL-model based on $\langle \omega, < \rangle$, then M has a run starting from $\langle q_0, \mathbf{0} \rangle$ and visiting q_r infinitely often. (Observe that this is not necessarily true for models based on arbitrary timelines.)

On the other hand, if M has a run $\big\langle \langle q_n, \mathbf{c}(n) \rangle : n < \omega \big\rangle$ that visits q_r infinitely often and $\mathbf{c}(0) = \mathbf{0}$, then we define a constant domain FOLTL-model $\mathfrak{M}_\infty^{fw} = \big\langle \langle \omega, < \rangle, \omega, I \big\rangle$ as follows. For all $n < \omega$ and $q \in Q$, let

$$\mathsf{S}^{I(n)} := \{n\}, \quad \mathsf{N}^{I(n)} := \{n+1\} \quad \text{and} \quad \mathsf{S}_q^{I(n)} := \begin{cases} \{n\}, & \text{if } q = q_n, \\ \emptyset, & \text{otherwise.} \end{cases} \tag{12}$$

Further, for all $i < N$ and $n < \omega$, we define inductively the sets $\mathsf{C}_i^{+I(n)}$ and $\mathsf{C}_i^{-I(n)}$. We let $\mathsf{C}_i^{+I(0)} = \mathsf{C}_i^{-I(0)} := \emptyset$, and then

$$\mathsf{C}_i^{+I(n+1)} := \begin{cases} \mathsf{C}_i^{+I(n)} \cup \{n\}, & \text{if } c_i(n+1) = c_i(n) + 1, \\ \mathsf{C}_i^{+I(n)}, & \text{otherwise.} \end{cases} \tag{13}$$

$$\mathsf{C}_i^{-I(n+1)} := \begin{cases} \mathsf{C}_i^{-I(n)} \cup \{min(\mathsf{C}_i^{+I(n)})\}, & \text{if } c_i(n+1) = c_i(n) - 1, \\ \mathsf{C}_i^{-I(n)}, & \text{otherwise.} \end{cases} \tag{14}$$

Then it is easy to check that $(\mathfrak{M}_\infty^{fw}, 0) \models^0$ (11), proving Theorem 1.

▶ **Theorem 2.** FOLTL$^{\neq}$*-validity is* Π_1^0*-hard*
1. *in decreasing domain models over* $\langle \omega, < \rangle$,
2. *in decreasing domain models over the class of all finite linear orders,*
3. *in models with finite decreasing domains over* $\langle \omega, < \rangle$.

We prove the theorem using a reduction of the following Σ_1^0-complete [20] problem:

CM reachability:

Given a counter machine M and two states q_0, q_r, is there a run from $\langle q_0, \mathbf{0} \rangle$ to some configuration $\langle q_r, \mathbf{c} \rangle$?

In order to prove **1**, define the formula reach^{dec} by taking

$$\mathsf{reach}^{dec} :: \quad \mathsf{S}_{q_r}(x) \wedge \forall x \, \Box_F^+ \big[\exists x \, \Diamond_F \mathsf{start}(x) \to \big(\neg \mathsf{start}(x) \wedge \forall x \bigwedge_{h \in H} \neg \mathsf{S}_h(x) \big) \big] \wedge \forall x \, \Box_F^+ \big(\mathsf{S}(x) \wedge \mathsf{start}(x) \to \mathsf{S}_{q_0}(x) \big).$$

The following claim is a consequence of Lemma 3:

▶ **Claim 2.1.** Suppose $\mathsf{diag}_\infty^{dec} \wedge \varphi_M^{bw\text{-}dec} \wedge \mathsf{reach}^{dec} \wedge \exists x \, \big(\mathsf{start}(x) \vee \Diamond_F \mathsf{start}(x) \big)$ is satisfiable in some decreasing domain FOLTL-model based on $\langle \omega, < \rangle$. Then there is a run of M starting with $\langle q_0, \mathbf{0} \rangle$ and reaching q_r.

On the other hand, if M has a run $\big\langle \langle q_n, \mathbf{c}(n) \rangle : n \leq K \big\rangle$ with $\mathbf{c}(0) = \mathbf{0}$ and $q_K = q_r$, then we define a constant domain FOLTL-model $\big\langle \langle \omega, < \rangle, \omega, I \big\rangle$ as follows. For all $n < \omega$, $q \in Q$,

$$\mathsf{S}^{I(n)} := \{n\}, \quad \mathsf{N}^{I(n)} := \{n+1\} \text{ and } \mathsf{S}_q^{I(n)} := \begin{cases} \{n\}, & \text{if } n \leq K \text{ and } q = q_{K-n}, \\ \{n\}, & \text{if } n > K \text{ and } q = q_h, \\ \emptyset, & \text{otherwise,} \end{cases} \tag{15}$$

for some fixed $h \in H$. Further, for all $i < N$ and $n < \omega$, we define inductively the sets $\mathsf{C}_i^{I(n)}$. We let $\mathsf{C}_i^{I(n)} := \emptyset$ whenever $n \geq K$, and then for every $n < K$, we let

$$\mathsf{C}_i^{I(K-n-1)} := \begin{cases} \mathsf{C}_i^{I(K-n)} \cup \{K-n\}, & \text{if } c_i(n+1) = c_i(n) + 1, \\ \mathsf{C}_i^{I(K-n)} - \{min(\mathsf{C}_i^{I(K-n)})\}, & \text{if } c_i(n+1) = c_i(n) - 1, \\ \mathsf{C}_i^{I(K-n)}, & \text{otherwise.} \end{cases} \tag{16}$$

Finally, let $\mathsf{start}^{I(K)} := \omega$, and $\mathsf{start}^{I(n)} := \emptyset$ for all $n \neq K$, $n < \omega$. Then it is not hard to check that $\mathsf{diag}_\infty^{dec} \wedge \varphi_M^{bw\text{-}dec} \wedge \mathsf{reach}^{dec} \wedge \exists x \, \big(\mathsf{start}(x) \vee \Diamond_F \mathsf{start}(x) \big)$ is satisfiable in this model.

However, $\mathsf{diag}_\infty^{dec}$ is clearly not satisfiable in models based on finite timelines, or in models with finite domains. Let $\mathsf{diag}_{fin}^{dec}$ be obtained from $\mathsf{diag}_\infty^{dec}$ by replacing the conjunct (2) with

$$\forall x \, \Box_F^+ \big(\mathsf{N}(x) \to \forall^{\neq} x \, \neg \mathsf{N}(x) \big) \wedge \forall x \, \Box_F^+ \big[\mathsf{N}(x) \wedge \neg \mathsf{start}(x) \to \big(\Diamond_F \mathsf{S}(x) \wedge \Box_F \Box_F \neg \mathsf{S}(x) \big) \big].$$

Now it is easy to see that $\mathsf{diag}^{dec}_{fin} \wedge \varphi^{bw\text{-}dec}_M \wedge \mathsf{reach}^{dec}$ is satisfiable in a decreasing domain FOLTL-model where either its timeline or all its domains are finite iff M has a run starting with $\langle q_0, \mathbf{0}\rangle$ and reaching q_r, completing the proof of Theorem 2.

▶ **Theorem 3.** FOLTL$^{\neq}$*-validity is Ackermann-hard in expanding domain models over the class of all finite linear orders.*

The decidability (and the finite expanding domain property) of this logic follows from its reducibility to certain propositional bimodal logics, see Theorem 11 in Section 7. Here we prove the lower bound in Theorem 3 by a reduction the following problem:

LCM reachability:

Given a counter machine M, a configuration $\sigma_0 = \langle q_0, \mathbf{0}\rangle$, and a state q_r, is there a lossy run from σ_0 to some configuration $\langle q_r, \mathbf{c}\rangle$?

It is shown in [24] that this problem, without the restriction that σ_0 has all-0 counters, is Ackermann-hard. It is not hard to see that this restriction does not matter: For every M and σ_0 one can define a machine M^{σ_0} that first performs incrementation steps filling the counters up to their 'σ_0-level', and then performs M's actions. Then M has a lossy run from σ_0 reaching q_r iff M^{σ_0} has a lossy run starting with all-0 counters and reaching q_r.

Let $\mathsf{diag}^{exp}_{fin}$ and reach^{exp} obtained from $\mathsf{diag}^{dec}_{fin}$ and reach^{dec}, respectively, by replacing all occurrences of the prefix $\forall x\, \Box^+_F$ with $\Box^+_F \forall x$. The next claim is a consequence of Lemma 4:

▶ **Claim 3.1.** If $\mathsf{diag}^{exp}_{fin} \wedge \varphi^{lossy}_M \wedge \mathsf{reach}^{exp}$ is satisfiable in an expanding domain FOLTL-model based on a finite linear order, then M has a lossy run starting with $\langle q_0, \mathbf{0}\rangle$ and reaching q_r.

On the other hand, suppose M has a lossy run $\big\langle \langle q_n, \mathbf{c}(n)\rangle : n \leq K\big\rangle$ with $\mathbf{c}(0) = \mathbf{0}$ and $q_K = q_r$. Then we can define a constant domain FOLTL-model $\big\langle \langle T, <\rangle, D, I\big\rangle$ that satisfies $\mathsf{diag}^{exp}_{fin} \wedge \varphi^{lossy}_M \wedge \mathsf{reach}^{exp}$ by taking $T = D = \{0, \dots, K\}$, the restriction of (15), $\mathsf{start}^{I(K)} := D$, $\mathsf{start}^{I(n)} := \emptyset$ for all $n < K$, $\mathsf{C}_i^{I(K)} := \emptyset$ for all $i < N$, and the following in place of (16), for all $i < N$ and $n < K$:

$$\mathsf{C}_i^{I(K-n-1)} := \begin{cases} \mathsf{C}_i^{I(K-n)} \cup \{K-n\}, & \text{if } c_i(n+1) = c_i(n) + 1, \\ \text{any subset of } \mathsf{C}_i^{I(K-n)} \text{ of size } c_i(n+1), & \text{if } c_i(n+1) \leq c_i(n). \end{cases}$$

This completes the proof of Theorem 3.

▶ **Theorem 4.** FOLTL$^{\neq}$*-validity is undecidable in expanding domain models over* $\langle \omega, <\rangle$.

We prove the theorem by a reduction of the following Π^0_1-complete problem [16, 18, 23] to the FOLTL$^{\neq}$-satisfiability problem in question:

LCM ω-reachability:

Given a counter machine M, a configuration $\sigma_0 = \langle q_0, \mathbf{0}\rangle$ and a state q_r, is it the case that for every $n < \omega$ M has a lossy run starting with σ_0 and visiting q_r at least n times?

(The idea of our reduction is similar to the one used in [16] for a formalism more expressive than FOLTL$^{\neq}$.) Take a fresh predicate symbol R, and define rec^{fw} as the conjunction of the following formulas:

$$\Box_F \Diamond_F \mathsf{start}(x) \wedge \Box_F \forall x \big(\mathsf{start}(x) \to \forall x\, \mathsf{start}(x)\big),$$
$$\Box_F \forall x \Big(\mathsf{start}(x) \to \exists x \big[\mathsf{R}(x) \wedge \Diamond_F \mathsf{S}(x) \wedge \Box_F \big(\Diamond_F \mathsf{S}(x) \to \neg \mathsf{start}(x)\big)\big]\Big),$$
$$\Box_F \forall x \big[\mathsf{R}(x) \to \Box_F \big(\mathsf{S}(x) \to \mathsf{S}_{q_r}(x)\big)\big],$$

$$\Box_F \forall x \Big[\mathsf{S}_{q_r}(x) \to \exists x \Big(\mathsf{R}(x) \wedge \Diamond_F \big(\mathsf{start}(x) \wedge \Diamond_F \mathsf{S}(x) \big) \wedge \Box_F \big[\mathsf{start}(x) \wedge \Diamond_F \mathsf{S}(x) \to \Box_F \big(\Diamond_F \mathsf{S}(x) \to \neg \mathsf{start}(x) \big) \big] \Big) \Big],$$

$$\Box_F^+ \forall x \, \big(\mathsf{R}(x) \to \Box_F \neg \mathsf{R}(x) \big).$$

▶ **Claim 4.1.** If $\mathsf{diag}_\infty^{exp} \wedge \mathsf{rec}^{fw}$ is satisfiable in some expanding domain FOLTL-model based on $\langle \omega, < \rangle$, then there is an infinite sequence $\langle k_n < \omega : n < \omega \rangle$ such that, for all $n < \omega$, $k_n \models \forall x \, \mathsf{start}(x)$, and if $n > 0$ then $|\{k : k_{n-1} < k \leq k_n \text{ and } k \models^{a_k} \mathsf{S}_{q_r}(x)\}| \geq n$.

(Observe that Claim 4.1 is not necessarily true for models based on arbitrary timelines.) Now the following claim is a consequence of Claim 4.1 and Lemma 4:

▶ **Claim 4.2.** Suppose that the formula

$$\mathsf{diag}_\infty^{exp} \wedge \varphi_M^{lossy} \wedge \mathsf{rec}^{fw} \wedge \Box_F \forall x \, \big(\mathsf{S}(x) \wedge \mathsf{start}(x) \to \mathsf{S}_{q_0}(x) \big) \wedge \Box_F \forall x \bigwedge_{h \in H} \neg \mathsf{S}_h(x) \tag{17}$$

is satisfiable in an expanding domain FOLTL-model based on $\langle \omega, < \rangle$. Then, for every $n < \omega$, M has a lossy run starting with $\langle q_0, \mathbf{0} \rangle$ and visiting q_r at least n times.

On the other hand, if for every $n < \omega$, M has a lossy run ρ_n of M starting with $\langle q_0, \mathbf{0} \rangle$ and visiting q_r at least n times, then (17) is satisfiable in the constant domain FOLTL-model sketched in Fig. 1, completing the proof of Theorem 4.

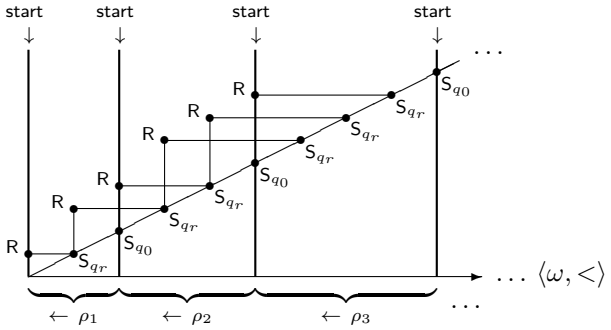


Figure 1 Sketch of a constant domain FOLTL-model based on $\langle \omega, < \rangle$ satisfying the formula (17).

5 FOLTL$^{\neq}$ over arbitrary linear orders

As FOLTL$^{\neq}$-validity in constant (expanding, decreasing) domain models over the class of all linear orders is recursively enumerable (see Theorem 10), we cannot expect 'CM reachability' or 'CM recurrence' to be reduced to its satisfiability problem. However, in the constant domain case at least, we can still reduce the following undecidable [20] problem:

CM non-termination:

Given a counter machine M and a state q_0, is there an infinite run starting from $\langle q_0, \mathbf{0} \rangle$?

Then Claim 1.1 and a FOLTL-model defined as in (12)–(14) give us the following:

▶ **Theorem 5.** FOLTL$^{\neq}$*-validity is undecidable in constant domain models over the class of all linear orders.*

Observe that a formula of the form $\Diamond_F \mathsf{S}(x) \wedge \Box_F \Box_F \neg \mathsf{S}(x)$ is clearly not satisfiable in any FOLTL-model based on a dense linear order, and so our formulas generating diagonals are not satisfiable in such a model either. However, below we show that the formulas used in Sections 3 and 4 can be modified to prove the following generalisation of Theorem 5:

▶ **Theorem 6.** FOLTL$^{\neq}$*-validity is undecidable in constant domain models over any class of linear orders containing a linear order that has an infinite ascending chain.*

We use a version of the 'interval trick' suggested in [22, 26, 9]. Take a fresh predicate symbol Tick and define a new temporal operator $\blacklozenge_F$ and its dual $\blacksquare_F$ by setting, for any FOLTL$^{\neq}$-formula $\psi(x)$,

$$\blacklozenge_F\psi(x) :: \big[\mathsf{Tick}(x) \to \Diamond_F\big(\neg\mathsf{Tick}(x) \wedge (\psi(x) \vee \Diamond_F\psi(x))\big)\big] \\ \wedge \big[\neg\mathsf{Tick}(x) \to \Diamond_F\big(\mathsf{Tick}(x) \wedge (\psi(x) \vee \Diamond_F\psi(x))\big)\big].$$

In order to properly simulate 'next time', we need the following property of $\mathsf{Tick}(x)$:

$$\big(\exists x\,\mathsf{Tick}(x) \leftrightarrow \forall x\,\mathsf{Tick}(x)\big) \wedge \Box_F\big(\exists x\,\mathsf{Tick}(x) \leftrightarrow \forall x\,\mathsf{Tick}(x)\big). \tag{18}$$

Suppose that $r \models (18)$ in some constant domain FOLTL-model $\big\langle\langle T, <\rangle, D, I\big\rangle$. We define a new relation $\prec$ on $T_r = \{t \in T : r < t\}$ by taking, for all $t, t' \in T_r$,

$$t \prec t' \quad \text{iff} \quad \exists z\,\big[t < z \le t' \text{ and, for all } a \in D,\ \big(t \models^a \mathsf{Tick}(x) \ \leftrightarrow\ z \models^a \neg\mathsf{Tick}(x)\big)\big].$$

It is straightforward to check that for all $t \in T_r$ and $a \in D$, $t \models^a \blacklozenge_F\psi$ iff there is $t' \in T_r$ with $t \prec t'$ and $t' \models^a \psi$. Also, $\prec$ is transitive and asymmetric, but $\langle T_r, \prec\rangle$ is not necessarily a linear order. Instead of trichotomy, we only have that either $t \prec t'$ or $t' \prec t$ or $t \sim t'$ hold, where $\sim$ is the following equivalence relation on T_r: $t \sim t'$ iff for all z with $\mathit{min}(t, t') \le z \le \mathit{max}(t, t')$ and all $a \in D$, $\big(z \models^a \mathsf{Tick}(x) \ \leftrightarrow\ \mathit{min}(t, t') \models^a \mathsf{Tick}(x)\big)$. We would like our predicates to be constant in any $\sim$-class. To achieve this, for a predicate symbol P, let $\mathsf{interval}_\mathsf{P}$ denote conjunction of (18) and the following formulas:

$$\forall x\,\Box_F\big(\mathsf{P}(x) \to \blacksquare_F\neg\mathsf{P}(x)\big),$$
$$\forall x\,\Box_F\big(\Diamond_F\mathsf{P}(x) \wedge \blacksquare_F\neg\mathsf{P}(x) \to \mathsf{P}(x)\big),$$
$$\forall x\,\Box_F\big(\mathsf{P}(x) \wedge \neg\blacklozenge_F\top(x) \to \Box_F\mathsf{P}(x)\big),$$
$$\forall x\,\Box_F\big(\mathsf{P}(x) \wedge \blacklozenge_F\top(x) \to \blacklozenge_F\mathsf{P}_{\mathit{next}}(x)\big),$$
$$\forall x\,\Box_F\big(\mathsf{P}(x) \to \Box_F(\blacklozenge_F\mathsf{P}_{\mathit{next}}(x) \to \mathsf{P}(x))\big),$$

where $\mathsf{P}_{\mathit{next}}$ is a fresh predicate symbol, and $\top(x)$ is a shorthand for $\mathsf{P}(x) \vee \neg\mathsf{P}(x)$.

▶ **Claim 6.1.** Suppose that $r \models \mathsf{interval}_\mathsf{P}$, and take $t, t' \in T_r$ with $t < t'$ and $t \sim t'$. Then, for all $a \in D$, $t \models^a \mathsf{P}(x)$ iff $t' \models^a \mathsf{P}(x)$.

Now we have the following generalisation of Claim 1.1:

▶ **Claim 6.2.** Let ϕ_M be obtained from $\mathsf{diag}^{dec}_\infty \wedge \varphi_M \wedge \mathsf{S}_{q_0}(x) \wedge \forall x\,\Box^+_F \bigwedge_{h\in H}\neg\mathsf{S}_h(x)$ by replacing each occurrence of $\Diamond_F$ and $\Box_F$ with $\blacklozenge_F$ and $\blacksquare_F$, and adding the conjuncts $\mathsf{interval}_\mathsf{P}$ for each occurring predicate symbol P. If $t_0 \models^{a_0} \phi_M$ in some constant domain FOLTL-model based on a linear order having an infinite ascending chain starting at t_0, then M has an infinite run starting from $\langle q_0, \mathbf{0}\rangle$.

For the other direction, suppose that M has an infinite run starting from $\langle q_0, \mathbf{0}\rangle$. Let $\langle T, <\rangle$ be a linear order in our class containing an infinite ascending chain $t_0 < \cdots < t_n < \ldots$. Take the constant domain FOLTL-model $\mathfrak{M}^{fw}_\infty$ defined in (12)–(14). We define a constant domain model $\mathfrak{N}^{fw}_\infty = \big\langle\langle T, <\rangle, \omega, J\big\rangle$ by taking, for all $t \in T$ and $\mathsf{P} \in \{\mathsf{N}, \mathsf{S}, \mathsf{C}^+_i, \mathsf{C}^-_i, \mathsf{S}_q\}_{i<N,\, q\in Q}$,

$$\mathsf{Tick}^{J(t)} = \begin{cases} \omega, & \text{if } t_{n+1} < t \le t_n, \text{ } n \text{ is even,} \\ \emptyset, & \text{otherwise,} \end{cases} \qquad \mathsf{P}^{J(t)} = \begin{cases} \mathsf{P}^{I(n)}, & \text{if } t_{n+1} < t \le t_n, \\ \emptyset, & \text{otherwise,} \end{cases}$$

$$\mathsf{P}^{J(t)}_{\mathit{next}} = \begin{cases} \mathsf{P}^{I(n)}, & \text{if either } n > 0 \text{ and } t_n < t \le t_{n-1}, \text{ or } n = 0 \text{ and } t > t_0, \\ \emptyset, & \text{otherwise.} \end{cases}$$

Then $(\mathfrak{N}^{fw}_\infty, t_0) \models^0 \phi_M$, completing the proof of Theorem 6.

6 FOLTL$^{\neq}$ over timelines with infinite descending chains

We say that a linear order $\langle T, < \rangle$ *has a rooted infinite descending chain* if there exists $\langle t_n \in T : n \leq \omega \rangle$ with $t_\omega < \cdots < t_n < \cdots < t_0$. In this section we show that, in FOLTL-models based on such linear orders, we can also simulate counter machine runs along a diagonal that is *generated* backward. Let $\mathsf{diag}_\infty^{bw\text{-}dec}$ be the conjunction of (2)–(3) and the following formulas:

$$\Diamond_F\big(\mathsf{S}(x) \wedge \mathsf{start}(x) \wedge \forall x\, \Box_F \neg \mathsf{S}(x)\big),$$
$$\forall x\, \Diamond_F \mathsf{N}(x),$$
$$\forall x\, \Box_F\big(\mathsf{N}(x) \to \exists x\, \mathsf{S}(x)\big),$$

and recall the formula $\varphi_M^{bw\text{-}dec}$ from Section 3. Then we have the following analogues of Lemmas 1 and 3:

▶ **Lemma 5.** Suppose that $r \models^{\alpha_0} \mathsf{diag}_\infty^{bw\text{-}dec}$ in some decreasing domain FOLTL-model $\big\langle \langle T, < \rangle, D_t, I \big\rangle_{t \in T}$. Then there are sequences $\langle \tau_n \in T : n < \omega \rangle$ and $\langle \alpha_n \in D_{\tau_n} : n < \omega \rangle$ such that $\tau_0 \models^{\alpha_0} \mathsf{start}(x)$, $t \models^a \neg \mathsf{S}(x)$ for all $t > \tau_0$ and $a \in D_t$, and the following hold, for all $n < \omega$ and $a \in D_{\tau_n}$: $r < \tau_n$, if $n > 0$ then τ_{n-1} is the immediate $<$-successor of τ_n, $\tau_n \models^a \mathsf{S}(x)$ iff $a = \alpha_n$, and $t_n \models^a \mathsf{N}(x)$ iff $n > 0$ and $a = \alpha_{n-1}$.

▶ **Lemma 6.** Suppose $r \models^{\alpha_0} \mathsf{diag}_\infty^{bw\text{-}dec} \wedge \varphi_M^{bw\text{-}dec}$ in a decreasing domain FOLTL-model $\big\langle \langle T, < \rangle, D_t, I \big\rangle_{t \in T}$. For all $i < N$ and all $n < \omega$, let

$$q_n := q, \text{ if } \tau_n \models^{\alpha_n} \mathsf{S}_q(x), \qquad c_i(n) := |\{a \in D_{\tau_n} : \tau_n \models^a \mathsf{C}_i(x)\}|.$$

Then $\big\langle \langle q_n, \mathbf{c}(n) \rangle : n < B \big\rangle$ is a well-defined run of M starting with all-0 counters, whenever $B \leq \omega$ is such that $\tau_n \models^{\alpha_n} \neg \mathsf{start}(x) \wedge \bigwedge_{h \in H} \neg \mathsf{S}_h(x)$, for every $n < B$.

Using these lemmas, first we prove the following generalisation of Theorem 5:

▶ **Theorem 7.** FOLTL$^{\neq}$*-validity is undecidable in decreasing domain models over the class of all linear orders.*

We reduce the 'CM non-termination' problem to FOLTL$^{\neq}$-satisfiability in the above class of models. On the one hand, Lemma 6 implies the 'backward' analogue of Claim 1.1:

▶ **Claim 7.1.** Suppose χ_M is satisfiable in a decreasing domain FOLTL-model, where

$$\chi_M :: \quad \mathsf{diag}_\infty^{bw\text{-}dec} \wedge \varphi_M^{bw\text{-}dec} \wedge \forall x\, \Box_F\big(\mathsf{S}(x) \wedge \mathsf{start}(x) \to \mathsf{S}_{q_0}(x)\big) \wedge \forall x\, \Box_F^+ \bigwedge_{h \in H} \neg \mathsf{S}_h(x)$$
$$\wedge\, \forall x\, \Box_F\big(\mathsf{start}(x) \to \Box_F \neg \mathsf{start}(x)\big). \tag{19}$$

Then M has an infinite run starting from $\langle q_0, \mathbf{0} \rangle$.

On the other hand, if M has an infinite run $\big\langle \langle q_n, \mathbf{c}(n) \rangle : n < \omega \big\rangle$ with $\mathbf{c}(0) = \mathbf{0}$, then we define a constant domain FOLTL-model $\mathfrak{N}_\infty^{bw} = \big\langle \langle \omega + 1, > \rangle, \omega, I \big\rangle$ as follows. Let $\mathsf{P}^{I(\omega)} = \emptyset$, for all $\mathsf{P} \in \{\mathsf{N}, \mathsf{S}, \mathsf{C}_i, \mathsf{S}_q\}_{i<N,\, q \in Q}$, and for all $n < \omega$ and $q \in Q$, let

$$\mathsf{S}^{I(n)} := \{n\}, \quad \mathsf{N}^{I(n)} := \begin{cases} \{n-1\}, & \text{if } n > 0, \\ \emptyset, & \text{if } n = 0, \end{cases} \quad \text{and} \quad \mathsf{S}_q^{I(n)} := \begin{cases} \{n\}, & \text{if } q_n = q, \\ \emptyset, & \text{otherwise.} \end{cases} \tag{20}$$

Further, for all $i < N$, $n < \omega$, we define inductively the sets $\mathsf{C}_i^{I(n)}$. We let $\mathsf{C}_i^{I(0)} := \emptyset$, and

$$\mathsf{C}_i^{I(n+1)} := \begin{cases} \mathsf{C}_i^{I(n)} \cup \{n\}, & \text{if } c_i(n+1) = c_i(n) + 1, \\ \mathsf{C}_i^{I(n)} - \{min(\mathsf{C}_i^{I(n)})\}, & \text{if } c_i(n+1) = c_i(n) - 1, \\ \mathsf{C}_i^{I(n)}, & \text{otherwise.} \end{cases} \tag{21}$$

Finally, let $\mathsf{start}^{I(0)} = \omega$, and $\mathsf{start}^{I(n)} = \emptyset$ for all $0 < n \leq \omega$. Then it is easy to check that $(\mathfrak{N}^{bw}_{\infty}, \omega) \models^0 (19)$, completing the proof of Theorem 7.

The proof of our next theorem uses the same counter machine problem, 'CM recurrence', as the proof of Theorem 1. We call a linear order *modally discrete* if there is no infinite sequence $\langle t_n \in T : n \leq \omega \rangle$ with $t_0 < t_1 < \cdots < t_n < \cdots < t_\omega$. Note that as the only temporal operators of FOLTL$^{\neq}$ are $\Diamond_F$ and $\Box_F$, 'full' discreteness (that is, having no two points with infinitely many points in between) is not FOLTL$^{\neq}$-expressible, but modal discreteness is (see e.g. [11]). Special cases of modally discrete linear orders are *Noetherian orders* (no ascending chains of points) and arbitrary models of the '$\Diamond_F\Box_F$-theory' of $\langle \omega, < \rangle$.

▶ **Theorem 8.** FOLTL$^{\neq}$*-validity is* Π^1_1*-hard in decreasing domain models over any class of modally discrete linear orders containing a linear order that has a rooted infinite descending chain.*

We define the formula rec^{bw} as the conjunction of the following formulas:

$$\forall x\, \Box_F\big(\mathsf{S}(x) \to \exists x\, \mathsf{R}(x)\big),$$
$$\forall x\, \Box_F\big(\mathsf{R}(x) \to \Box_F \neg \mathsf{S}(x)\big),$$
$$\forall x\, \big(\Diamond_F \mathsf{R}(x) \to \Box_F(\mathsf{S}(x) \to \mathsf{S}_{q_r}(x))\big),$$
$$\forall^{\neq} x\, \Box_F\big(\mathsf{S}(x) \to \exists x\, \mathsf{N}(x)\big).$$

In the following claim we use the diagonal generated backward in Lemma 5:

▶ **Claim 8.1.** Suppose that $r \models^{\alpha_0} \mathsf{diag}^{bw\text{-}dec}_{\infty} \wedge \mathsf{rec}^{bw}$ in some decreasing domain FOLTL-model based on a modally discrete linear order. Then there are infinitely many n such that $\tau_n \models^{\alpha_n} \mathsf{S}_{q_r}(x)$.

So by Claims 7.1 and 8.1, if $(19) \wedge \mathsf{rec}^{bw}$ is satisfiable in some decreasing domain FOLTL-model based on a modally discrete linear order, then M has a run starting from $\langle q_0, \mathbf{0} \rangle$ and visiting q_r infinitely often.

On the other hand, suppose that M has run $\big\langle \langle q_n, \mathbf{c}(n) \rangle : n < \omega \big\rangle$ such that $\mathbf{c}(0) = 0$ and $q_{k_n} = q_r$ for an infinite sequence $\langle k_n : n < \omega \rangle$. Let $\langle T, < \rangle$ be a modally discrete linear order in our class that has a rooted infinite descending chain $\tau_\omega < \cdots < \tau_n < \cdots < \tau_0$. We may assume that τ_n is the immediate $<$-successor of τ_{n+1}, for all $n < \omega$. Take the constant domain FOLTL-model $\mathfrak{N}^{bw}_{\infty}$ defined in (20)–(21). We define a constant domain model $\mathfrak{N} = \big\langle \langle T, < \rangle, \omega, J \big\rangle$ by taking, for all $t \in T$, $\mathsf{P} \in \{\mathsf{N}, \mathsf{S}, \mathsf{C}_i, \mathsf{start}, \mathsf{S}_q\}_{i<N,\, q \in Q}$,

$$\mathsf{P}^{J(t)} = \begin{cases} \mathsf{P}^{I(n)}, & \text{if } t = \tau_n,\ n < \omega, \\ \emptyset, & \text{otherwise,} \end{cases} \qquad \mathsf{R}^{J(t)} = \begin{cases} \{k_n\}, & \text{if } t = \tau_n,\ n < \omega, \\ \emptyset, & \text{otherwise.} \end{cases}$$

Then $(\mathfrak{N}, \tau_\omega) \models^0 (19) \wedge \mathsf{rec}^{bw}$, completing the proof of Theorem 8.

7 FOLTL$^{\neq}$ and propositional bimodal logics

There is a well-known connection between finite variable fragments of first-order temporal logics and propositional multimodal logics where the first-order quantifiers are simulated by **S5**-modalities [7]. Here we describe this connection for the version of FOLTL$^{\neq}$ that has $\exists^{\neq} x$ as its sole quantifier (see (1)).

Bimodal formulas are defined by the following grammar:

$$\varphi :: \quad \mathsf{P} \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond_0 \varphi \mid \Diamond_1 \varphi$$

where (with a slight abuse of notation) P ranges over an infinite set $\mathcal{P}$ of propositional variables. Then clearly there is a bijection * from FOLTL$^{\neq}$-formulas to bimodal formulas, mapping each $\mathsf{P}(x)$ to P, $\Diamond_F\phi$ to $\Diamond_0\phi^\star$, $\exists^{\neq}x\,\phi$ to $\Diamond_1\phi^\star$, and commuting with the Booleans.

Bimodal formulas are evaluated in *models* $\mathfrak{M} = \langle W, R_0, R_1, \nu\rangle$, where R_0, R_1 are binary relations over a nonempty set W, and ν is function from $\mathcal{P}$ to the subsets of W. We say that such an $\mathfrak{M}$ is a model *over* the bi-relational structure $\langle W, R_0, R_1\rangle$. For any model $\mathfrak{M}$, $w \in W$, and formula φ, we define the *truth-relation* $\mathfrak{M}, w \models \varphi$ (or just $w \models \varphi$ is $\mathfrak{M}$ is understood) by induction on φ:

- $w \models \mathsf{P}$ iff $w \in \nu(\mathsf{P})$, $w \models \neg\varphi$ iff $w \not\models \varphi$, $w \models \varphi \wedge \psi$ iff $w \models \varphi$ and $w \models \psi$,
- $w \models \Diamond_i\varphi$ iff there is $v \in W$ such that wR_iv and $v \models \varphi$, for $i = 0, 1$.

We say that φ is *true in a model* $\mathfrak{M}$, whenever $\mathfrak{M}, w \models \varphi$ holds for all $w \in W$. If for some set L of bimodal formulas, φ is true in $\mathfrak{M}$ for every φ in L, then we say that $\mathfrak{M}$ is a *model of* L. Given a class $\mathcal{C}$ of models, the *logic of* $\mathcal{C}$, denoted by $\mathsf{Log}\,\mathcal{C}$, is the set of all bimodal formulas that are true in each model from $\mathcal{C}$.

Every FOLTL-model $\mathfrak{M} = \big\langle\langle T, <\rangle, D_t, I\big\rangle_{t\in T}$ can be transformed to a modal model $\mathfrak{M}^\star = \langle W, R_0, R_1, \nu\rangle$, where $W = \{\langle t, a\rangle : t \in T,\, a \in D_t\}$, $\langle t, a\rangle R_0\langle t', a'\rangle$ iff $t < t'$ and $a = a'$, $\langle t, a\rangle R_1\langle t', a'\rangle$ iff $a \neq a'$ and $t = t'$, $\nu(\mathsf{P}) = \{\langle t, a\rangle : t \models^a \mathsf{P}(x)\}$. Such an R_0 is always transitive and weakly connected[2], and R_1 is a pseudo-equivalence relation[3]. So $\mathfrak{M}^\star$ is a model of the *fusion* (or *independent join*) $\mathbf{K4.3} \oplus \mathbf{Diff}$ of the unimodal logics $\mathbf{K4.3}$ (the logic of all models over transitive and weakly connected relations) and $\mathbf{Diff}$ (the logic of all models over pseudo-equivalence relations [25]). Using the methods of [17], it can be shown that in fact $\mathbf{K4.3} \oplus \mathbf{Diff} = \mathsf{Log}\{\mathfrak{M}^\star : \mathfrak{M}$ is a FOLTL-model$\}$, and so for any FOLTL$^{\neq}$-formula ϕ, ϕ is FOLTL$^{\neq}$-valid in arbitrary domain FOLTL-models over the class of all linear orders iff $\phi^\star$ belongs to the bimodal logic $\mathbf{K4.3} \oplus \mathbf{Diff}$. Therefore, the following theorem follows from the results of Wolter [29] and Spaan [26] on fusions:

▶ **Theorem 9.** FOLTL$^{\neq}$*-validity is decidable and* PSPACE*-hard in arbitrary domain models over the class of all linear orders.*

If $\mathfrak{M}$ is a (decreasing, expanding) constant domain FOLTL-model, then $\mathfrak{M}^\star$ is what is called in the literature [8, 7, 10, 17] a (*decreasing*, *expanding*) *product model*. So it is not hard to see that $\mathsf{Log}\{\mathfrak{M}^\star : \mathfrak{M}$ is a constant domain FOLTL-model$\}$ coincides with the *product logic* $\mathbf{K4.3} \times \mathbf{Diff}$, and so by Theorem 5 this bimodal logic is undecidable. Also, by similar results on modal product logics (see [8] or [7, Thm.3.17]), we obtain the following general theorem:

▶ **Theorem 10.** *If $\mathcal{C}$ is a class of linear orders that is definable by a recursive set of first-order sentences (in the language with a binary predicate and equality), then* FOLTL$^{\neq}$*-validity is recursively enumerable in constant, decreasing, or expanding domain models over $\mathcal{C}$.*

Further, Theorem 1 in [10] implies the following:

▶ **Theorem 11.** FOLTL$^{\neq}$*-validity is decidable and has the finite domain property in expanding domain models over the class of all finite linear orders.*

---

[2] A relation R is called *weakly connected* if $\forall xyz\big(xRy \wedge xRz \to (y = z \vee yRz \vee zRy)\big)$.

[3] A relation R is called a *pseudo-equivalence* if it is symmetric and $\forall xyz\big(xRy \wedge yRz \to (xRz \vee x = z)\big)$.

Observe that if $\mathfrak{M}$ is a constant domain FOLTL-model, then the two relations R_0 and R_1 of $\mathfrak{M}^\star$ commute. So $\mathfrak{M}^\star$ is a model of the bimodal logic

$$[\mathbf{K4.3}, \mathbf{Diff}] := \mathsf{Log}\{\langle W, R_0, R_1, \nu\rangle : R_0 \text{ is transitive and weakly connected,} \\ R_1 \text{ is a pseudo-equivalence, } R_0 \text{ and } R_1 \text{ commute}\}.$$

However, $[\mathbf{K4.3}, \mathbf{Diff}]$ is far from being equal to $\mathbf{K4.3}\times\mathbf{Diff}$. In fact, there are infinitely many logics in between, see [13]. So the following theorem generalises Theorem 5:

▶ **Theorem 12.** *No bimodal logic between* $[\mathbf{K4.3}, \mathbf{Diff}]$ *and* $\mathbf{K4.3}\times\mathbf{Diff}$ *is decidable.*

8 Conclusion and open problems

We have shown that FOLTL$^{\neq}$ is very complex over various classes of linear orders, whenever the models have constant, decreasing, or expanding domains. Several questions about expanding domain cases are left unanswered:

1) Is FOLTL$^{\neq}$ decidable in expanding domain models over the class of all linear orders?
2) Is FOLTL$^{\neq}$-validity recursively enumerable in expanding domain models over $\langle\omega, <\rangle$?
3) Is FOLTL$^{\neq}$-validity decidable or recursively enumerable in expanding domain models over the class of all modally discrete linear orders?

By generalising our techniques to the propositional bimodal setting, we have shown that the bimodal logic of commuting linear and pseudo-equivalence relations is undecidable. Related open questions are the following:

4) Is one half of commutativity between the **K4.3** and **Diff** modalities enough to obtain undecidability?
5) Is the bimodal logic $[\mathbf{K4}, \mathbf{Diff}]$ of commuting transitive and pseudo-equivalence relations decidable? Is the product logic $\mathbf{K4}\times\mathbf{Diff}$ decidable?
6) The bimodal reformulation of 1): Is the expanding product logic $\mathbf{K4.3}\times^{exp}\mathbf{Diff}$ decidable?

In our proofs we used reductions of counter machine problems. Other lower bound results about bimodal logics with grid-like models use reductions of tiling or Turing machine problems [7, 9, 22]. On the one hand, it is not hard to re-prove the same results using counter machine reductions. On the other, it seems tiling and Turing machine techniques require more control over the $\omega\times\omega$-grid than the limited expressivity of FOLTL$^{\neq}$ provides. In order to understand the boundary of each technique, it would be interesting to find tiling or Turing machine reductions for the results of the present paper.

References

1 R. Alur and T. Henzinger. A really temporal logic. *J. ACM*, 41:181–204, 1994.
2 E. Börger, E. Grädel, and Yu. Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.
3 J. Chomicki. Temporal query languages: a survey. In D. Gabbay and H.J. Ohlbach, editors, *Procs. ICTL-1994)*, volume 827 of *LNCS*, pages 506–534. Springer, 1994.
4 J. Chomicki and D. Niwinski. On the feasibility of checking temporal integrity constraints. *J. Computer and Systems Sciences*, 51:523–535, 1995.
5 A. Degtyarev, M. Fisher, and A. Lisitsa. Equality and monodic first-order temporal logic. *Studia Logica*, 72:147–156, 2002.
6 D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects, Volume 1*. Oxford University Press, 1994.
7 D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*, volume 148 of *Studies in Logic*. Elsevier, 2003.

8 D. Gabbay and V. Shehtman. Products of modal logics. Part I. *Journal of the IGPL*, 6:73–146, 1998.

9 D. Gabelaia, A. Kurucz, F. Wolter, and M. Zakharyaschev. Products of 'transitive' modal logics. *J. Symbolic Logic*, 70:993–1021, 2005.

10 D. Gabelaia, A. Kurucz, F. Wolter, and M. Zakharyaschev. Non-primitive recursive decidability of products of modal logics with expanding domains. *Ann. Pure Appl. Logic*, 142:245–268, 2006.

11 R. Goldblatt. *Logics of Time and Computation*. CSLI Lecture Notes, 1987.

12 E. Grädel, P. Kolaitis, and M. Vardi. On the decision problem for two-variable first order logic. *Bulletin of Symbolic Logic*, 3:53–69, 1997.

13 C. Hampson and A. Kurucz. Axiomatisation and decision problems of modal product logics with the difference operator. (manuscript), 2012.

14 I. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable fragments of first-order temporal logics. *Ann. Pure Appl. Logic*, 106:85–134, 2000.

15 I. Hodkinson, F. Wolter, and M. Zakharyaschev. Monodic fragments of first-order temporal logics: 2000–2001 A.D. In *Logic for Programming, Artificial Intelligence and Reasoning*, number 2250 in LNAI, pages 1–23. Springer, 2001.

16 B. Konev, F. Wolter, and M. Zakharyaschev. Temporal logics over transitive states. In R. Nieuwenhuis, editor, *Procs. CADE-20*, volume 3632 of *LNCS*, pages 182–203, 2005.

17 A. Kurucz and M. Zakharyaschev. A note on relativised products of modal logics. In P. Balbiani, N-Y. Suzuki., F. Wolter, and M. Zakharyaschev, editors, *Advances in Modal Logic, Volume 4*, pages 221–242. College Publications, 2003.

18 R. Mayr. Undecidable problems in unreliable computations. In G.H. Gonnet, D. Panario, and A. Viola, editors, *Procs. LATIN-2000*, volume 1776 of *LNCS*, pages 377–386, 2000.

19 S. Merz. Decidability and incompleteness results for first-order temporal logics of linear time. *J. Applied Non-Classical Logics*, 2:139–156, 1992.

20 M. Minsky. *Finite and infinite machines*. Prentice-Hall, 1967.

21 L. Pacholski, W. Szwast, and L. Tendera. Complexity results for first-order two-variable logic with counting. *SIAM J. Comput.*, 29:1083–1117, 2000.

22 M. Reynolds and M. Zakharyaschev. On the products of linear modal logics. *J. Logic and Computation*, 11:909–931, 2001.

23 P. Schnoebelen. Lossy counter machines decidability cheat sheet. In A. Kucera and I. Potapov, editors, *Procs. RP-2010*, volume 6227 of *LNCS*, pages 51–75. Springer, 2010.

24 P. Schnoebelen. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In P. Hlinený and A. Kucera, editors, *Procs. MFCS-2010*, volume 6281 of *LNCS*, pages 616–628. Springer, 2010.

25 K. Segerberg. A note on the logic of elsewhere. *Theoria*, 46:183–187, 1980.

26 E. Spaan. *Complexity of Modal Logics*. PhD thesis, Department of Mathematics and Computer Science, University of Amsterdam, 1993.

27 A. Szałas. Concerning the semantic consequence relation in first-order temporal logic. *Theor. Comput. Sci.*, 47(3):329–334, 1986.

28 A. Szałas and L. Holenderski. Incompleteness of first-order temporal logic with until. *Theor. Comput. Sci.*, 57:317–325, 1988.

29 F. Wolter. Fusions of modal logics revisited. In M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev, editors, *Advances in Modal Logic, Volume 1*, pages 361–379. CSLI Publications, 1998.

30 F. Wolter and M. Zakharyaschev. Modal description logics: modalizing roles. *Fundamenta Informaticae*, 39:411–438, 1999.

31 F. Wolter and M. Zakharyaschev. Axiomatizing the monodic fragment of first-order temporal logic. *Ann. Pure Appl. Logic*, 118:133–145, 2002.

On the locality of arb-invariant first-order logic with modulo counting quantifiers

Frederik Harwath and Nicole Schweikardt

Institut für Informatik, Goethe-Universität Frankfurt, Germany
{harwath,schweika}@cs.uni-frankfurt.de

Abstract

We study Gaifman and Hanf locality of an extension of first-order logic with modulo p counting quantifiers (FO+MOD$_p$, for short) with arbitrary numerical predicates. We require that the validity of formulas is independent of the particular interpretation of the numerical predicates and refer to such formulas as arb-invariant formulas. This paper gives a detailed picture of locality and non-locality properties of arb-invariant FO+MOD$_p$. For example, on the class of all finite structures, for any $p \geqslant 2$, arb-invariant FO+MOD$_p$ is neither Hanf nor Gaifman local with respect to a sublinear locality radius. However, in case that p is an odd prime power, it is *weakly* Gaifman local with a polylogarithmic locality radius. And when restricting attention to the class of string structures, for odd prime powers p, arb-invariant FO+MOD$_p$ is both Hanf and Gaifman local with a polylogarithmic locality radius. Our negative results build on examples of order-invariant FO+MOD$_p$ formulas presented in Niemistö's PhD thesis. Our positive results make use of the close connection between FO+MOD$_p$ and Boolean circuits built from NOT-gates and AND-, OR-, and MOD$_p$-gates of arbitrary fan-in.

1998 ACM Subject Classification F.4.1 Mathematical Logic, H.2.3 Languages (Query Languages), F.1.3 Complexity Measures and Classes

Keywords and phrases finite model theory, Gaifman and Hanf locality, first-order logic with modulo counting quantifiers, order-invariant and arb-invariant formulas, lower bounds in circuit complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.363

1 Introduction

Expressibility of logics over finite structures plays an important role in various areas of computer science. In descriptive complexity, logics are used to characterise complexity classes, and concerning databases, common query languages have well-known logical equivalents. These applications have motivated a systematic study of the expressive powers of logics on finite structures. The classical inexpressibility arguments for logics over finite structures (i.e., back-and-forth systems or Ehrenfeucht-Fraïssé games; cf. [9]) often involve nontrivial combinatorics. Notions of *locality* have been proposed as an alternative that allows to contain much of the hard combinatorial work in generic results.

The two best known notions of locality are *Gaifman locality* and *Hanf locality*, introduced in [8, 6]. A k-ary query is called *Gaifman local with locality radius* $f(n)$ if in a structure of cardinality n, the question whether a given tuple satisfies the query only depends on the isomorphism type of the tuple's neighbourhood of radius $f(n)$. A Boolean query is *Hanf local with locality radius* $f(n)$ if the question whether a structure of size n satisfies the query only depends on the number of occurrences of isomorphism types of neighbourhoods of radius $f(n)$. If a given logic is capable of defining only Gaifman or Hanf local queries with a sublinear locality radius, then this logic cannot express "non-local" queries such as, e.g., the

query asking whether two nodes of a graph are connected by a path, or the query asking whether a graph is acyclic (cf., e.g., the textbook [9]). It is well-known that first-order logic FO, as well as extensions of FO by various kinds of counting quantifiers, are Gaifman local and Hanf local with a constant locality radius [8, 6]. Also, locality properties of extensions of FO by invariant uses of order and arithmetic have been considered [7, 1].

Order-invariant and arb-invariant logics[1] were introduced to capture the data independence principle in databases: An implementation of a database query may exploit the order in which the database elements are stored in memory, and thus identify the elements with natural numbers on which arithmetic can be performed. But the use of order and arithmetic should be restricted in such a way that the result of the query does not depend on the particular order in which the data is stored. *Arb-invariant* formulas are formulas that can make use, apart from the relations present in a given structure, also of a linear order $<$ and arithmetic predicates such as $+$ or $\times$ induced by $<$, but only in such a way that the answer is independent of the particular linear order on a structure chosen for $<$. Arb-invariant formulas that only use the linear order, but no further arithmetic predicates, are called *order-invariant*. In [7] it was shown that order-invariant FO can express only queries that are Gaifman local with a constant locality radius, and from [1] we know that arb-invariant FO can express only queries that are Gaifman local with a polylogarithmic locality radius. The proof of [1] relies on a reduction using strong lower bound results from circuit complexity, concerning AC^0-circuits. Similar lower bounds are known also for the extension of AC^0-circuits by modulo p counting gates, for a prime power p [13]. This naturally raises the question whether the locality results from [1] can be generalised to the extension of FO by modulo p counting quantifiers ($\mathrm{FO}{+}\mathrm{MOD}_p$, for short), which precisely corresponds to AC^0-circuits with modulo p counting gates [3]. This question was the starting point for the investigations carried out in the present paper. Our results give a detailed picture of the locality and non-locality properties of order-invariant and arb-invariant $\mathrm{FO}{+}\mathrm{MOD}_p$.

For every natural number $p \geqslant 2$, order-invariant $\mathrm{FO}{+}\mathrm{MOD}_p$ is neither Hanf nor Gaifman local with a sublinear locality radius (Section 4 and Proposition 3.2). For *even* numbers $p \geqslant 2$, order-invariant $\mathrm{FO}{+}\mathrm{MOD}_p$ is not even *weakly* Gaifman local with a sublinear locality radius (Proposition 3.4). Here, *weak* Gaifman locality is a relaxed notion of Gaifman locality referring only to tuples with disjoint neighbourhoods (cf., [9]). However, for *odd prime powers* p we can show that arb-invariant $\mathrm{FO}{+}\mathrm{MOD}_p$ is weakly Gaifman local with a polylogarithmic locality radius (Theorem 3.5). For showing the latter result, we introduce a new locality notion called *shift locality*, for which we can prove for all prime powers p that arb-invariant $\mathrm{FO}{+}\mathrm{MOD}_p$ is shift local with a polylogarithmic locality radius (Theorem 3.7). Our proof relies on Smolensky's circuit lower bound [13]. Generalising our result from prime powers p to arbitrary numbers p can be expected to be difficult, since it would solve long-standing open questions in circuit complexity (Remark 3.13). When restricting attention to the class of *string structures*, we obtain for odd prime powers p that arb-invariant $\mathrm{FO}{+}\mathrm{MOD}_p$ is both Hanf and Gaifman local with a polylogarithmic locality radius (Theorem 4.3 and Corollary 4.4). On the other hand, for even numbers $p \geqslant 2$, already order-invariant $\mathrm{FO}{+}\mathrm{MOD}_p$ on string structures is neither Gaifman nor Hanf local with a sublinear locality radius (Proposition 3.4 and Section 4). This, in particular, implies that order-invariant $\mathrm{FO}{+}\mathrm{MOD}_p$ is strictly more expressive on strings than $\mathrm{FO}{+}\mathrm{MOD}_p$, refuting a conjecture of Benedikt and Segoufin [4].

The remainder of this paper is structured as follows: Section 2 fixes the basic notation,

[1] Strictly speaking, arb-invariant first-order logic is a "logical system" rather than a "logic", as its syntax is undecidable.

introduces the notions of order-invariant and arb-invariant FO+MOD$_p$ and recalls two examples of order-invariant FO+MOD$_p$-formulas from Niemistö's PhD-thesis [11]. Section 3 presents our results concerning Gaifman locality, weak Gaifman locality, and shift locality. Section 4 deals with Hanf locality on finite structures, and with Gaifman and Hanf locality on string structures.

Several details had to be omitted from this paper due to lack of space. These can be found in the full version of this paper, available at the authors' websites.

2 Preliminaries

Basic notation. We write $\mathbb{N}$ for the set of non-negative integers and let $\mathbb{N}_{\geqslant 1} := \mathbb{N} \setminus \{0\}$. For $n \in \mathbb{N}_{\geqslant 1}$ we write $[n]$ for the set $\{i \in \mathbb{N} : 0 \leqslant i < n\}$, i.e., $[n] = \{0, \ldots, n-1\}$. For integers i, j, p with $p \geqslant 2$, we write $i \equiv j \bmod p$ (and say that i is congruent j modulo p) iff there exists an integer k such that $i = j + kp$. For integers i, i', the term $(i+i' \bmod p)$ denotes the number $j \in [p]$ such that $i+i' \equiv j \bmod p$. Two natural numbers i and j are *coprime* if their greatest common divisor is 1. A number p is called a *prime power* if $p = \hat{p}^i$ for a prime $\hat{p}$ and an integer $i \geqslant 1$, and p is called an *odd* prime power if p's prime factor is different from 2 (i.e., p is odd). A number $r \in \mathbb{N}_{\geqslant 1}$ is called a *factor* of a natural number s, if there is a $t \in \mathbb{N}$ such that $s = rt$. We write $\log n$ to denote the logarithm of a number n with respect to base 2, and we often simply write $\log n$ instead of $\lfloor \log n \rfloor$.

For a finite set A we write $|A|$ to denote the cardinality of A. By 2^A we denote the power set of A, i.e., the set $\{Y : Y \subseteq A\}$. The set of all non-empty finite strings built from symbols in A is denoted A^+. We write $|w|$ for the length of a string $w \in A^+$. For an $a \in A$ we write $|w|_a$ for the number of occurrences of the letter a in the string w.

Structures. A *signature* σ is a set of relation symbols R, each of them associated with a fixed arity $ar(R) \in \mathbb{N}_{\geqslant 1}$. Throughout this paper, σ will usually denote a fixed finite signature.

A *σ-structure* $\mathcal{A}$ consists of a non-empty set A called the *universe* of $\mathcal{A}$, and a relation $R^{\mathcal{A}} \subseteq A^{ar(R)}$ for each relation symbol $R \in \sigma$. The *cardinality* of a σ-structure $\mathcal{A}$ is the cardinality of its universe. *Finite* σ-structures are σ-structures of finite cardinality. For σ-structures $\mathcal{A}$ and $\mathcal{B}$ and tuples $\overline{a} = (a_1, \ldots, a_k) \in A^k$ and $\overline{b} = (b_1, \ldots, b_k) \in B^k$ we write $(\mathcal{A}, \overline{a}) \cong (\mathcal{B}, \overline{b})$ to indicate that there is an isomorphism π from $\mathcal{A}$ to $\mathcal{B}$ that maps $\overline{a}$ to $\overline{b}$ (i.e., $\pi(a_i) = b_i$ for each $i \leqslant k$).

We represent strings over a finite alphabet Σ by successor-based structures as follows: We choose $\sigma_\Sigma := \{E\} \cup \{P_a : a \in \Sigma\}$, where E is a binary relation symbol and P_a is a unary relation symbol, for each $a \in \Sigma$. We represent a non-empty string $w \in \Sigma^+$ by the σ_Σ-structure $\mathcal{S}_w$, where the universe of $\mathcal{S}_w$ is the set $\{1, \ldots, |w|\}$ of positions of w, the edge relation $E^{\mathcal{S}_w}$ is the successor relation, i.e., $E^{\mathcal{S}_w} = \{(i, i+1) : 1 \leqslant i < |w|\}$, and for each $a \in \Sigma$, the set $P_a^{\mathcal{S}_w}$ consists of all positions of w that carry the letter a. Structures of the form $\mathcal{S}_w$ (for a string w) are called *string structures*.

In this paper, all *classes* $\mathfrak{C}$ of finite σ-structures will be closed under isomorphism, i.e., if $\mathcal{A}$ and $\mathcal{B}$ are isomorphic σ-structures, then $\mathcal{A} \in \mathfrak{C}$ iff $\mathcal{B} \in \mathfrak{C}$. We will write Σ-*strings* to denote the class of all σ_Σ-structures that represent strings in Σ^+ (i.e., Σ-*strings* is the closure under isomorphisms of the set $\{\mathcal{S}_w : w \in \Sigma^+\}$).

First-order logic with modulo counting quantifiers. We assume that the reader is familiar with basic concepts and notations concerning first-order logic and extensions thereof (cf., e.g., the textbooks [9, 5]). By $free(\varphi)$ we denote the set of all free variables of a formula

φ. A *sentence* is a formula φ with $\mathit{free}(\varphi) = \emptyset$. We often write $\varphi(\overline{x})$, for $\overline{x} = (x_1, \ldots, x_k)$, to indicate that $\mathit{free}(\varphi) = \{x_1, \ldots, x_k\}$. If $\mathcal{A}$ is a σ-structure and $\overline{a} = (a_1, \ldots, a_k) \in A^k$, we write $\mathcal{A} \models \varphi[\overline{a}]$ to indicate that the formula $\varphi(\overline{x})$ is satisfied in $\mathcal{A}$ when interpreting the free occurrences of the variables $x_1, \ldots, x_k$ with the elements $a_1, \ldots, a_k$.

We write $\mathrm{FO}(\sigma)$ to denote the class of all first-order formulas of signature σ. In this paper, we consider the extension of $\mathrm{FO}(\sigma)$ by *modulo counting quantifiers*, defined as follows: Let p be a natural number with $p \geqslant 2$. A *modulo p counting quantifier* is of the form $\exists^{i \bmod p}$ for some $i \in [p]$. A formula of the form $\exists^{i \bmod p} x\, \varphi(x, \overline{y})$ is satisfied by a σ-structure $\mathcal{A}$ and an interpretation $\overline{b} \in A^k$ of the variables $\overline{y}$ iff the number of elements $a \in A$ such that $\mathcal{A} \models \varphi[a, \overline{b}]$ is congruent i modulo p.

For a fixed natural number $p \geqslant 2$ we write $\mathrm{FO{+}MOD}_p(\sigma)$ to denote the extension of $\mathrm{FO}(\sigma)$ by modulo p counting quantifiers. I.e., $\mathrm{FO{+}MOD}_p(\sigma)$ is built from atomic formulas of the form $x_1{=}x_2$ and $R(x_1, \ldots, x_{ar(R)})$, for $R \in \sigma$ and variables $x_1, x_2, \ldots, x_{ar(R)}$, and closed under Boolean connectives $\wedge$, $\vee$, $\neg$, existential and universal first-order quantifiers $\exists$, $\forall$, and modulo p counting quantifiers $\exists^{i \bmod p}$, for $i \in [p]$. This logic has been studied in depth, see e.g., [14, 8, 3]. Note that if m is a multiple of p, then $\mathrm{FO{+}MOD}_m$ can express modulo p counting quantifiers, since $\exists^{i \bmod p} x\, \varphi(x, \overline{y})$ is equivalent to $\bigvee_{0 \leqslant j < m/p} \exists^{jp+i \bmod m} x\, \varphi(x, \overline{y})$.

Arb-invariant logics. We can extend the expressive power of a logic by allowing formulas to use, apart from the relation symbols present in the signature σ, also a linear order $<$, arithmetic predicates such as $+$ or $\times$, or arbitrary numerical predicates. By definition, an r-ary *numerical predicate* $P^{\mathbb{N}}$ is an r-ary relation on $\mathbb{N}$ (i.e., $P^{\mathbb{N}} \subseteq \mathbb{N}^r$). Two examples of numerical predicates are the linear order $<^{\mathbb{N}}$ consisting of all tuples $(a, b) \in \mathbb{N}^2$ with $a < b$, and the addition predicate $+^{\mathbb{N}}$ consisting of all triples $(a, b, c) \in \mathbb{N}^3$ with $a + b = c$.

To allow formulas to use numerical predicates, we fix the following notation: For every $r \in \mathbb{N}_{\geqslant 1}$ and every r-ary numerical predicate $P^{\mathbb{N}}$, let P be a new relation symbol of arity r ("new" meaning that P does not belong to σ). We write η_{arb} to denote the set of all the relation symbols P obtained this way, and let $\sigma_{\mathrm{arb}} := \sigma \cup \eta_{\mathrm{arb}}$ (the subscript "arb" stands for "arbitrary numerical predicates").

Next, we would like to allow $\mathrm{FO{+}MOD}_p(\sigma_{\mathrm{arb}})$-formulas to make meaningful statements about finite σ-structures. To this end, for a finite σ-structure $\mathcal{A}$, we consider embeddings ι of the universe of $\mathcal{A}$ into the initial segment of $\mathbb{N}$ of size $n = |A|$, i.e., the set $[n] = \{0, \ldots, n{-}1\}$.

▶ **Definition 2.1** (Embedding). Let $\mathcal{A}$ be a finite σ-structure, and let $n := |A|$.
An *embedding* ι of $\mathcal{A}$ is a bijection $\iota : A \to [n]$.

Given a finite σ-structure $\mathcal{A}$ and an embedding ι of $\mathcal{A}$, we can translate r-ary numerical predicates $P^{\mathbb{N}}$ into r-ary predicates on A as follows: $P^{\mathbb{N}}$ induces the r-ary predicate P^{ι} on A, consisting of all r-tuples $\overline{a} = (a_1, \ldots, a_r) \in A^r$ where $\iota(\overline{a}) = (\iota(a_1), \ldots, \iota(a_r)) \in P^{\mathbb{N}}$. In particular, the linear order $<^{\mathbb{N}}$ induces the linear order $<^{\iota}$ on A where for all $a, b \in A$ we have $a <^{\iota} b$ iff $\iota(a) < \iota(b)$.

The σ_{arb}-structure $\mathcal{A}^{\iota}$ associated with $\mathcal{A}$ and ι is the expansion of $\mathcal{A}$ by the predicates P^{ι} for all $P \in \eta_{\mathrm{arb}}$. I.e., $\mathcal{A}^{\iota}$ has the same universe as $\mathcal{A}$, all relation symbols $R \in \sigma$ are interpreted in $\mathcal{A}^{\iota}$ in the same way as in $\mathcal{A}$, and every numerical symbol $P \in \eta_{\mathrm{arb}}$ is interpreted by the relation P^{ι}.

To ensure that an $\mathrm{FO{+}MOD}_p(\sigma_{\mathrm{arb}})$-formula φ makes a meaningful statement about a σ-structure $\mathcal{A}$, we evaluate φ in $\mathcal{A}^{\iota}$, and we restrict attention to those formulas whose truth value is independent of the particular choice of the embedding ι. This is formalised by the following notion.

▶ **Definition 2.2** (Arb-invariance). Let $\varphi(\overline{x})$ be an FO+MOD$_p(\sigma_{\text{arb}})$-formula with k free variables, and let $\mathcal{A}$ be a finite σ-structure. The formula $\varphi(\overline{x})$ is *arb-invariant on* $\mathcal{A}$ if for *all* embeddings ι_1 and ι_2 of $\mathcal{A}$ and for all tuples $\overline{a} \in A^k$ we have: $\mathcal{A}^{\iota_1} \models \varphi[\overline{a}] \iff \mathcal{A}^{\iota_2} \models \varphi[\overline{a}]$.

Let $\varphi(\overline{x})$ be arb-invariant on $\mathcal{A}$. We write $\mathcal{A} \models \varphi[\overline{a}]$, if $\mathcal{A}^{\iota} \models \varphi[\overline{a}]$ for some (and hence every) embedding ι of $\mathcal{A}$.

▶ **Definition 2.3** (arb-inv-FO+MOD$_p$). An FO+MOD$_p(\sigma_{\text{arb}})$-formula $\varphi(\overline{x})$ is *arb-invariant on a class* $\mathfrak{C}$ of finite σ-structures, if $\varphi(\overline{x})$ is arb-invariant on every $\mathcal{A} \in \mathfrak{C}$. By arb-inv-FO+MOD$_p^{\mathfrak{C}}(\sigma)$ we denote the set of all FO+MOD$_p(\sigma_{\text{arb}})$-formulas that are arb-invariant on $\mathfrak{C}$.

$\varphi(\overline{x})$ is called *arb-invariant* if it is arb-invariant on the class of all finite σ-structures. We write arb-inv-FO+MOD$_p(\sigma)$ to denote the set of all arb-invariant FO+MOD$_p(\sigma_{\text{arb}})$-formulas.

▶ **Definition 2.4** (Order-invariance and $<$-inv-FO+MOD$_p$).
An arb-invariant formula that only uses the numerical predicate $<^{\mathbb{N}}$ is called *order-invariant*. By $<$-inv-FO+MOD$_p(\sigma)$ we denote the set of all arb-invariant FO+MOD$_p(\sigma \cup \{<\})$-formulas.

Next, we present two examples of $<$-inv-FO+MOD$_p(\sigma)$-sentences that were developed by Niemistö in [11] and that will be used later on in this paper.

▶ **Example 2.5** (Niemistö (Proposition 6.22 in [11])). Let $\sigma = \{E\}$ be the signature consisting of a binary relation symbol E. Proposition 6.22 of [11] presents an $<$-inv-FO+MOD$_2(\sigma)$-sentence $\varphi_{\textit{even cycles}}$ that is satisfied by exactly those finite σ-structures $\mathcal{A}$ that are disjoint unions of directed cycles where the number of cycles of even length is even.

▶ **Example 2.6** (Niemistö (Proposition 6.20 in [11])). Let $\sigma = \{E_1, E_2\}$ be the signature consisting of two binary relation symbols E_1 and E_2 and let $h, w \in \mathbb{N}$ with $h, w \geqslant 2$. The *torus* $\mathcal{A}_{h,w}$ and the *twisted torus* $\mathcal{B}_{h,w}$ of height h and width w are the σ-structures defined as follows (illustrations can be found on the left and in the middle of Figure 1).
The *torus of height h and width w* is the σ-structure $\mathcal{A}_{h,w}$ with universe $[h] \times [w]$ and relations

$$E_1^{\mathcal{A}_{h,w}} := \{ \big((i,j), (i{+}1 \bmod h, j)\big) : i \in [h], \ j \in [w] \}$$
$$E_2^{\mathcal{A}_{h,w}} := F_{h,w} \cup \{ \big((i,w{-}1), (i,0)\big) : i \in [h] \}$$

with $F_{h,w} := \{ \big((i,j), (i,j{+}1)\big) : i \in [h], \ j \in [w{-}1] \}$.
The *twisted torus of height h and width w* is the σ-structure $\mathcal{B}_{h,w}$ with universe $[h] \times [w]$ and relations

$$E_1^{\mathcal{B}_{h,w}} := E_1^{\mathcal{A}_{h,w}}$$
$$E_2^{\mathcal{B}_{h,w}} := F_{h,w} \cup \{ \big((i,w{-}1), (i{+}1 \bmod h, 0)\big) : i \in [h] \}.$$

Proposition 6.20 in [11] presents, for every $h \in \mathbb{N}$ with $h \geqslant 2$, an $<$-inv-FO+MOD$_h(\sigma)$-sentence $\varphi_{h\text{-}\textit{torus}}$ which, for every $w \in \mathbb{N}$ with $w \geqslant 2$, is satisfied by the torus $\mathcal{A}_{h,w}$, but not by the twisted torus $\mathcal{B}_{h,w}$.

3 Locality of queries

A *k-ary query* q is a mapping that associates with every finite σ-structure $\mathcal{A}$ a k-ary relation $q(\mathcal{A}) \subseteq A^k$, which is invariant under isomorphisms, i.e., if π is an isomorphism from a σ-structure $\mathcal{A}$ to a σ-structure $\mathcal{B}$, then for all $\overline{a} = (a_1, \ldots, a_k) \in A^k$ we have $\overline{a} \in q(\mathcal{A})$ iff $\pi(\overline{a}) = (\pi(a_1), \ldots, \pi(a_k)) \in q(\mathcal{B})$. If $\mathfrak{C}$ is a class of finite σ-structures, then every

arb-inv-FO+MOD$_p^{\mathfrak{C}}(\sigma)$-formula $\varphi(\overline{x})$ with k free variables defines a k-ary query q_φ on $\mathfrak{C}$ via $q_\varphi(\mathcal{A}) = \{\overline{a} \in A^k : \mathcal{A} \models \varphi[\overline{a}]\}$, for every σ-structure $\mathcal{A} \in \mathfrak{C}$.

The *Gaifman graph* of a σ-structure $\mathcal{A}$ is the undirected graph $\mathcal{G}(\mathcal{A})$ with vertex set A, where for any $a, b \in A$ with $a \neq b$ there is an undirected edge between a and b iff there is an $R \in \sigma$ and a tuple $(a_1, \ldots, a_{ar(R)}) \in R^{\mathcal{A}}$ such that $a, b \in \{a_1, \ldots, a_{ar(R)}\}$. The *distance* $dist^{\mathcal{A}}(a, b)$ between two elements $a, b \in A$ is the length of a shortest path between a and b in $\mathcal{G}(\mathcal{A})$. The distance $dist^{\mathcal{A}}(b, \overline{a})$ between an element $b \in A$ and a tuple $\overline{a} = (a_1, \ldots, a_k) \in A^k$ is the the minimum of $dist^{\mathcal{A}}(b, a_i)$ for all $i \in \{1, \ldots, k\}$. For every $r \in \mathbb{N}$, the *r-ball* $N_r^{\mathcal{A}}(\overline{a})$ around a tuple $\overline{a} \in A^k$ is the set of all elements b with $dist^{\mathcal{A}}(b, \overline{a}) \leqslant r$. The *$r$-neighbourhood* of $\overline{a}$ is the induced substructure $\mathcal{N}_r^{\mathcal{A}}(\overline{a})$ of $\mathcal{A}$ on $N_r^{\mathcal{A}}(\overline{a})$.

3.1 Gaifman locality

The notion of *Gaifman locality* is a standard tool for showing that particular queries are not definable in certain logics (cf., e.g., the textbook [9] for an overview).

▶ **Definition 3.1** (Gaifman locality)**.** Let $\mathfrak{C}$ be a class of finite σ-structures, $k \in \mathbb{N}_{\geqslant 1}$ and $f : \mathbb{N} \to \mathbb{N}$. A k-ary query q is *Gaifman $f(n)$-local on* $\mathfrak{C}$ if there is an $n_0 \in \mathbb{N}$ such that for every $n \in \mathbb{N}$ with $n \geqslant n_0$ and every σ-structure $\mathcal{A} \in \mathfrak{C}$ with $|A| = n$, the following is true for all k-tuples $\overline{a}, \overline{b} \in A^k$ with $(\mathcal{N}_{f(n)}^{\mathcal{A}}(\overline{a}), \overline{a}) \cong (\mathcal{N}_{f(n)}^{\mathcal{A}}(\overline{b}), \overline{b})$: $\quad \overline{a} \in q(\mathcal{A}) \iff \overline{b} \in q(\mathcal{A})$.
The query q is *Gaifman $f(n)$-local* if it is Gaifman $f(n)$-local on the class of all finite σ-structures.

I.e., in a σ-structure of cardinality n, a query that is Gaifman $f(n)$-local cannot distinguish between k-tuples of nodes whose neighbourhoods of radius $f(n)$ are isomorphic. The function $f(n)$ is called the *locality radius* of the query. It is well-known that queries definable in FO or FO+MOD$_p$ (for any $p \geqslant 2$) are Gaifman local with a constant locality radius [8]. The articles [7] and [1] generalised this to order-invariant FO (for constant locality radius) and arb-invariant FO (for polylogarithmic locality radius) in the following sense: Let $k \in \mathbb{N}_{\geqslant 1}$, and let q be a k-ary query. If q is definable in $<$-inv-FO(σ), then there is a $c \in \mathbb{N}$ such that q is Gaifman c-local. If q is definable in arb-inv-FO(σ), then there is a $c \in \mathbb{N}$ such that q is Gaifman $(\log n)^c$-local. However, for every $d \in \mathbb{N}$ there is a unary query q_d that is definable in arb-inv-FO$(\{E\})$ and that is not Gaifman $(\log n)^d$-local.

Somewhat surprisingly, using Example 2.6 one obtains that the Gaifman locality result cannot be generalised to order- or arb-invariant FO+MOD$_p$. In fact, $<$-inv-FO+MOD$_p$ can define queries that are not even Gaifman local with locality radius as big as $(\frac{n}{h}-2)$, for the smallest prime divisor h of p:

▶ **Proposition 3.2.** *Let $h \in \mathbb{N}$ with $h \geqslant 2$, and let $\sigma = \{R, E_1, E_2\}$ be a signature consisting of a unary relation symbol R and two binary relation symbols E_1, E_2. There exists a unary query q that is not Gaifman $(\frac{n}{h}-2)$-local, but definable in $<$-inv-FO+MOD$_p(\sigma)$, for every multiple $p \geqslant 2$ of h.*

Proof. Recall the $\{E_1, E_2\}$-structures $\mathcal{A}_{h,w}$ and $\mathcal{B}_{h,w}$ from Example 2.6 called *torus* and *twisted torus*, and recall the definition of the relation $F_{h,w}$. For $w \in \mathbb{N}$ with $w \geqslant 2$, the *pre-torus of height h and width w* is the σ-structure $\mathcal{C}_{h,w}$ with universe $[h] \times [w]$ and relations $R^{\mathcal{C}_{h,w}} := \{ (0, w-1) \}$, $E_1^{\mathcal{C}_{h,w}} := E_1^{\mathcal{A}_{h,w}}$, and $E_2^{\mathcal{C}_{h,w}} := F_{h,w}$ (see the rightmost part of Figure 1 for an illustration).

From Example 2.6 we obtain an $<$-inv-FO+MOD$_h(\{E_1, E_2\})$-sentence $\varphi_{h\text{-}torus}$ which, for every width $w \in \mathbb{N}$ with $w \geqslant 2$, is satisfied by $\mathcal{A}_{h,w}$, but not by $\mathcal{B}_{h,w}$. We modify $\varphi_{h\text{-}torus}$ in such a way that we obtain an $<$-inv-FO+MOD$_h(\sigma)$-formula $\psi(x)$ which, when evaluated

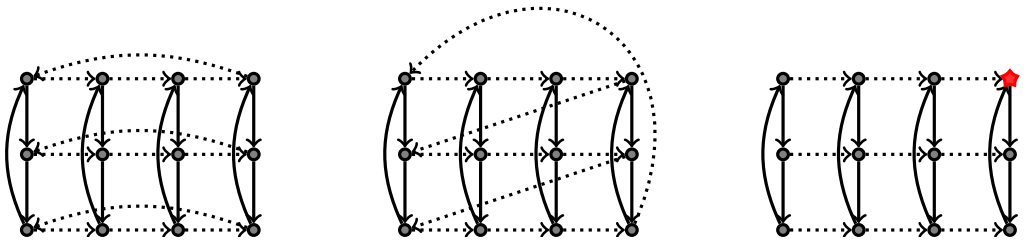


Figure 1 The torus $\mathcal{A}_{3,4}$ (left), the twisted torus $\mathcal{B}_{3,4}$ (middle), and the pre-torus $\mathcal{C}_{3,4}$ (right) of height 3 and width 4. The E_1- and E_2-edges are depicted by solid arcs and dotted arcs, respectively. The unique node in the relation $R^{\mathcal{C}_{3,4}}$ of the pre-torus is the rightmost node in the top row, depicted by a red star.

in the pre-torus $\mathcal{C}_{h,w}$ with x interpreted as the element $a := (0,0)$ (resp., as $b := (1,0)$), simulates $\varphi_{h\text{-}torus}$ evaluated on $\mathcal{A}_{h,w}$ (resp., on $\mathcal{B}_{h,w}$). To this end, we let $\psi(x)$ state that each of the following is satisfied:

- There is a unique element y_0 satisfying $R(y_0)$,
- there are elements $y_1, \ldots, y_{h-1}$ such that $E_1(y_i, y_{i+1 \bmod h})$ is true for all $i \in [h]$,
- there are elements $x_0, \ldots, x_{h-1}$ such that $x_0{=}x$ and $E_1(x_i, x_{i+1 \bmod h})$ is true for all $i \in [h]$,
- the formula φ' is satisfied, where φ' is obtained from $\varphi_{h\text{-}torus}$ by replacing every atom of the form $E_2(u,v)$ by the formula $\big(E_2(u,v) \ \vee \bigvee_{0 \leqslant i < h} \big(u{=}y_i \ \wedge \ v{=}x_i\big)\big)$.

Clearly, $\mathcal{C}_{h,w} \models \psi[a]$ (since $\mathcal{A}_{h,w} \models \varphi_{h\text{-torus}}$), and $\mathcal{C}_{h,w} \not\models \psi[b]$ (since $\mathcal{B}_{h,w} \not\models \varphi_{h\text{-torus}}$). Thus, $a \in q_\psi(\mathcal{C}_{h,w})$ and $b \notin q_\psi(\mathcal{C}_{h,w})$. Note that the $(w{-}2)$-neighbourhoods of a and b in the pre-torus $\mathcal{C}_{h,w}$ are isomorphic, i.e., $(\mathcal{N}^{\mathcal{C}_{h,w}}_{w-2}(a), a) \cong (\mathcal{N}^{\mathcal{C}_{h,w}}_{w-2}(b), b)$. The cardinality of $\mathcal{C}_{h,w}$ is $n := hw$, and hence $w{-}2 = \frac{n}{h}{-}2$. Thus, the query defined by $\psi(x)$ is not Gaifman $(\frac{n}{h}{-}2)$-local.

By Example 2.6, $\varphi_{h\text{-}torus}$ is order-invariant on the class of all finite $\{E_1, E_2\}$-structures. Therefore, the formula $\psi(x)$ is order-invariant on the class of all finite σ-structures. Note that $\psi(x)$ is definable in $<$-inv-FO+MOD$_h(\sigma)$, and hence also in $<$-inv-FO+MOD$_p(\sigma)$, for every multiple p of h. ◀

3.2 Weak Gaifman locality

Weak Gaifman locality (cf., [9]) is a relaxed notion of Gaifman locality where "$\overline{a} \in q(\mathcal{A}) \iff \overline{b} \in q(\mathcal{A})$" needs to be true only for those tuples $\overline{a}$ and $\overline{b}$ whose $f(n)$-neighbourhoods are disjoint.

▶ **Definition 3.3** (Weak Gaifman locality). Let $\mathfrak{C}$ be a class of finite σ-structures, $k \in \mathbb{N}_{\geqslant 1}$ and $f : \mathbb{N} \to \mathbb{N}$. A k-ary query q is *weakly Gaifman $f(n)$-local on* $\mathfrak{C}$ if there is an $n_0 \in \mathbb{N}$ such that for every $n \in \mathbb{N}$ with $n \geqslant n_0$ and every σ-structure $\mathcal{A} \in \mathfrak{C}$ with $|A| = n$, the following is true for all k-tuples $\overline{a}, \overline{b} \in A^k$ with $(\mathcal{N}^{\mathcal{A}}_{f(n)}(\overline{a}), \overline{a}) \cong (\mathcal{N}^{\mathcal{A}}_{f(n)}(\overline{b}), \overline{b})$ and $N^{\mathcal{A}}_{f(n)}(\overline{a}) \cap N^{\mathcal{A}}_{f(n)}(\overline{b}) = \emptyset$: $\overline{a} \in q(\mathcal{A}) \iff \overline{b} \in q(\mathcal{A})$. The query q is *weakly Gaifman $f(n)$-local* if it is weakly Gaifman $f(n)$-local on the class of all finite σ-structures.

Note that the example presented in the proof of Proposition 3.2 does not provide a counter-example to *weak* Gaifman locality, since the elements a and b considered in the proof of Proposition 3.2 are of distance 1, and thus their $f(n)$-neighbourhoods are not disjoint. However, using Example 2.5, one obtains a counter-example to *weak* Gaifman locality for

$<$-inv-FO+MOD$_p$ for *even* numbers p; see Proposition 6.23 in [11]. Here, we present a refinement of Niemistö's proof which provides a counter-example to weak Gaifman locality already for the restricted case of string structures.

▶ **Proposition 3.4.** *Let $\Sigma := \{0,1\}$, and let $\sigma_\Sigma = \{E, P_0, P_1\}$ be the signature used for representing strings over Σ. There exists a unary query q that is not weakly Gaifman $(\frac{n}{4}-1)$-local on Σ-strings, but definable in $<$-inv-FO+MOD$_p(\sigma_\Sigma)$, for every even number $p \geqslant 2$.*

Proof. For every $\ell \in \mathbb{N}_{\geqslant 1}$, let $\mathcal{A}_\ell$ and $\mathcal{B}_\ell$ be $\{E\}$-structures whose universe consists of 2ℓ vertices, the edge relation of $\mathcal{A}_\ell$ consists of two directed cycles of length ℓ, and the edge relation of $\mathcal{B}_\ell$ consists of a single directed cycle of length 2ℓ. Furthermore, we choose w_ℓ to be the string $1^\ell\,0^\ell\,1^\ell\,0^\ell$, and we let $a_\ell := \ell$ be the rightmost position of the first block of 1s, and $b_\ell := 3\ell$ the rightmost position of the second block of 1s.

From Example 2.5 we obtain an $<$-inv-FO+MOD$_2(\{E\})$-sentence $\varphi_{\textit{even cycles}}$ that is satisfied by a finite $\{E\}$-structure $\mathcal{A}$ iff $\mathcal{A}$ is a disjoint union of directed cycles where the number of cycles of even length is even. Thus, for every $\ell \in \mathbb{N}_{\geqslant 1}$ we have: $\mathcal{A}_\ell \models \varphi_{\textit{even cycles}}$ and $\mathcal{B}_\ell \not\models \varphi_{\textit{even cycles}}$. We modify the formula $\varphi_{\textit{even cycles}}$ in such a way that we obtain an $<$-inv-FO+MOD$_2(\sigma_\Sigma)$-formula $\psi(x)$ which, when evaluated in the σ_Σ-structure $\mathcal{S}_{w_\ell}$ representing the string w_ℓ with x interpreted as the position a_ℓ (respectively, the position b_ℓ), simulates $\varphi_{\textit{even cycles}}$ evaluated on $\mathcal{A}_\ell$ (respectively, on $\mathcal{B}_\ell$). To this end, we let $\psi(x)$ be a formula stating that each of the following is satisfied:

- There is a unique position $x' \neq x$ that carries the letter 1 such that the position directly to the right of x' carries the letter 0.
- There is a unique position y of in-degree 0, and this position carries the letter 1. Furthermore, there is a unique position y' that carries the letter 1, such that the position directly to the left of y' carries the letter 0.
- The formula φ' is satisfied, where φ' is obtained from $\varphi_{\textit{even cycles}}$ by relativisation of all quantifiers to positions that carry the letter 1, and by replacing every atom of the form $E(u,v)$ by the formula $\big(E(u,v) \vee \big(u{=}x \wedge v{=}y\big) \vee \big(u{=}x' \wedge v{=}y'\big)\big)$.

Clearly, for every $\ell \in \mathbb{N}_{\geqslant 1}$ we have: $\mathcal{S}_{w_\ell} \models \psi[a_\ell]$ (since $\mathcal{A}_\ell \models \varphi_{\textit{even cycles}}$), and $\mathcal{S}_{w_\ell} \not\models \psi[b_\ell]$ (since $\mathcal{B}_\ell \not\models \varphi_{\textit{even cycles}}$). Thus, $a_\ell \in q_\psi(\mathcal{S}_{w_\ell})$ and $b_\ell \notin q_\psi(\mathcal{S}_{w_\ell})$. Note that the $(\ell{-}1)$-neighbourhoods of a_ℓ and b_ℓ in $\mathcal{S}_{w_\ell}$ are disjoint and isomorphic. The cardinality of $\mathcal{S}_{w_\ell}$ is $n := 4\ell$, and hence $\ell{-}1 = \frac{n}{4}{-}1$. Thus, the query defined by $\psi(x)$ is not weakly Gaifman $(\frac{n}{4}{-}1)$-local. Since $\varphi_{\textit{even cycles}}$ is order-invariant on all finite $\{E\}$-structures, the formula $\psi(x)$ is order-invariant on the class of all finite σ_Σ-structures. Note that $\psi(x)$ is definable in $<$-inv-FO+MOD$_2(\sigma_\Sigma)$, and hence also in $<$-inv-FO+MOD$_p(\sigma_\Sigma)$, for every multiple p of 2. ◀

In light of Proposition 3.4 it is somewhat surprising that for *odd* numbers p, unary queries definable in $<$-inv-FO+MOD$_p$ are weakly Gaifman local with constant locality radius — this is a result obtained by Niemistö (see Corollary 6.37 in [11]). For *odd prime powers* p we can generalise this to k-ary queries definable in arb-inv-FO+MOD$_p$, when allowing polylogarithmic locality radius (note that we cannot hope for a smaller locality radius, since [1] provides, for every $d \in \mathbb{N}$, a unary query definable in arb-inv-FO$(\{E\})$ that is not weakly Gaifman $(\log n)^d$-local).

▶ **Theorem 3.5.** *Let $\mathfrak{C}$ be a class of finite σ-structures. Let $k \in \mathbb{N}_{\geqslant 1}$, let q be a k-ary query, and let p be an odd prime power. If q is definable in* arb-inv-FO+MOD$_p^{\mathfrak{C}}(\sigma)$ *on $\mathfrak{C}$, then there is a $c \in \mathbb{N}$ such that q is weakly Gaifman $(\log n)^c$-local on $\mathfrak{C}$.*

The proof of this theorem will be given in the next subsection, as an easy consequence of Theorem 3.7 below. A generalisation of Theorem 3.5 from odd prime powers to arbitrary odd numbers p would lead to new separations concerning circuit complexity classes and can therefore be expected to be rather difficult (see Remark 3.13).

3.3 Shift locality

The following notion of *shift locality* is a generalisation of the notion of *alternating Gaifman locality* introduced by Niemistö in [11].

▶ **Definition 3.6** (Shift locality)**.** Let $\mathfrak{C}$ be a class of finite σ-structures. Let $k, t \in \mathbb{N}_{\geqslant 1}$ with $t \geqslant 2$, and let $f : \mathbb{N} \to \mathbb{N}$. A kt-ary query q is *shift $f(n)$-local w.r.t. t on $\mathfrak{C}$* if there is an $n_0 \in \mathbb{N}$ such that for every $n \in \mathbb{N}$ with $n \geqslant n_0$ and every σ-structure $\mathcal{A} \in \mathfrak{C}$ with $|A| = n$, the following is true for all k-tuples $\overline{a}_0, \ldots, \overline{a}_{t-1} \in A^k$ with $(\mathcal{N}^{\mathcal{A}}_{f(n)}(\overline{a}_i), \overline{a}_i) \cong (\mathcal{N}^{\mathcal{A}}_{f(n)}(\overline{a}_j), \overline{a}_j)$ and $N^{\mathcal{A}}_{f(n)}(\overline{a}_i) \cap N^{\mathcal{A}}_{f(n)}(\overline{a}_j) = \emptyset$ for all $i, j \in [t]$ with $i \neq j$: $\quad (\overline{a}_0, \overline{a}_1 \ldots, \overline{a}_{t-1}) \in q(\mathcal{A}) \iff (\overline{a}_1, \ldots, \overline{a}_{t-1}, \overline{a}_0) \in q(\mathcal{A})$.
Query q is *shift $f(n)$-local w.r.t. t* if it is shift $f(n)$-local w.r.t. t on the class of all finite σ-structures.

In a technical lemma (Lemma 6.36 in [11]), Niemistö shows that for $k = 1$ and $p, t \in \mathbb{N}$ with $p, t \geqslant 2$ and p and t coprime, for every t-ary query q definable in $<$-inv-FO+MOD$_p(\sigma)$, there is a $c \in \mathbb{N}$ such that q is shift c-local w.r.t. t. Our next result deals with the general case of shift locality and the more expressive logic arb-inv-FO+MOD$_p(\sigma)$, when allowing polylogarithmic locality radius.

▶ **Theorem 3.7.** *Let $\mathfrak{C}$ be a class of finite σ-structures. Let $k, t \in \mathbb{N}_{\geqslant 1}$ with $t \geqslant 2$, let q be a kt-ary query, and let p be a prime power such that p and t are coprime. If q is definable in* arb-inv-FO+MOD$^{\mathfrak{C}}_p(\sigma)$ *on $\mathfrak{C}$, then there is a $c \in \mathbb{N}$ such that q is shift $(\log n)^c$-local w.r.t. t on $\mathfrak{C}$.*

Our proof of Theorem 3.7 relies on lower bounds achieved in circuit complexity. A generalisation of Theorem 3.7 from prime powers to arbitrary numbers $p \geqslant 2$ would lead to new separations of circuit complexity classes and can therefore be expected to be rather difficult (see Remark 3.13). Before giving the proof of Theorem 3.7, let us first point out that it immediately implies Theorem 3.5.

Proof of Theorem 3.5 (using Theorem 3.7). Let $\varphi(\overline{x})$ be an arb-inv-FO+MOD$^{\mathfrak{C}}_p(\sigma)$-formula with k free variables $\overline{x} = (x_1, \ldots, x_k)$, defining a k-ary query q_φ on $\mathfrak{C}$. Let $\overline{y} = (y_1, \ldots, y_k)$ be k variables different from the variables in $\overline{x}$. Then, $\psi(\overline{x}, \overline{y}) := \big(\, \varphi(\overline{x}) \wedge \bigwedge_{1 \leqslant i \leqslant k} y_i{=}y_i \,\big)$ is an arb-inv-FO+MOD$^{\mathfrak{C}}_p(\sigma)$-formula that defines a $2k$-ary query q_ψ. By Theorem 3.7, there exists a $c \in \mathbb{N}$ such that q_ψ is shift $(\log n)^c$-local w.r.t. $t := 2$ on $\mathfrak{C}$. It is straightforward to see that the shift $(\log n)^c$-locality of q_ψ w.r.t. $t = 2$ implies that the query q_φ is weakly Gaifman $(\log n)^c$-local. ◀

The remainder of this subsection is devoted to the proof of Theorem 3.7. We follow the overall method of [1] for the case of disjoint neighbourhoods (see [12] for an overview) and make use of the connection between arb-inv-FO+MOD$_p$ and MOD$_p$-circuits [3], along with a circuit lower bound by Smolensky [13].

We assume that the reader is familiar with basic notions and results in circuit complexity (cf., e.g., the textbook [2]). A MOD$_p$-gate returns the value 1 iff the number of ones at its inputs is congruent 0 modulo p. We consider Boolean MOD$_p$-*circuits* consisting of AND-,

OR-, and MOD_p-gates of unbounded fan-in, input gates, negated input gates, and constant gates **0** and **1**. More precisely, a MOD_p*-circuit with* m *input bits* is a directed acyclic graph whose vertices without ingoing edges are called *input gates* and are labelled with either **0**, **1**, w_ν, or $\neg w_\nu$ for $\nu \in \{1, \dots, m\}$, whose internal nodes are called *gates* and are labelled either AND or OR or MOD_p, and which has a distinguished vertex without outgoing edges called the *output gate*. A MOD_p-circuit C with m input bits naturally defines a function from $\{0,1\}^m$ to $\{0,1\}$. For an input string $w \in \{0,1\}^m$ we say that C *accepts* w if $C(w) = 1$. Accordingly, C *rejects* w if $C(w) = 0$. The *size* of a circuit is the number of gates it contains, and the *depth* is the length of the longest path from the output gate to one of the input gates.

Our proof of Theorem 3.7 relies on Smolensky's following circuit lower bound.

▶ **Theorem 3.8** (Smolensky [13] (see also [14]))**.** *Let p be a prime power. There exist numbers $\varepsilon, \ell > 0$ such that for every $d \in \mathbb{N}_{\geqslant 1}$ there is an $m_d \in \mathbb{N}_{\geqslant 1}$ such that for every $m \in \mathbb{N}$ with $m \geqslant m_d$ the following is true for every number r that has a prime factor different from p's prime factor: No MOD_p-circuit of depth d and size at most $2^{\varepsilon \sqrt[\ell d]{m}}$ accepts exactly those bitstrings $w \in \{0,1\}^m$ that contain a number of ones congruent 0 modulo r.*

In the literature, Smolensky's theorem is usually stated only for primes p. Note, however, that (for each fixed $k \in \mathbb{N}_{\geqslant 1}$) MOD_{p^k}-gates can easily be simulated by MOD_p-circuits of constant depth and polynomial size (cf., [14]), and hence Smolensky's theorem also holds for prime powers p, as stated in Theorem 3.8. It is still open whether an analogous result also holds for numbers p composed of more than one prime factor (see Chapter VIII of [14] and Chapter 14.4 of [2] for discussions on this).

To establish the connection between MOD_p-circuits and arb-inv-FO+$\mathrm{MOD}_p(\sigma)$, we need to represent σ-structures $\mathcal{A}$ and K-tuples $\overline{a} \in A^K$ (for $K \in \mathbb{N}$) by bitstrings. This is done in a straightforward way: Let $\sigma = \{R_1, \dots, R_{|\sigma|}\}$ and let $r_i := ar(R_i)$ for each $i \leqslant |\sigma|$. Consider a finite σ-structure $\mathcal{A}$ with $|A| = n$. Let ι be an embedding of $\mathcal{A}$ into $[n]$. For each $R_i \in \sigma$ we let $\mathrm{Rep}^\iota(R_i^{\mathcal{A}})$ be the bitstring of length n^{r_i} whose j-th bit is 1 iff the j-th smallest element in A^{r_i} w.r.t. the lexicographic order associated with $<^\iota$ belongs to the relation $R_i^{\mathcal{A}}$. Similarly, for each component a_i of a K-tuple $\overline{a} = (a_1, \dots, a_K) \in A^K$ we let $\mathrm{Rep}^\iota(a_i)$ be the bitstring of length n whose j-th bit is 1 iff a_i is the j-th smallest element of A w.r.t. $<^\iota$. Finally, we let

$$\mathrm{Rep}^\iota(\mathcal{A}, \overline{a}) \;:=\; \mathrm{Rep}^\iota(R_1^{\mathcal{A}}) \;\cdots\; \mathrm{Rep}^\iota(R_{|\sigma|}^{\mathcal{A}})\; \mathrm{Rep}^\iota(a_1) \;\cdots\; \mathrm{Rep}^\iota(a_K)$$

be the *binary representation of* $(\mathcal{A}, \overline{a})$ *w.r.t.* ι. Note that, independently of ι, the length of the bitstring $\mathrm{Rep}^\iota(\mathcal{A}, \overline{a})$ is $\lambda_K^\sigma(n) := \sum_{i=1}^{|\sigma|} n^{r_i} + Kn$.

The connection between FO+$\mathrm{MOD}_p(\sigma_{\mathrm{arb}})$ and MOD_p-circuits is obtained by the following result.

▶ **Theorem 3.9** (implicit in [3] (see also [14]))**.** *Let σ be a finite relational signature, let $K \in \mathbb{N}$, and let $p \in \mathbb{N}$ with $p \geqslant 2$. For every* FO+$\mathrm{MOD}_p(\sigma_{arb})$*-formula $\varphi(\overline{x})$ with K free variables there exist numbers $d, s \in \mathbb{N}$ such that for every $n \in \mathbb{N}_{\geqslant 1}$ there is a* MOD_p*-circuit C_n with $\lambda_K^\sigma(n)$ input bits, depth d, and size n^s such that the following is true for all σ-structures $\mathcal{A}$ with $|A| = n$, all $\overline{a} \in A^K$, and all embeddings ι of $\mathcal{A}$ into $[n]$: C_n accepts* $\mathrm{Rep}^\iota(\mathcal{A}, \overline{a}) \iff \mathcal{A}^\iota \models \varphi[\overline{a}]$.

Our proof of Theorem 3.7 uses the next two technical lemmas. To simplify notation, let $\vec{a}^{(0)} := (\overline{a}_0, \overline{a}_1, \dots, \overline{a}_{t-1})$ and $\vec{a}^{(i)} := (\overline{a}_i, \overline{a}_{i+1}, \dots, \overline{a}_{t-1}, \overline{a}_0, \overline{a}_1, \dots, \overline{a}_{i-1})$, for all $i \in [t]$ with $i \geqslant 1$.

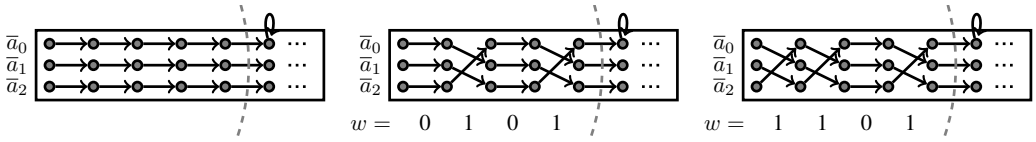


Figure 2 Illustration of a structure $\mathcal{A}$ (left) and structures $\mathcal{A}_w$ for two different bitstrings w.

▶ **Lemma 3.10.** *Let $m, k, t \in \mathbb{N}_{\geqslant 1}$ with $t \geqslant 2$. Let $\mathcal{A}$ be a finite σ-structure with $n := |A|$. For each $i \in [t]$ let $\bar{a}_i \in A^k$ such that for all $i, j \in [t]$ with $i \neq j$ we have $(\mathcal{N}_m^{\mathcal{A}}(\bar{a}_i), \bar{a}_i) \cong (\mathcal{N}_m^{\mathcal{A}}(\bar{a}_j), \bar{a}_j)$ and $N_m^{\mathcal{A}}(\bar{a}_i) \cap N_m^{\mathcal{A}}(\bar{a}_j) = \emptyset$. Let $p \in \mathbb{N}$ with $p \geqslant 2$. Let C be a MOD_p-circuit with $\lambda_{kt}^{\sigma}(n)$ input bits such that: (a) C accepts $\mathrm{Rep}^{\iota_1}(\mathcal{A}, \vec{a}^{(i)})$ iff it accepts $\mathrm{Rep}^{\iota_2}(\mathcal{A}, \vec{a}^{(i)})$, for all embeddings ι_1 and ι_2 of $\mathcal{A}$ and for every $i \in [t]$, and (b) C accepts $\mathrm{Rep}^{\iota}(\mathcal{A}, \vec{a}^{(0)})$ and rejects $\mathrm{Rep}^{\iota}(\mathcal{A}, \vec{a}^{(1)})$, for every embedding ι of $\mathcal{A}$.*

There exists a MOD_p-circuit $\tilde{C}$ with m input bits, such that: (c) $\tilde{C}$ has the same depth and size as C, (d) for all $w, w' \in \{0,1\}^m$ with $|w|_1 \equiv |w'|_1 \bmod t$, $\tilde{C}$ accepts w iff it accepts w', and (e) $\tilde{C}$ accepts all $w \in \{0,1\}^m$ with $|w|_1 \equiv 0 \bmod t$ and rejects all $w \in \{0,1\}^m$ with $|w|_1 \equiv 1 \bmod t$.

Proof. Let $I \subset [t]$ be the set containing $i \in [t]$ iff C accepts $\mathrm{Rep}^{\iota_1}(\mathcal{A}, \vec{a}^{(i)})$ for some (i.e., due to property (a) of C, *every*) embedding ι_1 of $\mathcal{A}$. By property (b) of C, we know that $0 \in I$ and $1 \notin I$.

For the remainder of this proof, fix an embedding ι of $\mathcal{A}$ into $[n]$. Note that ι is also an embedding of any other σ-structure that has the same universe as $\mathcal{A}$. For every $w \in \{0,1\}^m$, we will define a σ-structure $\mathcal{A}_w$ with the same universe as $\mathcal{A}$, which has the following property for every $i \in [t]$:

$$\text{If } \ |w|_1 \equiv i \bmod t, \quad \text{then } \ (\mathcal{A}_w, \vec{a}^{(0)}) \ \cong \ (\mathcal{A}, \vec{a}^{(i)}). \tag{1}$$

Note that if $(\mathcal{A}_w, \vec{a}^{(0)}) \cong (\mathcal{A}, \vec{a}^{(i)})$, then there is an embedding ι_1 such that $\mathrm{Rep}^{\iota_1}(\mathcal{A}, \vec{a}^{(i)}) = \mathrm{Rep}^{\iota}(\mathcal{A}_w, \vec{a}^{(0)})$. Hence, due to property (a), C accepts $\mathrm{Rep}^{\iota}(\mathcal{A}_w, \vec{a}^{(0)})$ iff it accepts $\mathrm{Rep}^{\iota}(\mathcal{A}, \vec{a}^{(i)})$.

The circuit $\tilde{C}$ will be constructed so that on input $w \in \{0,1\}^m$ it does the same as circuit C does on input $\mathrm{Rep}^{\iota}(\mathcal{A}_w, \vec{a}^{(0)})$. Thus, the following is true for every $w \in \{0,1\}^m$ and the particular number $i \in [t]$ such that $|w|_1 \equiv i \bmod t$:

$$\tilde{C} \text{ accepts } w \iff C \text{ accepts } \mathrm{Rep}^{\iota}(\mathcal{A}_w, \vec{a}^{(0)}) \iff C \text{ accepts } \mathrm{Rep}^{\iota}(\mathcal{A}, \vec{a}^{(i)}) \iff i \in I.$$

This immediately implies that $\tilde{C}$ satisfies property (d); and since $0 \in I$ and $1 \notin I$, the circuit $\tilde{C}$ also satisfies property (e).

Definition of $\mathcal{A}_w$: For each $j \in [t]$, we partition $N_m^{\mathcal{A}}(\bar{a}_j)$ into *shells* $S_\nu(\bar{a}_j) := \{x \in A : \mathit{dist}^{\mathcal{A}}(x, \bar{a}_j) = \nu\}$, for all $\nu \in \{0, \ldots, m\}$. We write S_ν for the set $S_\nu(\bar{a}_0) \cup \cdots \cup S_\nu(\bar{a}_{t-1})$. For each $j \in [t]$ let π_j be an isomorphism from $(\mathcal{N}_m^{\mathcal{A}}(\bar{a}_j), \bar{a}_j)$ to $(\mathcal{N}_m^{\mathcal{A}}(\bar{a}_{(j+1 \bmod t)}), \bar{a}_{(j+1 \bmod t)})$. Note that $\pi_j(S_\nu(\bar{a}_j)) = S_\nu(\bar{a}_{(j+1 \bmod t)})$ for each $j \in [t]$ and each $\nu \leqslant m$.

For a bitstring $w = w_1 \cdots w_m \in \{0,1\}^m$ the structure $\mathcal{A}_w$ has the same universe as $\mathcal{A}$. For each $R \in \sigma$ of arity r, the relation $R^{\mathcal{A}_w}$ is obtained from $R^{\mathcal{A}}$ as follows: We start with $R^{\mathcal{A}_w} := \emptyset$, and then for each tuple $\bar{c} \in R^{\mathcal{A}}$ we insert the tuple $\bar{c}_w$ into $R^{\mathcal{A}_w}$, where $\bar{c}_w$ is defined as follows:

- If $\bar{c} \notin (S_{\nu-1} \cup S_\nu)^r$ for any $\nu \leqslant m$, or $\bar{c} \in S_\nu^r$ for some $\nu \leqslant m$, then $\bar{c}_w := \bar{c}$.
- Otherwise, if $\bar{c} \in (S_{\nu-1} \cup S_\nu)^r$ for some $\nu \leqslant m$, then note that (since $\bar{c} \in R^{\mathcal{A}}$), there is a unique $j \in [t]$ such that $\bar{c} \in (S_{\nu-1}(\bar{a}_j) \cup S_\nu(\bar{a}_j))^r$ (since $N_m^{\mathcal{A}}(\bar{a}_j) \cap N_m^{\mathcal{A}}(\bar{a}_{j'}) = \emptyset$, for all

$j, j' \in [t]$ with $j \neq j'$). To keep the notation simple, assume that $\overline{c} = (\overline{c}_{\nu-1}, \overline{c}_\nu)$, where all elements of $\overline{c}_{\nu-1}$ belong to $S_{\nu-1}(\overline{a}_j)$ and all elements of $\overline{c}_\nu$ belong to $S_\nu(\overline{a}_j)$. We define $\overline{c}_w$ depending on the ν-th bit w_ν of w: If $w_\nu = 0$, then $\overline{c}_w := \overline{c}$. If $w_\nu = 1$, then $\overline{c}_w := (\overline{c}_{\nu-1}, \pi_j(\overline{c}_\nu))$.

Note that for every $\nu \in \{1, \dots, m\}$ with $w_\nu = 1$, this construction enforces that the role that was formerly played by shell $S_\nu(\overline{a}_j)$ is afterwards played by shell $S_\nu(\overline{a}_{(j+1 \bmod t)})$; see Figure 2 for an illustration. It is easy to verify that if $|w|_1 \equiv i \bmod t$, then $(\mathcal{A}_w, \vec{a}^{(0)}) \cong (\mathcal{A}, \vec{a}^{(i)})$.

Construction of $\tilde{C}$: The circuit $\tilde{C}$ is obtained from C by replacing the input gates of C in a way that mirrors the construction of $\mathcal{A}_w$ above, in the same way as done in [1]: Each input gate g of C is replaced either by a constant gate $\mathbf{0}$ or $\mathbf{1}$ or by a new input gate w_ν or its negation $\neg w_\nu$. In particular, $\tilde{C}$ has the same depth as C, and the size of $\tilde{C}$ is smaller than or equal to the size of C. ◀

▶ **Lemma 3.11.** *Let $m, d, M, t, p \in \mathbb{N}_{\geqslant 1}$ with $m > 9$ and $p, t \geqslant 2$ such that p and t are coprime. Let $\tilde{C}$ be a MOD_p-circuit of depth d and size M which has the property that for all words $w, w' \in \{0,1\}^m$ with $|w|_1 \equiv |w'|_1 \bmod t$, it accepts w iff it accepts w'. Furthermore, let $\tilde{C}$ accept all $w \in \{0,1\}^m$ with $|w|_1 \equiv 0 \bmod t$, and reject all $w \in \{0,1\}^m$ with $|w|_1 \equiv 1 \bmod t$.*

There is a MOD_p-circuit $\hat{C}$ of depth $(d{+}6)$ and size $(tM{+}2m^t)$ which, for some factor $r \geqslant 2$ of t, accepts exactly those bitstrings $w \in \{0,1\}^m$ where $|w|_1 \equiv 0 \bmod r$.

Proof. We let $b = b_0 b_1 \cdots b_{t-1}$ be the bitstring of length t where, for every $j \in [t]$ we have $b_j = 1$ iff $\tilde{C}$ accepts bitstrings $w \in \{0,1\}^m$ with $|w|_1 \equiv j \bmod t$.

For a bitstring $w \in \{0,1\}^m$ with $|w|_0 \geqslant t{-}1$, we let $\mathit{pattern}(w) = a_0 a_1 \cdots a_{t-1} \in \{0,1\}^t$ with $a_j = 1$ iff $\tilde{C}$ accepts the bitstring obtained from w by replacing the first j zeros with ones. Note that if $|w|_1 \equiv i \bmod t$, then $\mathit{pattern}(w) = b_i b_{i+1} \cdots b_{t-1} b_0 \cdots b_{i-1}$.

▶ **Claim.** *There is a factor $r \geqslant 2$ of t such that for all $w \in \{0,1\}^m$ with $|w|_0 \geqslant t{-}1$ we have: $\mathit{pattern}(w) = b \iff |w|_1 \equiv 0 \bmod r$.*

Proof. In case that $\mathit{pattern}(w) = b$ iff $|w|_1 \equiv 0 \bmod t$, we are done by choosing $r := t$. In case that there is a w with $\mathit{pattern}(w) = b$ and $|w|_1 \equiv i \bmod t$ for an $i \in \{1, \dots, t{-}1\}$, we know that $b_0 b_1 \cdots b_{t-1} = b_i b_{i+1} \cdots b_{t-1} b_0 \cdots b_{i-1}$. Thus, for $x := b_0 \cdots b_{i-1}$ and $y := b_i \cdots b_{t-1}$ we have $b = xy = yx$, and $x, y \in \{0,1\}^+$.

A basic result in word combinatorics (see Proposition 1.3.2 in [10]) states that two words $x, y \in \{0,1\}^+$ commute (i.e., $xy = yx$) iff they are powers of the same word (i.e., there is a $z \in \{0,1\}^+$ and $\nu, \mu \in \mathbb{N}_{\geqslant 1}$ such that $x = z^\nu$ and $y = z^\mu$). We choose $z \in \{0,1\}^+$ of minimal length such that $b = z^s$ for some $s \in \mathbb{N}$. Clearly, $|z| \geqslant 2$, since by assumption we have $b_0 b_1 = 10$.

Since z is of minimal length, it is straightforward to see that for every $w \in \{0,1\}^m$ with $|w|_0 \geqslant t{-}1$ we have: $\mathit{pattern}(w) = z^s \iff |w|_1 \equiv 0 \bmod |z|$. ◀

We choose r according to the claim. Obviously, the following is true for every $w \in \{0,1\}^m$:

$$|w|_1 \equiv 0 \bmod r \iff \begin{cases} (1) & |w|_0 \geqslant t{-}1 \text{ and } \mathit{pattern}(w) = b, \text{ or} \\ (2) & \text{there is a } j \in [t{-}1] \text{ with } m{-}j \equiv 0 \bmod r \text{ such that } |w|_0 = j. \end{cases}$$

To complete the proof of Lemma 3.11, it therefore suffices to construct circuits $C_{(1)}$ and $C_{(2)}$ testing for (1) and (2), respectively, and to let $\hat{C}$ be the disjunction of $C_{(1)}$ and $C_{(2)}$. It is an easy exercise to construct these circuits, using the given circuit $\tilde{C}$, in such a way that the resulting circuit $\hat{C}$ has depth $\leqslant (d{+}6)$ and size $\leqslant (tM + 2m^t)$. ◀

We are now ready for the proof of Theorem 3.7.

Proof of Theorem 3.7. Let q be a kt-ary query defined on $\mathfrak{C}$ by an arb-inv-FO+MOD$_p^{\mathfrak{C}}(\sigma)$-formula $\varphi(\overline{x}_0,\ldots,\overline{x}_{t-1})$, where $\overline{x}_i$ is a k-tuple of variables, for each $i\in[t]$. By Theorem 3.9, there exist numbers $d,s\in\mathbb{N}$ such that for every $n\in\mathbb{N}_{\geqslant 1}$ there is a MOD$_p$-circuit C_n with $\lambda_{kt}^{\sigma}(n)$ input bits, depth d, and size n^s such that the following is true for all σ-structures $\mathcal{A}\in\mathfrak{C}$ with $|A|=n$, all k-tuples $\overline{a}_0,\ldots,\overline{a}_{t-1}\in A^k$, and all embeddings ι of $\mathcal{A}$ into $[n]$:

$$C_n \text{ accepts } \mathrm{Rep}^{\iota}(\mathcal{A},\overline{a}_0,\ldots,\overline{a}_{t-1}) \iff \mathcal{A}^{\iota}\models\varphi[\overline{a}_0,\ldots,\overline{a}_{t-1}] \iff \mathcal{A}\models\varphi[\overline{a}_0,\ldots,\overline{a}_{t-1}].$$

For contradiction, assume that for every $c\in\mathbb{N}$ the query q is *not* shift $(\log n)^c$-local w.r.t. t on $\mathfrak{C}$. Thus, in particular for $c:=2\ell(d{+}6)$ (with ℓ chosen as in Theorem 3.8), we obtain that for all $n_0\in\mathbb{N}$ there is an $n\geqslant n_0$, and a σ-structure $\mathcal{A}\in\mathfrak{C}$ with $|A|=n$, and k-tuples $\overline{a}_0,\ldots,\overline{a}_{t-1}\in A^k$ such that for $m:=(\log n)^c=(\log n)^{2\ell(d+6)}$ we have:

- $(\mathcal{N}_m^{\mathcal{A}}(\overline{a}_i),\overline{a}_i)\cong(\mathcal{N}_m^{\mathcal{A}}(\overline{a}_j),\overline{a}_j)$ and $N_m^{\mathcal{A}}(\overline{a}_i)\cap N_m^{\mathcal{A}}(\overline{a}_j)=\emptyset$ for all $i,j\in[t]$ with $i\neq j$, and
- $\mathcal{A}\models\varphi[\overline{a}_0,\overline{a}_1,\ldots,\overline{a}_{t-1}]$ and $\mathcal{A}\not\models\varphi[\overline{a}_1,\ldots,\overline{a}_{t-1},\overline{a}_0]$.

We fix $n\in\mathbb{N}$ sufficiently large such that, for $\hat{d}:=(d{+}6)$ and ε and $m_{\hat{d}}$ chosen as in Theorem 3.8, we have for $m=(\log n)^c$ that $m>9$, $m\geqslant m_{\hat{d}}$, and $n^{\varepsilon\log n}>tn^s+2(\log n)^{ct}$.

Clearly, C_n is a MOD$_p$-circuit with $\lambda_{kt}^{\sigma}(n)$ input bits which, for every $i\in[t]$ and all embeddings ι_1 and ι_2 of $\mathcal{A}$ accepts $\mathrm{Rep}^{\iota_1}(\mathcal{A},\vec{a}^{(i)})$ iff it accepts $\mathrm{Rep}^{\iota_2}(\mathcal{A},\vec{a}^{(i)})$. Furthermore, C_n accepts $\mathrm{Rep}^{\iota}(\mathcal{A},\vec{a}^{(0)})$ and rejects $\mathrm{Rep}^{\iota}(\mathcal{A},\vec{a}^{(1)})$, for every embedding ι of $\mathcal{A}$. Thus, from Lemma 3.10 we obtain a MOD$_p$-circuit $\tilde{C}$ on m input bits, with depth d and size n^s, such that $\tilde{C}$ has the property that for all words $w,w'\in\{0,1\}^m$ with $|w|_1\equiv|w'|_1 \bmod t$, it accepts w iff it accepts w'. Furthermore, $\tilde{C}$ accepts all $w\in\{0,1\}^m$ with $|w|\equiv 0 \bmod t$ and rejects all $w\in\{0,1\}^m$ with $|w|\equiv 1 \bmod t$. From Lemma 3.11, we therefore obtain a MOD$_p$-circuit $\hat{C}$ of depth $\hat{d}:=(d{+}6)$ and size $(tn^s+2m^t)=(tn^s+2(\log n)^{ct})$ which, for some factor $r\geqslant 2$ of t, accepts exactly those bitstrings $w\in\{0,1\}^m$ where $|w|_1\equiv 0 \bmod r$.

Since p and t are coprime by assumption, and $r\geqslant 2$ is a factor of t, we know that r has a prime factor different from p's prime factor. Thus, from Theorem 3.8 (for $\varepsilon,\ell,m_{\hat{d}}$ chosen as in Theorem 3.8, and for $m\geqslant m_{\hat{d}}$) we know that the size $(tn^s+2(\log n)^{ct})$ of $\hat{C}$ must be bigger than $2^{\varepsilon\,\ell\sqrt[\hat{d}]{m}}$. However, we chose $m=(\log n)^c=(\log n)^{2\ell\hat{d}}$, and hence $2^{\varepsilon\,\ell\sqrt[\hat{d}]{m}}=2^{\varepsilon\cdot(\log n)^2}=n^{\varepsilon\cdot\log n}>tn^s+2(\log n)^{ct}$ for all sufficiently large n — a contradiction! Thus, the proof of Theorem 3.7 is complete. ◀

3.4 Applications

In the same way as Gaifman locality (cf., e.g., [9]), also shift locality can be used for showing that certain queries are not expressible in particular logics. The first example query we consider here is the reachability query *reach* which associates, with every finite directed graph $\mathcal{A}=(A,E^{\mathcal{A}})$, the relation

$$\mathit{reach}(\mathcal{A}):=\{(a,b)\,:\,\mathcal{A}\text{ contains a directed path from node }a\text{ to node }b\}.$$

▶ **Proposition 3.12.** *Let $\sigma=\{E\}$ consist of a binary relation symbol E. Let $p,t\in\mathbb{N}$ with $p,t\geqslant 2$ be chosen such that every t-ary query q definable in* arb-inv-FO+MOD$_p(\sigma)$ *is shift $f_q(n)$-local w.r.t. t, for a function $f_q:\mathbb{N}\to\mathbb{N}$ where $f_q(n)\leqslant(\frac{n}{2t}-\frac{1}{2})$ for all sufficiently large n. Then, the reachability query is not definable in* arb-inv-FO+MOD$_p(\sigma)$.

Proof. Assume, for contradiction, that *reach* is definable by an arb-inv-FO+MOD$_p(\sigma)$-formula $\varrho(x,y)$. Then, $\psi(x_0,\ldots,x_{t-1}):=\varrho(x_0,x_1)\wedge\varrho(x_1,x_2)\wedge\cdots\wedge\varrho(x_{t-2},x_{t-1})$ is

an arb-inv-FO+MOD$_p(\sigma)$-formula expressing in a finite graph $\mathcal{A}$, that there is a directed path from node x_i to node x_{i+1}, for every $i \in [t{-}1]$. Let q be the t-ary query defined by $\psi(x_0, \ldots, x_{t-1})$. By assumption, this query is shift $f_q(n)$-local w.r.t. t, for a function f_q with $f_q(n) \leqslant \frac{n}{2t} - \frac{1}{2}$ for all sufficiently large n.

Now, consider for each $\ell \in \mathbb{N}_{\geqslant 1}$ the graph $\mathcal{A}_\ell$ consisting of a single directed path $v_1 \to v_2 \to \cdots \to v_{t(2\ell+1)}$ on $t{\cdot}(2\ell{+}1)$ nodes. For each $i \in [t]$ let $a_i := v_{i(2\ell+1)+(\ell+1)}$. Then, the ℓ-neighbourhoods of the a_i, for $i \in [t]$, are pairwise disjoint and isomorphic. The cardinality of $\mathcal{A}_\ell$ is $n := t{\cdot}(2\ell{+}1)$, and thus $\ell = \frac{n}{2t} - \frac{1}{2} \geqslant f_q(n)$. Since q is shift $f_q(n)$-local w.r.t. t, we obtain that $\mathcal{A}_\ell \models \psi[a_0, a_1, \ldots, a_{t-1}] \iff \mathcal{A}_\ell \models \psi[a_1, \ldots, a_{t-1}, a_0]$. But in $\mathcal{A}_\ell$ there is a directed path from a_i to a_{i+1} for every $i \in [t{-}1]$, but there is no directed path from a_{t-1} to a_0. Due to the choice of ψ, we have that $\mathcal{A}_\ell \models \psi[a_0, a_1, \ldots, a_{t-1}]$ but $\mathcal{A}_\ell \not\models \psi[a_1, \ldots, a_{t-1}, a_0]$ — a contradiction! ◀

As an immediate consequence of Proposition 3.12 and Theorem 3.7 we obtain (the known fact) that the reachability query is not definable in arb-inv-FO+MOD$_p(\{E\})$, for any prime power p. Using similar constructions, it is an easy exercise to show that none of the following queries is definable in arb-inv-FO+MOD$_p(\{E\})$, for any prime power p:

- *cycle*$(\mathcal{A}) := \{a \in A : a$ is a node that lies on a cycle of the graph $\mathcal{A} = (A, E^{\mathcal{A}})\}$,
- *triangle-reach*$(\mathcal{A}) := \{a \in A : a$ is reachable from a triangle in the graph $\mathcal{A} = (A, E^{\mathcal{A}})\}$,
- *same-distance*$(\mathcal{A}) := \{(a, b, c) \in A^3 : \mathit{dist}^{\mathcal{A}}(a, c) = \mathit{dist}^{\mathcal{A}}(b, c)\}$.

We close with a remark explaining why it can be expected to be difficult to generalise Theorem 3.5 and Theorem 3.7 from prime powers p to arbitrary numbers $p \geqslant 2$.

▶ **Remark 3.13.** Assume, we could generalise Theorem 3.7 from prime powers p to arbitrary numbers $p \geqslant 2$. By Proposition 3.12, we would then obtain that the reachability query is not definable in arb-inv-FO+MOD$_p(\{E\})$, for any $p \in \mathbb{N}$ with $p \geqslant 2$. The "opposite direction" of Theorem 3.9, obtained in [3], would then tell us that the reachability query is not computable in ACC^0. Here, ACC$^0 = \bigcup_{p \geqslant 2}$ AC$^0[p]$, where AC$^0[p]$ is the class of all problems computable by a family of constant depth, polynomial size MOD$_p$-circuits. Since the reachability query can be computed in nondeterministic logarithmic space, we would thus obtain that NLOGSPACE $\not\subseteq$ ACC^0. This would constitute a major breakthrough in computational complexity: The current state-of-the-art (see [15] for a recent survey) states that NEXP $\not\subseteq$ ACC^0, but does not know a problem in PTIME that provably does not belong to ACC^0.

Similarly, a generalisation of Theorem 3.5 to all odd numbers p would imply that the reachability query is not definable in AC$^0[p]$, for any odd number p. Also this is currently not known.

4 Hanf locality and locality on string structures

For giving the precise definition of Hanf locality, we need the following notation: As in [9], for σ-structures $\mathcal{A}$ and $\mathcal{B}$, for k-tuples $\overline{a} \in A^k$ and $\overline{b} \in B^k$, and for an $r \in \mathbb{N}$, we write $(\mathcal{A}, \overline{a}) \leftrightarrows_r (\mathcal{B}, \overline{b})$ (or simply $\mathcal{A} \leftrightarrows_r \mathcal{B}$ in case that $k{=}0$) if there is a bijection $\beta : A \to B$ such that $(\mathcal{N}_r^{\mathcal{A}}(\overline{a}c), \overline{a}c) \cong (\mathcal{N}_r^{\mathcal{B}}(\overline{b}\beta(c)), \overline{b}\beta(c))$ is true for every $c \in A$.

▶ **Definition 4.1** (Hanf locality)**.** Let $\mathfrak{C}$ be a class of finite σ-structures, $k \in \mathbb{N}$, and $f : \mathbb{N} \to \mathbb{N}$. A k-ary query q is *Hanf $f(n)$-local on* $\mathfrak{C}$ if there is an $n_0 \in \mathbb{N}$ such that for every $n \in \mathbb{N}$ with $n \geqslant n_0$ and all σ-structures $\mathcal{A}, \mathcal{B} \in \mathfrak{C}$ with $|A| = |B| = n$, the following is true for all k-tuples $\overline{a} \in A^k$ and $\overline{b} \in B^k$ with $(\mathcal{A}, \overline{a}) \leftrightarrows_{f(n)} (\mathcal{B}, \overline{b})$: $\quad \overline{a} \in q(\mathcal{A}) \iff \overline{b} \in q(\mathcal{B})$.

The query q is called *Hanf $f(n)$-local* if it is Hanf $f(n)$-local on the class of all finite σ-structures.

Hanf locality is an even stronger locality notion than Gaifman locality:

▶ **Theorem 4.2** (Hella, Libkin, Nurmonen [8]). *Let $\mathfrak{C}$ be a class of finite σ-structures and let $f : \mathbb{N} \to \mathbb{N}$. Let $k \in \mathbb{N}_{\geqslant 1}$ and let q be a k-ary query. If q is Hanf $f(n)$-local on $\mathfrak{C}$, then q is Gaifman $(3f(n)+1)$-local on $\mathfrak{C}$.*

It is well-known that queries definable in FO or FO+MOD$_p$ (for any $p \geqslant 2$) are Hanf local with a constant locality radius [6, 8]. For order-invariant or arb-invariant FO it is still open whether they are Hanf local with respect to any sublinear locality radius. As an immediate consequence of Proposition 3.2 and Theorem 4.2 one obtains for every $p \in \mathbb{N}$ with $p \geqslant 2$ that order-invariant FO+MOD$_p$ is *not* Hanf local with respect to any sublinear locality radius.

For the restricted case of string structures, Benedikt and Segoufin [4] have shown that on Σ-*strings* order-invariant FO has the same expressive power as FO and thus is Hanf local with constant locality radius (in fact, [4] obtains the same result also for finite labelled ranked trees). In [1] it was shown that every query definable in arb-invariant FO on Σ-*strings* is Hanf local with polylogarithmic locality radius, and that in the worst case the locality radius can indeed be of polylogarithmic size. As an immediate consequence of Proposition 3.4 and Theorem 4.2 we obtain that for $\Sigma := \{0, 1\}$ there is a unary query q that is *not* Hanf $(\frac{n}{4}-1)$-local on Σ-*strings*, but definable in $<$-inv-FO+MOD$_p(\sigma_\Sigma)$ for every *even* number $p \geqslant 2$. A modification of the proof of Proposition 3.4 also leads to an example of a *Boolean* query (i.e., a 0-ary query) that is not Hanf $(\frac{n-1}{8})$-local on Σ-*strings* for $\Sigma := \{0, 1, 2\}$. Together with the Hanf locality of FO+MOD$_p$, this implies that the result of Benedikt and Segoufin [4] cannot be lifted from order-invariant FO to order-invariant FO+MOD$_p$ on Σ-*strings*, for even numbers $p \geqslant 2$, thus refuting a conjecture of [4].

From Niemistö's Corollary 7.25 in [11] it follows that for *odd* numbers p, order-invariant FO+MOD$_p(\sigma_\Sigma)$ on Σ-*strings* has exactly the same expressive power as FO+MOD$_{\mathrm{PFC}(p)}(\sigma_\Sigma)$, where PFC$(p)$ is the set of all numbers whose prime factors are prime factors of p, and FO+MOD$_{\mathrm{PFC}(p)}$ is first-order logic with modulo m counting quantifiers for all $m \in \mathrm{PFC}(p)$.

The present section's main result shows that for *odd prime powers* p, the Hanf locality result of [1] can be generalised from arb-invariant FO to arb-invariant FO+MOD$_p$ on Σ-*strings*:

▶ **Theorem 4.3.** *Let Σ be a finite alphabet. Let $k \in \mathbb{N}$, let q be a k-ary query, and let p be an odd prime power. If q is definable in* arb-inv-FO+MOD$_p^{\Sigma\text{-strings}}(\sigma_\Sigma)$ *on Σ-strings, then there is a $c \in \mathbb{N}$ such that q is Hanf $(\log n)^c$-local on Σ-strings.*

Together with Theorem 4.2 this implies (general instead of weak) Gaifman locality on Σ-*strings*:

▶ **Corollary 4.4.** *Let Σ be a finite alphabet. Let $k \in \mathbb{N}_{\geqslant 1}$, let q be a k-ary query, and let p be an odd prime power. If q is definable in* arb-inv-FO+MOD$_p^{\Sigma\text{-strings}}(\sigma_\Sigma)$ *on Σ-strings, then there is a $c \in \mathbb{N}$ such that q is Gaifman $(\log n)^c$-local on Σ-strings.*

Note that this corollary does not contradict the non-locality result of Proposition 3.2, as the counter-example given in the proof of that proposition is not a string structure.

The remainder of this section is devoted to the proof of Theorem 4.3. We follow the overall approach of [1]. The crucial step is to prove Theorem 4.3 for queries q of arity $k = 0$; the case for queries of arity $k \geqslant 1$ can easily be reduced to the case for queries of arity 0 by adding k extra symbols to the alphabet (see Section 5.3 in [1] for details).

Note that a 0-ary query q defines the string-language $L_q := \{w \in \Sigma^+ : () \in q(\mathcal{S}_w)\}$, where () denotes the unique tuple of arity 0. The language L_q is called *Hanf $f(n)$-local* iff q is Hanf $f(n)$-local on Σ-*strings*. For proving Theorem 4.3 for the case $k = 0$, we consider the following notion.

▶ **Definition 4.5** (Disjoint swaps [1]). Let $r \in \mathbb{N}$ and let $w \in \Sigma^+$ be a string over a finite alphabet Σ. A string $w' \in \Sigma^+$ is obtained from w by a *disjoint r-swap operation* if there exist strings $\mathtt{x}, \mathtt{u}, \mathtt{y}, \mathtt{v}, \mathtt{z}$ such that $w = \mathtt{xuyvz}$ and $w' = \mathtt{xvyuz}$, and for the positions i, j, i', j' of w just before $\mathtt{u}, \mathtt{y}, \mathtt{v}, \mathtt{z}$ the following is true: The neighbourhoods $N_r^{\mathcal{S}_w}(i)$, $N_r^{\mathcal{S}_w}(j)$, $N_r^{\mathcal{S}_w}(i')$, $N_r^{\mathcal{S}_w}(j')$ are pairwise disjoint, and $(\mathcal{N}_r^{\mathcal{S}_w}(i), i) \cong (\mathcal{N}_r^{\mathcal{S}_w}(i'), i')$ and $(\mathcal{N}_r^{\mathcal{S}_w}(j), j) \cong (\mathcal{N}_r^{\mathcal{S}_w}(j'), j')$.

Let $f : \mathbb{N} \to \mathbb{N}$. A string-language $L \subseteq \Sigma^+$ is *closed under disjoint $f(n)$-swaps* if there exists an $n_0 \in \mathbb{N}$ such that for every $n \in \mathbb{N}_{\geqslant 1}$ with $n \geqslant n_0$, all strings $w \in \Sigma^+$ of length n, and all strings w' obtained from w by a disjoint $f(n)$-swap operation, we have: $w \in L \iff w' \in L$.

It was shown in [1] (see Proposition 5.7, Lemma 5.2, and the proof of Theorem 5.1 in [1]) that if a language $L \subseteq \Sigma^+$ is closed under disjoint $(\log n)^d$-swaps, for some $d \in \mathbb{N}$, then it is Hanf $(\log n)^c$-local on Σ-*strings*, for some $c > d$. Hence, the following lemma immediately implies Theorem 4.3 for the case $k = 0$.

▶ **Lemma 4.6.** *Let Σ be a finite alphabet, let $L \subseteq \Sigma^+$, and let p be an odd prime power. If L is definable, on Σ-strings, by an* arb-inv-FO+MOD$_p^{\Sigma\text{-}strings}(\sigma_\Sigma)$*-sentence, then there exists a constant $d \in \mathbb{N}$ such that L is closed under disjoint $(\log n)^d$-swaps.*

Proof sketch. We proceed in the same way as in the proof of Proposition 5.5 in [1], which obtained the analogue of Lemma 4.6 for arb-inv-FO$^{\Sigma\text{-}strings}(\sigma_\Sigma)$-sentences. By contradiction, assume that there does not exist any $d \in \mathbb{N}$ such that the language L is closed under disjoint $(\log n)^d$-swaps. Then, for any choice of $d, n_0 \in \mathbb{N}$ there exist strings w and w' of length $n \geqslant n_0$ which witness the violation of the "closure under disjoint $(\log n)^d$-swaps" property.

The proof of [1] then proceeds as follows: Choose an appropriate extension $\tilde{\sigma}$ of the signature σ_Σ, define a suitable $\tilde{\sigma}$-structure $\mathcal{A}_w$, and modify the sentence φ defining L to obtain a formula $\psi(\overline{x})$, so that for suitably chosen tuples $\overline{a}$ and $\overline{a}'$ of elements in A_w, the following is true: On $\mathcal{A}_w$, the formula $\psi(\overline{x})$ simulates the evaluation of φ in $\mathcal{S}_w$ (resp., $\mathcal{S}_{w'}$) when assigning the values $\overline{a}$ (resp., $\overline{a}'$) to the variables $\overline{x}$. In [1], $\psi(\overline{x})$ and $\mathcal{A}_w$ are constructed in such a way that $\overline{a}$ and $\overline{a}'$ witness a violation of the Gaifman locality (with a polylogarithmic locality radius) of arb-invariant FO$(\tilde{\sigma})$.

However, in the present case we consider arb-invariant FO+MOD$_p(\tilde{\sigma})$, and thus we only have available the *weak* Gaifman locality result stated in Theorem 3.5. Therefore, we have to choose $\mathcal{A}_w$ and $\psi(\overline{x})$ more carefully, so that we can conclude by using only *weak* Gaifman locality. ◀

References

1 M. Anderson, D. van Melkebeek, N. Schweikardt, and L. Segoufin. Locality from circuit lower bounds. *SIAM Journal on Computing*, 41(6):1481–1523, 2012.

2 S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge Univ. Press, 2009.

3 D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990.

4 M. Benedikt and L. Segoufin. Towards a characterization of order-invariant queries over tame structures. *Journal of Symbolic Logic*, 74(1):168–186, 2009.

5 H.-D. Ebbinghaus and J. Flum. *Finite model theory*. Springer, 1999.

6 R. Fagin, L. J. Stockmeyer, and M. Y. Vardi. On Monadic NP vs. Monadic co-NP. *Information and Computation*, 120(1):78–92, 1995.

7 M. Grohe and T. Schwentick. Locality of order-invariant first-order formulas. *ACM Transactions on Computational Logic*, 1(1):112–130, 2000.

8 L. Hella, L. Libkin, and J. Nurmonen. Notions of locality and their logical characterizations over finite models. *Journal of Symbolic Logic*, 64(4):1751–1773, 1999.

9 L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

10 M. Lothaire. *Combinatorics on words*. Cambridge University Press, 1984.

11 H. Niemistö. *Locality and Order-Invariant Logics*. PhD thesis, Department of Mathematics and Statistics, University of Helsinki, 2007.

12 N. Schweikardt. A short tutorial on order-invariant first-order logic. In *Proc. 8th Int'l Computer Science Symposium in Russia (CSR'13)*, pages 112–126, 2013.

13 R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proc. STOC'87*, pages 77–82, 1987.

14 H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, 1994.

15 R. Williams. Guest column: A casual tour around a circuit complexity bound. *SIGACT News*, 42(3):54–76, 2011.

When is Metric Temporal Logic Expressively Complete?*

Paul Hunter

Department of Computer Science
Université Libre de Bruxelles
Belgium
paul.hunter@ulb.ac.be

Abstract

A seminal result of Kamp is that over the reals Linear Temporal Logic (LTL) has the same expressive power as first-order logic with binary order relation $<$ and monadic predicates. A key question is whether there exists an analogue of Kamp's theorem for Metric Temporal Logic (MTL) – a generalization of LTL in which the Until and Since modalities are annotated with intervals that express metric constraints. Hirshfeld and Rabinovich gave a negative answer, showing that first-order logic with binary order relation $<$ and unary function $+1$ is strictly more expressive than MTL with integer constants. However, a recent result of Hunter, Ouaknine and Worrell shows that when rational timing constants are added to both languages, MTL has the same expressive power as first-order logic, giving a positive answer. In this paper we generalize these results by giving a precise characterization of those sets of constants for which MTL and first-order logic have the same expressive power. We also show that full first-order expressiveness can be recovered with the addition of counting modalities, strongly supporting the assertion of Hirshfeld and Rabinovich that Q2MLO is one of the most expressive decidable fragments of $\mathrm{FO}(<, +1)$.

1998 ACM Subject Classification F.4.1 Mathematical Logic, I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases Metric Temporal Logic, Expressive power, First-order logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.380

1 Introduction

One of the best-known and most widely studied logics in specification and verification is *Linear Temporal Logic (LTL)*: temporal logic with the modalities *Until* and *Since*. For discrete-time systems one considers interpretations of LTL over the integers $(\mathbb{Z}, <)$, and for continuous-time systems one considers interpretations over the reals $(\mathbb{R}, <)$. A celebrated result of Kamp [14] is that, over both $(\mathbb{Z}, <)$ and $(\mathbb{R}, <)$, LTL has the same expressiveness as the *Monadic Logic of Order* ($\mathrm{FO}(<)$): first-order logic with binary order relation $<$ and uninterpreted monadic predicates. Thus we can benefit from the appealing variable-free syntax and elementary decision procedures of LTL, while retaining the expressiveness and canonicity of first-order logic.

Over the reals $\mathrm{FO}(<)$ cannot express quantitative properties such as "every request is followed by a response within one time unit". This motivates the introduction of *Monadic Logic of Order and Metric*, which augments $\mathrm{FO}(<)$ with a family of unary function symbols

* This work was supported by the ERC inVEST project.

$+c$, for all $c \in \mathbb{R}$. We consider fragments of this logic by restricting the unary functions to some set $\mathcal{K} \subseteq \mathbb{R}$ of timing constants, and we denote this by $\mathrm{FO}_{\mathcal{K}}$. The traditional choice for $\mathcal{K}$ is $\mathbb{Z}$ (or, equivalently, $\{1\}$); however it seems more natural in the continuous time setting to have sets of timing constants that are not discrete. In [13] Hunter *et al.* considered rational timing constants, but sets of constants involving irrationals such as $\{1, \sqrt{2}\}$ or $\mathbb{R}$ have practical application: for example, in the specification of systems with two or more timing devices which are initially synchronized but have independent unit time length. In this paper we consider arbitrary subsets of $\mathbb{R}$ for $\mathcal{K}$; however we observe that with simple arithmetic any integer linear combination of elements in $\mathcal{K}$ can be derived as a unary function. Thus we restrict our attention to sets that are closed under integer linear combinations, that is, additive subgroups of $\mathbb{R}$.

There have been a variety of proposals for quantitative temporal logics, with modalities definable in $\mathrm{FO}_{\mathcal{K}}$ (see, e.g., [1, 2, 3, 7, 8, 9, 13]). Typically these temporal logics can be seen as quantitative extensions of LTL. However, until [13] there was no fully satisfactory counterpart to Kamp's theorem in the quantitative setting.

The best-known quantitative temporal logic is *Metric Temporal Logic (MTL)*, introduced over 20 years ago in [15]. MTL arises by annotating the temporal modalities of LTL with real intervals representing metric constraints. Again we consider fragments by restricting the endpoints of the intervals to some $\mathcal{K} \subseteq \mathbb{R}$, and as we are interested in various choices of $\mathcal{K}$ we denote this as $\mathrm{MTL}_{\mathcal{K}}$. Since the $\mathrm{MTL}_{\mathcal{K}}$ operators are definable in $\mathrm{FO}_{\mathcal{K}}$, it is immediate that one can translate $\mathrm{MTL}_{\mathcal{K}}$ into $\mathrm{FO}_{\mathcal{K}}$. The main question addressed by this paper is when does the converse apply?

Several previous results, illustrating that the question is non-trivial, can be succinctly specified with our notation:

- Kamp [14]: $\mathrm{MTL}_{\{0\}} = \mathrm{LTL} = \mathrm{FO}(<) = \mathrm{FO}_{\{0\}}$.
- Hirshfeld and Rabinovich [11]: $\mathrm{MTL}_{\mathbb{Z}} \neq \mathrm{FO}_{\{1\}} = \mathrm{FO}_{\mathbb{Z}}$.
- Hunter, Ouaknine and Worrell [13]: $\mathrm{MTL}_{\mathbb{Q}} = \mathrm{FO}_{\mathbb{Q}}$.

The first main result of this paper generalizes these results by giving a precise characterization of when $\mathrm{MTL}_{\mathcal{K}}$ is expressively complete.

▶ **Theorem 1.** *Let $\mathcal{K}$ be an additive subgroup of $\mathbb{R}$. Then $MTL_{\mathcal{K}} = FO_{\mathcal{K}}$ if and only if $\mathcal{K}$ is dense.*

Two consequences of this theorem are that $\mathrm{MTL}_{\mathbb{R}}$ is expressively complete (for the Monadic Logic of Order and Metric), and, in contrast to $\mathrm{MTL}_{\mathbb{Z}} \neq \mathrm{FO}_{\{1\}}$, MTL with interval endpoints taken from $\mathbb{Z}[\sqrt{2}] = \{a + b\sqrt{2} : a, b \in \mathbb{Z}\}$ is able to express all of $\mathrm{FO}_{\{1,\sqrt{2}\}}$.

Our proof for the "if" part of this theorem is similar to the strategy used in [13]: we prove expressive completeness for bounded formulas and then use a generalization of the concept of *separation*, introduced by Gabbay [4], to extend this to all formulas. However our proof departs significantly from that result in a number of areas. In [13], the authors used scaling to reduce the problem from $\mathrm{FO}_{\mathbb{Q}}$ formulas to $\mathrm{FO}_{\mathbb{Z}}$ formulas. For general sets of constants however, this is not always possible: both in the scaling operation, which may no longer be reversible, and in the reduction to $\mathrm{FO}_{\mathbb{Z}}$. We overcome this by introducing a more general notion of separability in Lemma 13 which can account for unary functions other than $+1$. Secondly, the regularity of the integers was exploited to remove occurrences of the $+1$ function, resulting in $\mathrm{FO}_{\{0\}}$ formulas restricted to unit intervals. In our setting there is no obvious interval length that will achieve this. Instead we introduce a normal form for $\mathrm{FO}_{\mathcal{K}}$ formulas that enables us to remove the $+c$ functions and reduce the problem to $\mathrm{FO}_{\{0\}}$ formulas restricted to bounded intervals.

It follows from our proof of Theorem 1 and the result of [11] that if $\mathrm{MTL}_K \neq \mathrm{FO}_K$ then even with a (possibly infinite) set of arbitrary additional modal operators of bounded quantifier depth the inequality remains. Examples of separating formula are $\mathbf{C}_n\ \varphi$ and $\overline{\mathbf{C}}_n\ \varphi$, for sufficiently large n, where $\mathbf{C}_n$ is the modal operator which asserts that a formula is satisfied at least n distinct times in the next time interval and $\overline{\mathbf{C}}_n$ is its temporal dual. Our second main result is to show that for expressive completeness it is sufficient to add the (infinite) set of these counting operators. That is, if we define $\mathrm{MTL}_{\mathbb{Z}}$+C as the logic of $\mathrm{MTL}_{\mathbb{Z}}$ with the additional operators $\{\mathbf{C}_n, \overline{\mathbf{C}}_n \colon n \in \mathbb{N}\}$, then

▶ **Theorem 2.** *$\mathrm{MTL}_{\mathbb{Z}}$+C has the same expressive power as $\mathrm{FO}_{\mathbb{Z}}$.*

In [9] Hirshfeld and Rabinovich considered the addition of counting modalities to MITL: Metric Temporal Logic without singleton (punctual) intervals. They showed the resulting logic had the same expressive power as Q2MLO, a decidable fragment of $\mathrm{FO}_{\{1\}}$. Our result supports their claim that Q2MLO is one of the most expressive decidable fragments of $\mathrm{FO}_{\{1\}}$: by adding the operators $\Diamond_{\{1\}}X$ (X occurs in exactly one time unit) and $\Diamond\!\!\!\!-_{\{1\}}X$ (X occurred exactly one time unit ago) the resulting logic has the full expressive power of $\mathrm{FO}_{\{1\}}$.

2 Preliminaries

In this section we define the concepts and notation used throughout the paper.

We say $\mathcal{K} \subseteq \mathbb{R}$ is an *additive subgroup of* $\mathbb{R}$ if it is non-empty and closed under addition and unary minus, and we say $\mathcal{K}$ is *dense* if for all $a < b \in \mathcal{K}$, there exists $c \in \mathcal{K}$ such that $a < c < b$. We write $\mathcal{K}_{\geq 0}$ for the set $\{c : c \in \mathcal{K} \text{ and } c \geq 0\}$.

First-order logic

Formulas of *Monadic Logic of Order and Metric with constants $\mathcal{K}$ ($\mathrm{FO}_{\mathcal{K}}$)* are first-order formulas over a signature with a binary relation symbol $<$, an infinite collection of unary predicate symbols $P_1, P_2, \ldots$, and a (possibly infinite) family of unary function symbols $+c$, $c \in \mathcal{K}$. Formally, the terms of $\mathrm{FO}_{\mathcal{K}}$ are generated by the grammar $t ::= x \mid t + c$, where x is a variable and $c \in \mathcal{K}$. Formulas of $\mathrm{FO}_{\mathcal{K}}$ are given by the following syntax:

$$\varphi ::= \mathbf{true} \mid P_i(t) \mid t = t \mid t < t \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x\, \varphi\,,$$

where x denotes a variable and t a term. When $\mathcal{K}$ is an additive subgroup it suffices to consider only terms of the form $x + c$, $c \in \mathcal{K}$.

We consider interpretations of $\mathrm{FO}_{\mathcal{K}}$ over the real line, $\mathbb{R}$, with the natural interpretations of $<$ and $+c$. It follows that a structure for $\mathrm{FO}_{\mathcal{K}}$ is determined by an interpretation of the monadic predicates.

Given terms $t_1 = x_1 + c_1$ and $t_2 = x_2 + c_2$, we define $\mathrm{Bet}_{\mathcal{K}}(t_1, t_2)$ to consist of the $\mathrm{FO}_{\mathcal{K} \cup \{c_1, c_2\}}$ formulas in which

(i) each subformula $\exists z\, \psi$ has the form $\exists z\, ((t_1 < z < t_2) \wedge \chi)$, i.e., each quantifier is relativized to the open interval between t_1 and t_2;
(ii) in each atomic subformula $P(t)$ the term t is a bound occurrence of a variable;
(iii) For $i \in \{1, 2\}$, if $c_i \notin \mathcal{K}$ then $+c_i$ only occurs in t_i.

These conditions ensure that a formula in $\mathrm{Bet}_{\mathcal{K}}(t_1, t_2)$ is essentially a $\mathrm{FO}_{\mathcal{K}}$ formula that only refers to the values of monadic predicates on points in the open interval (t_1, t_2). Clause (iii) allows us to use endpoints not necessarily definable in $\mathrm{FO}_{\mathcal{K}}$. We say that a formula $\varphi(x)$ in $\mathrm{Bet}_{\mathcal{K}}(x - N, x + N)$ is *N-bounded*.

Metric Temporal Logic

Given a set $\boldsymbol{P}$ of atomic propositions, the formulas of *Metric Temporal Logic with constants* $\mathcal{K}$ *(MTL*$_\mathcal{K}$*)* are built from $\boldsymbol{P}$ using boolean connectives and time-constrained versions of the *Until* and *Since* operators $\mathbf{U}$ and $\mathbf{S}$ as follows:

$$\varphi ::= \mathbf{true} \mid P \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \,\mathbf{U}_I\, \varphi \mid \varphi \,\mathbf{S}_I\, \varphi\,,$$

where $P \in \boldsymbol{P}$ and $I \subseteq (0,\infty)$ is an interval with endpoints in $\mathcal{K}_{\geq 0} \cup \{\infty\}$.

Intuitively, the meaning of $\varphi_1 \,\mathbf{U}_I\, \varphi_2$ is that φ_2 will hold at some time in the interval I, and until then φ_1 holds. More precisely, the semantics of MTL$_\mathcal{K}$ are defined as follows. A *signal* is a function $f : \mathbb{R} \to 2^{\boldsymbol{P}}$. Given a signal f and $r \in \mathbb{R}$, we define the satisfaction relation $f, r \models \varphi$ by induction over φ as follows:

- $f, r \models p$ iff $p \in f(r)$,
- $f, r \models \neg\varphi$ iff $f, r \not\models \varphi$,
- $f, r \models \varphi_1 \wedge \varphi_2$ iff $f, r \models \varphi_1$ and $f, r \models \varphi_2$,
- $f, r \models \varphi_1 \,\mathbf{U}_I\, \varphi_2$ iff there exists $t > r$ such that $t - r \in I$, $f, t \models \varphi_2$ and $f, u \models \varphi_1$ for all $u \in (r, t)$,
- $f, r \models \varphi_1 \,\mathbf{S}_I\, \varphi_2$ iff there exists $t < r$ such that $r - t \in I$, $f, t \models \varphi_2$ and $f, u \models \varphi_1$ for all $u \in (t, r)$.

LTL can be seen as a restriction of MTL with only the interval $I = (0,\infty)$, so in particular LTL $=$ MTL$_{\{0\}}$. MITL is a restriction of MTL$_\mathbb{Z}$ where singleton intervals, that is intervals of the form $\{c\}$, do not occur in the $\mathbf{U}$ and $\mathbf{S}$ operators.

We say the $\mathbf{U}_I$ and $\mathbf{S}_I$ operators are *bounded* if I is bounded, otherwise we say that the operators are *unbounded*.

We introduce the derived connectives $\Diamond_I\varphi := \mathbf{true}\,\mathbf{U}_I\,\varphi$ (φ will be true at some point in interval I) and $\Diamond\!\!\!\!-_I\varphi := \mathbf{true}\,\mathbf{S}_I\,\varphi$ (φ was true at some point in interval I in the past). We also have the dual connectives $\Box_I\varphi := \neg\Diamond_I\neg\varphi$ (φ will hold at all times in interval I in the future) and $\boxminus_I := \neg\Diamond\!\!\!\!-_I\neg\varphi$ (φ was true at all times in interval I in the past).

Counting modalities

The counting modalities $\mathbf{C}_n\,\varphi$ and $\overline{\mathbf{C}}_n\,\varphi$ are defined for all $n \in \mathbb{N}$ and are interpreted as φ will be true for at least n distinct occasions in the next/previous time unit. That is, for any signal f and $r \in \mathbb{R}$:

- $f, r \models \mathbf{C}_n\,\varphi$ iff there exists $r_1 < \cdots < r_n \in (r, r+1)$ with $f, r_i \models \varphi$ for all i.
- $f, r \models \overline{\mathbf{C}}_n\,\varphi$ iff there exists $r_1 < \cdots < r_n \in (r-1, r)$ with $f, r_i \models \varphi$ for all i.

We define *MTL*$_\mathcal{K}$ *with counting (MTL*$_\mathcal{K}$*+C)* to be the extension of MTL$_\mathcal{K}$ by the operations $\{\mathbf{C}_n, \overline{\mathbf{C}}_n \colon n \in \mathbb{N}\}$.

Expressive Equivalence

Given a set $\boldsymbol{P} = \{P_1, \ldots, P_m\}$ of monadic predicates, a signal $f : \mathbb{R} \to 2^{\boldsymbol{P}}$ defines an interpretation of each P_i, where $P_i(r)$ iff $P_i \in f(r)$. As observed earlier, this is sufficient to define the model-theoretic semantics of FO$_\mathcal{K}$, enabling us to relate the semantics of FO$_\mathcal{K}$ and MTL$_\mathcal{K}$.

Let $\varphi(x)$ be an FO$_\mathcal{K}$ formula with one free variable and ψ an MTL$_\mathcal{K}$ formula. We say φ and ψ are *equivalent* if for all signals f and $r \in \mathbb{R}$:

$$f \models \varphi[r] \iff f, r \models \psi.$$

We say $\mathrm{MTL}_{\mathcal{K}}$ and $\mathrm{FO}_{\mathcal{K}}$ have the same expressive power, written $\mathrm{MTL}_{\mathcal{K}} = \mathrm{FO}_{\mathcal{K}}$, if for all formulas with one free variable $\varphi(x) \in \mathrm{FO}_{\mathcal{K}}$ there is an equivalent formula $\varphi^\dagger \in \mathrm{MTL}_{\mathcal{K}}$ and vice versa.

3 Characterization of expressively complete MTL

The goal of this section is to prove:

▶ **Theorem 1.** *Let $\mathcal{K}$ be an additive subgroup of $\mathbb{R}$. Then $MTL_{\mathcal{K}} = FO_{\mathcal{K}}$ if and only if $\mathcal{K}$ is dense.*

First we consider the "only if" direction. Central to this is the following easily proven result:

▶ **Lemma 3.** *Let $\mathcal{K}$ be an additive subgroup of $\mathbb{R}$. If $\mathcal{K}$ is not dense then $\mathcal{K} = \epsilon\mathbb{Z}$ for some $\epsilon > 0$.*

It now follows by a simple scaling argument and the result $\mathrm{MTL}_{\mathbb{Z}} \neq \mathrm{FO}_{\mathbb{Z}}$ [11] that if $\mathcal{K}$ is not dense then $\mathrm{MTL}_{\mathcal{K}} \neq \mathrm{FO}_{\mathcal{K}}$. We refer the reader to the full version of the paper for details.

In fact [11] showed a much stronger result: even with (possibly infinite) additional arbitrary modal operators of bounded quantifier depth $\mathrm{MTL}_{\mathbb{Z}}$ cannot fully express $\mathrm{FO}_{\mathbb{Z}}$. This result clearly carries over to $\mathcal{K} = \epsilon\mathbb{Z}$, thus in the non-dense case $\mathrm{MTL}_{\mathcal{K}}$ is "quite far" from $\mathrm{FO}_{\mathcal{K}}$.

▶ **Corollary 4.** *Let $\mathcal{K}$ be a non-dense additive subgroup of $\mathbb{R}$. With additional arbitrary modal operators of bounded quantifier depth $MTL_{\mathcal{K}}$ cannot fully express $FO_{\mathcal{K}}$.*

Returning to the "if" direction in the proof of Theorem 1, we focus on the non-trivial case ($\mathcal{K}$ infinite), as the trivial case $\mathcal{K} = \{0\}$ is covered by Kamp's theorem [14]. As mentioned earlier, our strategy parallels the proof of expressive completeness of $\mathrm{MTL}_{\mathbb{Q}}$ in [13]: we first show expressive completeness for bounded formulas, and then, using a refinement of syntactic separation [4, 5, 13], extend this to all $\mathrm{FO}_{\mathcal{K}}$ formulas.

3.1 Expressive completeness for bounded formulas

To show that bounded $\mathrm{FO}_{\mathcal{K}}$ formulas can be expressed by $\mathrm{MTL}_{\mathcal{K}}$ we proceed in a similar manner to [13].

Step 1. We first remove any occurrence of a unary $+c$ function applied to a bound variable.

Step 2. Using a composition argument (see e.g. [6, 10]) we then reduce the problem to showing expressive completeness for formulas in $\mathrm{Bet}_{\{0\}}(x, x + c)$.

Step 3. Exploiting a normal form of [6] and the denseness of $\mathcal{K}$ we show how an $\mathrm{MTL}_{\mathcal{K}}$ formula can express any formula in $\mathrm{Bet}_{\{0\}}(x, x+c)$, and hence any bounded formula.

Our proof differs significantly to that of [13] notably at Steps 1 and 2. In [13] the authors were able to scale $\mathrm{FO}_{\mathbb{Q}}$ formulas to $\mathrm{FO}_{\{1\}}$ and then use the regularity of the integers to reduce the problem to formulas in $\mathrm{Bet}_{\{0\}}(x, x + 1)$ (so-called *unit-formulas*). For more general $\mathcal{K}$ however neither of these steps are applicable so instead we introduce a normal form for $\mathrm{FO}_{\mathcal{K}}$ formulas which simplifies the removal of the unary functions.

Step 1. Removing unary functions

Given an N-bounded $\mathrm{FO}_{\mathcal{K}}$ formula with one free variable x, we show that it is equivalent to a N'-bounded formula (over a possibly larger set of monadic predicates, suitably interpreted) in which the unary functions are only applied to x. We can remove occurrences of unary functions within the scope of monadic predicates by introducing new predicates. That is, we replace $P(y+c)$ with $P^c(y)$, the intended interpretation of P^c being $\{r : r+c \in P\}$. We will later replace $P^c(y)$ with $\Diamond_{\{c\}}P$ when completing the translation to $\mathrm{MTL}_{\mathcal{K}}$. Thus it suffices to demonstrate how to remove the unary functions from the scope of the $<$ relation. For this we introduce a normal form where all inequality constraints are replaced with interval inclusions and the intervals satisfy the following hierarchical condition: if y is quantified to $(x+c, z+c')$ then all intervals involving y and a variable that was free when y was quantified are affine translations of $(x+c, y)$ or $(y, z+c')$. We note that the results of this section apply for any additive subgroup $\mathcal{K} \subseteq \mathbb{R}$.

▶ **Definition 5.** An *interval-guarded formula* is a $\mathrm{FO}_{\mathcal{K}}$-formula such that all quantifiers are of the form $\exists x \in (y+c, y'+c')$ where y, y' are free variables and $c, c' \in \mathcal{K}$. A *Hierarchical Interval Formula (HIF)* is an interval-guarded $\mathrm{FO}_{\mathcal{K}}$-formula defined inductively as follows.

- Any $<$-free, quantifier-free $\mathrm{FO}_{\mathcal{K}}$-formula is a HIF;
- If φ_1, φ_2 are HIFs then so are $\neg\varphi_1$ and $\varphi_1 \vee \varphi_2$; and
- If $\varphi(\overline{x}, y)$ is a HIF and there exists $x_l, x_r \in \overline{x}$ and $c_l, c_r \in \mathcal{K}$ such that the only intervals in φ involving y and a free variable are of the form $(x_l + c_l + c, y + c)$ or $(y + c, x_r + c_r + c)$ for some $c \in \mathcal{K}$, then $\exists y \in (x_l + c_l, x_r + c_r).\varphi(\overline{x}, y)$ is a HIF.

Note that if $\varphi(\overline{x}, y)$ is a HIF then so is $\varphi(\overline{x}, u)$ for any term u involving variables from $\overline{x}$. As an example, consider the following interval-guarded $\mathrm{FO}_{\{1\}}$-formula:

$$\varphi(x) = \exists y \in (x, x+1).\exists z \in (y, y+1).\psi(x, y, z).$$

This is not a HIF as neither endpoint of the interval $(y, y+1)$ corresponds to an endpoint of $(x, x+1)$, the interval defining y. However, it is easy to see that $\varphi(x)$ is equivalent to:

$$\varphi'(x) = \exists y \in (x, x+1).\big(\exists z \in (y, x+1).\psi(x, y, z) \vee \psi(x, y, x+1) \vee \exists z \in (x+1, y+1).\psi(x, y, z)\big)$$

which is a HIF. Indeed, HIFs are a normal form for N-bounded $\mathrm{FO}_{\mathcal{K}}$ formulas with one free variable:

▶ **Lemma 6.** *Every N-bounded $FO_{\mathcal{K}}$ formula with one free variable is equivalent to a HIF.*

To prove this, the following property of HIFs that allows us to break up intervals will be useful:

▶ **Lemma 7.** *Let $\varphi(\overline{x}) = \exists y \in (s, t).\psi(\overline{x}, y)$ be a HIF. Then for any term u involving variables from $\overline{x}$, the following equivalence holds:*

$$u \in (s, t) \wedge \varphi(\overline{x}) \qquad \longleftrightarrow \qquad u \in (s, t) \wedge \big(\theta^{<}(\overline{x}) \vee \theta^{=}(\overline{x}) \vee \theta^{>}(\overline{x})\big)$$

where $\theta^{<}(\overline{x}) = \exists y \in (s, u).\psi^{<}(\overline{x}, y)$ and $\theta^{>}(\overline{x}) = \exists y \in (u, t).\psi^{>}(\overline{x}, y)$ are HIFs and $\theta^{=}(\overline{x})$ is a HIF with strictly smaller quantifier depth than φ.

Proof. Clearly $u \in (s, t) \wedge \varphi(\overline{x})$ is equivalent to

$$u \in (s, t) \wedge \big(\exists y \in (s, u).\psi(\overline{x}, y) \vee \psi(\overline{x}, u) \vee \exists y \in (u, t).\psi(\overline{x}, y)\big).$$

As ψ has strictly smaller quantifier depth than φ, defining $\theta^{=}(\overline{x}) = \psi(\overline{x}, u)$ suffices. We focus on the first disjunct to define $\theta^{<}$, the definition of $\theta^{>}$ from the third disjunct is analogous. We proceed by induction on the quantifier depth of ψ. If ψ is quantifier-free then it is a HIF so set $\theta^{<} = \exists y \in (s,u).\psi$. As φ is a HIF, the only inductive case that is not straightforward is if $\psi = \exists z \in (y+c, t+c).\chi(\overline{x}, y, z)$ for some $c \in \mathcal{K}$ (the case $\psi = \forall z \in (y+c, t+c).\chi(\overline{x}, y, z)$ is also handled similarly). Applying the induction hypothesis yields:

$$\begin{aligned} u \in (s,t) \wedge y \in (s,u) \wedge \psi &\equiv u \in (s,t) \wedge u + c \in (y+c, t+c) \wedge s < y \wedge \psi \\ &\equiv u \in (s,t) \wedge u + c \in (y+c, t+c) \wedge s < y \wedge \\ &\quad \big(\exists z \in (y+c, u+c).\eta^{<} \vee \eta^{=} \vee \exists z \in (u+c, t+c).\eta^{>}\big) \\ &\equiv u \in (s,t) \wedge y \in (s,u) \wedge \\ &\quad \big(\exists z \in (y+c, u+c)\eta^{<} \vee \eta^{=} \vee \exists z \in (u+c, t+c)\eta^{>}\big) \end{aligned}$$

where $\eta^{<}$, $\eta^{=}$ and $\eta^{>}$ are HIFs. It follows that $\theta^{<} = \exists y \in (s,u).\psi$ is equivalent to a HIF. ◀

We now show that every N-bounded $\mathrm{FO}_{\mathcal{K}}$ formula with one free variable is equivalent to a HIF. As with the example above, the idea is to successively break up bad intervals from the innermost quantified variables, using Lemma 7 to ensure that such breaking up does not introduce more bad intervals on already processed subformulas. To simplify the procedure, we start with a more general statement.

▶ **Lemma 8.** *Every $FO_{\mathcal{K}}$ formula $\psi(\overline{x}) \in \mathrm{Bet}_{\mathcal{K}}(x_0 - N, x_0 + N)$ is equivalent to a disjunction $\bigvee_i \big(\kappa_i(\overline{x}) \wedge \varphi_i(\overline{x})\big)$ where each κ_i is a conjunction of constraints of the form $x_j + c < x_k + c'$ and each φ_i is a HIF.*

Proof. We prove this by induction on the quantifier depth of ψ. We can remove the equality predicate by substitution (and induction on the number of variables), so for simplicity we assume that all inequalities are strict and occur within the scope of an even number of negations. In particular, we see that if the result holds for ψ then it also holds for $\neg\psi$ as negations of inequality constraints are also inequality constraints and negations of HIFs are also HIFs. Now if ψ is quantifier-free the result follows by taking a disjunctive normal form of ψ. So suppose $\psi = \exists y \varphi(\overline{x}, y)$. By the induction hypothesis we have $\varphi(\overline{x}, y)$ is equivalent to $\bigvee_i(\kappa_i(\overline{x}, y) \wedge \varphi_i(\overline{x}, y))$, so ψ is equivalent to

$$\bigvee_i (\kappa_i'(\overline{x}) \wedge \exists y \bigwedge_{j=0}^{n} y < x_j + c_j \wedge \bigwedge_{j=0}^{n} y > x_j + c_j' \wedge \varphi_i(\overline{x}, y)).$$

For technical reasons that will become clear shortly, we need to remove from each φ_i intervals of the form $(y + c, y + c')$. To do this, we observe that, by the pigeon-hole principle, $x_0 + n(c' - c) \in (y + c, y + c')$ for some $n \in \mathbb{Z}$. As $\psi \in \mathrm{Bet}_{\mathcal{K}}(x_0 - N, x_0 + N)$ we have $c - N < n(c' - c) < c' + N$, so there are a finite number of possibilities for n, and as $c, c' \in \mathcal{K}$, $n(c' - c) \in \mathcal{K}$. Thus for each interval $I = (y + c, y + c')$ occurring in φ_i we take a disjunction over all integers n in $(\frac{c-N}{c'-c}, \frac{c'+N}{c'-c})$, add the constraints $y + c < x_0 + n(c - c') < y + c'$, and use Lemma 7 to remove I. We also assume that all constraints amongst $\overline{x}$ and y implicitly defined[1] by φ_i are included in the conjunction of inequalities κ_i'.

The idea is to now take a disjunction over all possible choices for the greatest lower bound, $x_l + c_l$, and the least upper bound, $x_r + c_r'$, for y. This adds some additional constraints (e.g.

[1] For example $\exists z \in (x, y)$ implicitly implies $x < y$

$x_l + c_l > x_j + c_j$ for all $j \neq l$) which we add to κ'_i in each disjunct. Now ψ is equivalent to

$$\bigvee_{i'} (\kappa''_{i'}(\overline{x}) \wedge \exists y \in (x_l + c_l, x_r + c'_r)\, \varphi_i(\overline{x}, y)).$$

We next apply Lemma 7 to transform $\exists y \in (x_l + c_l, x_r + c'_r)\, \varphi_i(\overline{x}, y)$ into a HIF. Technically we apply it several times, once for each interval defined by free variables bounded above by $y + c$ and not bounded below by $x_l + c_l + c$ and once for each interval defined by free variables bounded below by $y + c$ and not bounded above by $x_r + c'_r$. The assumptions that there is no interval of the form $(y + c, y + c')$ and that all constraints implicitly defined by φ_i are included in κ_i together with the additional constraints imposed by the choice of x_l and x_r guarantee that $x_l + c_l + c$ is an element of any interval bounded above by $y + c$ and $x_r + c'_r + c$ is an element of any interval bounded below by $y + c$. Thus Lemma 7 guarantees that in the resulting HIF, φ'_i, all intervals involving y and some free variable are either of the form $(x_l + c_l + c, y + c)$ or $(y + c, x_r + c'_r + c)$. Thus $\exists y \in (x_l + c_l, x_r + c'_r)\, \varphi'_i(\overline{x}, y)$ is a HIF. ◀

Lemma 6 now follows as a corollary as inequality constraints over one variable can be trivially resolved.

The final stage of this step is to remove the application of unary functions to all bound variables.

▶ **Lemma 9.** *Let $\mathcal{K}$ be an additive subgroup of $\mathbb{R}$ and $\varphi(x)$ be an N-bounded $FO_{\mathcal{K}}$ formula with one free variable. Then $\varphi(x)$ is equivalent to an N'-bounded $FO_{\mathcal{K}}$ formula $\varphi'(x)$ in which the unary functions are only applied to x.*

Proof. Let us say there is a *violation* if a unary function is applied to a variable other than x. Following Lemma 6 and the comments at the start of the section it suffices to consider HIFs and remove all violations from intervals. We proceed from any maximal subformula of $\varphi(x)$, $\psi(x, \overline{y}) = \exists z \in (s, t).\theta(x, \overline{y}, z)$ where there is a violation, say $t = y_j + c$ (the case for $s = y_j + c$ being similar). Consider $\psi' = \exists z' \in (s - c, t - c).\theta(x, \overline{y}, z' + c)$. ψ' is clearly equivalent to ψ and is $(N + c)$-bounded. It suffices to show that $s - c$ is not a violation as this implies all violations in ψ' occur in proper subformulas and the result then follows by induction. The critical case is if $s = y_k + c'$. Then, as φ is a HIF and y_j and y_k are bound in φ, it follows that $j \neq k$. Suppose $j < k$. Then $y_j + c - c'$ must have been an endpoint on the interval constraining y_k at the point where y_k was quantified. As ψ is maximal, it follows that $c = c'$. Likewise if $k < j$. Therefore $s - c$ is not a violation. ◀

Step 2. Reduction to $\mathrm{Bet}_{\{0\}}(x, x + c)$ formulas

Suppose now $\varphi(x)$ is an N-bounded $\mathrm{FO}_{\mathcal{K}}$ formula in which the unary functions are only applied to x. Let $c_0 < c_1 < \ldots < c_n$ be the constants in $\mathcal{K}$ (including 0) corresponding to the unary functions that are applied to x. Let $\varphi'(\overline{z})$ be the formula resulting from replacing each term $x + c_i$ with a new variable z_i. Then $\varphi(x)$ is equivalent to $\exists \overline{z}.(z_0 < \cdots < z_n) \wedge \varphi'(\overline{z}) \wedge \bigwedge(z_i = x + c_i)$. Moreover, φ' does not contain any unary functions and is thus a formula of $\mathrm{FO}_{\{0\}}$. A standard model-theoretic argument (see [14, 6, 10]) shows that $(z_0 < \cdots < z_n) \wedge \varphi'(\overline{z})$ can be written as a finite disjunction of formulas of the form $\bigwedge_{i=0}^{n} \psi_i(z_i) \wedge \bigwedge_{i=0}^{n-1} \chi_i(z_i, z_{i+1})$ where each ψ_i is a boolean combination of monadic predicates and each $\chi_i \in \mathrm{Bet}_{\{0\}}(z_i, z_{i+1})$. Thus $\varphi(x)$ can be written as a finite disjunction of formulas of the form

$$\bigwedge_{i=0}^{n} \psi_i(x + c_i) \wedge \bigwedge_{i=0}^{n-1} \chi_i(x + c_i, x + c_{i+1}).$$

Now $\psi_i(x + c_i)$ is clearly expressible by the $\mathrm{MTL}_{\mathcal{K}}$ formula $\Diamond_{\{c_i\}}\psi_i^\dagger$, where $\psi_i^\dagger$ is the obvious translation of $\psi_i(x)$ to $\mathrm{MTL}_{\mathcal{K}}$. Likewise, if $\chi_i^\dagger$ were an $\mathrm{MTL}_{\mathcal{K}}$ formula expressing $\chi_i(x, x + c_{i+1} - c_i)$ then $\Diamond_{\{c_i\}}\chi_i^\dagger$ would be an $\mathrm{MTL}_{\mathcal{K}}$ formula expressing $\chi_i(x + c_i, x + c_{i+1})$. Thus we have reduced the problem of expressing N-bounded $\mathrm{FO}_{\mathcal{K}}$ formulas to expressing every formula in $\mathrm{Bet}_{\{0\}}(x, x + c)$.

Step 3. Expressive completeness for bounded formulas

Critical to this step is the following definition and lemma from [6].

A *decomposition formula* $\delta(x, y)$ is any formula of the form

$$\begin{aligned} x < y \wedge \exists z_0 \ldots \exists z_n \, (x = z_0 < \cdots < z_n = y) \\ \wedge \bigwedge \{\varphi_i(z_i) : 0 < i < n\} \\ \wedge \bigwedge \{\forall u \, ((z_{i-1} < u < z_i) \to \psi_i(u)) : 0 < i \leq n\} \end{aligned}$$

where φ_i and ψ_i are LTL formulas regarded as unary predicates.

▶ **Lemma 10** ([6]). *Over any domain with a complete linear order, every formula $\psi(x, y)$ in $\mathrm{Bet}_{\{0\}}(x, y)$ is equivalent to a boolean combination of decomposition formulas $\delta(x, y)$.*

It follows that it suffices to show $\mathrm{MTL}_{\mathcal{K}}$ is able to express a decomposition formula. The proof of this result very closely follows the proof in [13], so we only outline the ideas and refer the reader to the full version of the paper for the details.

▶ **Lemma 11.** *Any decomposition formula $\delta(x, x + c)$ is equivalent to an $MTL_{\mathcal{K}}$ formula.*

Proof (sketch). The proof is by induction on n, the number of existential quantifiers in $\delta(x, x + c)$. We divide the interval $(x, x + c)$ into small intervals of width $\nu \in \mathcal{K}$ where $0 < \nu \leq \frac{c}{2n}$. The fact that $\mathcal{K}$ is non-trivial and dense guarantees that ν exists. We then consider three cases depending on where the witnesses for the existential quantifiers of δ lie (taking a disjunction to cover all cases). If all witnesses lie in a single interval in the first half of $(x, x + c)$ then we can assert in $\mathrm{MTL}_{\mathcal{K}}$: ψ_1 holds until some point in the interval, then subsequent witness points occur within ν time units of the previous one. If instead all witnesses lie in a single interval in the second half of $(x, x + c)$ we assert: In c time units ψ_n would have held since a point in the interval, and each witness point was preceded within ν time units by another. Finally, if there is some k such that $x + k\nu$ separates the witnesses, we divide $\delta(x, x + c)$ into a $\mathrm{Bet}_{\{0\}}(x, x + k\nu)$ formula and a $\mathrm{Bet}_{\{0\}}(x + k\nu, x + c)$ formula and apply the inductive hypothesis. ◀

Combining Kamp's Theorem [14] and the results of this section yields:

▶ **Lemma 12.** *Let $\mathcal{K}$ be a dense additive subgroup of $\mathbb{R}$. Any N-bounded $FO_{\mathcal{K}}$ formula with one free variable is equivalent to an $MTL_{\mathcal{K}}$ formula.*

3.2 Syntactic separation of $\mathrm{MTL}_{\mathcal{K}}$

Having established that $\mathrm{MTL}_{\mathcal{K}}$ can express N-bounded $\mathrm{FO}_{\mathcal{K}}$ formulas when $\mathcal{K}$ is dense we now turn to extending the result to all $\mathrm{FO}_{\mathcal{K}}$. Our results for this section hold for all non-trivial additive subgroups $\mathcal{K}$.

The notion of *separation* was introduced by Gabbay in [4] where he showed that every LTL formula can be equivalently rewritten as a boolean combination of formulas, each of

which depends only on the past, present or future. This was later extended to LTL over the reals in [5]. Hunter, Ouaknine and Worrell [13] extended this idea for the metric setting, showing that each $\mathrm{MTL}_{\mathbb{Q}}$ formula can be equivalently rewritten as a boolean combination of formulas, each of which depends only on the distant past, bounded present, or distant future.

Here we use a similar approach; however we need to refine the definition of distant past and distant future in order to use the separation property in Section 3.3. This refinement is, however, simple enough that the proof of separability of $\mathrm{MTL}_{\mathbb{Q}}$ in [13] can largely be used and we need only indicate the two places where adjustments need to be made to account for our more general setting. The complete proof can be found in the full version of the paper.

Recall from [13] the inductive definitions of *future-reach* $fr : \mathrm{MTL}_{\mathcal{K}} \to \mathcal{K} \cup \{\infty\}$ and *past-reach* $pr : \mathrm{MTL}_{\mathcal{K}} \to \mathcal{K} \cup \{\infty\}$

- $fr(p) = pr(p) = 0$ for all propositions p,
- $fr(\mathbf{true}) = pr(\mathbf{true}) = 0$,
- $fr(\neg\varphi) = fr(\varphi)$, $pr(\neg\varphi) = pr(\varphi)$,
- $fr(\varphi \wedge \psi) = \max\{fr(\varphi), fr(\psi)\}$,
- $pr(\varphi \wedge \psi) = \max\{pr(\varphi), pr(\psi)\}$,
- If $n = \inf(I)$ and $m = \sup(I)$:
  - $fr(\varphi\ \mathbf{U}_I\ \psi) = m + \max\{fr(\varphi), fr(\psi)\}$,
  - $pr(\varphi\ \mathbf{S}_I\ \psi) = m + \max\{pr(\varphi), pr(\psi)\}$,
  - $fr(\varphi\ \mathbf{S}_I\ \psi) = \max\{fr(\varphi), fr(\psi) - n\}$,
  - $pr(\varphi\ \mathbf{U}_I\ \psi) = \max\{pr(\varphi), pr(\psi) - n\}$.

Our separation result is then:

▶ **Lemma 13.** *Let $\mathcal{K}$ be a non-trivial additive subgroup of $\mathbb{R}$. For any $c \in \mathcal{K}_{\geq 0}$, every $MTL_{\mathcal{K}}$ formula is equivalent to a boolean combination of:*

- $\Diamond_{\{N\}}\varphi$ *where* $pr(\varphi) < N - c$ *for some* $N \in \mathcal{K}$,
- $\diamondsuit_{\{N\}}\varphi$ *where* $fr(\varphi) < N - c$ *for some* $N \in \mathcal{K}$, *and*
- φ *where all intervals occurring in the temporal operators are bounded.*

Proof (sketch). The proof follows directly from the proof of the separability of $\mathrm{MTL}_{\mathbb{Q}}$ in [13] as only few assumptions were made about the underlying set of constants, which we now address.

- For the equivalence defining K^+ and K^- as bounded formulas, we instead need to use: $K^+(\varphi) \leftrightarrow \neg(\neg\varphi\ \mathbf{U}_{<\nu}\ \mathbf{true})$ and $K^-(\varphi) \leftrightarrow \neg(\neg\varphi\ \mathbf{S}_{<\nu}\ \mathbf{true})$, where $\nu \in \mathcal{K}$ is such that $\nu > 0$. Note that as $\mathcal{K}$ is non-trivial such a ν exists.
- In Step 3 (*Completing the separation*) N was chosen so that $N > pr(\theta) + 1$. Now we choose $N \in \mathcal{K}$ such that $N > pr(\theta) + c$. Note that again as $\mathcal{K}$ is non-trivial such a choice is always possible.

◀

3.3 Expressive completeness for $\mathrm{FO}_{\mathcal{K}}$

We now use Lemmas 12 and 13 to complete the proof of Theorem 1. Our argument is similar to other expressive completeness results based on separation – we refer the reader to [5, 12, 13]. Let $\varphi(x)$ be a $\mathrm{FO}_{\mathcal{K}}$ formula. We prove by induction on the quantifier depth of $\varphi(x)$ that it is equivalent to an $\mathrm{MTL}_{\mathcal{K}}$ formula.

Base case

All atoms are of the form $P_i(x)$, $x = x$, $x < x$, $x + c = x$. We replace these by P_i, **true**, **false**, **false** respectively and obtain an $\mathrm{MTL}_{\mathcal{K}}$ formula which is clearly equivalent to φ.

Inductive case

Without loss of generality we may assume $\varphi = \exists y.\psi(x,y)$. We would like to remove x from ψ. To this end we take a disjunction over all possible choices for $\gamma : \{P_1(x), \dots P_m(x)\} \to \{\mathbf{true}, \mathbf{false}\}$, and use γ to determine the value of $P_i(x)$ in each disjunct via the formula $\theta_\gamma := \bigwedge_{i=1}^{m}(P_i(x) \leftrightarrow \gamma(P_i))$. Thus we can equivalently write φ in the form $\bigvee_\gamma \left(\theta_\gamma(x) \wedge \exists y.\psi_\gamma(x,y)\right)$, where the propositions $P_i(x)$ do not appear in the ψ_γ.

Now in each ψ_γ, we may assume, after some arithmetic, x appears only in atoms of the form $x = z$, $x < z$, $x > z$ and $x + c = z$ for some variable z. We next introduce new monadic propositions $P_=$, $P_<$, $P_>$, and F_c for all c such that there is an atom $x + c = z$, and replace each of the atoms containing x in ψ_γ with the corresponding proposition. That is, $x = z$ becomes $P_=(z)$, $x < z$ becomes $P_<(z)$ and so on. This yields a formula $\psi'_\gamma(y)$ in which x does not occur, such that $\psi'_\gamma(y)$ has the same truth value as $\psi_\gamma(x,y)$ for suitable interpretations of the new propositions.

By the induction hypothesis, for each γ there is an $\mathrm{MTL}_\mathcal{K}$ formula $\theta^\dagger_\gamma$ equivalent to $\theta_\gamma(x)$, and an $\mathrm{MTL}_\mathcal{K}$ formula $\psi^\dagger_\gamma$ equivalent to $\psi'_\gamma(y)$. Then our original formula φ has the same truth value at each point x as

$$\varphi' := \bigvee_\gamma \left(\theta^\dagger_\gamma \wedge (\diamondsuit\psi^\dagger_\gamma \vee \psi^\dagger_\gamma \vee \diamondsuit\psi^\dagger_\gamma)\right)$$

for suitable interpretations of $P_=$, $P_<$, $P_>$ and the F_c.

Let $c_{\max} \in \mathcal{K}$ be the largest, in absolute value, element of $\mathcal{K}$ for which the propositional variable F_c was introduced. By Lemma 13, φ' is equivalent to a boolean combination of formulas

(I) $\diamondsuit_{\{N\}}\theta$ where $pr(\theta) < N - |c_{\max}|$,
(II) $\diamondsuit_{\{N\}}\theta$ where $fr(\theta) < N - |c_{\max}|$, and
(III) θ where all intervals occurring in the temporal operators are bounded.

Now in formulas of type (I) above, we know the intended value of each of the propositional variables $P_=, P_<, P_>$ and F_c: they are all **false** except $P_<$, which is **true**. So we can replace these propositional atoms by **true** and **false** as appropriate and obtain an equivalent $\mathrm{MTL}_\mathcal{K}$ formula which does not mention the new variables. Likewise we know the value of each of propositional variables in formulas of type (II): all are **false** except $P_>$, which is **true**; so we can again obtain an equivalent $\mathrm{MTL}_\mathcal{K}$ formula which does not mention the new variables. It remains to deal with each of the bounded formulas, θ. As $\mathrm{MTL}_\mathcal{K}$ is definable in $\mathrm{FO}_\mathcal{K}$, there exists a formula $\theta^*(x) \in \mathrm{FO}_\mathcal{K}$, with predicates from $\{P_=, P_<, P_>, F_c\}$, equivalent to θ. It is clear that as θ is bounded, there is an N such that θ^* is N-bounded. We now unsubstitute each of the introduced propositional variables. That is, replace in $\theta^*(x)$ all occurrences of $P_=(z)$ with $z = x$, all occurrences of $P_<(z)$ with $x < z$ etc. The result is an equivalent formula $\theta^+(x) \in \mathrm{FO}_\mathcal{K}$, which is still N-bounded as we have not removed any constraints on the variables of θ^*. From Lemma 12, it follows that there exists an $\mathrm{MTL}_\mathcal{K}$ formula δ that is equivalent to θ^+, i.e., equivalent to θ.

4 Expressive completeness of $\mathrm{MTL}_\mathbb{Z}$ with counting

In this section we show

▶ **Theorem 2.** *$MTL_\mathbb{Z}+C$ has the same expressive power as $FO_\mathbb{Z}$.*

In fact we show a slightly stronger result involving an extension of Q2MLO (see [10]) by punctuality quantifiers.

▶ **Definition 14.** *Q2MLO with punctuality (PQ2MLO)* is an extension of $\mathrm{FO}_{\{0\}}$ (and a restriction of $\mathrm{FO}_{\{1\}}$) defined by the following syntax:

$$\varphi ::= \mathbf{true} \mid P_i(x) \mid x < y \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x\, \varphi \mid \exists_x^{x+1} y\, \psi \mid \exists_{x-1}^{x} y\, \psi \mid \Diamond_1^x y.\, \chi \mid \Diamond\!\!\!-_1^x y.\, \chi\,,$$

where x and y denote variables, ψ denotes a PQ2MLO formula with two free variables x and y, and χ denotes a PQ2MLO formula with one free variable, y. *Q2MLO* is the restriction of PQ2MLO to formulas that do not contain the punctual quantifiers $\Diamond_1^x$ and $\Diamond\!\!\!-_1^x$.

The quantifiers $\exists_x^{x+1} y$, $\exists_{x-1}^{x} y$, $\Diamond_1^x y$ and $\Diamond\!\!\!-_1^x y$ are interpreted as $\exists y \in (x, x+1)$, $\exists y \in (x-1, x)$, $\exists y.\, (y = x+1)$ and $\exists y.\, (y = x-1)$ respectively.

▶ **Theorem 15.** *$FO_{\mathbb{Z}}$, PQ2MLO and $MTL_{\mathbb{Z}}$+C all have the same expressive power.*

It is clear that $\mathrm{FO}_{\mathbb{Z}}$ is at least as expressive as the other two. To show the equivalence of PQ2MLO and $\mathrm{MTL}_{\mathbb{Z}}$+C we use the following result of [10].

▶ **Theorem 16** ([10])**.** *MITL with counting has the same expressive power as Q2MLO.*

We also observe that if $\varphi(y)$ is a formula of PQ2MLO that is equivalent to $\varphi' \in \mathrm{MTL}_{\mathbb{Z}}$+C then $\Diamond_1^x y\varphi(y)$ is equivalent to $\Diamond_{\{1\}}\varphi'$ and $\Diamond\!\!\!-_1^x y\varphi(y)$ is equivalent to $\Diamond\!\!\!-_{\{1\}}\varphi'$. The result then follows by induction on the nesting depth of the punctual operators ($\Diamond_1^x y$ / $\Diamond\!\!\!-_1^x y$ and $\Diamond_{\{1\}}$ / $\Diamond\!\!\!-_{\{1\}}$) and Theorem 16.

It remains to show any formula in $\mathrm{FO}_{\mathbb{Z}}$ has an equivalent PQ2MLO formula. Using similar arguments to the previous section, it is sufficient to derive analogues of Lemma 11 for $\mathrm{FO}_{\{1\}}$ formulas and Lemma 13 for $\mathrm{MTL}_{\mathbb{Z}}$+C.

4.1 Expressive equivalence of bounded formulas

In order to show every bounded $\mathrm{FO}_{\mathbb{Z}}$ formula can be expressed by a PQ2MLO formula, the results of Section 3.1 imply that we need only consider $\mathrm{FO}_{\{1\}}$ formulas of the form $\delta(x) = \delta(x, x+1)$ where:

$$\begin{aligned}\delta(x,y) = \exists z_0 \ldots \exists z_n\, (x = z_0 < \cdots < z_n = y)\\ \wedge \bigwedge\{\varphi_i(z_i) : 0 < i < n\}\\ \wedge \bigwedge\{\forall u\, ((z_{i-1} < u < z_i) \to \psi_i(u)) : 0 < i \leq n\}.\end{aligned}$$

Now for $1 \leq j \leq 2n-1$ let

$$\begin{aligned}\delta_j(x,y) = \exists z_0 \ldots \exists z_k\, (x = z_0 < \cdots < z_k = y)\\ \wedge \bigwedge\{\varphi_i(z_i) : 0 < i \leq \lfloor \frac{j}{2} \rfloor\}\\ \wedge \bigwedge\{\forall u\, ((z_{i-1} < u < z_i) \to \psi_i(u)) : 0 < i \leq k\},\end{aligned}$$

where $k = \lceil \frac{j}{2} \rceil$. That is, $\delta_j(x,y)$ is the formula obtained by restricting $\delta(x)$ to the first j formulas of $\psi_1, \varphi_1, \psi_2, \varphi_2, \ldots, \psi_n$. Now consider the PQ2MLO formula:

$$\delta'(x) \quad = \quad \forall_x^{x+1}\, u.\, \bigvee_{i=1}^{2n-1} \delta_i(x,u) \;\wedge\; \Diamond_1^x y. \exists_{y-1}^{y} u. \delta(u,y).$$

The following result provides the suitable analogue of Lemma 11.

▶ **Lemma 17.** *$\delta(x)$ is equivalent to $\delta'(x)$.*

Proof. $\delta(x) \Rightarrow \delta'(x)$. Let $x_0, \ldots, x_n \in [x, x+1]$ be witnesses for the existential quantifiers in δ. From the definition of δ_i, if $u \in (x_i, x_{i+1})$ (for $0 \leq i < n$) then $\delta_{2i+1}(x, u)$ holds. Further, if $u = x_i$ (for $1 \leq i < n$) then $\delta_{2i}(x, u)$ holds. Thus the first conjunct of δ' is satisfied for all $u \in (x, x+1)$. Any $u \in (x, x_1)$ is a witness for $\exists_x^{x+1} u.\delta(u, x+1)$, and as $x_1 \leq x+1$, $u \in (y-1, y)$ where $y = x+1$. Thus the second conjunct holds and $\delta'(z)$ is satisfied.

$\delta'(x) \Rightarrow \delta(x)$. From the second conjunct of δ', $\delta(u, x+1)$ is satisfied for some $u \in (x, x+1)$. Let $x_0, \ldots, x_n \in [x, x+1]$ be the witnesses for $\delta(u, x+1)$. Now take any $v \in (x_{n-1}, x_n)$. From the first conjunct of δ', there is some $r \leq 2n-1$ such that $\delta_r(x, v)$ is satisfied. Note that if $r = 2n-1$ then as $\psi_n(x)$ holds for all $y \in (x_{n-1}, x_n)$ and $v \in (x_{n-1}, x_n)$, $\delta_{2n-1}(x, v)$ can be extended to the whole interval $[x, x+1)$, and thus $\delta(x)$ holds. So assume $r < 2n-1$, and let $x'_0, \ldots, x'_{r'} \in [x, v]$ be the witnesses for $\delta_r(x, v)$ where $r' = \lceil \frac{r}{2} \rceil < n$. Let m be the smallest index such that $x_m < x'_m$. As $x_{n-1} < v = x'_{r'}$ such an index must exist. Then we claim that $x, x'_1, \ldots, x'_{m-1}, x_m, x_{m+1}, \ldots, x_{n-1}, x+1$ are witnesses for $\delta(x)$. Every interval I defined by these witnesses, except (x'_{m-1}, x_m), is either an interval defined by witnesses of $\delta_r(x, x+1)$ or an interval defined by witnesses of $\delta(u, x+1)$, so all points in I satisfy ψ_i or φ_i as required. For the remaining interval, we observe that $(x'_{m-1}, x_m) \subseteq (x'_{m-1}, x'_m)$, thus all points satisfy ψ_{m-1} as required. Thus $\delta(x)$ is satisfied.

◀

4.2 Syntactic separation of $\mathbf{MTL}_{\mathbb{Z}}$+C

We now derive an extension of Lemma 13 for $\mathrm{MTL}_{\mathbb{Z}}$ with counting. We first extend the past-reach and future-reach functions as follows:

- $fr(\mathbf{C}_n\ \varphi) = 1 + fr(\varphi)$ and $fr(\overline{\mathbf{C}}_n\ \varphi) = fr(\varphi)$ for all n.
- $pr(\mathbf{C}_n\ \varphi) = pr(\varphi)$ and $pr(\overline{\mathbf{C}}_n\ \varphi) = 1 + pr(\varphi)$ for all n.

Our separation result for $\mathrm{MTL}_{\mathcal{K}}$ with counting is a generalization of Lemma 13 for any set $\mathcal{K}$ such that $1 \in \mathcal{K}$ (so that fr and pr are correctly defined).

▶ **Lemma 18.** *Let $\mathcal{K}$ be an additive subgroup of $\mathbb{R}$ such that $1 \in \mathcal{K}$. For any $c \in \mathcal{K}$, every $MTL_{\mathcal{K}}$+C formula is equivalent to a boolean combination of:*

- *$\Diamond_{\{N\}}\varphi$ where $pr(\varphi) < N - c$,*
- *$\overleftarrow{\Diamond}_{\{N\}}\varphi$ where $fr(\varphi) < N - c$, and*
- *φ where all intervals occurring in the temporal operators are bounded.*

The proof of Lemma 18 proceeds in the same way as the proof of Lemma 13. In that proof, the unbounded temporal operators were removed from the scope of bounded temporal operators, then the separation result of Gabbay [4] is applied treating the formulas in the scope of bounded temporal operators as atomic propositions. Here we also include formulas in the scope of the counting modalities as atomic propositions, thus it suffices to show how unbounded until and since operators can be removed from the scope of the counting modalities. This follows by induction from the following observation:

▶ **Lemma 19.** *For all $n \in \mathbb{N}$, the following equivalences and their temporal duals hold over all signals.*

(i) $\quad \mathbf{C}_1\ \varphi \quad \longleftrightarrow \quad \Diamond_1 \varphi$

$$\text{(ii)}\quad \mathbf{C}_n\,(\chi \wedge (\varphi\ \mathbf{U}\ \psi)) \quad\longleftrightarrow\quad \begin{array}{c}\mathbf{C}_n\,(\chi \wedge (\varphi\ \mathbf{U}_{<1}\ \psi))\\ \vee\\ \Diamond_{\{1\}}(\varphi \wedge (\varphi\ \mathbf{U}\ \psi)) \wedge\\ \bigvee_{k=0}^{n-1}\big[\,\mathbf{C}_k\,(\chi \wedge (\varphi\ \mathbf{U}_{<1}\ \psi))\\ \wedge\ \mathbf{C}_{n-k}\,(\chi \wedge \Box_{<1}(\varphi \wedge \neg\psi))\big]\end{array}$$

$$\text{(iii)}\quad \mathbf{C}_n\,(\chi \wedge \Box\varphi) \quad\longleftrightarrow\quad \mathbf{C}_n\,(\chi \wedge \Box_{<1}\varphi) \wedge \Box_{[1,\infty)}\varphi$$

$$\text{(iv)}\quad \mathbf{C}_n\,(\chi \wedge (\varphi\ \mathbf{S}\ \psi)) \quad\longleftrightarrow\quad \begin{array}{c}\mathbf{C}_n\,(\chi \wedge (\varphi\ \mathbf{S}_{<1}\ \psi))\\ \vee\\ (\varphi \wedge (\varphi\ \mathbf{S}\ \psi)) \wedge\\ \bigvee_{k=0}^{n-1}\big[\,\mathbf{C}_k\,(\chi \wedge (\varphi\ \mathbf{S}_{<1}\ \psi))\\ \wedge\ \mathbf{C}_{n-k}\,(\chi \wedge \boxminus_{<1}(\varphi \wedge \neg\psi))\big]\end{array}$$

$$\text{(v)}\quad \mathbf{C}_n\,(\chi \wedge \boxminus\varphi) \quad\longleftrightarrow\quad \mathbf{C}_n\,(\chi \wedge \boxminus_{<1}\varphi) \wedge \varphi \wedge \boxminus\varphi$$

4.3 Equivalence of $\mathrm{MTL}_{\mathbb{Z}}$+C, PQ2MLO and $\mathrm{FO}_{\mathbb{Z}}$

To complete the proof of Theorem 15 and hence Theorem 2 we apply the arguments of Section 3.3 together with Lemmas 17 and 18. We need only observe that $\mathrm{MTL}_{\mathbb{Z}}$+C formulas of type (III), that is where all intervals occurring in the temporal operators are bounded, are themselves bounded when translated to $\mathrm{FO}_{\mathbb{Z}}$. However, this follows directly from the definition of the counting modalities as they are defined by bounded formulas.

5 Conclusion and further work

We have given a precise characterization of the sets of timing constants $\mathcal{K}$ for which $\mathrm{MTL}_{\mathcal{K}}$ and $\mathrm{FO}_{\mathcal{K}}$ have the same expressive power. We have also shown that adding counting modalities to $\mathrm{MTL}_{\mathbb{Z}}$ (and punctuality modalities to Q2MLO) yields the full expressive power of $\mathrm{FO}_{\mathbb{Z}}$. This result can also be extended to other $\mathrm{MTL}_{\mathcal{K}}$ that are not as expressive as their first-order counterparts by adding the ability to count in the smallest definable non-zero interval (which, from the characterization, is known to exist).

Whilst most logics considered here have undecidable satisfaction tests, the cost of the translation in terms of formula size would still be an interesting area of exploration. Another area of ongoing work is an exploration of the notion of metric separation, and whether a suitable analogue of Gabbay's Theorem [4] can be derived.

References

1 R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In *REX Workshop*, volume 600 of *Lecture Notes in Computer Science*. Springer, 1991.

2 R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.

3 R. Alur and T. A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–204, 1994.

4 D. M. Gabbay. Expressive functional completeness in tense logic. In U. Monnich, editor, *Aspects of Philosophical Logic*, pages 91–117. Reidel, 1981.

5 D. M. Gabbay, I. M. Hodkinson, and M. A. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects, volume 1.* Clarendon Press, Oxford, 1994.

6 D. M. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal basis of fairness. In *Proceedings of POPL*. ACM Press, 1980.

7 T. A. Henzinger. It's about time: Real-time logics reviewed. In *Proceedings of CONCUR 98*, volume 1466 of *Lecture Notes in Computer Science*, pages 439–454. Springer, 1998.

8 T. A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In *Proceedings of ICALP 98*, volume 1443 of *Lecture Notes in Computer Science*. Springer, 1998.

9 Y. Hirshfeld and A. Rabinovich. Logics for real time: Decidability and complexity. *Fundam. Inform.*, 62(1), 2004.

10 Y. Hirshfeld and A. Rabinovich. An expressive temporal logic for real time. In *Proceedings of MFCS 06*, pages 492–504, 2006.

11 Y. Hirshfeld and A. Rabinovich. Expressiveness of metric modalities for continuous time. *Logical Methods in Computer Science*, 3(1), 2007.

12 I. M. Hodkinson and M. A. Reynolds. Separation - past, present, and future. In *We Will Show Them! (2)*, pages 117–142. College Publications, 2005.

13 P. Hunter, J. Ouaknine, and J. Worrell. Expressive completeness for metric temporal logic. In *Proceedings of LICS*, pages 349–357, 2013.

14 H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, 1968.

15 R. Koymans. Specifying real-time properties with Metric Temporal Logic. *Real-Time Systems*, 2(4), 1990.

Proving Strong Normalisation via Non-deterministic Translations into Klop's Extended λ-Calculus

Kentaro Kikuchi

RIEC, Tohoku University,
Katahira 2-1-1, Aoba-ku, Sendai 980-8577, Japan
kentaro@nue.riec.tohoku.ac.jp

Abstract

In this paper we present strong normalisation proofs using a technique of non-deterministic translations into Klop's extended λ-calculus. We first illustrate the technique by showing strong normalisation of a typed calculus that corresponds to natural deduction with general elimination rules. Then we study its explicit substitution version, the type-free calculus of which does not satisfy PSN with respect to reduction of the original calculus; nevertheless it is shown that typed terms are strongly normalising with respect to reduction of the explicit substitution calculus. In the same framework we prove strong normalisation of Sørensen and Urzyczyn's cut-elimination system in intuitionistic sequent calculus.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Strong normalisation, Klop's extended λ-calculus, Explicit substitution, Cut-elimination

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.395

1 Introduction

It is common to prove strong normalisation of a reduction system by a mapping into a set equipped with a well-founded order, e.g. $(\mathbb{N}, >)$. In the field of λ-calculus, it is also common to use a translation from terms of a calculus into λ-terms that are known to be strongly normalising, e.g. simply typed λ-terms. Such a translation is usually a (deterministic) function, and sometimes gives rise to difficulty in preserving a reduction step of the original calculus in one or more reduction steps of λ-calculus, in particular when the translation involves substitution.

In [19, 20], Lengrand developed a technique to cope with this sort of problem, where the translation from terms of the original calculus is not into λ-terms but into $\lambda I_{[,]}$-terms of [18] with additional pairing constructs. Moreover, it is defined to be non-deterministic (i.e. to be a relation rather than a function) so that an arbitrary term can be added as the second element of the pairing constructs inserted at random places. One can thus retain those terms which would disappear if translated by a function, and preserve reduction steps that take place within those terms. (For a survey on different techniques concerning $\lambda I_{[,]}$ to infer normalisation properties, see, e.g. [8].)

In this paper we first illustrate the technique by proving strong normalisation of typed terms of a calculus that corresponds to natural deduction with general elimination rules [26]. Although the same result has already been shown by different methods (e.g. [12, 22, 24]), our proof will help the reader to understand the contents of the later part of the paper with results that have not been obtained by those methods.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 395–414

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In the latter half of the paper, we apply the technique to systems with explicit substitutions [1]. We study a modification of the explicit substitution calculus introduced by Nakazawa [22], for which he mentioned the difficulty in proving strong normalisation of typed terms. We explain why the method in [22] does not work for the modified system, and prove strong normalisation of typed terms using a non-deterministic translation into $\lambda I_{[,]}$-calculus. The proof method provides a general framework for showing strong normalisation of systems with various reduction rules on the same terms, which include proof terms for intuitionistic sequent calculus. We illustrate the framework with an extension of Sørensen and Urzyczyn's cut-elimination system [27].

Since Melliès [21] gave an unexpected counter-example, strong normalisation for explicit substitution calculi has been widely studied. For composition-free systems, the methods in [6, 4, 5] are standard. They work even for type-free calculi to prove the Preservation of Strong Normalisation (PSN) property, which states that if a term is strongly normalising in the original calculus without explicit substitutions then it is also strongly normalising in the explicit substitution calculus. This property, however, does not hold for the calculi we treat in this paper. So we use techniques from [19, 20] to prove strong normalisation of typed terms in the explicit substitution calculus, without relying on the result of the original calculus. A similar proof can be found in [16] for the restricted case of proof terms for intuitionistic sequent calculus. In this paper, the definition of the non-deterministic translation is extended and improved from the one in [16]. In [27], a method closely related to the one in [16] has been developed. However, it introduces Klop's pairing constructs not only for λ-terms but also for proof terms for sequent calculus, which leads to complications.

In this paper we will refer to the modification of the system in [22] as $\lambda \mathsf{x}_g$, which makes substitution of the original calculus λ_g explicit in the style of $\lambda \mathsf{x}$ [6]. As mentioned above, the calculus $\lambda \mathsf{x}_g$ does not satisfy PSN with respect to λ_g, but it does not mean a flaw of $\lambda \mathsf{x}_g$. To put it briefly, the reason is that λ_g only implements some specific strategies. (For more details, see Remark in Subsection 3.2.)

The main subject of the paper is the technique of non-deterministic translations into $\lambda I_{[,]}$-calculus. The technique was originally developed for proving PSN of an explicit substitution calculus with composition [14], and later applied to a local cut-elimination procedure that simulates β-reduction [16]. However, the proofs for those systems are not so accessible to readers who are working in other fields. In this paper we explain the key ideas of the technique, separating them from the formalism of explicit substitution calculi. This amounts to extending the range of application of the technique, e.g. to proof of strong normalisation for $\lambda\mu$-calculus [25], solving the so-called erasing continuation problem [23]. (cf. [17])

The paper is organised as follows. In Section 2 we recall the definitions of λ_g-calculus and $\lambda I_{[,]}$-calculus, and prove strong normalisation of typed λ_g-terms. In Section 3 we extend the syntax of λ_g-calculus by explicit substitution, and prove strong normalisation of typed terms by extending the method in Section 2. In Section 4 we apply the proof method to Sørensen and Urzyczyn's cut-elimination system.

2 Strong normalisation for λ_g-calculus

This section provides a survey of the method of proving strong normalisation through a non-deterministic translation into Klop's $\lambda I_{[,]}$-calculus. Although the original formalisation by Lengrand was explained using an explicit substitution calculus with composition [14] (which is the only example to which the technique is applied in [19, 20]), here we apply the method to simply typed λ_g-calculus without explicit substitution.

2.1 λ_g-calculus

λ_g-calculus is introduced as a term calculus corresponding to natural deduction with general elimination rules [26]. It has been studied, e.g. in [11, 22]. Typed terms of the calculus can also be seen as term representation of proofs in a fragment of intuitionistic sequent calculus. We first define the syntax of the type-free version of the calculus.

▶ **Definition 1** (Grammar of λ_g)**.** The set Λ_g of terms of the λ_g-calculus is defined by the following grammar:

$$M, N, P ::= x \mid \lambda x.M \mid M[N, x.P]$$

An element of Λ_g is called a λ_g*-term*. The notions of free and bound variables are defined as usual, with an additional clause that the variable x in $M[N, x.P]$ binds the free occurrences of x in P. The set of free variables of a λ_g-term M is denoted by $\mathsf{FV}(M)$. The symbol $\equiv$ denotes syntactical equality modulo α-conversion, and $\{\_/\_\}\_$ is used for usual capture-free substitution.

▶ **Definition 2** (Reduction system of λ_g)**.** The reduction rules are:

$$\begin{array}{lrcl} (\beta_g) & (\lambda x.M)[N, y.P] & \to & \{\{N/x\}M/y\}P \\ (\pi_g) & M[N, y.P][N', y'.P'] & \to & M[N, y.P[N', y'.P']] \end{array}$$

The reduction relation $\longrightarrow_{\beta_g,\pi_g}$ is defined by the contextual closure of the rules (β_g) and (π_g). We use $\longrightarrow^{+}_{\beta_g,\pi_g}$ for its transitive closure, and $\longrightarrow^{*}_{\beta_g,\pi_g}$ for its reflexive transitive closure. The set of λ_g-terms that are strongly normalising with respect to $\longrightarrow_{\beta_g,\pi_g}$ is denoted by $\mathsf{SN}^{\beta_g,\pi_g}$. These kinds of notations are also used for the notions of other reductions in this paper.

The type assignment system for λ_g-terms is defined by the rules in Figure 1. A *typing context* is defined as a finite set of pairs $\{x_1 : A_1, \ldots, x_n : A_n\}$ where the variables are pairwise distinct. The typing context $\Gamma, x : A$ denotes the union $\Gamma \cup \{x : A\}$ where x does not appear in Γ. We write $\Gamma \vdash_{\lambda_g} M : A$ if $\Gamma \vdash M : A$ is derivable with the rules of Figure 1. We also write $\Gamma \vdash_{\lambda} t : A$ if $\Gamma \vdash t : A$ is derivable with the standard rules of the simply typed λ-calculus.

$$\frac{}{\Gamma, x : A \vdash x : A}\ (\mathsf{Var}) \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}\ (\mathsf{Abs})$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A \quad \Gamma, y : B \vdash P : C}{\Gamma \vdash M[N, y.P] : C}\ (\mathsf{GApp})$$

Figure 1 Type assignment system for λ_g-terms.

The reduction rules (β_g) and (π_g), when applied to typed terms, correspond to transformation of typing derivations. In Figure 2 we show the transformation corresponding to (π_g).

2.2 $\lambda I_{[,]}$-calculus

In this subsection we recall the definition and some properties of Klop's extended λ-calculus, which is referred to as $\lambda I_{[,]}$ in [18].

$$\dfrac{\dfrac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A \quad \Gamma, y : B \vdash P : C \to D}{\Gamma \vdash M[N, y.P] : C \to D} \qquad \Gamma \vdash N' : C \quad \Gamma, y' : D \vdash P' : E}{\Gamma \vdash M[N, y.P][N', y'.P'] : E}$$

is transformed into

$$\dfrac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A \quad \dfrac{\Gamma, y : B \vdash P : C \to D \quad \Gamma' \vdash N' : C \quad \Gamma', y' : D \vdash P' : E}{\Gamma, y : B \vdash P[N', y'.P'] : E}}{\Gamma \vdash M[N, y.P[N', y'.P']] : E}$$

where $\Gamma' = \Gamma, y : B$ and $y : B$ is added by weakening.

Figure 2 Derivation transformation corresponding to (π_g).

▶ **Definition 3** (Grammar of $\lambda I_{[,]}$)**.** The set $\Lambda I_{[,]}$ of terms of the $\lambda I_{[,]}$-calculus is defined by the following grammar:

$$T, U ::= x \mid \lambda x.T \mid T\,U \mid [T, U]$$

with the additional restriction that every abstraction $\lambda x.T$ satisfies $x \in \mathsf{FV}(T)$.

We denote lists of $\lambda I_{[,]}$-terms using vectors, and if $\overrightarrow{T} = T_1, \ldots, T_n$ then $[U, \overrightarrow{T}]$ denotes $[\ldots[U, T_1], \ldots, T_n]$ when $n \geq 1$, and U when $n = 0$.

The following property is straightforward by induction on terms.

▶ **Lemma 4** (Stability under substitution [18])**.**
If $T, U \in \Lambda I_{[,]}$, then $\{U/x\}T \in \Lambda I_{[,]}$.

▶ **Definition 5** (Reduction system of $\lambda I_{[,]}$)**.** The reduction rules are:

$$\begin{array}{lrcl} (\beta) & (\lambda x.T)\,U & \to & \{U/x\}T \\ (\pi) & [T, U]\,T' & \to & [T\,T', U] \end{array}$$

The following remark is straightforward [18]:

▶ **Lemma 6.** *If $T \longrightarrow_{\beta,\pi} T'$ then $\mathsf{FV}(T) = \mathsf{FV}(T')$ and $\{T/x\}U \longrightarrow^{+}_{\beta,\pi} \{T'/x\}U$ provided that $x \in \mathsf{FV}(U)$.*

Now we recall from [19, 20] an encoding of λ-calculus into $\lambda I_{[,]}$:

▶ **Definition 7** (Encoding of λ-calculus into $\lambda I_{[,]}$)**.** We encode the λ-calculus into $\lambda I_{[,]}$ as follows:

$$\begin{array}{lcll} \mathsf{i}(x) & := & x & \\ \mathsf{i}(\lambda x.t) & := & \lambda x.\mathsf{i}(t) & \text{if } x \in \mathsf{FV}(t) \\ \mathsf{i}(\lambda x.t) & := & \lambda x.[\mathsf{i}(t), x] & \text{if } x \notin \mathsf{FV}(t) \\ \mathsf{i}(t\,u) & := & \mathsf{i}(t)\mathsf{i}(u) & \end{array}$$

Note that this encoding is different from Klop's ι [18] in that the latter does not make the case distinction for abstractions.

A crucial property of the encoding, on which all strong normalisation results in this paper depend, is the following:

▶ **Theorem 8** ([19, 20])**.** *For any λ-term t, if $t \in \mathcal{SN}^{\beta}$ then $\mathsf{i}(t) \in \mathcal{SN}^{\beta,\pi}$.*

2.3 Strong normalisation of typed λ_g-terms

Our aim of this section is to show that all typed λ_g-terms are strong normalising with respect to β_g, π_g-reduction. The result has already been proved in several ways (e.g. [12, 22, 24]), but the strong normalisation results in later sections have not been obtained by those methods.

A naive attempt is to reduce the problem to the strong normalisation of β-reduction in the simply typed λ-calculus, using the translation $\mathcal{F}$ that maps all terms $M[N, y.P]$ into $\{\mathcal{F}(M)\mathcal{F}(N)/y\}\mathcal{F}(P)$. However, this translation does not necessarily preserve one or more reduction steps; for instance, if $M \longrightarrow_{\beta_g,\pi_g} M'$ then $M[N, y.z] \longrightarrow_{\beta_g,\pi_g} M'[N, y.z]$, but $\mathcal{F}(M[N, y.z]) \equiv z \equiv \mathcal{F}(M'[N, y.z])$. So for our purpose some modification of the translation is needed. Here we introduce the following one, taking account of the free occurrences of y in P for $M[N, y.P]$.

▶ **Definition 9** (Encoding of λ_g into λ-calculus)**.** We encode the λ_g into λ-calculus as follows:

$$\begin{array}{lcll} \mathcal{G}(x) & := & x & \\ \mathcal{G}(\lambda x.M) & := & \lambda x.\mathcal{G}(M) & \\ \mathcal{G}(M[N, y.P]) & := & \{\mathcal{G}(M)\mathcal{G}(N)/y\}\mathcal{G}(P) & \text{if } y \in \mathsf{FV}(P) \\ \mathcal{G}(M[N, y.P]) & := & (\lambda y.\mathcal{G}(P))(\mathcal{G}(M)\mathcal{G}(N)) & \text{if } y \notin \mathsf{FV}(P) \end{array}$$

Unfortunately, this encoding does not allow simulation of reduction.

▶ **Example 10.** Let $M_1 \equiv m[n, y.(\lambda x.z)[y[z, w.w], v.v]]$ and $N_1 \equiv m[n, y.z]$. Then $M_1 \longrightarrow_{\beta_g} N_1$ holds, but for their encodings $\mathcal{G}(M_1) \equiv (\lambda x.z)(mnz)$ and $\mathcal{G}(N_1) \equiv (\lambda y.z)(mn)$, the former cannot reduce to the latter.

The above encoding will be used not for simulation of reduction but for the lifting of a λ_g-term to be proved strongly normalising. For simulation, we use as the target calculus $\lambda I_{[,]}$ instead of λ-calculus, and the translation is now defined to be non-deterministic. In Figure 3 we give the inductive definition of the relation $\mathcal{H}$ between λ_g-terms and $\lambda I_{[,]}$-terms.

$$\frac{}{x \;\mathcal{H}\; x} \qquad \frac{M \;\mathcal{H}\; T \quad N \;\mathcal{H}\; U \quad P \;\mathcal{H}\; S}{M[N, y.P] \;\mathcal{H}\; \{T\,U/y\}S}\; y \in \mathsf{FV}(S)$$

$$\frac{M \;\mathcal{H}\; T}{\lambda x.M \;\mathcal{H}\; \lambda x.T}\; x \in \mathsf{FV}(T) \qquad \frac{M \;\mathcal{H}\; T}{M \;\mathcal{H}\; [T, U]}\; U \in \Lambda I_{[,]}$$

Figure 3 Relation between λ_g & $\lambda I_{[,]}$.

▶ **Lemma 11.** *If $M \;\mathcal{H}\; T$, then*

1. $\mathsf{FV}(M) \subseteq \mathsf{FV}(T)$
2. $T \in \Lambda I_{[,]}$
3. $x \notin \mathsf{FV}(M)$ *and* $U \in \Lambda I_{[,]}$ *implies* $M \;\mathcal{H}\; \{U/x\}T$
4. $\{y/x\}M \;\mathcal{H}\; \{y/x\}T$

▶ **Example 12.** $M_1 \equiv m[n, y.(\lambda x.z)[y[z, w.w], v.v]] \;\mathcal{H}\; (\lambda x.[z, x])(mnz)$ and $N_1 \equiv m[n, y.z] \;\mathcal{H}\; [z, mnz]$ as shown in Figures 4 and 5. Note that $(\lambda x.[z, x])(mnz)$ β-reduces to $[z, mnz]$ in contrast with the encodings in Example 10. The point is that yz, which corresponds to $y[z, w.w]$ discarded by β_g-reduction from M_1, is retained in the $\lambda I_{[,]}$-term $[z, yz]$ in Figure 5.

$$\dfrac{\dfrac{}{m \mathrel{\mathcal{H}} m} \quad \dfrac{}{n \mathrel{\mathcal{H}} n} \quad \dfrac{\dfrac{\dfrac{}{z \mathrel{\mathcal{H}} z}}{z \mathrel{\mathcal{H}} [z,x]}}{\lambda x.z \mathrel{\mathcal{H}} \lambda x.[z,x]} \quad \dfrac{\dfrac{}{y \mathrel{\mathcal{H}} y} \quad \dfrac{}{z \mathrel{\mathcal{H}} z} \quad \dfrac{}{w \mathrel{\mathcal{H}} w}}{y[z,w.w] \mathrel{\mathcal{H}} \{yz/w\}w} \quad \dfrac{}{v \mathrel{\mathcal{H}} v}}{(\lambda x.z)[y[z,w.w],v.v] \mathrel{\mathcal{H}} \{(\lambda x.[z,x])(yz)/v\}v}}{m[n,y.(\lambda x.z)[y[z,w.w],v.v]] \mathrel{\mathcal{H}} \{mn/y\}((\lambda x.[z,x])(yz))}$$

Figure 4 Derivation of $m[n,y.(\lambda x.z)[y[z,w.w],v.v]] \mathrel{\mathcal{H}} (\lambda x.[z,x])(mnz)$.

$$\dfrac{\dfrac{}{m \mathrel{\mathcal{H}} m} \quad \dfrac{}{n \mathrel{\mathcal{H}} n} \quad \dfrac{\dfrac{}{z \mathrel{\mathcal{H}} z}}{z \mathrel{\mathcal{H}} [z,yz]}}{m[n,y.z] \mathrel{\mathcal{H}} \{mn/y\}[z,yz]}$$

Figure 5 Derivation of $m[n,y.z] \mathrel{\mathcal{H}} [z,mnz]$.

Now our aim is to show that reduction in λ_g is simulated in $\lambda I_{[,]}$ through $\mathcal{H}$. For this we need the following lemma.

▶ **Lemma 13.** *If $M \mathrel{\mathcal{H}} T$ and $N \mathrel{\mathcal{H}} U$, then $\{N/x\}M \mathrel{\mathcal{H}} \{U/x\}T$.*

Proof. By induction on the derivation of $M \mathrel{\mathcal{H}} T$. Here we only consider the case where the last applied rule of the derivation is

$$\dfrac{M' \mathrel{\mathcal{H}} T' \quad N' \mathrel{\mathcal{H}} U' \quad P \mathrel{\mathcal{H}} S}{M'[N',y.P] \mathrel{\mathcal{H}} \{T'U'/y\}S} \; y \in \mathsf{FV}(S)$$

Then we have

$$\dfrac{\overset{\text{I.H.}}{\{N/x\}M' \mathrel{\mathcal{H}} \{U/x\}T'} \quad \overset{\text{I.H.}}{\{N/x\}N' \mathrel{\mathcal{H}} \{U/x\}U'} \quad \overset{\text{I.H.}}{\{N/x\}P \mathrel{\mathcal{H}} \{U/x\}S}}{\begin{array}{c}\{N/x\}M'[\{N/x\}N',y.\{N/x\}P] \mathrel{\mathcal{H}} \{\{U/x\}T'\{U/x\}U'/y\}\{U/x\}S \\ \| \\ \{N/x\}(M'[N',y.P]) \mathrel{\mathcal{H}} \{U/x\}\{T'U'/y\}S\end{array}}$$

◀

Now we are in a position to prove the simulation theorem in $\lambda I_{[,]}$.

▶ **Theorem 14** (Simulation in $\lambda I_{[,]}$)**.** *Suppose $M \mathrel{\mathcal{H}} T$.*

1. *If $M \longrightarrow_{\beta_g} N$ then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{+}_{\beta,\pi} U$.*
2. *If $M \longrightarrow_{\pi_g} N$ then $N \mathrel{\mathcal{H}} T$.*

Proof. By induction on the derivation of $M \mathrel{\mathcal{H}} T$.

- The case of the rule
$$\dfrac{}{x \mathrel{\mathcal{H}} x}$$
is vacuous.
- For the rule
$$\dfrac{M \mathrel{\mathcal{H}} T}{M \mathrel{\mathcal{H}} [T,U]} \; U \in \Lambda I_{[,]}$$
we simply apply the induction hypothesis.

- For the rule
$$\frac{M \mathrel{\mathcal{H}} T}{\lambda x.M \mathrel{\mathcal{H}} \lambda x.T}\ x \in \mathsf{FV}(T)$$
the reduction must take place within M, so we can apply the induction hypothesis, remembering that reduction in $\lambda I_{[,]}$ preserves free variables (Lemma 6), so the side-condition remains satisfied.
- The interesting case is
$$\frac{M \mathrel{\mathcal{H}} T \quad N \mathrel{\mathcal{H}} U \quad P \mathrel{\mathcal{H}} S}{M[N, y.P] \mathrel{\mathcal{H}} \{T\,U/y\}S}\ y \in \mathsf{FV}(S)$$
If the reduction takes place within M, N or P, we apply the induction hypothesis again, and the side-condition remains satisfied. Moreover, a β_g-reduction step in M or N is simulated by at least one reduction step from T or U and that step is preserved in the reduction of $\{T\,U/y\}S$ since $y \in \mathsf{FV}(S)$.
Otherwise, the reduction takes place at the root. We inspect the two cases, noting that the form of the last part of the derivation is determined by the redex.
1. $(\lambda x.M)[N, y.P] \longrightarrow_{\beta_g} \{\{N/x\}M/y\}P$. Then the derivation has the form
$$\dfrac{\dfrac{\dfrac{M \mathrel{\mathcal{H}} T}{\lambda x.M \mathrel{\mathcal{H}} \lambda x.T}\ x \in \mathsf{FV}(T)}{\lambda x.M \mathrel{\mathcal{H}} [\lambda x.T, \overrightarrow{R}]} \quad N \mathrel{\mathcal{H}} U \quad P \mathrel{\mathcal{H}} S}{(\lambda x.M)[N, y.P] \mathrel{\mathcal{H}} \{[\lambda x.T, \overrightarrow{R}]U/y\}S}\ y \in \mathsf{FV}(S)$$
By applying Lemma 13 twice, we have
$$\dfrac{\dfrac{\dfrac{N \mathrel{\mathcal{H}} U \quad M \mathrel{\mathcal{H}} T}{\{N/x\}M \mathrel{\mathcal{H}} \{U/x\}T}\ \text{Lemma 13}}{\{N/x\}M \mathrel{\mathcal{H}} [\{U/x\}T, \overrightarrow{R}]} \quad P \mathrel{\mathcal{H}} S}{\{\{N/x\}M/y\}P \mathrel{\mathcal{H}} \{[\{U/x\}T, \overrightarrow{R}]/y\}S}\ \text{Lemma 13}$$
Since $y \in \mathsf{FV}(S)$, we have $\{[\lambda x.T, \overrightarrow{R}]U/y\}S \longrightarrow^{+}_{\beta,\pi} \{[\{U/x\}T, \overrightarrow{R}]/y\}S$ as required.
2. $M[N, y.P][N', y'.P'] \longrightarrow_{\pi_g} M[N, y.P[N', y'.P']]$. In this case, the derivation has the form
$$\dfrac{\dfrac{\dfrac{M \mathrel{\mathcal{H}} T \quad N \mathrel{\mathcal{H}} U \quad P \mathrel{\mathcal{H}} S}{M[N, y.P] \mathrel{\mathcal{H}} \{T\,U/y\}S}\ y \in \mathsf{FV}(S)}{M[N, y.P] \mathrel{\mathcal{H}} [\{T\,U/y\}S, \overrightarrow{R}]} \quad N' \mathrel{\mathcal{H}} U' \quad P' \mathrel{\mathcal{H}} S'}{M[N, y.P][N', y'.P'] \mathrel{\mathcal{H}} \{[\{T\,U/y\}S, \overrightarrow{R}]U'/y'\}S'}\ y' \in \mathsf{FV}(S')$$
Then we have
$$\dfrac{M \mathrel{\mathcal{H}} T \quad N \mathrel{\mathcal{H}} U \quad \dfrac{\dfrac{P \mathrel{\mathcal{H}} S}{P \mathrel{\mathcal{H}} [S, \overrightarrow{R}]} \quad N' \mathrel{\mathcal{H}} U' \quad P' \mathrel{\mathcal{H}} S'}{P[N', y'.P'] \mathrel{\mathcal{H}} \{[S, \overrightarrow{R}]U'/y'\}S'}\ y' \in \mathsf{FV}(S')}{M[N, y.P[N', y'.P']] \mathrel{\mathcal{H}} \{T\,U/y\}\{[S, \overrightarrow{R}]U'/y'\}S'}\ y \in \mathsf{FV}(S'')$$
$$\Vert\!\Vert$$
$$M[N, y.P[N', y'.P']] \mathrel{\mathcal{H}} \{[\{T\,U/y\}S, \overrightarrow{R}]U'/y'\}S'$$
where $S'' \equiv \{[S, \overrightarrow{R}]U'/y'\}S'$. Since $y' \in \mathsf{FV}(S')$ and $y \in \mathsf{FV}(S)$, we have $y \in \mathsf{FV}(S'')$.
◀

To prove the strong normalisation of any typed λ_g-term, we lift it to a $\lambda I_{[,]}$-term through the encodings $\mathcal{G}$ and i.

▶ **Lemma 15.** *For any λ_g-term M, there exists a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$.*

Proof. By induction on M. (For the details, see Appendix A.) ◀

Finally we show that π_g-reduction is strongly normalising.

▶ **Lemma 16.** *$\longrightarrow_{\pi_g}$ is strongly normalising.*

Proof. We define a map $h : \Lambda_g \longrightarrow \mathbb{N}$ as follows: $h(x) := 1$, $h(\lambda x.M) := h(M)$, and $h(M[N, y.P]) := h(M) \times (h(N) + h(P))$. Then observe that if $M \longrightarrow_{\pi_g} N$ then $h(M) > h(N)$. ◀

Now we can prove the strong normalisation theorem of typed λ_g-terms.

▶ **Theorem 17** (Strong normalisation)**.**
For any λ_g-term M, if $\Gamma \vdash_{\lambda_g} M : A$ then $M \in \mathit{SN}^{\beta_g,\pi_g}$.

Proof. Suppose there is an infinite β_g, π_g-reduction sequence from M. Since π_g-reduction is strongly normalising (Lemma 16), the sequence has infinitely many β_g-reduction steps.

Now, from $\Gamma \vdash_{\lambda_g} M : A$, we have $\Gamma \vdash_{\lambda} \mathcal{G}(M) : A$, so by the strong normalisation of typed λ-terms, $\mathcal{G}(M) \in \mathsf{SN}^{\beta}$. Hence by Theorem 8, $\mathsf{i}(\mathcal{G}(M)) \in \mathsf{SN}^{\beta,\pi}$.

By Lemma 15, there is a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$. Then, applying Theorem 14 to each β_g, π_g-reduction step of the infinite reduction sequence from M, we have an infinite β, π-reduction sequence

$$T \longrightarrow^+_{\beta,\pi} T_1 \longrightarrow^+_{\beta,\pi} T_2 \longrightarrow^+_{\beta,\pi} \cdots$$

which is a contradiction. ◀

3 Strong normalisation for $\lambda \mathsf{x}_g$-calculus

In the following we extend the syntax of λ_g-calculus by explicit substitution and study properties of the calculus. Strong normalisation of typed terms is proved using an extension of the non-deterministic translation in the previous section.

3.1 $\lambda \mathsf{x}_g$-calculus

In this subsection we introduce a modification of the explicit substitution calculus in [22], which we call $\lambda \mathsf{x}_g$-calculus. As shown in [22], typed terms of the calculus are isomorphic to proofs in intuitionistic sequent calculus modulo a term quotient. First we define the syntax of the type-free calculus.

▶ **Definition 18** (Grammar of $\lambda \mathsf{x}_g$)**.** The set $\Lambda \mathsf{x}_g$ of terms of the $\lambda \mathsf{x}_g$-calculus is defined by the following grammar:

$$M, N, P ::= x \mid \lambda x.M \mid M[N, x.P] \mid \langle M/x \rangle N$$

The notions of free and bound variables are extended from those for λ_g by the clause that the variable x in $\langle M/x \rangle N$ binds the free occurrences of x in N.

▶ **Definition 19** (Reduction system of $\lambda \mathsf{x}_g$)**.** The reduction rules are:

(1)	$\langle M/x\rangle y \to y$	$(x \not\equiv y)$
(2)	$\langle M/x\rangle x \to M$	
(3)	$\langle M/x\rangle(\lambda y.N) \to \lambda y.\langle M/x\rangle N$	
(4)	$\langle M/x\rangle(y[N,z.P]) \to y[\langle M/x\rangle N, z.\langle M/x\rangle P]$	$(x \not\equiv y)$
(5)	$\langle M/x\rangle(x[N,z.P]) \to M[\langle M/x\rangle N, z.\langle M/x\rangle P]$	$(x \in \mathsf{FV}([N,z.P]))$
(6)	$\langle M/x\rangle(Q[N,z.P]) \to (\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]$	(Q is not a variable)
(7)	$\langle M/x\rangle(x[N,z.P]) \to M[N,z.P]$	$(x \notin \mathsf{FV}([N,z.P]))$
(B_1)	$(\lambda y.M)[N,z.P] \to \langle N/y\rangle\langle M/z\rangle P$	
(B_2)	$(\lambda y.M)[N,z.P] \to \langle\langle N/y\rangle M/z\rangle P$	
(Pi)	$M[N,z.P][N',z'.P'] \to M[N,z.P[N',z'.P']]$	

The reduction relation $\longrightarrow_{\lambda \mathsf{x}_g}$ is defined by the contextual closure of all the reduction rules. We define two subsystems of $\lambda \mathsf{x}_g$: the system B consists of the rules (B_1) and (B_2), and the system x consists of the rules (1)-(7) and (Pi).

▶ Remark. The reduction rules of Λ_{gx} in [22] are the rules (1)-(7), (B_1) and the following:

$$(\mathsf{Pi}') \quad M[N,z.P][N',z'.P'] \to M[N,z.\langle P/x\rangle(x[N',z'.P'])] \quad (x \notin \mathsf{FV}([N',z'.P']))$$

Note that the system x in [22] does not include the above rule (Pi′), while our system x includes the rule (Pi).

The type assignment system for $\lambda \mathsf{x}_g$-terms is defined by the rules in Figure 1 and the following:

$$\frac{\Gamma \vdash M : A \qquad \Gamma, x : A \vdash N : B}{\Gamma \vdash \langle M/x\rangle N : B}\ (\mathsf{Sub})$$

We write $\Gamma \vdash_{\lambda \mathsf{x}_g} M : A$ if $\Gamma \vdash M : A$ is derivable with those rules.

When applied to typed terms, the reduction rules correspond to transformation of typing derivations. In Figure 6 we show the transformation corresponding to (B_1) and (B_2).

$$\frac{\dfrac{\Gamma, y : A \vdash M : B}{\Gamma \vdash \lambda y.M : A \to B} \qquad \Gamma \vdash N : A \qquad \Gamma, z : B \vdash P : C}{\Gamma \vdash (\lambda y.M)[N,z.P] : C}$$

is transformed into

$$\frac{\Gamma \vdash N : A \qquad \dfrac{\Gamma, y : A \vdash M : B \qquad \Gamma, y : A, z : B \vdash P : C}{\Gamma, y : A \vdash \langle M/z\rangle P : B}}{\Gamma \vdash \langle N/y\rangle\langle M/z\rangle P : C}\ \text{by } (\mathsf{B}_1)$$

and

$$\frac{\dfrac{\Gamma \vdash N : A \qquad \Gamma, y : A \vdash M : B}{\Gamma \vdash \langle N/y\rangle M : B} \qquad \Gamma, z : B \vdash P : C}{\Gamma \vdash \langle\langle N/y\rangle M/z\rangle P : C}\ \text{by } (\mathsf{B}_2)$$

Figure 6 Derivation transformation corresponding to (B_1) and (B_2).

3.2 Failure of PSN with respect to β_g, π_g-reduction

The main result of this paper is the strong normalisation theorem of typed $\lambda \mathsf{x}_g$-terms. It has been proved by Nakazawa [22] for the case where the reduction system does not include the

rule (B_2). He also mentioned the difficulty in proving strong normalisation in the presence of (B_2). In this subsection we explain why the method in [22] does not work for the system with (B_2).

A standard method of proving strong normalisation of explicit substitution calculi [6, 4, 5] uses projection onto normal forms of the substitution subcalculus. Those normal forms are terms without explicit substitution, and the proof relies on the strong normalisation result of the original calculus without explicit substitution. Such a proof works even for type-free calculi to show the property called Preservation of Strong Normalisation (PSN), which states that if a term is strongly normalising with respect to reduction of the original calculus then it is also strongly normalising in the explicit substitution calculus. However, this property does not hold between λ_g-calculus and $\lambda\mathsf{x}_g$-calculus.

▶ **Example 20.** $\lambda\mathsf{x}_g$-calculus does not satisfy PSN with respect to β_g, π_g-reduction as the following example shows. Let $\omega \equiv \lambda y.y[y, v.v]$. Then

$$\omega[\omega, z.x] \longrightarrow_{\beta_g} \{\{\omega/y\}(y[y, v.v])/z\}x \equiv x$$

Since this is the only β_g, π_g-reduction sequence from $\omega[\omega, z.x]$, it is in $\mathsf{SN}^{\beta_g,\pi_g}$. However,

$$\begin{array}{ll} \omega[\omega, z.x] & \longrightarrow_{\mathsf{B}_2} \langle\langle\omega/y\rangle(y[y, v.v])/z\rangle x \\ & \longrightarrow^*_{\lambda\mathsf{x}_g} \langle\omega[\omega, v.v]/z\rangle x \\ & \longrightarrow_{\mathsf{B}_2} \cdots \end{array}$$

Hence $\omega[\omega, z.x] \notin \mathsf{SN}^{\lambda\mathsf{x}_g}$.

In spite of the above fact, strong normalisation of typed $\lambda\mathsf{x}_g$-terms may be proved, but then one cannot use a proof method that would yield at the same time PSN of the type-free calculus with respect to β_g, π_g-reduction. Specifically, a standard method as in [22], which projects $\lambda\mathsf{x}_g$-terms onto λ_g-terms and relies on the result of strong normalisation of β_g, π_g-reduction, does not work.

▶ Remark. An intended meaning of the β_g-rule $(\lambda y.M)[N, z.P] \to \{\{N/y\}M/z\}P$ of λ_g-calculus is that the function $\lambda y.M$ is applied to the argument N and then the result of the application is passed to the continuation $z.P$. In the type-free case, the computation of the application may not produce any result as seen in the example above, but even so, the term $\{N/y\}M$ is substituted for z in P; in particular, when z does not occur free in P, the term $\{N/y\}M$ is discarded. This means that λ_g-calculus can not express a natural operational semantics that passes to the continuation the result of the application after computing it, but only implement some specific strategies. On the other hand, $\lambda\mathsf{x}_g$-calculus and other formalisms like $\overline{\lambda}\mu\widetilde{\mu}$ [7] allow for such a natural operational semantics. (Those calculi do not satisfy PSN with respect to β_g, π_g-reduction, but they satisfy PSN with respect to β-reduction in an isomorphic image of the λ-calculus through appropriate embeddings.)

3.3 Strong normalisation of typed $\lambda\mathsf{x}_g$-terms

Our proof of strong normalisation of typed $\lambda\mathsf{x}_g$-terms proceeds in a similar pattern to Section 2 except for the treatment of explicit substitution. To deal with explicit substitution we use another technique from [19, 20] with the notions of safe and minimal reductions.

▶ **Definition 21.** A reduction step is *minimal* if every proper subterm of the redex is in $\mathsf{SN}^{\lambda\mathsf{x}_g}$. A minimal reduction step is *safe* if the redex itself is in $\mathsf{SN}^{\lambda\mathsf{x}_g}$, and *unsafe* if not.

We can restrict infinite $\lambda \mathsf{x}_g$-reduction sequences to those consisting only of minimal reduction steps.

▶ **Lemma 22.** *If $M \notin \mathit{SN}^{\lambda \mathsf{x}_g}$ then there exists an infinite $\lambda \mathsf{x}_g$-reduction sequence starting from M such that all the reduction steps are minimal.*

Proof. It suffices to take in each reduction step an innermost redex of the ones that preserve the possibility of infinite reduction. (Such a reduction sequence is called a minimal infinite reduction sequence, e.g. in [3].) ◀

▶ **Definition 23.** Let h be a subsystem of $\lambda \mathsf{x}_g$, and let $M \longrightarrow_{\mathsf{h}} N$. We write $M \longrightarrow_{\mathsf{minh}} N$ (resp. $M \longrightarrow_{\mathsf{safeh}} N$) to denote that the reduction step $M \longrightarrow_{\mathsf{h}} N$ is minimal (resp. safe) (where minimality is with respect to $\mathsf{SN}^{\lambda \mathsf{x}_g}$ and not for the subsystem h).

A crucial point of our proof is that we divide minimal reduction steps into two kinds: One is those which are simulated in $\lambda I_{[,]}$ so that one or more reduction steps are preserved, as β_g-reduction steps in Section 2. The other is those which are strongly normalising in $\lambda \mathsf{x}_g$ and simulated in $\lambda I_{[,]}$ where one or more reduction steps are not necessarily preserved, as π_g-reduction steps in Section 2. In this section we take unsafe B-reduction steps as the former and the rest (i.e. reduction steps by $\longrightarrow_{\mathsf{safeB,minx}}$) as the latter.

To show that $\longrightarrow_{\mathsf{safeB,minx}}$ is strongly normalising, we briefly recall the lexicographic path ordering [13]. For a more detailed description and proofs, the reader is referred to, e.g. [2].

▶ **Definition 24** (Lexicographic path ordering)**.** Let $\succ$ be a transitive and irreflexive ordering on the set of function symbols in a first-order signature, and let $s \equiv f(s_1, \ldots, s_m)$ and $t \equiv g(t_1, \ldots, t_n)$ be terms over the signature. Then $s >_{\mathsf{lpo}} t$, if one of the following holds:

1. $s_i \equiv t$ or $s_i >_{\mathsf{lpo}} t$ for some $i = 1, \ldots, m$,
2. $f \succ g$ and $s >_{\mathsf{lpo}} t_j$ for all $j = 1, \ldots, n$,
3. $f \equiv g$, $s >_{\mathsf{lpo}} t_j$ for all $j = 1, \ldots, n$, and $s_1 \equiv t_1, \ldots, s_{i-1} \equiv t_{i-1}, s_i >_{\mathsf{lpo}} t_i$ for some $i = 1, \ldots, m$.

▶ **Theorem 25.** *$>_{\mathsf{lpo}}$ is well-founded if and only if $\succ$ is well-founded.*

Now we encode $\lambda \mathsf{x}_g$-terms into a first-order syntax given by the following ordered infinite signature:

$$\mathsf{sub}(\_,\_) \succ \mathsf{gapp}(\_,\_,\_) \succ \mathsf{abs}(\_) \succ \mathsf{c}^{(m,n)}$$

where for every $m, n \in \mathbb{N}$, there is a constant $\mathsf{c}^{(m,n)}$. Those constants are all below $\mathsf{abs}(\_)$, and the precedence between them is given by $\mathsf{c}^{(m,n)} \succ \mathsf{c}^{(m',n')}$ if $(m,n) > (m',n')$ lexicographically. Then the precedence relation is well-founded, and so $>_{\mathsf{lpo}}$ induced on the first-order terms is also well-founded.

Let M be a $\lambda \mathsf{x}_g$-term with $M \in \mathsf{SN}^{\lambda \mathsf{x}_g}$. We define $w(M)$ as $(\mathsf{maxred}(M), |M|)$ where $\mathsf{maxred}(M)$ is the maximal length of all $\lambda \mathsf{x}_g$-reduction sequences starting from M, and $|M|$ is the size of M. Then the aforementioned encoding is given in Figure 7.

▶ **Lemma 26.** *If $M \longrightarrow_{\mathsf{safeB,minx}} M'$ then $\overline{M} >_{\mathsf{lpo}} \overline{M'}$. Hence, $\longrightarrow_{\mathsf{safeB,minx}}$ is strongly normalising.*

Proof. By induction on the derivation of the reduction step. For the details, see Appendix B. ◀

The relation $\mathcal{H}$ between $\lambda \mathsf{x}_g$-terms and $\lambda I_{[,]}$-terms is inductively defined by the rules in Figure 3 and the following:

$$\frac{M \mathrel{\mathcal{H}} T \qquad N \mathrel{\mathcal{H}} U}{\langle M/x \rangle N \mathrel{\mathcal{H}} \{T/x\}U} \; x \in \mathsf{FV}(U) \vee M \in \mathsf{SN}^{\lambda \mathsf{x}_g}$$

$$
\begin{array}{lcll}
\overline{M} & := & \mathsf{c}^{w(M)} & \text{if } M \in \mathsf{SN}^{\lambda \mathsf{x}_g} \\
\text{otherwise} & & & \\
\overline{\lambda x.M} & := & \mathsf{abs}(\overline{M}) & \\
\overline{M[N, y.P]} & := & \mathsf{gapp}(\overline{M}, \overline{N}, \overline{P}) & \\
\overline{\langle M/x \rangle N} & := & \mathsf{sub}(\overline{M}, \overline{N}) &
\end{array}
$$

Figure 7 Encoding of $\lambda \mathsf{x}_g$ into a first-order syntax.

The side-condition of the above rule is designed so that an unsafe B-reduction step in M is simulated by at least one reduction step in $\{T/x\}U$ (cf. the first paragraph of page 413). It is also closely related to the notion of decent term in [27, Definition 4.4]. Note that in the presence of the above rule, Lemma 11 (0a) no longer holds.

▶ **Theorem 27** (Simulation in $\lambda I_{[,]}$)**.** *Suppose $M \mathrel{\mathcal{H}} T$.*

1. *If $M \longrightarrow_{\mathit{minB}} N$ and the reduction step is unsafe then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{+}_{\beta,\pi} U$.*
2. *If $M \longrightarrow_{\mathit{minB}} N$ and the reduction step is safe then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{*}_{\beta,\pi} U$.*
3. *If $M \longrightarrow_{\mathit{minx}} N$ then $N \mathrel{\mathcal{H}} T$.*

Proof. By induction on the derivation of $M \mathrel{\mathcal{H}} T$. A detailed proof is found in Appendix B. ◀

▶ **Definition 28** (Encoding of $\lambda \mathsf{x}_g$ into λ-calculus)**.** We encode $\lambda \mathsf{x}_g$ into λ-calculus, extending the definition of $\mathcal{G}$ (Definition 9) by

$$
\begin{array}{lcll}
\mathcal{G}(\langle M/x \rangle N) & := & \{\mathcal{G}(M)/x\}\mathcal{G}(N) & \text{if } x \in \mathsf{FV}(N) \\
\mathcal{G}(\langle M/x \rangle N) & := & (\lambda x.\mathcal{G}(N))\,\mathcal{G}(M) & \text{if } x \notin \mathsf{FV}(N)
\end{array}
$$

As in the previous section, we lift any $\lambda \mathsf{x}_g$-term to a $\lambda I_{[,]}$-term through $\mathcal{G}$ and i.

▶ **Lemma 29.** *For any $\lambda \mathsf{x}_g$-term M, there exists a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^{*}_{\beta,\pi} T$.*

Proof. By induction on M. (For the details, see Appendix B.) ◀

Now we can prove the strong normalisation theorem of typed $\lambda \mathsf{x}_g$-terms.

▶ **Theorem 30** (Strong normalisation)**.**
For any $\lambda \mathsf{x}_g$-term M, if $\Gamma \vdash_{\lambda \mathsf{x}_g} M : A$ then $M \in \mathit{SN}^{\lambda \mathsf{x}_g}$.

Proof. Suppose $M \notin \mathsf{SN}^{\lambda \mathsf{x}_g}$. Then by Lemma 22, there exists an infinite $\lambda \mathsf{x}_g$-reduction sequence starting from M such that all the reduction steps are minimal. Since $\longrightarrow_{\mathsf{safeB},\mathsf{minx}}$ is strongly normalising (Lemma 26), the sequence has infinitely many unsafe B-reduction steps.

Now, from $\Gamma \vdash_{\lambda \mathsf{x}_g} M : A$, we have $\Gamma \vdash_{\lambda} \mathcal{G}(M) : A$, so by the strong normalisation of typed λ-terms, $\mathcal{G}(M) \in \mathsf{SN}^{\beta}$. Hence by Theorem 8, $\mathsf{i}(\mathcal{G}(M)) \in \mathsf{SN}^{\beta,\pi}$.

By Lemma 29, there is a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^{*}_{\beta,\pi} T$. Then, applying Theorem 27 to each minimal reduction step of the infinite $\lambda \mathsf{x}_g$-reduction sequence from M, we have an infinite β,π-reduction sequence, which is a contradiction. ◀

4 Application to other systems

The proof method in the previous section provides a general framework for showing strong normalisation of systems on $\lambda\mathsf{x}_g$-terms with various reduction rules. In this section we illustrate that with an extension of Sørensen and Urzyczyn's cut-elimination system in intuitionistic sequent calculus [27].

▶ **Definition 31** (Reduction system of $\lambda\mathsf{x}_g^{\mathsf{SU}}$)**.** The reduction rules of $\lambda\mathsf{x}_g^{\mathsf{SU}}$ are the rules (1)-(4) of $\lambda\mathsf{x}_g$ (Definition 19) and the following:

(8) $\langle y/x\rangle(x[N,z.P]) \to \langle y/x\rangle(y[N,z.P])$
(9) $\langle y[N,z.P]/x\rangle(x[N',z'.P']) \to y[N,z.\langle P/x\rangle(x[N',z'.P'])]$
(B_3) $\langle \lambda y.M/x\rangle(x[N,z.P]) \to \langle \lambda y.M/x\rangle\langle\langle N/y\rangle M/z\rangle P$

The reduction relation $\longrightarrow_{\lambda\mathsf{x}_g^{\mathsf{SU}}}$ is defined by the contextual closure of those reduction rules. The subsystem x^{SU} consists of the rules (1)-(4), (8) and (9).

▶ Remark. The reduction rules of the system in [27, page 920] are the same as those of $\lambda\mathsf{x}_g^{\mathsf{SU}}$, but the terms are restricted to those such that M is a variable in $M[N,y.P]$.

The notions of minimal, safe and unsafe reductions and the encoding into the first-order syntax are defined as in the case of $\lambda\mathsf{x}_g$. We define $d(M)$ as the number of subterms of M that have the form $\langle y/x\rangle(x[N,z.P])$. Then we can prove the following lemma and the simulation theorem.

▶ **Lemma 32.** *Let* $\mathsf{h} := \mathsf{safeB}_3, \mathsf{minx}^{\mathsf{SU}}$. *If* $M \longrightarrow_{\mathsf{h}} M'$ *then* $\overline{M} >_{\mathsf{lpo}} \overline{M'}$ *or* $\overline{M} = \overline{M'}$ *and* $d(M) > d(M')$. *Hence,* $\longrightarrow_{\mathsf{h}}$ *is strongly normalising.*

Proof. By induction on the derivation of the reduction step. ◀

▶ **Theorem 33** (Simulation in $\lambda I_{[,]}$)**.** *Suppose* $M \mathcal{H} T$.

1. *If* $M \longrightarrow_{\mathsf{minB}_3} N$ *and the reduction step is unsafe then there exists* U *such that* $N \mathcal{H} U$ *and* $T \longrightarrow^{+}_{\beta,\pi} U$.
2. *If* $M \longrightarrow_{\mathsf{minB}_3} N$ *and the reduction step is safe then there exists* U *such that* $N \mathcal{H} U$ *and* $T \longrightarrow^{*}_{\beta,\pi} U$.
3. *If* $M \longrightarrow_{\mathsf{minx}^{\mathsf{SU}}} N$ *then* $N \mathcal{H} T$.

Proof. By induction on the derivation of $M \mathcal{H} T$. ◀

Using the above lemma and theorem, we can prove strong normalisation of typed $\lambda\mathsf{x}_g$-terms with respect to reduction of $\lambda\mathsf{x}_g^{\mathsf{SU}}$.

▶ **Theorem 34** (Strong normalisation)**.**
For any $\lambda\mathsf{x}_g$*-term* M, *if* $\Gamma \vdash_{\lambda\mathsf{x}_g} M : A$ *then* $M \in \mathcal{SN}^{\lambda\mathsf{x}_g^{\mathsf{SU}}}$.

Proof. Similar to the proof of Theorem 30. ◀

Since proof terms for intuitionistic sequent calculus have the same type in the type assignment system of [27] and in ours, it follows that the cut-elimination procedure is strongly normalising.

As we have seen, in our framework, proving strong normalisation of systems with various reduction rules on $\lambda\mathsf{x}_g$-terms consists in

- taking an appropriate subsystem h that is strongly normalising, as in Lemma 32
- proving the simulation theorem in $\lambda I_{[,]}$

In the case of $\lambda\mathsf{x}_g$ and $\lambda\mathsf{x}_g^{\mathsf{SU}}$, we can in fact prove a stronger result than Theorems 30 and 34 that typed $\lambda\mathsf{x}_g$-terms are strongly normalising with respect to $\longrightarrow_{\lambda\mathsf{x}_g,\lambda\mathsf{x}_g^{\mathsf{SU}}}$, taking $\mathsf{h} := \mathsf{safe}(\mathsf{B},\mathsf{B}_3),\mathsf{min}(\mathsf{x},\mathsf{x}^{\mathsf{SU}})$.

5 Conclusion and related work

We have presented proofs of strong normalisation of typed terms using non-deterministic translations into Klop's $\lambda I_{[,]}$-calculus. The method has worked for the explicit substitution calculus in [22] extended with the rule (B_2) as well as the cut-elimination system in [27]. The proof method provides a general framework for showing strong normalisation of various reduction systems on $\lambda\mathsf{x}_g$-terms.

As regards related work, the CGPS-translation [22, 9] has been used for proving strong normalisation of calculi that correspond to proof systems with general elimination rules. (Those calculi do not have step-by-step reduction of explicit substitutions.) It aims to simulate every reduction step of the calculi by at least one β-reduction step in the λ-calculus. On the other hand, the method in this paper makes such reduction steps as few as possible, i.e., in the case of $\lambda\mathsf{x}_g$, only unsafe B-reduction steps have to be simulated by at least one β,π-reduction step in $\lambda I_{[,]}$ (cf. the remark after Definition 23). This is an essential part of our proof of strong normalisation for explicit substitution calculi.

The cut-elimination system in [27] is not intended to simulate β-reduction and was so far difficult to classify among, and relate to, other cut-elimination procedures for sequent calculus. Our proof of strong normalisation in the general framework helps to shed some light on such an exotic system. The strong normalisation proof in [27] introduces Klop's pairing constructs not only for λ-terms but also for proof terms for sequent calculus. This leads to complications, and in this sense, our approach is simpler than theirs.

Recent work by Espírito Santo and Pinto [10] has introduced some variants of intuitionistic sequent calculi (without step-by-step reduction of explicit substitutions). Reduction of those calculi is directly simulated by the explicit substitution calculus in [15], so that strong normalisation of the calculi follows from that of the calculus in [15]. On the other hand, the explicit substitution calculi we studied in this paper do not seem to be directly simulated by the calculus in [15], since it is not easy to simulate a local cut-elimination procedure in sequent calculus by an explicit substitution calculus for the usual λ-calculus.

Acknowledgements. I would like to thank the anonymous reviewers for valuable comments. I also thank Stéphane Lengrand for valuable discussions. The figures of the derivations have been drawn using Makoto Tatsuta's `proof.sty` macros.

References

1 M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *J. Funct. Programming*, 1(4):375–416, 1991.

2 F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

3 Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *J. Funct. Programming*, 6(5):699–722, 1996.

4 R. Bloo. *Preservation of Termination for Explicit Substitution.* PhD thesis, Eindhoven University of Technology, 1997.

5 R. Bloo and H. Geuvers. Explicit substitution: On the edge of strong normalization. *Theoret. Comput. Sci.*, 211(1-2):375–395, 1999.

6 R. Bloo and K. H. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *Proc. of CSN'95 (Computing Science in the Netherlands)*, 62–72, 1995.

7 P.-L. Curien and H. Herbelin. The duality of computation. In *Proc. of ICFP'00*, 233–243, 2000.

8 I. L. Gørtz, S. Reuss, and M. H. Sørensen. Strong normalization from weak normalization by translation into the lambda-I-calculus. *Higher-Order and Symbolic Computation*, 16(3):253–285, 2003.

9 J. Espírito Santo, R. Matthes, and L. Pinto. Continuation-passing style and strong normalisation for intuitionistic sequent calculi. *Logical Methods in Computer Science*, 5(2), 2009.

10 J. Espírito Santo and L. Pinto. A calculus of multiary sequent terms. *ACM Trans. Comput. Log.*, 12(3):22, 2011.

11 F. Joachimski and R. Matthes. Standardization and confluence for a lambda calculus with generalized applications. In *Proc. of RTA'00*, LNCS 1833, 141–155, 2000.

12 F. Joachimski and R. Matthes. Short proofs of normalization for the simply-typed λ-calculus, permutative conversions and Gödel's T. *Arch. Math. Log.*, 42(1):59–87, 2003.

13 S. Kamin and J.-J. Lévy. Attempts for generalizing the recursive path orderings. 1980. Handwritten paper, University of Illinois.

14 D. Kesner and S. Lengrand. Resource operators for λ-calculus. *Inform. and Comput.*, 205(4):419–473, 2007.

15 D. Kesner A theory of explicit substitutions with safe and full composition. *Logical Methods in Computer Science*, 5(3), 2009.

16 K. Kikuchi and S. Lengrand. Strong normalisation of cut-elimination that simulates β-reduction. In *Proc. of FoSSaCS'08*, LNCS 4962, 380–394, 2008.

17 K. Kikuchi. Non-deterministic CPS-translation for $\lambda\mu$-calculus. 2013. Manuscript. Available at `http://www.nue.riec.tohoku.ac.jp/user/kentaro/cpslm`.

18 J. W. Klop. *Combinatory Reduction Systems*, volume 127 of *Mathematical Centre Tracts*. CWI, 1980. PhD Thesis.

19 S. Lengrand. Induction principles as the foundation of the theory of normalisation: concepts and techniques. Technical report, Université Paris 7, March 2005. Available at `http://hal.ccsd.cnrs.fr/ccsd-00004358`.

20 S. Lengrand. *Normalisation & Equivalence in Proof Theory & Type Theory*. PhD thesis, Université Paris 7 & University of St Andrews, 2006.

21 P.-A. Melliès. Typed λ-calculi with explicit substitution may not terminate. In *Proc. of TLCA'95*, LNCS 902, 328–334, 1995.

22 K. Nakazawa. An isomorphism between cut-elimination procedure and proof reduction. In *Proc. of TLCA'07*, LNCS 4583, 336–350, 2007.

23 K. Nakazawa and M. Tatsuta. Strong normalization proof with CPS-translation for second order classical natural deduction. *J. of Symbolic Logic*, 68(3):851–859, 2003. Corrigendum: vol. 68 (2003), no. 4, pp. 1415–1416.

24 K. Nakazawa and M. Tatsuta. Strong normalization of classical natural deduction with disjunctions. *Ann. Pure Appl. Logic*, 153(1–3):21–37, 2008.

25 M. Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *Proc. of LPAR'92*, LNCS 624, 190–201, 1992.

26 J. von Plato. Natural deduction with general elimination rules. *Arch. Math. Log.*, 40(7):541–567, 2001.

27 M. H. Sørensen and P. Urzyczyn. Strong cut-elimination in sequent calculus using Klop's ι-translation and perpetual reductions. *J. of Symbolic Logic*, 73(3):919–932, 2008.

A Proof in Section 2

In this appendix we give a proof of Lemma 15 in Section 2.

First, note the following facts:

- For any λ_g-term M, $\mathsf{FV}(\mathcal{G}(M)) = \mathsf{FV}(M)$.
- For any λ-term t, $\mathsf{FV}(\mathsf{i}(t)) = \mathsf{FV}(t)$.
- For any λ-terms t and u, $\mathsf{i}(\{u/x\}t) = \{\mathsf{i}(u)/x\}\mathsf{i}(t)$.

▶ **Lemma 15.** *For any λ_g-term M, there exists a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$.*

Proof. By induction on M. The case where M is a variable is straightforward. We consider the remaining two cases.

- For $\lambda x.M$, we have

$$\mathsf{i}(\mathcal{G}(\lambda x.M)) = \mathsf{i}(\lambda x.\mathcal{G}(M)) = \begin{cases} \lambda x.\mathsf{i}(\mathcal{G}(M)) & \text{if } x \in \mathsf{FV}(\mathcal{G}(M)) \\ \lambda x.[\mathsf{i}(\mathcal{G}(M)), x] & \text{if } x \notin \mathsf{FV}(\mathcal{G}(M)) \end{cases}$$

  By the induction hypothesis, there is a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$.
  - If $x \in \mathsf{FV}(\mathcal{G}(M))$ then $\mathsf{i}(\mathcal{G}(\lambda x.M)) = \lambda x.\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} \lambda x.T$, and since $\mathsf{FV}(\mathcal{G}(M)) = \mathsf{FV}(\mathsf{i}(\mathcal{G}(M))) = \mathsf{FV}(T)$, we have $x \in \mathsf{FV}(T)$. From $M \mathrel{\mathcal{H}} T$, we have $\lambda x.M \mathrel{\mathcal{H}} \lambda x.T$.
  - If $x \notin \mathsf{FV}(\mathcal{G}(M))$ then $\mathsf{i}(\mathcal{G}(\lambda x.M)) = \lambda x.[\mathsf{i}(\mathcal{G}(M)), x] \longrightarrow^*_{\beta,\pi} \lambda x.[T, x]$. From $M \mathrel{\mathcal{H}} T$, we have $M \mathrel{\mathcal{H}} [T, x]$, and hence $\lambda x.M \mathrel{\mathcal{H}} \lambda x.[T, x]$.
- For $M[N, y.P]$, we have

$$\mathsf{i}(\mathcal{G}(M[N, y.P])) = \begin{cases} \{\mathsf{i}(\mathcal{G}(M))\,\mathsf{i}(\mathcal{G}(N))/y\}\mathsf{i}(\mathcal{G}(P)) & \text{if } y \in \mathsf{FV}(P) \\ (\lambda y.[\mathsf{i}(\mathcal{G}(P)), y])(\mathsf{i}(\mathcal{G}(M))\,\mathsf{i}(\mathcal{G}(N))) & \text{if } y \notin \mathsf{FV}(P) \end{cases}$$

  By the induction hypothesis, there are $\lambda I_{[,]}$-terms T, U and S such that (a) $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$, (b) $N \mathrel{\mathcal{H}} U$ and $\mathsf{i}(\mathcal{G}(N)) \longrightarrow^*_{\beta,\pi} U$, and (c) $P \mathrel{\mathcal{H}} S$ and $\mathsf{i}(\mathcal{G}(P)) \longrightarrow^*_{\beta,\pi} S$.
  - If $y \in \mathsf{FV}(P)$ then $\mathsf{i}(\mathcal{G}(M[N, y.P])) = \{\mathsf{i}(\mathcal{G}(M))\,\mathsf{i}(\mathcal{G}(N))/y\}\mathsf{i}(\mathcal{G}(P)) \longrightarrow^*_{\beta,\pi} \{T\,U/y\}S$. Since $\mathsf{FV}(P) = \mathsf{FV}(\mathsf{i}(\mathcal{G}(P))) = \mathsf{FV}(S)$, we have $y \in \mathsf{FV}(S)$. Hence, from (a), (b) and (c), we have $M[N, y.P] \mathrel{\mathcal{H}} \{T\,U/y\}S$.
  - If $y \notin \mathsf{FV}(P)$ then $\mathsf{i}(\mathcal{G}(M[N, y.P])) = (\lambda y.[\mathsf{i}(\mathcal{G}(P)), y])(\mathsf{i}(\mathcal{G}(M))\,\mathsf{i}(\mathcal{G}(N))) \longrightarrow^*_{\beta,\pi} (\lambda y.[S, y])(T\,U) \longrightarrow_\beta \{T\,U/y\}[S, y]$. From (c), we have $P \mathrel{\mathcal{H}} [S, y]$ and hence $M[N, y.P] \mathrel{\mathcal{H}} \{T\,U/y\}[S, y]$.

◀

B Proofs in Section 3

In this appendix we give proofs of Lemmas 26 and 29, and Theorem 27 in Section 3.

In the proof below we use the following fact:

- If N is a proper subterm of M then $w(M) > w(N)$ and hence $\mathsf{c}^{w(M)} > \mathsf{c}^{w(N)}$.

▶ **Lemma 26.** *If $M \longrightarrow_{\mathsf{safeB},\mathsf{minx}} M'$ then $\overline{M} >_{\mathsf{lpo}} \overline{M'}$. Hence, $\longrightarrow_{\mathsf{safeB},\mathsf{minx}}$ is strongly normalising.*

Proof. By induction on the derivation of the reduction step. First we consider the cases where the reduction takes place at the root. If the reduction step is safe, i.e. if the redex itself is in $\mathsf{SN}^{\lambda\mathsf{x}_g}$, then $\overline{M} \equiv \mathsf{c}^{w(M)} >_{\mathsf{lpo}} \mathsf{c}^{w(M')} \equiv \overline{M'}$. So let the reduction step be $\longrightarrow_{\mathsf{minx}}$ where the redex is not in $\mathsf{SN}^{\lambda\mathsf{x}_g}$.

(1) $\langle M/x\rangle y \longrightarrow_{\mathsf{minx}} y \quad (x \not\equiv y)$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle y} & = \mathsf{sub}(\overline{M}, \overline{y}) \\ \text{RHS}: & \overline{y} & = \overline{y} \end{array}$$

(2) $\langle M/x\rangle x \longrightarrow_{\mathsf{minx}} M$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle x} & = \mathsf{sub}(\overline{M}, \overline{x}) \\ \text{RHS}: & \overline{M} & = \overline{M} \end{array}$$

(3) $\langle M/x\rangle(\lambda y.N) \longrightarrow_{\mathsf{minx}} \lambda y.\langle M/x\rangle N$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle(\lambda y.N)} & = \mathsf{sub}(\overline{M}, \overline{\lambda y.N}) \\ & & = \mathsf{sub}(\overline{M}, \mathsf{c}^{w(\lambda y.N)}) \\ \text{RHS}: & \overline{\lambda y.\langle M/x\rangle N} & = \mathsf{abs}(\overline{\langle M/x\rangle N}) \\ & & = \mathsf{abs}(\mathsf{sub}(\overline{M}, \overline{N})) \\ & & = \mathsf{abs}(\mathsf{sub}(\overline{M}, \mathsf{c}^{w(N)})) \end{array}$$

(4) $\langle M/x\rangle(y[N, z.P]) \longrightarrow_{\mathsf{minx}} y[\langle M/x\rangle N, z.\langle M/x\rangle P] \quad (x \not\equiv y)$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle(y[N, z.P])} & = \mathsf{sub}(\overline{M}, \overline{y[N, z.P]}) \\ & & = \mathsf{sub}(\overline{M}, \mathsf{c}^{w(y[N,z.P])}) \\ \text{RHS}: & \overline{y[\langle M/x\rangle N, z.\langle M/x\rangle P]} & \leq \mathsf{gapp}(\overline{y}, \overline{\langle M/x\rangle N}, \overline{\langle M/x\rangle P}) \\ & & \leq \mathsf{gapp}(\overline{y}, \mathsf{sub}(\overline{M}, \overline{N}), \mathsf{sub}(\overline{M}, \overline{P})) \\ & & = \mathsf{gapp}(\mathsf{c}^{w(y)}, \mathsf{sub}(\overline{M}, \mathsf{c}^{w(N)}), \mathsf{sub}(\overline{M}, \mathsf{c}^{w(P)})) \end{array}$$

where $\leq$ is used for $= \cup <_{\mathsf{lpo}}$ to deal with the cases where some of the subterms of RHS are already in $\mathsf{SN}^{\lambda x_g}$, in which cases those subterms M are encoded as $\mathsf{c}^{w(M)}$.

(5) $\langle M/x\rangle(x[N, z.P]) \longrightarrow_{\mathsf{minx}} M[\langle M/x\rangle N, z.\langle M/x\rangle P] \quad (x \in \mathsf{FV}([N, z.P]))$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle(x[N, z.P])} & = \mathsf{sub}(\overline{M}, \overline{x[N, y.P]}) \\ & & = \mathsf{sub}(\overline{M}, \mathsf{c}^{w(x[N,y.P])}) \\ \text{RHS}: & \overline{M[\langle M/x\rangle N, z.\langle M/x\rangle P]} & \leq \mathsf{gapp}(\overline{M}, \overline{\langle M/x\rangle N}, \overline{\langle M/x\rangle P}) \\ & & \leq \mathsf{gapp}(\overline{M}, \mathsf{sub}(\overline{M}, \overline{N}), \mathsf{sub}(\overline{M}, \overline{P})) \\ & & = \mathsf{gapp}(\overline{M}, \mathsf{sub}(\overline{M}, \mathsf{c}^{w(N)}), \mathsf{sub}(\overline{M}, \mathsf{c}^{w(P)}))) \end{array}$$

(6) $\langle M/x\rangle(Q[N, z.P]) \longrightarrow_{\mathsf{minx}} (\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P] \quad (Q \text{ is not a variable})$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle(Q[N, z.P])} & = \mathsf{sub}(\overline{M}, \overline{Q[N, y.P]}) \\ & & = \mathsf{sub}(\overline{M}, \mathsf{c}^{w(Q[N,y.P])}) \\ \text{RHS}: & \overline{(\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]} & \leq \mathsf{gapp}(\overline{\langle M/x\rangle Q}, \overline{\langle M/x\rangle N}, \overline{\langle M/x\rangle P}) \\ & & \leq \mathsf{gapp}(\mathsf{sub}(\overline{M}, \overline{Q}), \mathsf{sub}(\overline{M}, \overline{N}), \mathsf{sub}(\overline{M}, \overline{P})) \\ & & = \mathsf{gapp}(\mathsf{sub}(\overline{M}, \mathsf{c}^{w(Q)}), \mathsf{sub}(\overline{M}, \mathsf{c}^{w(N)}), \mathsf{sub}(\overline{M}, \mathsf{c}^{w(P)})) \end{array}$$

(7) $\langle M/x\rangle(x[N, z.P]) \longrightarrow_{\mathsf{minx}} M[N, z.P] \quad (x \notin \mathsf{FV}([N, z.P]))$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle(x[N, z.P])} & = \mathsf{sub}(\overline{M}, \overline{x[N, z.P]}) \\ & & = \mathsf{sub}(\overline{M}, \mathsf{c}^{w(x[N,z.P])}) \\ \text{RHS}: & \overline{M[N, z.P]} & \leq \mathsf{gapp}(\overline{M}, \overline{N}, \overline{P}) \\ & & = \mathsf{gapp}(\overline{M}, \mathsf{c}^{w(N)}, \mathsf{c}^{w(P)}) \end{array}$$

(Pi) $M[N, z.P][N', z'.P'] \longrightarrow_{\mathsf{min}\times} M[N, z.P[N', z'.P']]$

$$\begin{array}{lll} \text{LHS}: & \overline{M[N, z.P][N', z'.P']} & = \mathsf{gapp}(\overline{M[N, z.P]}, \overline{N'}, \overline{P'}) \\ & & = \mathsf{gapp}(\mathsf{c}^{w(M[N,z.P])}, \overline{N'}, \overline{P'}) \\ \text{RHS}: & \overline{M[N, z.P[N', z'.P']]} & \leq \mathsf{gapp}(\overline{M}, \overline{N}, \overline{P[N', z'.P']}) \\ & & \leq \mathsf{gapp}(\overline{M}, \overline{N}, \mathsf{gapp}(\overline{P}, \overline{N'}, \overline{P'})) \\ & & = \mathsf{gapp}(\mathsf{c}^{w(M)}, \mathsf{c}^{w(N)}, \mathsf{gapp}(\mathsf{c}^{w(P)}, \overline{N'}, \overline{P'})) \end{array}$$

The cases where the reduction is not at the root are easily proved by the induction hypothesis, since $>_{\mathsf{lpo}}$ is context-closed. ◀

▶ **Theorem 27** (Simulation in $\lambda I_{[,]}$)**.** *Suppose $M \mathrel{\mathcal{H}} T$.*

1. *If $M \longrightarrow_{minB} N$ and the reduction step is unsafe then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{+}_{\beta,\pi} U$.*
2. *If $M \longrightarrow_{minB} N$ and the reduction step is safe then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{*}_{\beta,\pi} U$.*
3. *If $M \longrightarrow_{minx} N$ then $N \mathrel{\mathcal{H}} T$.*

Proof. By induction on the derivation of $M \mathrel{\mathcal{H}} T$. Here we consider the cases where the reduction takes place at the root and those where the derivation ends with the rule for explicit substitution. (The other cases are proved in the same way as in the proof of Theorem 14.)

First we inspect the case where one of (B_1), (B_2) and (Pi) takes place at the root. Note that, by minimality, M, N and P (in the rules below) are in $\mathsf{SN}^{\lambda x_g}$.

(B_1) $(\lambda y.M)[N, z.P] \longrightarrow_{\mathsf{minB}} \langle N/y\rangle\langle M/z\rangle P$. In this case, the derivation has the form

$$\dfrac{\dfrac{\dfrac{M \mathrel{\mathcal{H}} T}{\lambda y.M \mathrel{\mathcal{H}} \lambda y.T}\; y \in \mathsf{FV}(T)}{\lambda y.M \mathrel{\mathcal{H}} [\lambda y.T, \overrightarrow{R}]} \quad N \mathrel{\mathcal{H}} U \quad P \mathrel{\mathcal{H}} S}{(\lambda y.M)[N, z.P] \mathrel{\mathcal{H}} \{[\lambda y.T, \overrightarrow{R}]U/z\}S}\; z \in \mathsf{FV}(S)$$

Then we have

$$\dfrac{N \mathrel{\mathcal{H}} U \quad \dfrac{\dfrac{M \mathrel{\mathcal{H}} T}{M \mathrel{\mathcal{H}} [T, \overrightarrow{R}]} \quad P \mathrel{\mathcal{H}} S}{\langle M/z\rangle P \mathrel{\mathcal{H}} \{[T, \overrightarrow{R}]/z\}S}}{\langle N/y\rangle\langle M/z\rangle P \mathrel{\mathcal{H}} \{U/y\}\{[T, \overrightarrow{R}]/z\}S \equiv \{[\{U/y\}T, \overrightarrow{R}]/z\}S}$$

Since $z \in \mathsf{FV}(S)$, we have $\{[\lambda y.T, \overrightarrow{R}]U/z\}S \longrightarrow^{+}_{\beta,\pi} \{[\{U/y\}T, \overrightarrow{R}]/z\}S$ as required.

(B_2) $(\lambda y.M)[N, z.P] \longrightarrow_{\mathsf{minB}} \langle\langle N/y\rangle M/z\rangle P$. In this case, the derivation has the same form as the case (B_1). Then we have

$$\dfrac{\dfrac{\dfrac{N \mathrel{\mathcal{H}} U \quad M \mathrel{\mathcal{H}} T}{\langle N/y\rangle M \mathrel{\mathcal{H}} \{U/y\}T}}{\langle N/y\rangle M \mathrel{\mathcal{H}} [\{U/y\}T, \overrightarrow{R}]} \quad P \mathrel{\mathcal{H}} S}{\langle\langle N/y\rangle M/z\rangle P \mathrel{\mathcal{H}} \{[\{U/y\}T, \overrightarrow{R}]/z\}S}$$

Again, since $z \in \mathsf{FV}(S)$, we have $\{[\lambda y.T, \overrightarrow{R}]U/z\}S \longrightarrow^{+}_{\beta,\pi} \{[\{U/y\}T, \overrightarrow{R}]/z\}S$ as required.

(Pi) $M[N, z.P][N', z'.P'] \longrightarrow_{\mathsf{min}\times} M[N, z.P[N', z'.P']]$. This case is proved in the same way as the case (π_g) in the proof of Theorem 14.

Next we consider the case where the last applied rule of the derivation is

$$\frac{M \mathrel{\mathcal{H}} T \quad N \mathrel{\mathcal{H}} U}{\langle M/x\rangle N \mathrel{\mathcal{H}} \{T/x\}U}\ x \in \mathsf{FV}(U) \vee M \in \mathsf{SN}^{\lambda \mathsf{x}_g}$$

If the reduction takes place within M or N, we apply the induction hypothesis, remembering that reduction in $\lambda I_{[,]}$ preserves free variables (Lemma 6), so the side-condition remains satisfied. Moreover, an unsafe B-reduction in M is simulated by at least one reduction step from T. (Indeed, since the B-reduction is unsafe, we know that $M \notin \mathsf{SN}^{\lambda \mathsf{x}_g}$ and hence we must have $x \in \mathsf{FV}(U)$.) The simulating reduction step from T is therefore preserved in the reduction of $\{T/x\}U$. This is the precise point where the distinction between safe and unsafe reductions plays its role.

Otherwise, we have a (minimal) root reduction and the case analysis below inspects some of the rules. Note that, by minimality, both M and N (in the rule above) are in $\mathsf{SN}^{\lambda \mathsf{x}_g}$.

(1) $\langle M/x\rangle y \longrightarrow_{\mathsf{minx}} y$ $(x \not\equiv y)$. In this case, the derivation has the form

$$\frac{M \mathrel{\mathcal{H}} T \quad \dfrac{\overline{y \mathrel{\mathcal{H}} y}}{y \mathrel{\mathcal{H}} [y, \overrightarrow{R}]}}{\langle M/x\rangle y \mathrel{\mathcal{H}} \{T/x\}[y, \overrightarrow{R}] \equiv [y, \{T/x\}\overrightarrow{R}]}$$

Then we have

$$\frac{\overline{y \mathrel{\mathcal{H}} y}}{y \mathrel{\mathcal{H}} [y, \{T/x\}\overrightarrow{R}]}$$

(2) $\langle M/x\rangle x \longrightarrow_{\mathsf{minx}} M$. In this case, the derivation has the form

$$\frac{M \mathrel{\mathcal{H}} T \quad \dfrac{\overline{x \mathrel{\mathcal{H}} x}}{x \mathrel{\mathcal{H}} [x, \overrightarrow{R}]}}{\langle M/x\rangle x \mathrel{\mathcal{H}} \{T/x\}[x, \overrightarrow{R}] \equiv [T, \{T/x\}\overrightarrow{R}]}$$

Then we have

$$\frac{M \mathrel{\mathcal{H}} T}{M \mathrel{\mathcal{H}} [T, \{T/x\}\overrightarrow{R}]}$$

(3) $\langle M/x\rangle(\lambda y.N) \longrightarrow_{\mathsf{minx}} \lambda y.\langle M/x\rangle N$. In this case, the derivation has the form

$$\frac{M \mathrel{\mathcal{H}} T \quad \dfrac{\dfrac{N \mathrel{\mathcal{H}} U}{\lambda y.N \mathrel{\mathcal{H}} \lambda y.U}\ y \in \mathsf{FV}(U)}{\lambda y.N \mathrel{\mathcal{H}} [\lambda y.U, \overrightarrow{R}]}}{\langle M/x\rangle(\lambda y.N) \mathrel{\mathcal{H}} \{T/x\}[\lambda y.U, \overrightarrow{R}] \equiv [\lambda y.\{T/x\}U, \{T/x\}\overrightarrow{R}]}$$

Then we have

$$\frac{\dfrac{\dfrac{M \mathrel{\mathcal{H}} T \quad N \mathrel{\mathcal{H}} U}{\langle M/x\rangle N \mathrel{\mathcal{H}} \{T/x\}U}}{\lambda y.\langle M/x\rangle N \mathrel{\mathcal{H}} \lambda y.\{T/x\}U}\ y \in \mathsf{FV}(\{T/x\}U)}{\lambda y.\langle M/x\rangle N \mathrel{\mathcal{H}} [\lambda y.\{T/x\}U, \{T/x\}\overrightarrow{R}]}$$

(6) $\langle M/x\rangle(Q[N, z.P]) \longrightarrow_{\mathsf{minx}} (\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]$ (Q is not a variable). In this case, the derivation has the form

$$\frac{M \mathrel{\mathcal{H}} T \quad \dfrac{\dfrac{Q \mathrel{\mathcal{H}} T' \quad N \mathrel{\mathcal{H}} U \quad P \mathrel{\mathcal{H}} S}{Q[N, z.P] \mathrel{\mathcal{H}} \{T'U/z\}S}\ z \in \mathsf{FV}(S)}{Q[N, z.P] \mathrel{\mathcal{H}} [\{T'U/z\}S, \overrightarrow{R}]}}{\langle M/x\rangle(Q[N, z.P]) \mathrel{\mathcal{H}} \{T/x\}[\{T'U/z\}S, \overrightarrow{R}]}$$

Then we have

$$\dfrac{\dfrac{\dfrac{M\ \mathcal{H}\ T \quad Q\ \mathcal{H}\ T'}{\langle M/x\rangle Q\ \mathcal{H}\ \{T/x\}T'} \quad \dfrac{M\ \mathcal{H}\ T \quad N\ \mathcal{H}\ U}{\langle M/x\rangle N\ \mathcal{H}\ \{T/x\}U} \quad \dfrac{M\ \mathcal{H}\ T \quad P\ \mathcal{H}\ S}{\langle M/x\rangle P\ \mathcal{H}\ \{T/x\}S}}{(\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]\ \mathcal{H}\ \{\{T/x\}T'\{T/x\}U/z\}\{T/x\}S}\ z \in \mathsf{FV}(\{T/x\}S)}{(\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]\ \mathcal{H}\ [\{\{T/x\}T'\{T/x\}U/z\}\{T/x\}S, \{T/x\}\overrightarrow{R}]}$$
$$\Vert\!\Vert$$
$$(\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]\ \mathcal{H}\ \{T/x\}[\{T'U/z\}S, \overrightarrow{R}]$$

(7) $\langle M/x\rangle(x[N, z.P]) \longrightarrow_{\mathsf{minx}} M[N, z.P]$ $(x \notin \mathsf{FV}([N, z.P]))$. In this case, the derivation has the form

$$\dfrac{\dfrac{M\ \mathcal{H}\ T \qquad \dfrac{\dfrac{x\ \mathcal{H}\ T' \quad N\ \mathcal{H}\ U \quad P\ \mathcal{H}\ S}{x[N, z.P]\ \mathcal{H}\ \{T'U/z\}S}\ z \in \mathsf{FV}(S)}{x[N, z.P]\ \mathcal{H}\ [\{T'U/z\}S, \overrightarrow{R}]}}{\langle M/x\rangle(x[N, z.P])\ \mathcal{H}\ \{T/x\}[\{T'U/z\}S, \overrightarrow{R}]}$$

where $T' \equiv [x, \overrightarrow{R'}]$. Then we have

$$\dfrac{\dfrac{\dfrac{M\ \mathcal{H}\ T}{M\ \mathcal{H}\ [T, \{T/x\}\overrightarrow{R'}]}}{M\ \mathcal{H}\ \{T/x\}T'} \quad \dfrac{N\ \mathcal{H}\ U}{N\ \mathcal{H}\ \{T/x\}U}\ \text{Lemma 11 (0c)} \quad \dfrac{P\ \mathcal{H}\ S}{P\ \mathcal{H}\ \{T/x\}S}\ \text{Lemma 11 (0c)}}{\dfrac{M[N, z.P]\ \mathcal{H}\ \{\{T/x\}T'\{T/x\}U/z\}\{T/x\}S}{M[N, z.P]\ \mathcal{H}\ [\{\{T/x\}T'\{T/x\}U/z\}\{T/x\}S, \{T/x\}\overrightarrow{R}]}}\ z \in \mathsf{FV}(\{T/x\}S)$$
$$\Vert\!\Vert$$
$$M[N, z.P]\ \mathcal{H}\ \{T/x\}[\{T'U/z\}S, \overrightarrow{R}]$$

◀

▶ **Lemma 29.** *For any λx_g-term M, there exists a $\lambda I_{[,]}$-term T such that $M\ \mathcal{H}\ T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$.*

Proof. By induction on M. Here we consider the case of explicit substitution. (The other cases are proved in the same way as in the proof of Lemma 15.) Then we have

$$\mathsf{i}(\mathcal{G}(\langle M/x\rangle N)) = \begin{cases} \{\mathsf{i}(\mathcal{G}(M))/x\}\mathsf{i}(\mathcal{G}(N)) & \text{if } x \in \mathsf{FV}(N) \\ (\lambda x.[\mathsf{i}(\mathcal{G}(N)), x])\,\mathsf{i}(\mathcal{G}(M)) & \text{if } x \notin \mathsf{FV}(N) \end{cases}$$

By the induction hypothesis, there are $\lambda I_{[,]}$-terms T and U such that (a) $M\ \mathcal{H}\ T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$, and (b) $N\ \mathcal{H}\ U$ and $\mathsf{i}(\mathcal{G}(N)) \longrightarrow^*_{\beta,\pi} U$.

- If $x \in \mathsf{FV}(N)$ then $\mathsf{i}(\mathcal{G}(\langle M/x\rangle N)) = \{\mathsf{i}(\mathcal{G}(M))/x\}\mathsf{i}(\mathcal{G}(N)) \longrightarrow^*_{\beta,\pi} \{T/x\}U$. Since $\mathsf{FV}(N) = \mathsf{FV}(\mathsf{i}(\mathcal{G}(N))) = \mathsf{FV}(U)$, we have $x \in \mathsf{FV}(U)$. Hence, from (a) and (b), we have $\langle M/x\rangle N\ \mathcal{H}\ \{T/x\}U$.
- If $x \notin \mathsf{FV}(N)$ then $\mathsf{i}(\mathcal{G}(\langle M/x\rangle N)) = (\lambda x.[\mathsf{i}(\mathcal{G}(N)), x])\,\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} (\lambda x.[U, x])T \longrightarrow_\beta \{T/x\}[U, x]$. From (b), we have $N\ \mathcal{H}\ [U, x]$ and hence $\langle M/x\rangle N\ \mathcal{H}\ \{T/x\}[U, x]$.

◀

Kleene Algebra with Products and Iteration Theories

Dexter Kozen and Konstantinos Mamouras

Computer Science Department, Cornell University, Ithaca, NY 14853, USA
{kozen,mamouras}@cs.cornell.edu

Abstract

We develop a typed equational system that subsumes both iteration theories and typed Kleene algebra in a common framework. Our approach is based on cartesian categories endowed with commutative strong monads to handle nondeterminism.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Kleene algebra, typed Kleene algebra, iteration theories

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.415

1 Introduction

In the realm of equational systems for reasoning about iteration, two chief complementary bodies of work stand out. One of these is *iteration theories* (IT), the subject of the extensive monograph of Bloom and Ésik [4] as well as many other authors (see the cited literature). The primary motivation for iteration theories is to capture in abstract form the equational properties of iteration on structures that arise in domain theory and program semantics, such as continuous functions on ordered sets. Of central interest is the dagger operation †, a kind of parameterized least fixpoint operator, that when applied to an object representing a simultaneous system of equations gives an object representing the least solution of those equations. Much of the work on iteration theories involves axiomatizing or otherwise characterizing the equational theory of iteration as captured by †. Complete axiomatizations have been provided [7, 9, 10] as well as other algebraic and categorical characterizations [2, 3].

Bloom and Ésik claim that "...the notion of an iteration theory seems to axiomatize the equational properties of all computationally interesting structures..." [6]. This is true to a certain extent, certainly if one is interested only in structures that arise in domain theory and programming language semantics. However, it is not the entire story.

Another approach to equational reasoning about iteration that has met with some success over the years is the notion of *Kleene algebra* (KA), the algebra of regular expressions. KA has a long history going back to the original paper of Kleene [12] and was further developed by Conway, who coined the name Kleene algebra in his 1971 monograph [8]. It has since been studied by many authors. KA relies on an iteration operator * that characterizes iteration in a different way from †. Its principal models are not those of domain theory, but rather basic algebraic objects such as sets of strings (in which * gives the Kleene asterate operation), binary relations (in which * gives reflexive transitive closure), and other structures with applications in shortest path algorithms on graphs and geometry of convex sets. Complete axiomatizations and complexity analyses have been given; the regular sets of strings over an alphabet A form the free KA on generators A in much the same way that the rational $\Sigma_\perp$-trees form the free IT on a signature Σ.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca ; pp. 415–431

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Although the two systems fulfill many of the same objectives and are related at some level, there are many technical and stylistic differences. Whereas iteration theories are based on Lawvere theories, a categorical concept, Kleene algebra operates primarily at a level of abstraction one click down. For this reason, KA may be somewhat more accessible. KA has been shown to be useful in several nontrivial static analysis and verification tasks (see e.g. [16, 23]). Also, KA can model nondeterministic computation, whereas IT is primarily deterministic.

Nevertheless, both systems have claimed to capture the notion of iteration in a fundamental way, and it is interesting to ask whether they can somehow be reconciled. This is the investigation that we have undertaken in this paper. We start with the observation that ITs use the objects of a category to represent types. Technically the objects of interest in ITs are morphisms $f : n \to m$ in a category whose objects are natural numbers, and the morphism $f : n \to m$ is meant to model functions $f : A^m \to A^n$ (the arrows are reversed for technical reasons). Thus ITs might be captured by a version of KA with types. Although the primary version of KA is untyped, there is a notion of typed KA [21], although it only has types of the form $A \to B$, whereas to subsume IT it would need products as well. The presence of products allow ITs to capture parameterized fixpoints through the rule

$$\frac{f : n \to n + m}{f^\dagger : n \to m}$$

giving the parameterized least fixpoint $f^\dagger : A^m \to A^n$ of a parameterized function $f : A^m \times A^n \to A^n$. This would be possible to capture in KA if the typed version had products, which it does not. On the other hand, KA allows the modeling of nondeterministic computation, which IT does not, at least not in any obvious way. Thus to capture both systems, it would seem that we need to extend the type system of typed KA, or extend the categorical framework of IT to handle nondeterminism, or both.

The result of our investigation is a common categorical framework based on cartesian categories (categories with products) combined with a monadic treatment of nondeterminism. Types are represented by objects in the category, and we identify the appropriate axioms in the form of typed equations that allow equational reasoning on the morphisms. Our framework captures iteration as represented in ITs and KAs in a common language. We show how to define the KA operations as enrichments on the morphisms and how to define † in terms of *. Our main contributions are as follows.

- **Commutative strong monads.** To accommodate nondeterminism, we need to lift the computation to the Kleisli category of a monad representing nondeterminate values. However, the ordinary powerset monad does not suffice for this purpose, as it does not interact well with non-strictness. We axiomatize the relevant properties for an arbitrary *commutative strong monad*, where (i) the property of *commutativity* captures the idea that the computation of a pair can be done in either order, and (ii) *strength* refers to tensorial strength, which axiomatizes the interaction of pairs with the nondeterminism monad.
- **Lazy pairs**. We need to model non-strict (lazy) evaluation of programs in the presence of products. Ordinarily, a pair $\langle x, \bot\rangle$ would be $\bot$ by strictness. This requires the development of the concept of *lazy pairs* and its categorical axiomatization. Intuitively, in the case of *eager pairs*, the computation of a pair $\langle v, \bot\rangle$, where v is a value and $\bot$ denotes a diverging computation, would also be diverging, i.e., $\langle v, \bot\rangle = \bot$. This makes it impossible to recover the left component v of the pair: $\langle v, \bot\rangle; \pi_1 = \bot$.
- **Simplified axiomatization of commutative strong monads and lazy pairs**. We have given a simplified axiomatization of commutative monads with lazy pairs in terms of a certain operator ψ that captures the interaction of these concepts in a very concise form, much simpler than the axiomatizations of the two of them separately. This is an adaptation

of a construction that can be found in the work of Kock in the 1970s [13, 15]. We use this extensively in our development to simplify arguments.

- **Deterministic arrows**. Certain properties work only for deterministic computations. We show how to capture the necessary properties of determinism in the Kleisli category. A separate syntactic arrow provides a convenient notation for reasoning about deterministic computation in the underlying category when working in the Kleisli category, and we provide an axiomatization of the necessary properties.
- **Lifting the cartesian structure**. We show how the cartesian structure (pairing and projections) in the underlying category can be lifted in a smooth way to corresponding operations in the Kleisli category.
- **Capturing nondeterminism**. We give three equivalent ways of capturing (angelic) nondeterminism in the homsets of the Kleisli category of a monad. These characterizations make essential use of cartesian structure of the base category.
- **Capturing IT and KA**. We show how to enrich the homsets of the Kleisli category with the KA operations, including *, to obtain typed KA with products, and that all the axioms of KA (except the strictness axiom) are satisfied. Sequential composition is modeled by Kleisli composition. We also show that Park theories [9] are subsumed by KA with products. This is our main result.
- **Model theory**. Finally, we show that two particular monads, the *lowerset monad* and the *ideal completion monad*, provide natural concrete models in that they are commutative strong monads with lazy pairs. The ideal completion monad involves ideal completion in ω-complete partial orders (ω-CPOs) and models nondeterminism in those structures.

2 Commutative Strong Monads with Lazy Pairs

As a first step in the development of our category-theoretic framework, we define axiomatically the base cartesian category of "non-strict functions and values", where the notion of divergence is made explicit. We handle nondeterminism with a commutative strong monad, for which the formation of lazy pairs is implemented with just one very natural axiom that intuitively says: "forming a pair of a non-diverging computation with a diverging computation allows one to recover the non-diverging component". The Kleisli category of the monad, which we think of as a category of "non-strict programs and computations", is symmetric monoidal. In fact, it possesses more structure, which we will exploit by making the notion of deterministic arrow explicit.

In order to model non-strict (lazy) evaluation of programs and lazy pairs we consider our base category to be a *cartesian category $\mathcal{C}$ with bottom elements*. For an object X, we write the identity for X as $\mathrm{id}_X : X \to X$. The composition operation is written as ; and the operands are given in diagrammatic order: the composite of the two arrows $f : X \to Y$ and $g : Y \to Z$ is written $f; g : X \to Z$. For objects X and Y, we denote by $X \times Y$ their product with corresponding left and right projections $\pi_1^{XY} : X \times Y \to X$ and $\pi_2^{XY} : X \times Y \to Y$ respectively. The pairing operation $\langle \cdot, \cdot \rangle$ takes two arrows $f : X \to Y$ and $g : X \to Z$ with the same domain and produces an arrow $\langle f, g \rangle : X \to Y \times Z$. The terminal object is denoted by $\mathbb{1}$. We write $\bot_X : X \to \mathbb{1}$ for the unique arrow from X to $\mathbb{1}$. The equational axioms

$$\langle f, g \rangle; \pi_1 = f \qquad \langle f, g \rangle; \pi_2 = g \qquad \langle h; \pi_1, h; \pi_2 \rangle = h \qquad f = \bot_X$$

say that the operations $\times, \pi_1, \pi_2, \langle \cdot, \cdot \rangle, \mathbb{1}, \bot$ endow $\mathcal{C}$ with cartesian structure. For every object X, the *bottom element* of X is written as $\bot_{\mathbb{1}X} : \mathbb{1} \to X$. Define the *bottom morphism* $\bot_{XY}$ from X to Y by $\bot_{XY} := \bot_X; \bot_{\mathbb{1}Y}$. The bottom morphisms satisfy the axiom $f; \bot_{YZ} = \bot_{XZ} : X \to Z$. An arrow $f : X \to Y$ satisfying the equation $\bot_{\mathbb{1}X}; f = \bot_{\mathbb{1}Y}$

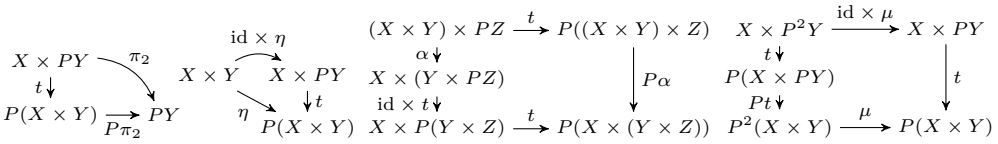


Figure 1 Axioms for tensorial strength $t : X \times PY \to P(X \times Y)$ of the monad (P, η, μ).

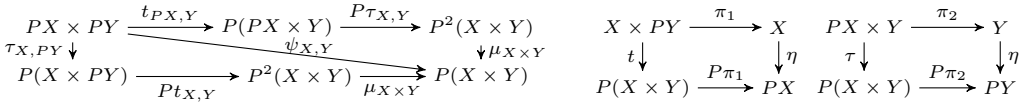


Figure 2 Commutativity axiom for the strong monad $(P, \eta, \mu), t$ (diagram on the left), and lazy pairs axiom in terms of t and equivalently in terms of τ (two diagrams on the right).

is called *strict*.

A *monad* (P, η, μ) over $\mathcal{C}$ consists of an endofunctor $P : \mathcal{C} \to \mathcal{C}$, and natural transformations $\eta_X : X \to PX$ and $\mu_X : P^2X \to PX$, called the *unit* and *multiplication* of the monad respectively, that satisfy: $P\mu_X; \mu_X = \mu_{PX}; \mu_X$, $\eta_{PX}; \mu_X = \mathrm{id}_{PX}$, and $P\eta_X; \mu_X = \mathrm{id}_{PX}$.

A monad (P, η, μ) is *strong* with *tensorial strength* $t_{X,Y} : X \times PY \to P(X \times Y)$ if t is a natural transformation satisfying the axioms of Figure 1. The arrow $\alpha_{X,Y,Z} : (X \times Y) \times Z \to X \times (Y \times Z)$ is the natural isomorphism defined as $\alpha := \langle \pi_1; \pi_1, \langle \pi_1; \pi_2, \pi_2 \rangle \rangle$. We define the *dual tensorial strength* $\tau_{X,Y} : PX \times Y \to P(X \times Y)$ as $\tau_{X,Y} = \mathsf{s}_{PX,Y}; t_{Y,X}; P\mathsf{s}_{Y,X}$, where $\mathsf{s}_{X,Y} : X \times Y \to Y \times X$ is the natural isomorphism given by $\mathsf{s}_{X,Y} = \langle \pi_2, \pi_1 \rangle$. The properties satisfied by t imply that τ is also a natural transformation and it satisfies the obvious dual axioms. For all objects X, Y define the morphism $\psi_{X,Y} : PX \times PY \to P(X \times Y)$ as $\psi_{X,Y} = t_{PX,Y}; P\tau_{X,Y}; \mu_{X \times Y}$. Using the fact that t and τ are natural transformations, it can be shown that ψ is also a natural transformation.

A strong monad $(P, \eta, \mu), t$ is *commutative* if the diagram on the left side of Fig. 2 commutes. Intuitively, the commutativity condition says that when computing a pair it does not matter in which order the components are computed. We are interested in strong monads that model *lazy pairs*. In the case of eager pairs, the computation of a pair $\langle v, \bot \rangle$, where v is a value and $\bot$ denotes a diverging computation, would also be diverging, i.e., $\langle v, \bot \rangle = \bot$. Therefore, it is not possible to recover the left component v of the pair: $\langle v, \bot \rangle; \pi_1 = \bot$. So, for lazy pairs we need an additional axiom, which can be given equivalently in terms of t or τ. The "lazy pairs" axiom says that the two diagrams on the right side of Fig. 2 commute.

▶ **Definition 1.** We say that $(P, \eta, \mu), t$ is a *commutative strong monad with lazy pairs* over $\mathcal{C}$ if (P, η, μ) is a monad with tensorial strength t so that the commutativity axiom of Figure 2 and the lazy pairs axiom of Figure 2 hold.

A commutative monad with lazy pairs can be equivalently given in terms of ψ. In Figure 3 we give properties that ψ satisfies, when it is defined in terms of t (see [13]). Conversely, we consider a monad (P, η, μ) over $\mathcal{C}$ together with a natural transformation $\psi : PX \times PY \to P(X \times Y)$ satisfying the axioms of Figure 3. Then, we can define t as $t_{X,Y} = (\eta_X \times \mathrm{id}_{PY}); \psi_{X,Y}$ and recover all the axioms we had given for $(P, \eta, \mu), t$ (see [15]). It has been proved by Anders Kock in [14] (Theorem 2.1) that for a commutative strong monad the axiom $\psi; \langle P\pi_1, P\pi_2 \rangle = \mathrm{id}$ is equivalent to $P\mathbb{1} \cong \mathbb{1}$, which says that P preserves final objects. We have opted for the former axiom in our definition, because it corresponds immediately to the intuition for the formation of lazy pairs.

The *Kleisli category* $\mathcal{C}_P$ has the same objects as $\mathcal{C}$. For all objects X, Y the homset $\mathcal{C}_P(X, Y)$ is equal to the homset $\mathcal{C}(X, PY)$. We use the notation $f : X \rightharpoonup Y$ for an arrow in

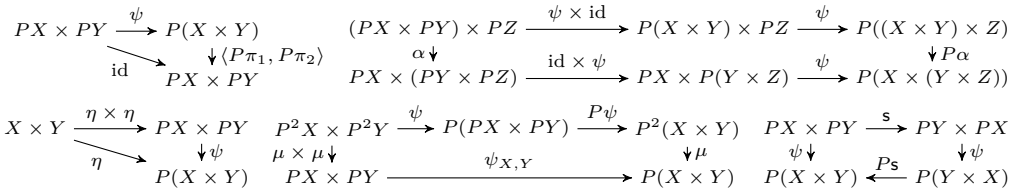


Figure 3 Commutative strong monad with lazy pairs (given in terms of ψ).

$\mathcal{C}_P(X,Y)$. The composition operation in $\mathcal{C}_P$ is the *Kleisli composition* operation, denoted $;$, and defined as $f;g := f;Pg;\mu_Z : X \rightharpoonup Z$. For object X, the identity in $\mathcal{C}_P$ is $\eta_X : X \rightharpoonup X$. The equations $\eta_X;f = f = f;\eta_Y$ and $(f;g);h = f;(g;h)$ state that $\mathcal{C}_P$ is a category and they can be shown from the definitions and the monad axioms.

We define the map $H = (-;\eta)$ from the category $\mathcal{C}$ to the Kleisli category $\mathcal{C}_P$ by $HX := X$ and $Hf := f;\eta_Y : X \rightharpoonup Y$. We verify that H is a functor. First, note that it sends the identity $\mathrm{id}_X : X \to X$ of $\mathcal{C}$ to the identity $H\mathrm{id}_X = \eta_X : X \rightharpoonup X$ of $\mathcal{C}_P$. Moreover, the equation $H(f;g) = Hf;Hg : X \rightharpoonup Z$ holds, which says that H commutes with composition. So, H is a functor from $\mathcal{C}$ to $\mathcal{C}_P$, which we will call here the *unit functor* of the monad. The functor H is the left adjoint of the Kleisli adjunction for the monad.

We say that an arrow $f : X \rightharpoonup Y$ of $\mathcal{C}_P$ is a *deterministic arrow* if there exists an arrow $f' : X \to Y$ of $\mathcal{C}$ such that $f = f';\eta_Y = Hf' : X \to PY$. So, the deterministic arrows of the Kleisli category are exactly the image of the arrows of $\mathcal{C}$ under the unit functor H. We indicate that f is a deterministic arrow of $\mathcal{C}_P$ by writing $f : X \to\!\!\!\!\bullet\; Y$. The Kleisli composite of two deterministic arrows is also deterministic. In our notation, if $f : X \to\!\!\!\!\bullet\; Y$ and $g : Y \to\!\!\!\!\bullet\; Z$ then $f;g : X \to\!\!\!\!\bullet\; Z$. The identity $\eta_X : X \to\!\!\!\!\bullet\; X$ is a deterministic arrow because $\eta_X = H\mathrm{id}_X$.

For the rest of this section, we assume that $(P,\eta,\mu),t$ is a commutative strong monad over $\mathcal{C}$ with lazy pairs. We will define "Kleisli versions" of projections, the pairing operation, and the product functor. We will prove useful properties that they satisfy. The notion of deterministic arrow turns out to be relevant.

We define the *Kleisli pairing* operation $\langle\!\langle \cdot,\cdot \rangle\!\rangle$ in $\mathcal{C}_P$ and the *Kleisli projections* (left and right) as follows: $\langle\!\langle f,g \rangle\!\rangle := \langle f,g\rangle;\psi : X \rightharpoonup Y \times Z$ for $f : X \rightharpoonup Y$, $g : X \rightharpoonup Z$, and $\varpi_1 := H\pi_1 : X \times Y \to\!\!\!\!\bullet\; X$, $\varpi_2 := H\pi_2 : X \times Y \to\!\!\!\!\bullet\; Y$. The Kleisli projections are deterministic arrows. If $f : X \to\!\!\!\!\bullet\; Y$ and $g : X \to\!\!\!\!\bullet\; Z$ are deterministic arrows, then so is $\langle\!\langle f,g \rangle\!\rangle$. This is an immediate consequence of the equation $H\langle f,g\rangle = \langle\!\langle Hf,Hg \rangle\!\rangle : X \rightharpoonup Y \times Z$, which states that H commutes with the pairing operation.

▶ **Theorem 2.** *The following typed equations for Kleisli projections/pairing hold:*

$$\langle\!\langle f,g \rangle\!\rangle;\varpi_1 = f : X \rightharpoonup Y \qquad \langle\!\langle h;\varpi_1, h;\varpi_2 \rangle\!\rangle = h : X \to\!\!\!\!\bullet\; Y \times Z$$
$$\langle\!\langle f,g \rangle\!\rangle;\varpi_2 = g : X \rightharpoonup Z \qquad f;\langle\!\langle g_1,g_2 \rangle\!\rangle = \langle\!\langle f;g_1, f;g_2 \rangle\!\rangle : X \rightharpoonup Z_1 \times Z_2$$

For the bottom-right equation, there is the extra hypothesis that $f : X \to\!\!\!\!\bullet\; Y$ is deterministic.

We define the operation $\otimes$ on $\mathcal{C}_P$, which we call *Kleisli product functor*, by $f_1 \otimes f_2 := (f_1 \times f_2);\psi : X_1 \times X_2 \rightharpoonup Y_1 \times Y_2$. Equivalently, we can define the Kleisli product as $f_1 \otimes f_2 = \langle\!\langle \varpi_1;f_1, \varpi_2;f_2 \rangle\!\rangle$. We observe that H commutes with the product functor, that is, $H(f \times g) = Hf \otimes Hg$. An easy consequence of the above results is that the Kleisli category $\mathcal{C}_P$ has symmetric monoidal structure given by $\otimes$ (tensor or monoidal product), $\mathbb{1}$ (identity object), $H\alpha : (X \otimes Y) \otimes Z \to\!\!\!\!\bullet\; X \otimes (Y \otimes Z)$ (associator), $\varpi_2 : \mathbb{1} \otimes X \to\!\!\!\!\bullet\; X$ (left unitor), $\varpi_1 : X \otimes \mathbb{1} \to\!\!\!\!\bullet\; X$ (right unitor), and $H\mathsf{s}_{X,Y} : X \otimes Y \to\!\!\!\!\bullet\; Y \otimes X$ (commutativity constraint).

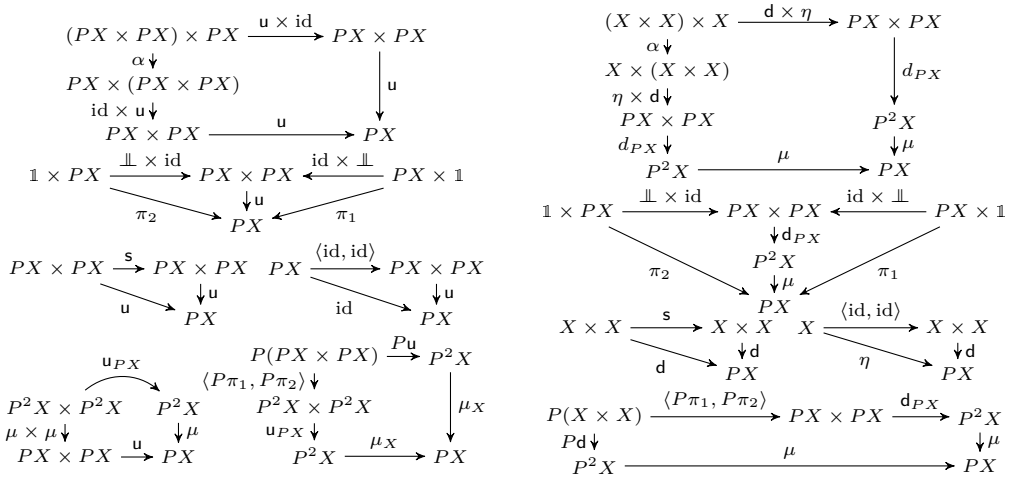


Figure 4 Axioms for $\mathsf{u}_X : PX \times PX \to PX$ (left side) and for $\mathsf{d}_X : X \times X \to PX$ (right side).

3 Nondeterministic Monads

In this section we look at three equivalent ways of endowing with (angelic) nondeterministic structure the homsets of the Kleisli category of a monad. See also [11], where the similar notion of additive monad is considered. The proofs of equivalence we present make essential use of the cartesian structure of the base category. The axiomatizations are simple and intuitive in that they correspond to familiar properties of the elementary operations of "binary union" and "formation of unordered pair", thus allowing us to easily identify models.

▶ **Definition 3.** Let (P, η, μ) be a monad over the category $\mathcal{C}$, let $\perp\!\!\!\perp_{XY} : X \to PY$ be a family of morphisms, and $+$ be a binary operation on $\mathcal{C}(X, PY)$ which we call *(nondeterministic) choice*. We say that $(P, \eta, \mu), +, \perp\!\!\!\perp$ is a *nondeterministic monad* if the axioms

$$(f+g)+h = f+(g+h) \qquad f+\perp\!\!\!\perp = f \qquad f;(g_1+g_2) = f;g_1 + f;g_2 \qquad f;\perp\!\!\!\perp = \perp\!\!\!\perp$$
$$f+g = g+f \qquad f+f = f \qquad (f_1+f_2);g = f_1;g + f_2;g$$

are satisfied. The above axioms state that every homset $\mathcal{C}(X, PY)$ is a commutative idempotent monoid w.r.t. $+$ and $\perp\!\!\!\perp$. Additionally, $+$ distributes over Kleisli composition, and $\perp\!\!\!\perp$ is a right annihilator for $;$.

Assuming the category $\mathcal{C}$ is cartesian, we will give an equivalent definition of the nondeterministic monad in terms of a natural transformation $\mathsf{u}_X : PX \times PX \to PX$, which we can intuitively think of as *binary union*. Then, we will also derive another equivalent definition of the nondeterministic monad in terms of a natural transformation $\mathsf{d}_X : X \times X \to PX$, which we think of as an operation that forms an *unordered pair* of two elements.

▶ **Theorem 4.** *Let $\mathcal{C}$ be a category with cartesian structure given by $\times, \pi_1, \pi_2, \langle\cdot,\cdot\rangle, \mathbb{1}, \bot$.*

- *Suppose $(P, \eta, \mu), +, \perp\!\!\!\perp$ is a nondeterministic monad. Define $\mathsf{u}_X := \pi_1 + \pi_2 : PX \times PX \to PX$. Then, u_X is natural and the diagrams on the left side of Figure 4 commute.*
- *Conversely, suppose that (P, η, μ) is a monad, $\mathsf{u}_X : PX \times PX \to PX$ is natural, and $\perp\!\!\!\perp_{\mathbb{1}X} : \mathbb{1} \to PX$ is a family of morphisms, so that the axioms on the left side of Figure 4 are satisfied. Define the operation $+$ on the homset $\mathcal{C}(X, PY)$ by $f + g := \langle f, g\rangle; \mathsf{u}_Y : X \to PY$, and $\perp\!\!\!\perp_{XY} := \bot_X; \perp\!\!\!\perp_{\mathbb{1}Y} : X \to PY$. Then, $(P, \eta, \mu), +, \perp\!\!\!\perp$ is a nondeterministic monad.*

It is easy to see that the two constructions described in Theorem 4 are mutually inverse. From the choice operation $+$ we obtain $\mathsf{u}_X = \pi_1 + \pi_2 : PX \times PX \to PX$ and then a new

choice operation $\vee$ given by $f \vee g = \langle f, g\rangle; \mathsf{u}_X$. But we have that $f \vee g = \langle f, g\rangle; (\pi_1 + \pi_2) = \big(\langle f, g\rangle; \eta\big); (\pi_1 + \pi_2) = \langle f, g\rangle; \pi_1 + \langle f, g\rangle; \pi_2 = f + g$. For the other direction, we notice that from $\mathsf{u}_X : PX \times PX \to PX$ we obtain a choice operation $+$ and then a new binary union natural transformation $\hat{\mathsf{u}}_X = \pi_1 + \pi_2 = \langle \pi_1, \pi_2\rangle; \mathsf{u}_X = \mathrm{id}; \mathsf{u}_X = \mathsf{u}_X$.

▶ **Theorem 5.** *Let $\mathcal{C}$ be a category with cartesian structure given by $\times, \pi_1, \pi_2, \langle\cdot,\cdot\rangle, \mathbb{1}, \bot$. Let (P, η, μ) be a monad over $\mathcal{C}$, and $\perp\!\!\!\perp_{1X} : \mathbb{1} \to PX$ be a family of morphisms.*

- *Suppose that $\mathsf{u}_X : PX \times PX \to PX$ is natural and the diagrams on the left side of Figure 4 commute. Define $\mathsf{d}_X := (\eta_X \times \eta_X); \mathsf{u}_X : X \times X \to PX$. Then, d_X is natural and satisfies the axioms on the right side of Figure 4.*
- *Conversely, suppose that $\mathsf{d}_X : X \times X \to PX$ is natural and the diagrams on the right side of Figure 4 commute. Define $\mathsf{u}_X := \mathsf{d}_{PX}; \mu_X : PX \times PX \to PX$. Then, u is natural and satisfies the axioms on the left side of Figure 4.*

Again, the constructions described in Theorem 5 are mutually inverse. In one direction, we have $\mathsf{u}_X \mapsto \mathsf{d}_X = (\eta_X \times \eta_X); \mathsf{u}_X \mapsto \hat{\mathsf{u}}_X = \mathsf{d}_{PX}; \mu_X$, where $\hat{\mathsf{u}}_X = (\eta_{PX} \times \eta_{PX}); \mathsf{u}_{PX}; \mu_X = (\eta_{PX} \times \eta_{PX}); (\mu_X \times \mu_X); \mathsf{u}_X = \mathsf{u}_X$ using one of the axioms for u. In the other direction, we have $\mathsf{d}_X \mapsto \mathsf{u}_X = \mathsf{d}_{PX}; \mu_X \mapsto \hat{\mathsf{d}}_X = (\eta_X \times \eta_X); \mathsf{u}_X$ and therefore $\hat{\mathsf{d}}_X = (\eta_X \times \eta_X); \mathsf{d}_{PX}; \mu_X = \mathsf{d}_X; P\eta_X; \mu_X = \mathsf{d}_X$, since d is a natural transformation.

4 Nondeterministic Strong Monad with Iteration

In §3 we investigated the nondeterministic structure of the Kleisli category of a monad in isolation from products. This is not sufficient for our purposes, because we also want to capture the interaction between nondeterminism and products. For example, we would expect to be able to derive the property $\langle a + b, c\rangle = \langle a, c\rangle + \langle b, c\rangle$ for pairs with a nondeterminate component. So, we consider an additional axiom that relates the tensorial strength with the nondeterministic structure (Theorem 6). Then, we proceed to give our main definition of a nondeterministic strong monad with iteration, which puts together the axioms for all the symbols of our algebraic signature, including those for iteration (taken from [18]). The rest of the section is devoted to identifying models. We first consider the lowerset monad over pointed posets, which generalizes the familiar relational semantics of programs to the setting of lazy evaluation with lazy pairs. Then, we investigate the ideal completion monad over ω-CPOs, which is a kind of lower powerdomain construction. We derive several properties for this monad that will be essential for the main technical result of this paper: embedding the theory of † in KA with products.

▶ **Theorem 6.** *Let $\mathcal{C}$ be a category with cartesian structure given by $\times, \pi_1, \pi_2, \langle\cdot,\cdot\rangle, \mathbb{1}, \bot$. Let $(P, \eta, \mu), t$ be a commutative strong monad over $\mathcal{C}$ with tensorial strength $t_{X,Y} : X \times PY \to P(X \times Y)$. Assume additionally that (P, η, μ) is a nondeterministic monad together with $+$ and $\perp\!\!\!\perp$, and also that the axiom $\langle \mathrm{id}, f + g\rangle; t = \langle \mathrm{id}, f\rangle; t + \langle \mathrm{id}, g\rangle; t$ holds. Then, the equations $\langle\!\langle f_1 + f_2, g\rangle\!\rangle = \langle\!\langle f_1, g\rangle\!\rangle + \langle\!\langle f_2, g\rangle\!\rangle$ and $\langle\!\langle f, g_1 + g_2\rangle\!\rangle = \langle\!\langle f, g_1\rangle\!\rangle + \langle\!\langle f, g_2\rangle\!\rangle$ hold.*

The stipulated axiom $\langle \mathrm{id}, f + g\rangle; t = \langle \mathrm{id}, f\rangle; t + \langle \mathrm{id}, g\rangle; t$ in Theorem 6 relates the nondeterministic structure, as given by the choice operation $+$, with the tensorial strength $t : X \times PY \to P(X \times Y)$. An equivalent characterization can be given in terms of the natural transformation $\mathsf{u}_X : PX \times PX \to PX$ as shown in Figure 5, where $\kappa = \langle \mathrm{id} \times \pi_1, \mathrm{id} \times \pi_2\rangle : X \times (Y \times Z) \to (X \times Y) \times (X \times Z)$. In the set-theoretic models that we will consider, the equation corresponding to the right diagram of Figure 5 simply states that $A \times (B \cup C) = (A \times B) \cup (A \times C)$ for sets A, B, C. Similarly, we can also give an

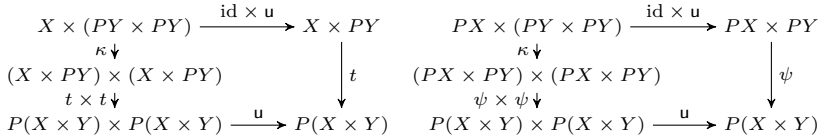


Figure 5 Axiom relating the binary union natural transformation $\mathsf{u}_X : PX \times PX \to PX$ with the tensorial strength $t : X \times PY \to P(X \times Y)$ or, equivalently, with $\psi : PX \times PY \to P(X \times Y)$.

Table 1 Kleisli category of a nondeterministic strong monad with iteration.

$$\begin{array}{rcl}
\eta_X ; f = f & \langle\!\langle f_1, f_2 \rangle\!\rangle ; \varpi_i = f_i & f = \mathbb{\perp}_{X\mathbb{1}} \\
f ; \eta_Y = f & \langle\!\langle h ; \varpi_1, h ; \varpi_2 \rangle\!\rangle = h \ (h \text{ det.}) & f ; \mathbb{\perp}_{YZ} = \mathbb{\perp}_{XZ} \\
(f ; g) ; h = f ; (g ; h) & f \text{ det.: } f ; \langle\!\langle g_1, g_2 \rangle\!\rangle = \langle\!\langle f ; g_1, f ; g_2 \rangle\!\rangle & \\
 & (f_1 \otimes f_2) ; (g_1 \otimes g_2) = (f_1 ; g_1) \otimes (f_2 ; g_2) & \\
(f + g) + h = f + (g + h) & f ; (g_1 + g_2) = f ; g_1 + f ; g_2 & \eta_X + f ; f^* \leq f^* \\
f + g = g + f & (f_1 + f_2) ; g = f_1 ; g + f_2 ; g & \eta_X + f^* ; f \leq f^* \\
f + \mathbb{\perp} = f & \langle\!\langle h ; \varpi_1, h ; \varpi_2 \rangle\!\rangle \geq h & f ; g \leq g \Rightarrow f^* ; g \leq g \\
f + f = f & \langle\!\langle f_1 + f_2, g \rangle\!\rangle = \langle\!\langle f_1, g \rangle\!\rangle + \langle\!\langle f_2, g \rangle\!\rangle & g ; f \leq g \Rightarrow g ; f^* \leq g \\
 & \langle\!\langle f, g_1 + g_2 \rangle\!\rangle = \langle\!\langle f, g_1 \rangle\!\rangle + \langle\!\langle f, g_2 \rangle\!\rangle &
\end{array}$$

equivalent characterization in terms of $\mathsf{d}_X : X \times X \to PX$ according to the equation $(\mathrm{id}_X \times \mathsf{d}_Y) ; t_{X,Y} = \kappa ; \mathsf{d}_{X \times Y} : X \times (Y \times Y) \to P(X \times Y)$.

▶ **Definition 7.** Let $\mathcal{C}$ be a category with cartesian structure and bottom elements. We say that $(P, \eta, \mu), \psi, \mathsf{u}, {}^*$ is a *nondeterministic strong monad with iteration* if:

(i) $(P, \eta, \mu), \psi$ is a commutative strong monad with lazy pairs.
(ii) $(P, \eta, \mu), \mathsf{u}, \mathbb{\perp}$ is a nondeterministic monad, where $\mathbb{\perp}_{XY} = \perp_{XY} ; \eta_Y : X \to PY$.
(iii) For all $f, g : X \to PY$, the equation $\langle \mathrm{id}, f + g \rangle ; t = \langle \mathrm{id}, f \rangle ; t + \langle \mathrm{id}, g \rangle ; t$ holds, where t is the tensorial strength induced by ψ and $+$ is the choice operation induced by u.
(iv) The axiom $\mathrm{id}_{P(X \times Y)} \leq \langle P\pi_1, P\pi_2 \rangle ; \psi$ holds, where $\leq$ is the order induced by $+$.
(v) The *iteration* operation * sends $f : X \to PX$ to $f^* : X \to PX$. The axioms

$$\frac{f : X \rightharpoonup X}{\eta_X + f ; f^* \leq f^*} \qquad \frac{f : X \rightharpoonup X}{\eta_X + f^* ; f \leq f^*} \qquad \frac{f : X \rightharpoonup X \quad g : X \rightharpoonup Y}{f ; g \leq g \Rightarrow f^* ; g \leq g} \qquad \frac{g : X \rightharpoonup Y \quad f : Y \rightharpoonup Y}{g ; f \leq g \Rightarrow g ; f^* \leq g}$$

are satisfied. These are the axioms for the Kleene star operation introduced in [18, 19].

▶ **Theorem 8.** *Let $\mathcal{C}$ be a category with cartesian structure and bottom elements. Let $(P, \eta, \mu), \psi, \mathsf{u}, {}^*$ be a nondeterministic strong monad with iteration. The Kleisli category $\mathcal{C}_P$ with composition $;$ and identities η, together with the operations $\varpi_1, \varpi_2, \langle\!\langle \cdot, \cdot \rangle\!\rangle, \mathbb{\perp}, +, {}^*$ (Kleisli projections, pairing, bottoms, choice, and iteration) satisfies the axioms of Table 1.*

A reasonable question to consider is whether there is any interaction between the iteration operation * and the tensorial strength. Even though it is not necessary for our purposes here, we note that two extra very natural axioms for the interaction between (non)determinism and Kleisli products together with the * axioms are sufficient to capture the interaction between * and ψ considered in [11]. We elaborate on this in the appendix.

4.1 The Lowerset Monad

In the usual relational interpretation of programs, a (strict) nondeterministic program $f : X \rightharpoonup Y$ is interpreted as a morphism in the category **Rel** of sets and binary relations. The category **Rel** is isomorphic to the Kleisli category $\mathbf{Set}_\wp$ of the powerset monad $\wp$ over the category **Set** of sets and total functions. In order to model lazy evaluation, the category $\mathbf{Set}_\wp$ (together with the operations of Kleisli composition and cartesian product) is not appropriate, since we need an explicit notion of divergence, non-strictness, and lazy pairs.

Instead, we consider the category **Pposet** of *pointed posets* (partial orders with a bottom element) and monotone functions, in which we can interpret non-strict deterministic programs that form lazy pairs. The bottom element $\bot_X$ of a pointed poset X denotes divergence. The arrows in **Pposet** can be partial, in the sense that they can send a non-bottom element to bottom. For an object $(X, \leq)$ of **Pposet**, we understand the partial order $\leq$ as follows: $x \leq y$ intuitively means that x "has more diverging components" than y.

Pposet is a cartesian category with bottom elements. The product $X \times Y$ of two objects X, Y is the cartesian product together with the pointwise partial order. So, the bottom element of $X \times Y$ is $\bot_{X\times Y} = \langle \bot_X, \bot_Y \rangle$. The projections π_1 and π_2 are given by $\pi_i(x_1, x_2) = x_i$. The pairing operation $\langle \cdot, \cdot \rangle$ is defined as $\langle f, g \rangle = \lambda x \in X.\langle f(x), g(x) \rangle$ for $f : X \to Y$ and $g : X \to Z$. The terminal object is some singleton poset $\mathbb{1} = \{\bot_{\mathbb{1}}\}$. The bottom global element $\bot_{\mathbb{1}X} : \mathbb{1} \to X$ is the map that sends $\bot_{\mathbb{1}}$ to the bottom $\bot_X$ of X. So, $\bot_{XY} = \bot_X ; \bot_{\mathbb{1}Y}$ is the function that always diverges. We define the pointwise partial order $\leq$ on every homset $\mathbf{Pposet}(X, Y)$. Then, $\bot_{XY}$ is the bottom element of $\mathbf{Pposet}(X, Y)$.

▶ **Definition 9** (lowerset monad)**.** Let $(X, \leq)$ be a pointed poset, the bottom element of which is denoted $\bot_X$. For a subset $S \subseteq X$ define $\downarrow S = \{y \in X \mid y \leq x \text{ for some } x \in S\}$ to be the lowerset of X generated by S. For $x \in X$, we write $\downarrow x$ to mean $\downarrow\{x\}$. Define $\wp_\downarrow X$ to be the set of all non-empty lowersets of X. We observe that $\wp_\downarrow X$ is a complete lattice w.r.t. set inclusion. The top element is X and the bottom is $\{\bot_X\}$. The join is set-theoretic union and the meet is set-theoretic intersection. It follows that the homset $\mathbf{Pposet}(X, \wp_\downarrow Y)$ (w.r.t. the pointwise order induced by $\wp_\downarrow Y, \subseteq$) is also a complete lattice. We extend $\wp_\downarrow$ to an endofunctor $\mathbf{Pposet} \to \mathbf{Pposet}$ by putting $(\wp_\downarrow f)(S) := \downarrow\{f(x) \mid x \in S\}$ for every $S \in \wp_\downarrow X$. Together with the families of maps $\eta_X : x \in X \mapsto \downarrow x \in \wp_\downarrow X$ and $\mu_X : \mathcal{S} \in \wp^2_\downarrow X \mapsto \bigcup \mathcal{S} \in \wp_\downarrow X$ it forms a monad over **Pposet**. We call $(\wp_\downarrow, \eta, \mu)$ the *lowerset monad* over **Pposet**. Kleisli composition is given by $(f ; g)(x) = \bigcup_{y \in f(x)} g(y)$.

▶ **Theorem 10.** *Define* $\psi_{X,Y} : \wp_\downarrow X \times \wp_\downarrow Y \to \wp_\downarrow(X \times Y)$ *by* $(S_1, S_2) \mapsto S_1 \times S_2$ *and* $\mathsf{u}_X : \wp_\downarrow X \times \wp_\downarrow X \to \wp_\downarrow X$ *by* $(S_1, S_2) \mapsto S_1 \cup S_2$*. For* $f : X \to \wp_\downarrow X$*, define* $f^i : X \to \wp_\downarrow X$ *by induction:* $f^0 = \eta_X$ *and* $f^{i+1} = f^i ; f$*. Put* $f^* := \bigvee_{i<\omega} f^i = \lambda x \in X. \bigcup_{i<\omega} f^i(x)$*, where* $\bigvee$ *is the join of the complete lattice* $\boldsymbol{Pposet}(X, \wp_\downarrow X)$*. The lowerset monad* $(\wp_\downarrow, \eta, \mu)$ *over* ***Pposet****, together with* $\psi, \mathsf{u}, ^*$*, is a nondeterministic strong monad with iteration.*

4.2 The Ideal Completion Monad

An *ω-complete partial order* (ω-CPO) is a partially ordered set $(X, \leq)$ that has a least element $\bot_X$ and is *ω-complete* in the sense that every ω-chain (countable chain) $x_0 \leq x_1 \leq \cdots$ has a supremum $\sup_i x_i$. A function $f : X \to Y$ between ω-CPOs is called *ω-continuous* if it preserves suprema of ω-chains. That is, for every ω-chain $x_0 \leq x_1 \leq \cdots$ in X, $f(\sup_i x_i) = \sup_i f(x_i)$. An ω-continuous function is monotone. An ω-continuous function is *strict* if $f(\bot) = \bot$. If X, Y are ω-CPOs, then so is their cartesian product $X \times Y$ under the componentwise order: $(x_1, x_2) \leq (y_1, y_2)$ iff $x_1 \leq y_1 \wedge x_2 \leq y_2$. The least element is $\bot_{X\times Y} = (\bot_X, \bot_Y)$. For an ω-chain $(x_0, y_0) \leq (x_1, y_1) \leq \cdots$ in $X \times Y$, $\sup_i(x_i, y_i) = (\sup_i x_i, \sup_i y_i)$. We denote by $[X \to Y]$ the ω-CPO of all ω-continuous functions from X to Y ordered pointwise: $f \leq g$ iff $\forall x \in P.\ f(x) \leq g(x)$. The bottom element is $\bot_{XY} = \lambda x \in X.\bot_Y$. The supremum of a chain $f_0 \leq f_1 \leq \cdots$ in $[X \to Y]$ is $\sup_i f_i = \lambda x \in X. \sup_i f_i(x)$ and it is ω-continuous. The operations ; and $\langle \cdot, \cdot \rangle$ are monotone in all arguments. The ω-continuous functions on ω-CPOs are closed under well-typed composition and pairing and contain all identities and projections. Thus, ω-CPOs and ω-continuous functions form a cartesian category with bottom elements denoted **CPO**.

Table 2 Typing rules for KA with products.

$$\frac{f : X \to\!\!\!\!\bullet\; Y}{f : X \rightharpoonup Y} \qquad \frac{f : X \rightharpoonup Y \qquad g : Y \rightharpoonup Z}{f;g : X \rightharpoonup Z} \qquad \frac{f : X \rightharpoonup Y \qquad g : X \rightharpoonup Z}{\langle f, g\rangle : X \rightharpoonup Y \times Z} \qquad \frac{f, g : X \rightharpoonup Y}{f + g : X \rightharpoonup Y}$$

$$\frac{f : X \to\!\!\!\!\bullet\; Y \qquad g : Y \to\!\!\!\!\bullet\; Z}{f;g : X \to\!\!\!\!\bullet\; Z} \qquad \frac{f : X \to\!\!\!\!\bullet\; Y \qquad g : X \to\!\!\!\!\bullet\; Z}{\langle f, g\rangle : X \to\!\!\!\!\bullet\; Y \times Z} \qquad \frac{f : X \rightharpoonup X}{f^* : X \rightharpoonup X}$$

Let $(X, \leq)$ be an ω-CPO. A subset $I \subseteq X$ is called an *ideal* of X if it is a non-empty lower set and is closed under suprema of ω-chains. The set of all ideals of X is denoted $\mathfrak{I}X$. We denote by $\mathrm{cl}_X(S)$ the smallest ideal containing $S \subseteq X$. This is an operation of type $\mathrm{cl}_X : \wp X \to \mathfrak{I}X$. We also write $\mathrm{cl}(S)$ instead of $\mathrm{cl}_X(S)$ when no confusion arises. We say that $\mathrm{cl}(S)$ is the *ideal generated by* S. The set $\downarrow x$ is an ideal and we call it the *principal ideal* generated by x. If $x_1 \leq x_2 \leq \ldots$ is an ω-chain in X, then $\mathrm{cl}[\{x_1, x_2, \ldots\}] = \downarrow \sup_i x_i$. The set $\mathfrak{I}X$ of all ideals of an ω-CPO X is a complete lattice w.r.t. set-theoretic inclusion. The meet $\bigwedge$ is set-theoretic intersection and the join $\bigvee$ is the ideal generated by the set-theoretic union. The bottom element is $\{\bot_X\}$ and the top element is X. If I, J are ideals of X, then so is $I \cup J$. In particular, $\mathrm{cl}(S_1 \cup S_2) = \mathrm{cl}(S_1) \cup \mathrm{cl}(S_2)$. Let X, Y be ω-CPOs. If I is an ideal of X and J is an ideal of Y, then $I \times J$ is an ideal of the ω-CPO $X \times Y$. If K is an ideal of $X \times Y$, then the left and right projections of K are ideals of X and Y respectively.

▶ **Definition 11** (the ideal completion monad)**.** We extend $\mathfrak{I}$ to an endofunctor $\mathbf{CPO} \to \mathbf{CPO}$ by putting $(\mathfrak{I}f)(S) := \mathrm{cl}_Y[f(S)] = \mathrm{cl}_Y\{f(x) \mid x \in S\}$, for every $S \in \mathfrak{I}X$, where $f : X \to Y$. We define the family of functions $\eta_X : X \to \mathfrak{I}X$ by $\eta_X(x) := \downarrow x$, and $\mu_X : \mathfrak{I}^2 X \to \mathfrak{I}X$ by $\mu_X(\mathcal{S}) := \bigvee \mathcal{S}$. Now, we claim that $(\mathfrak{I}, \eta, \mu)$ is a monad, called the *ideal completion monad* of **CPO**. Kleisli composition can be given as $(f \mathbin{;} g)(x) = \bigvee_{y \in f(x)} g(y)$.

▶ **Theorem 12.** *Define* $\psi_{X,Y} : \mathfrak{I}X \times \mathfrak{I}Y \to \mathfrak{I}(X \times Y)$ *as* $(I, J) \mapsto I \times J$ *and* $\mathsf{u}_X : \mathfrak{I}X \times \mathfrak{I}X \to \mathfrak{I}X$ *as* $(I, J) \mapsto I \cup J$*. Also define the iteration operation by* $f^* := \bigvee_{i<\omega} f^i = \lambda x \in X.\ \bigvee_{i<\omega} f^i(x)$*. The ideal completion monad* $(\mathfrak{I}, \eta, \mu)$ *over the category* **CPO***, together with* $\psi, \mathsf{u}, ^*$*, is a nondeterministic strong monad with iteration.*

5 Typed Kleene Algebra with Products

Let Ω be a set of *atomic types*. Let $\mathbb{1} \notin \Omega$ be a special constant called the *unit type*. The set of *types over* Ω, denoted $\mathsf{Types}(\Omega)$, is the set of terms freely generated by Ω and $\mathbb{1}$ under the binary product type constructor $\times$. The terms of the language are typed. The types of terms are expressions of the form $X \rightharpoonup Y$, where $X, Y \in \mathsf{Types}(\Omega)$. We indicate the type of a term by writing $f : X \rightharpoonup Y$. Some of the terms will be designated as *deterministic* by writing $f : X \to\!\!\!\!\bullet\; Y$. We think of $X \to\!\!\!\!\bullet\; Y$ as a subtype of $X \rightharpoonup Y$ and hence we include the typing rule given in the first column of Table 2.

Let H be a set of *atomic arrows*, each endowed with a fixed type $h : X \rightharpoonup Y$. We write $h : X \to\!\!\!\!\bullet\; Y$ for a deterministic atomic arrow of H. Let id, π_1, π_2, $\bot$ be special deterministic polymorphic constants called *identities*, *left projections*, *right projections*, and *bottoms*, respectively. Where necessary, we use subscripts or superscripts to denote the specialization at a particular type; e.g., $\mathrm{id}_X : X \to\!\!\!\!\bullet\; X$, $\pi_1^{XY} : X \times Y \to\!\!\!\!\bullet\; X$, or $\bot_{XY} : X \to\!\!\!\!\bullet\; Y$. Let $;$ and $\langle \cdot, \cdot \rangle$ be polymorphic constructors called *composition* and *pairing*, respectively, satisfying the typing rules shown in Table 2. Note that compositions $f; g$ are written in diagrammatic order. Composition and pairing preserve determinism. We add to the language the polymorphic constructors $+$ and *, called *(nondeterministic) choice* and *iteration* respectively, with the typing rules given in Table 2. Choice and iteration introduce nondeterminism.

Table 3 Axioms of Park and KA.

Park	KA
$f \le g \wedge g \le h \Rightarrow f \le h$ $f \le g \wedge g \le f \Rightarrow f = g$ $\bot \le f$ and $f \le f$	$f + (g + h) = (f + g) + h$ $f + g = g + f$ $f + \bot = f = f + f$
$f;(g;h) = (f;g);h$ $\mathrm{id};f = f = f;\mathrm{id}$	$f;(g;h) = (f;g);h$ $f;\mathrm{id} = f = f;\mathrm{id}$
$g_1 \le g_2 \Rightarrow f;g_1 \le f;g_2$ $f_1 \le f_2 \Rightarrow f_1;g \le f_2;g$	$f;(g_1 + g_2) = f;g_1 + f;g_2$ $(f_1 + f_2);g = f_1;g + f_2;g$
	$f;\bot = \bot$
$\langle f^\dagger, \mathrm{id}_Y\rangle;f \le f^\dagger$ $\langle g, \mathrm{id}_Y\rangle;f \le g \Rightarrow f^\dagger \le g$ $g;f^\dagger \le [(\mathrm{id}_X \times g);f]^\dagger$	$\mathrm{id} + f;f^* \le f^* \quad f;g \le g \to f^*;g \le g$ $\mathrm{id} + f^*;f \le f^* \quad g;f \le g \to g;f^* \le g$
$f = \bot_X : X \to 1 \quad \langle h;\pi_1, h;\pi_2\rangle = h$ $\langle f_1, f_2\rangle;\pi_i = f_i$	$f = \bot_X : X \to 1 \quad \langle h;\pi_1, h;\pi_2\rangle = h$ (det. h) $\langle f_1, f_2\rangle;\pi_i = f_i$
	$(f_1 \times f_2);(g_1 \times g_2) = (f_1;g_1) \times (f_2;g_2)$ $h \le \langle h;\pi_1, h;\pi_2\rangle$ $f;\langle g_1, g_2\rangle = \langle f;g_1, f;g_2\rangle$ (det. f)
$f_1 \le f_2 \Rightarrow \langle f_1, g\rangle \le \langle f_2, g\rangle$ $g_1 \le g_2 \Rightarrow \langle f, g_1\rangle \le \langle f, g_2\rangle$	$\langle f_1 + f_2, g\rangle = \langle f_1, g\rangle + \langle f_2, g\rangle$ $\langle f, g_1 + g_2\rangle = \langle f, g_1\rangle + \langle f, g_2\rangle$

▶ **Definition 13.** A *typed Kleene algebra with products* is a multi-sorted algebraic structure $\mathcal{K} = \big(\mathcal{K}_0, \times, \mathbb{1}, \mathcal{K}_1, \mathcal{K}_1^d, \mathsf{dom}, \mathsf{cod}, +, ;, ^*, \bot, \mathrm{id}, \langle\cdot,\cdot\rangle, \pi_1, \pi_2\big)$, where $\mathcal{K}_0$ is the set of *objects*, $\mathcal{K}_1$ is the set of *arrows* or *elements*, and $\mathcal{K}_1^d \subseteq \mathcal{K}_1$ is the set of *deterministic arrows*.

- The operations dom and cod (called *domain* and *codomain*) map arrows to objects.
- The *type* of an element f of $\mathcal{K}$ is the expression $X \rightharpoonup Y$, where $X = \mathsf{dom} f$ and $Y = \mathsf{cod} f$. We write $f : X \rightharpoonup Y$ to denote this. If $f \in \mathcal{K}_1^d$, we write $f : X \to Y$.
- The distinguished *product* operation $\times$ is a function $\times : \mathcal{K}_0 \times \mathcal{K}_0 \to \mathcal{K}_0$. The object $\mathbb{1} \in \mathcal{K}_0$ is the distinguished *terminal object* of the structure.
- The polymorphic operations and constants $+$, $;$, *, $\bot$, id, $\langle\cdot,\cdot\rangle$, π_1, and π_2 satisfy the expected typing rules.
- Additionally, the structure is a model of the well-typed instances of the axioms given in the second column of Table 3. The partial order $\le$ is induced by $+$: $f \le g$ iff $f + g = g$. The product $\times$ is defined by: $f \times g = \langle \pi_1; f, \pi_2; g\rangle$.

We have included all the axioms of KA [18, 19] except for the *strictness* axiom $\bot;f = \bot$.

We will denote by KA the quasi-equational system of typed Kleene algebra with products. Notice that (up to a slight change in notation to make it less cumbersome), the axioms satisfied by a typed Kleene algebra with products are exactly those given in Table 1. So, any small subcategory of the Kleisli category of a nondeterministic strong monad with iteration that is closed under the appropriate operations is a typed Kleene algebra with products.

6 Iteration Theories

The language of iteration theories consists of *atomic typed actions* $h : n \to m$, where n, m are natural numbers, and polymorphic operation symbols ; (composition), id (identity), ι (injection), $[-,-,\cdots,-]$ (cotupling), $\bot$ (bottom), † (dagger). The typing rules for the language are the following: $\mathrm{id}_n : n \to n$, $\iota_i^n : 1 \to n$, $\bot_{nm} : n \to m$, and

$$\frac{f : n \to m \qquad g : m \to p}{f;g : n \to p} \qquad \frac{f_i : 1 \to m \qquad i = 1, \ldots, n}{[f_1, f_2, \ldots, f_n] : n \to m} \qquad \frac{f : n \to n + p}{f^\dagger : n \to p}$$

The *typed terms* $f : n \to m$ of the language are built from the atomic actions and the operation symbols according to the typing rules.

Consider now the category **CPO** of ω-CPOs and ω-continuous maps, which will provide the *standard interpretation* for the language of iteration theories. An interpretation $[\![\cdot]\!]$ in **CPO** consists of an ω-CPO A and a mapping of every atomic symbol $h : n \to m$ to a morphism $[\![h]\!] : A^m \to A^n$ in **CPO**, where A^k denotes the k-fold associative cartesian product of A. The identity symbol $\mathrm{id}_n : n \to n$ is interpreted as the identity function $[\![\mathrm{id}_n]\!] : A^n \to A^n$. The injection symbol $\iota_i^n : 1 \to n$ is interpreted as the i-th projection $[\![\iota_i^n]\!] : A^n \to A$. The bottom symbol $\bot_{nm} : n \to m$ is interpreted as the least function $[\![\bot_{nm}]\!] : A^m \to A^n$ of $\mathbf{CPO}(A^m, A^n)$. Now, ; and $[-, -, \cdots, -]$ are interpreted as function composition and tupling respectively. For functions $f_i : 1 \to m$, $i = 1, \ldots, n$, the function denoted by $[f_1, \ldots, f_n] : n \to m$ is given by $[\![[f_1, \ldots, f_n]]\!](\bar{x}) := \langle [\![f_1]\!](\bar{x}), \ldots, [\![f_n]\!](\bar{x})\rangle$, for every $\bar{x} \in A^m$. Every ω-continuous function $\phi : X \to X$ in **CPO** has a least fixpoint, which is the supremum of the ω-chain $\bot \le \phi(\bot) \le \phi^2(\bot) \le \cdots \le \phi^n(\bot) \le \cdots$, where $\phi^0 = \mathrm{id}$ and $\phi^{n+1} = \phi^n; \phi$. We denote the least fixpoint of ϕ by $\mu(\phi) = \sup_i \phi^i(\bot)$. For an ω-continuous function $\phi : X \times Y \to X$, we define the function $\phi^\dagger : Y \to X$ by $\phi^\dagger(y) := \mu(\phi_y) = \sup_i \phi_y^i(\bot)$, where $\phi_y = \lambda x \in X.\phi(x, y) : X \to X$. The function $\phi^\dagger : Y \to X$ is also ω-continuous. We call † the *parametric fixpoint operation*. The operation † is monotone. We interpret the dagger symbol † of the language as parametric fixpoint †: for $f : n \to n + p$ we define $[\![f^\dagger]\!] = \big([\![f]\!] : A^n \times A^p \to A^n\big)^\dagger : A^p \to A^n$.

Every homset $\mathbf{CPO}(X, Y)$ is equipped with the pointwise partial order $\le$. For terms s, t we write $\mathbf{CPO} \models s \le t$ to mean that $[\![s]\!] \le [\![t]\!]$ for every interpretation $[\![\cdot]\!]$ of the language in **CPO**. Define $\mathsf{Th}(\mathbf{CPO})$ to be the set of all valid inequalities over **CPO**, that is, $\mathsf{Th}(\mathbf{CPO}) := \{s \le t \mid \mathbf{CPO} \models s \le t\}$, where s, t are terms of the language of iteration theories. $\mathsf{Th}(\mathbf{CPO})$ is the "(in)equational theory of iteration", in the words of Ésik [9].

▶ **Definition 14.** Define Park to be the universal Horn system (including equality) with: axioms for categories, axioms asserting that ι, $[-, -, \cdots, -]$, and $\bot$ give associative categorical coproducts, axioms stating that $\le$ is a partial order, that ; and $[-, -, \cdots, -]$ are motonone in all arguments w.r.t. $\le$, and that $\bot_{nm} : n \to m$ is the least element of $\mathrm{Hom}(n, m)$, and axioms for the dagger operation:

$$\frac{f : n \to n + p}{f; [f^\dagger, \mathrm{id}_p] \le f^\dagger : n \to p} \qquad \frac{f : n \to n + p \quad g : n \to p}{f; [g, \mathrm{id}_p] \le g \Rightarrow f^\dagger \le g} \qquad \frac{f : n \to n + p \quad g : p \to q}{f^\dagger; g \le [f; (\mathrm{id}_n \oplus g)]^\dagger : n \to q}$$

where the copairing operation $[-, -]$ is induced by the cotupling operation $[-, -, \ldots, -]$ in the obvious way. The three dagger axioms are called *pre-fixpoint inequality*, *least pre-fixpoint implication* or *Park induction rule*, and *parameter inequality* respectively.

▶ **Theorem 15** (Ésik [9])**.** *Park axiomatizes $\mathsf{Th}(\mathbf{CPO})$, that is, $\mathbf{CPO} \models t_1 \le t_2$ iff $\mathsf{Park} \vdash t_1 \le t_2$, for all terms t_1, t_2 in the language of iteration theories.*

The choice of a language with coproducts and copairing/injection symbols is confusing, because the standard models we are interested in here are models of functions where the symbols are interpreted as products and pairing/projections respectively. Moreover, there is no reason to collapse isomorphic products, such as $X \times (Y \times Z) \cong (X \times Y) \times Z$ or $1 \times X \cong X$. In fact, this only complicates the technical presentation of proofs.

So, we consider for the rest of the paper (as is also done in [6, 26]) that the language of iteration theories is instead as follows: For a set Ω of atomic types, let $\mathsf{Types}(\Omega)$ be the set freely generated by Ω, $\mathbb{1} \notin \Omega$, and the product constructor $\times$. The terms of the language are typed, e.g., $f : X \to Y$, where $X, Y \in \mathsf{Types}(\Omega)$. Each atomic arrow has a fixed type $h : X \to Y$. We have polymorphic constants π_1^{XY}, π_2^{XY}, id_X, $\bot_{XY}$ and polymorphic constructors ;, $\langle\cdot, \cdot\rangle$, †. The typing rules are as usual with the exception of the rule for †: for $f : X \times Y \to X$ we have $f^\dagger : Y \to X$. Now, a standard interpretation $[\![\cdot]\!]$ in **CPO** assigns

an ω-CPO to each base type and an ω-continuous function to each atomic action. This extends in the obvious way to all terms of the language. In particular, the dagger symbol is interpreted as parametric fixpoint, e.g., $[\![f^\dagger]\!] = [\![f]\!]^\dagger : [\![Y]\!] \to [\![X]\!]$ for $f : X \times Y \to X$. See the first column of Table 3 for the axioms of Park in the language with products.

7 Embedding the Equational Theory of Iteration in KA

We augment the system of KA that we presented in Section 5 with an additional typing rule about the preservation of determinism:

$$\frac{g : X \nrightarrow Y \qquad f : Y \nrightarrow Y \qquad g \leq g; f : X \rightharpoonup Y}{g; f^* : X \nrightarrow Y}. \tag{7}$$

We note that this rule is not valid in all Kleene algebras, but it is valid in the Kleisli category $\mathbf{CPO}_\mathfrak{I}$ of the ideal completion monad over **CPO**. For a term $f : X \times Y \rightharpoonup X$ of KA, we define the abbreviation $f^\ddagger := \langle \bot_{YX}, \mathrm{id}_Y \rangle; \langle f, \pi_2 \rangle^*; \pi_1 : Y \rightharpoonup X$. We call ‡ the *derived dagger operation* in KA. Using the rule (7) we can derive in KA the *dagger typing rule*: if $f : X \times Y \nrightarrow X$ then $f^\ddagger : Y \nrightarrow X$. The dagger typing rule states that the derived dagger operation preserves determinism.

▶ **Definition 16** (translation)**.** We define a *translation* $[\cdot]$ from the language of iteration theories to the language of KA with products. All atomic action symbols and atomic constants are translated as deterministic symbols of the same type. E.g., for $h : X \to Y$ we have $[h] = h : X \nrightarrow Y$, and $[\pi_1] = \pi_1 : X \times Y \nrightarrow X$. The dagger is translated as

$$[f^\dagger] := f^\ddagger = \langle \bot, \mathrm{id} \rangle; \langle [f], \pi_2 \rangle^*; \pi_1 : Y \nrightarrow X$$

The translation function $[\cdot]$ commutes with the rest of the operation symbols.

▶ **Theorem 17** (Completeness and soundness)**.** *Let $t \leq t'$ be an inequality in the language of Park. Then,* $\mathbf{CPO} \models t \leq t'$ *iff* $\mathsf{KA} \vdash [t] \leq [t']$.

Proof sketch. For completeness we use Theorem 15 and show how to obtain the Park axioms in KA. For soundness we exhibit an operation-preserving embedding of **CPO** in $\mathbf{CPO}_\mathfrak{I}$. ◀

Even though the above result was developed using Park theories, which have a universal Horn axiomatization, we also obtain a result for the equational theory of iteration theories. Recall the definition (see e.g. [7]): an *iteration theory* is a cartesian category with a dagger operation that satisfies the equations valid in **CPO**. So, the above theorem also says that the equational theory of iteration theories is embedded in KA.

8 Related Work

The work by Goncharov [11] is closely related to ours. He defines *additive (strong) monads* and *Kleene monads* axiomatically. Calculi for an extended metalanguage of effects are defined and completeness/incompleteness results are obtained. Our notion of a "nondeterministic strong monad with iteration" is different from that of a Kleene monad: we consider non-strict programs that form lazy pairs, and we axiomatize iteration quasi-equationally. The absence of the strictness axiom $\bot; f = \bot$ from our axiomatization and the use of lazy pairs are essential for our encoding of fixpoints. In particular, the axioms stipulated in [11] would force all the parametric fixpoints to be equal to $\bot$, because in that system $\langle \bot, \mathrm{id} \rangle; \langle f, \pi_2 \rangle^*; \pi_1 = \bot; \langle f, \pi_2 \rangle^*; \pi_1 = \bot$. So, the models we are investigating in the present work are crucially different from the models considered in [11].

The work on Hoare powerdomains [1, 25], which give models of angelic nondeterministic computations, is related. The (lower) Hoare powerdomain of a domain is formed by taking

all the ideals of the domain, where by ideal we mean here the nonempty Scott-closed subsets of the domain. In the present work, we identify models of the axiomatically defined "nondeterministic monad with iteration," which are similar to and simpler than the construction of Hoare powerdomains over DCPOs. We first identify a simple model: the lowerset monad over the category of posets with bottom elements. Then, we also prove that the ideal completion monad over the category of ω-CPOs is a model.

There is a long line of work, primarily by Bloom and Ésik, under the name of "iteration theories" or the "(in)equational theory of iteration" (see e.g. [4, 6, 7, 9, 26] and references therein), which is intimately related to the work on Kleene algebra [8, 16–23] in general and the present work in particular. The axioms of iteration theories capture the equational properties of fixpoints in several classes of structures relevant to computer science. For example, they capture the equational theory of ω-continuous functions between ω-CPOs, where the algebraic signature includes symbols for composition, pairing, and parametric fixpoints. Several different equational axiomatizations have been considered in the literature, all of which require substantial effort to parse and understand. By allowing quasi-equations, simpler axiomatizations can be found. Many examples of iteration theories involve functions on posets, so it is a natural question to look for complete axiomatizations of the valid inequalities over classes of structures that are of interest, e.g., structures of ω-continuous functions over ω-CPOs. One such universal Horn axiomatization is given in [9]. This axiomatization includes two inequalities and one implication for the $\dagger$ operation, which are both intuitive and easy to memorize. We note that in the work on iteration theories, the issue of how (non)determinism interacts with pairs, which is central in the present work, is not handled at all.

Of particular relevance to the relationship between iteration theories and Kleene algebra are the works on the so-called "matrix iteration theories" [4, 5, 7]. They are cartesian categories in which the homsets are commutative monoids with respect to an operation $+$, which distributes over composition. This also induces cocartesian structure and allows an easy translation between the dagger (parametric fixpoint) operation and Kleene star. However, this translation is not sound for the classes of structures we consider. In particular, the $+$ symbol cannot be interpreted as nondeterministic choice when $\langle \cdot, \cdot \rangle$ is interpreted as pairing: the axioms imply the property $\langle a, \bot \rangle + \langle \bot, b \rangle = \langle a, b \rangle$, which is not meaningful for programs. The translation of the dagger operation in the language of KA that we give here is crucially different, and is in fact sound for the class of matrix iteration theories as well.

There is a connection between our characterizations of nondeterministic structure and the work of Plotkin and Power on algebraic operations (see e.g. [24] for an overview). The theory of algebraic operations provides a uniform semantics of computational effects by considering primitive operations of type $f_X : (PX)^n \to PX$ that are the source of effects. The binary union operation $\mathsf{u}_X : PX \times PX \to PX$ that we consider is an algebraic operation in this sense. Our purpose here, however, is very different: we axiomatize u_X and establish equivalence with other axiomatizations of nondeterministic structure that do not fit in the framework of algebraic operations, e.g., with $\mathsf{d}_X : X \times X \to PX$. In particular, for the equivalence results that we develop here, there does not seem to be any known fact about algebraic operations that can be invoked to simplify our proofs.

Our work here builds directly upon the existing work on Kleene algebra [8, 17–20, 22]. The crucial axioms for the iteration operation * are taken from [18]. The system of KA we present is a typed Kleene algebra in the sense of [21] extended with products that satisfy weaker axioms than those of categorical products.

9 Conclusion and Future Work

In the present work we reconcile the notions of iteration captured by the star operation * of KA and the dagger operation † of IT. We present and investigate a system of typed KA with products, in which the notion of a deterministic program turns out to be of importance. We work in the framework of cartesian categories combined with commutative strong monads to treat (angelic) nondeterminism. We have adapted an axiomatization of commutative strong monads from the work of Kock [13, 15] to our setting. We have described three equivalent ways, in the presence of cartesian structure, of capturing nondeterminism. We have identified two concrete monads, the monad of lowersets of pointed posets and the monad of ideals of ω-CPOs, as models. The main technical result of our paper is a translation of † in terms of * that gives an embedding of the (in)equational theory of † in KA.

The present work has been a first step in presenting a higher-order system of typed Kleene algebra. We would like to investigate what properties of recursion can be captured in such a higher-order system and how this would relate to the investigations of [6].

References

1 Samson Abramsky and Achim Jung. Domain theory, 1994.
2 J. Adámek, S. Milius, and J. Velebil. Elgot algebras. *Log. Meth. Comp. Sci.*, 2:1–31, 2006.
3 J. Adámek, S. Milius, and J. Velebil. Elgot theories: a new perspective of the equational properties of iteration. *Math. Structures Comput. Sci.*, 21(2):417–480, 2011.
4 S. L. Bloom and Z. Ésik. *Iteration Theories*. Springer, 1993.
5 S. L. Bloom and Z. Ésik. Matrix and matricial iteration theories, part I. *Journal of Computer and System Sciences*, 46(3):381 – 408, 1993.
6 S. L. Bloom and Z. Ésik. Fixed-point operations on ccc's. part I. *TCS*, 155(1):1–38, 1996.
7 S. L. Bloom and Z. Ésik. The equational logic of fixed points. *TCS*, 179(1–2):1–60, 1997.
8 John Horton Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.
9 Z. Ésik. Completeness of Park induction. *Theor. Comput. Sci.*, 177(1):217–283, 1997.
10 S. Ginali. Regular trees and the free iterative theory. *JCSS*, 18:228–242, 1979.
11 Sergey Goncharov. *Kleene Monads*. PhD thesis, Universität Bremen, 2010.
12 S. C. Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, 1956.
13 A. Kock. Monads on symmetric monoidal closed categories. *Ar. der Math.*, 21:1–10, 1970.
14 A. Kock. Bilinearity and cartesian closed monads. *Mathematica Scand.*, 29:161–174, 1971.
15 A. Kock. Strong functors and monoidal monads. *Archiv der Math.*, 23:113–120, 1972.
16 L. Kot and D. Kozen. Kleene algebra and bytecode verification. *ENTCS*, 141(1):221–236, 2005.
17 D. Kozen. On Kleene algebras and closed semirings. MFCS '90, pages 26–47, 1990.
18 D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *Proceedings of 6th Annual IEEE Symp. on Logic in Comp. Sci.*, pages 214–225, 1991.
19 D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.
20 D. Kozen. Kleene algebra with tests. *ACM Trans. Prog. Lang. Syst.*, 19(3):427–443, 1997.
21 D. Kozen. Typed Kleene algebra. Technical report, Cornell University, 1998.
22 D. Kozen. On Hoare logic and Kleene algebra with tests. *TOCL*, 1(1):60–76, 2000.
23 D. Kozen and M.-C. Patron. Certification of compiler optimizations using Kleene algebra with tests. In *Proceed. of the First Intern. Conf. on Comput. Logic*, pages 568–582, 2000.
24 G. Plotkin and J. Power. Computational effects and operations. *ENTCS*, 73:149–163, 2004.
25 Gordon Plotkin. Domains, 1983. Pisa notes on domain theory.
26 A. Simpson and G. Plotkin. Complete axioms for categorical fixed-point operators. In *Proceedings of 15th Annual IEEE Symp. on Logic in Comp. Sci.*, pages 30 –41, 2000.

A Interaction between tensorial strength and iteration

In [11] an axiom is considered that relates the iteration operation μ with the tensorial strength of the nondeterministic monad:

$$\mu f.\big((\mathrm{id}_X \times p); t_{X,Y} + f; (\mathrm{id}_X \times q); t_{X,Y}\big) = \big(\mathrm{id}_X \times \mu f.(p + f; q)\big); t_{X,Y},$$

where $p : X \to PY$ and $q : Y \to PY$ (composition ; binds tighter than Kleisli composition $;$, which in turn binds tighter than $+$). We rewrite the above axiom in terms of ψ using the property $t_{X,Y} = (\eta_X \times \mathrm{id}); \psi_{X,Y}$:

$$\mu f.\big((\eta_X \times p); \psi_{X,Y} + f; (\eta_X \times q); \psi_{X,Y}\big) = \big(\eta_X \times \mu f.(p + f; q)\big); \psi_{X,Y}.$$

Recall the definition of the Kleisli product functor: $f \otimes g = (f \times g); \psi$. So, we can rewrite the above equation as

$$\mu f.\big((\eta_X \otimes p) + f; (\eta_X \otimes q)\big) = \eta_X \otimes \mu f.(p + f; q).$$

The expression $\mu f.\phi(f)$ is meant to denote the least fixpoint of the map $f \mapsto \phi(f)$. So, we think intuitively of the expression $\mu f.(p + f; q)$ as denoting the supremum of the chain $p \leq p + p; q \leq p + p; q + p; q; q \leq \cdots$, which in our language would be represented by $p; q^*$. Using this correspondence, the axiom finally becomes

$$(\eta_X \otimes p); (\eta_X \otimes q)^* = \eta_X \otimes p; q^*$$

in the language of KA. We will identify two very fundamental axioms that together with the KA axioms allow us to prove the above equation.

Before we state the result, we give some intuition for the extra axioms we will assume. In the categorical models we have been investigating the uniqueness property $\langle h; \varpi_1, h; \varpi_2\rangle = h$ holds whenever the arrow $h : X \rightharpoonup Y \times Z$ is deterministic, but does not hold for arbitrary h. The reason for this is illustrated by the following example:

$$S = \{(a, c), (b, d)\} : \mathbb{1} \to P(X \times Y) \qquad \begin{aligned} S; \varpi_1 &= \{a, b\} : \mathbb{1} \to PX \\ S; \varpi_2 &= \{c, d\} : \mathbb{1} \to PY \end{aligned}$$

and $\langle S; \varpi_1, S; \varpi_2\rangle = \{(a, c), (a, d), (b, c), (b, d)\} \neq S$. However, if we change the example so that $S; \varpi_1$ is deterministic, then the uniqueness axiom is satisfied.

$$S = \{(a, c), (a, d)\} : \mathbb{1} \to P(X \times Y) \qquad \begin{aligned} S; \varpi_1 &= \{a\} : \mathbb{1} \to PX \\ S; \varpi_2 &= \{c, d\} : \mathbb{1} \to PY \end{aligned}$$

Notice that $S; \varpi_1 = \{a\}$ is deterministic and $\langle S; \varpi_1, S; \varpi_2\rangle = S$, even though S is not deterministic. This example motivates the rules of Table 4.

▶ **Proposition 18.** *Every nondeterministic monad with iteration that additionally satisfies the two rules of Table 4 also satisfies the equation*

$$(\eta_X \otimes p); (\eta_X \otimes q)^* = \eta_X \otimes p; q^*.$$

Table 4 Additional rules for the interaction between (non)determinism and Kleisli products.

$$\frac{f : X \rightharpoonup Y \times Z \qquad f; \varpi_1 : X \to Y}{\langle f; \varpi_1, f; \varpi_2\rangle = f : X \rightharpoonup Y \times Z} \qquad \frac{f : X \rightharpoonup Y \times Z \qquad f; \varpi_2 : X \to Z}{\langle f; \varpi_1, f; \varpi_2\rangle = f : X \rightharpoonup Y \times Z}$$

Proof. Call L the left-hand side and R the right-hand side of the equation we have to prove. The inequality $L \leq R$ is easily seen to be provable from the axioms for *. It suffices to show that $\eta_X \otimes p \leq \eta_X \otimes p; q^*$, which holds because $p \leq p; q^*$, and also that

$$\begin{aligned}(\eta_X \otimes p; q^*); (\eta_X \otimes q) &= \eta_X; \eta_X \otimes p; q^*; q && [\otimes \text{ functor}]\\ &\leq \eta_X \otimes p; q^*. && [\text{star axiom}]\end{aligned}$$

We will see now that the inequality $R \leq L$ is also provable from the star axioms, by making use of the rules given in Table 4. First, we claim that

$$L; \varpi_1 = (\eta_X \otimes p); (\eta_X \otimes q)^*; \varpi_1 = \varpi_1.$$

Since $(\eta_X \otimes q); \varpi_1 = \varpi_1; \eta_X = \varpi_1$, we have that $(\eta_X \otimes q); \varpi_1 \leq \varpi_1$ and hence $(\eta_X \otimes q)^*; \varpi_1 \leq \varpi_1$. It follows that $L; \varpi_1 \leq (\eta_X \otimes p); \varpi_1 = \varpi_1$. Moreover,

$$\varpi_1 = (\eta_X \otimes p); \varpi_1 \leq (\eta_X \otimes p); (\eta_X \otimes q)^*; \varpi_1 = L; \varpi_1.$$

We have thus shown that $L; \varpi_1 = \varpi_1$ is deterministic. So, the first rule of Table 4 gives us that

$$(\eta_X \otimes p); (\eta_X \otimes q)^* = \langle\!\langle \varpi_1, (\eta_X \otimes p); (\eta_X \otimes q)^*; \varpi_2 \rangle\!\rangle.$$

We also have by definition of $\otimes$ that $\eta_X \otimes p; q^* = \langle\!\langle \varpi_1, \varpi_2; p; q^* \rangle\!\rangle$. It suffices to show that

$$\begin{aligned}\varpi_2; p; q^* \leq (\eta_X \otimes p); (\eta_X \otimes q)^*; \varpi_2 &\Longleftarrow (\eta_X \otimes p); \varpi_2; q^* \leq (\eta_X \otimes p); (\eta_X \otimes q)^*; \varpi_2\\ &\Longleftarrow \varpi_2; q^* \leq (\eta_X \otimes q)^*; \varpi_2,\end{aligned}$$

which is implied by $\varpi_2 \leq (\eta_X \otimes q)^*; \varpi_2$ (it clearly holds) and

$$(\eta_X \otimes q)^*; \varpi_2; q = (\eta_X \otimes q)^*; (\eta_X \otimes q); \varpi_2 \leq (\eta_X \otimes q)^*; \varpi_2. \qquad \blacktriangleleft$$

So, the above proposition says, somewhat informally, that the interaction between iteration and tensorial strength that is stipulated in [11] is a consequence of the properties of * and of two fundamental axioms that involve only determinism judgments and Kleisli products. The axioms of Table 4 are sound for the models we consider here, as well as for the powerset monad $\wp$ over **Set**, which is a model of the axioms considered in [11]. We have not included these axioms in the definition of a nondeterministic strong monad with iteration, because they are not necessary for our investigations.

Internalizing Relational Parametricity in the Extensional Calculus of Constructions

Neelakantan R. Krishnaswami and Derek Dreyer

Max Planck Institute for Software Systems (MPI-SWS)
Kaiserslautern and Saarbrücken, Germany
{neelk,dreyer}@mpi-sws.org

Abstract

We give the first relationally parametric model of the extensional calculus of constructions. Our model remains as simple as traditional PER models of types, but unlike them, it additionally permits the relating of terms that implement abstract types in different ways. Using our model, we can validate the soundness of quotient types, as well as derive strong equality axioms for Church-encoded data, such as the usual induction principles for Church naturals and booleans, and the eta law for strong dependent pair types. Furthermore, we show that such equivalences, justified by relationally parametric reasoning, may soundly be internalized (*i.e.,* added as equality axioms to our type theory). Thus, we demonstrate that it is possible to interpret equality in a dependently-typed setting using parametricity. The key idea behind our approach is to interpret types as so-called *quasi-PERs* (or *zigzag-complete relations*), which enable us to model the symmetry and transitivity of equality while at the same time allowing abstract types with different representations to be equated.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Relational parametricity, dependent types, quasi-PERs

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.432

1 Introduction

Reynolds [23] famously introduced the concept of *relational parametricity* with a fable about data abstraction. Professors Bessel and Descartes, each teaching a class on complex numbers, defined them differently in the first lecture, the former using polar coordinates and the latter using (of course) cartesian coordinates. But despite accidentally trading sections after the first lecture, they never taught their students anything false, since after the first class, both professors proved all their theorems in terms of the defined operations on complex numbers, and never in terms of their underlying coordinate representation.

Reynolds formalized this idea by giving a semantics for System F in which each type denoted not just a set of well-formed terms, but a *logical relation* between them, defined recursively on the type structure of the language. Then, the fact that well-typed client programs were insensitive to a specific choice of implementation could be formalized in terms of their taking logically related inputs to logically related results. Since the two constructions of the complex numbers share the same interface, and it is easy to show they are logically related at that interface, any client of the interface must return equivalent results regardless of which implementation of the interface is used. Hence, parametricity gives a way of modelling a general notion of *representation-independent* program equivalence, significantly coarser (and thus more flexible) than standard notions of set-theoretic equality.

Subsequently, Plotkin and Abadi [22] showed how to build a logic in which parametricity could be used to prove the equivalence of System F programs. Plotkin-Abadi logic is a

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 432–451

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

system in the LCF tradition, where an external logic is used to reason about the behavior of programs written in a particular programming language. This logic lets us prove, for example, that the polar and cartesian representations of the complex numbers are indeed equal at a suitable existential type.

More recently, there has been a great deal of interest in unifying programming languages with their program logic, by means of dependent type theory [21]. Dependent type systems allow types to mention program terms and thereby state strong invariants about the behavior of programs. The ability to put strong preconditions on functions means that we can support operations that might be unsafe in general (*e.g.,* unchecked array access) without compromising the safety of the programming language.

However, the notion of equality available in dependent type theory has historically been limited. In intensional type theories (*e.g.,* [2]), equality is given purely as the β-equality of terms. Even the more generous approach of extensional type theory [20, 1] still equips each set with an intrinsic notion of equality at the time of its definition. The approach of fixing a notion of equality is at odds with the "outside view" of equality suggested by relational parametricity, where equivalence is determined *relative* to the operations exported to a client. This limitation is especially galling given the recent work of Bernardy *et al.* [6], who show that the *syntax* of type theory is wholly compatible with parametricity—every well-typed term in the calculus of constructions respects the relationally parametric interpretation associated with its type—but there is no way to internalize this fact.

As a result, though it is possible to *define* many types such as existentials, coproducts, and dependent pairs, it is not possible to prove that they satisfy the expected equational properties (*e.g.,* η-rules). This is particularly frustrating in a dependently-typed setting, where being able to internalize parametricity properties would greatly facilitate verification.

In this paper, we give the first relationally parametric model of an extensional variant of the calculus of constructions [11], by means of a realizability-style interpretation of type theory [14]. Our model lets us prove equalities and induction principles using parametricity-based reasoning, then internalize these properties as new axioms. In other words, we can *internalize relational parametricity* into our dependent type theory.

We interpret the types of the calculus of constructions by means of a logical relations model, and interpret the identity type using the relations our model defines. However, an off-the-shelf logical relation is only guaranteed to be a reflexive congruence for well-typed terms (the *fundamental property*). Logical relations in general are not necessarily symmetric or transitive, both of which are needed to use the relation as a model of *equality*.

A common approach to gaining symmetry and transitivity is to require the relations interpreting types to be partial equivalence relations (PERs), which are symmetric and transitive by definition. However, we pay a high price for mandating symmetry: relating terms with different representations (*e.g.,* Peano and binary numbers) is no longer possible, since we can no longer relate different things on each side of the relation. Consequently, we can no longer prove representation independence results in a natural way.

Traditionally, this difficulty is resolved by giving a Reynolds-style relational semantics for the language *together* with a PER model in a mutually recursive construction, and proving a correspondence between the two (the *identity extension lemma*). However, this approach more than doubles the work involved in defining a model, which hits particularly hard when facing the complex models of dependent type theory. Moreover, to our knowledge, no such parametric PER models of dependent type theory have yet been developed.

Our key innovation is to model types instead using so-called *quasi-PERs* (a.k.a. *difunctional relations*, or *zigzag-complete relations*), which generalize PERs to the asymmetric

$$
\begin{array}{llll}
\kappa & ::= & * \mid \Pi x : X.\, \kappa \mid \Pi \alpha : \kappa.\, \kappa' & \text{Kinds} \\
X, A & ::= & \Pi \alpha : \kappa.\, X \mid \Pi x : X.\, Y \mid e =_X e' & \text{Types} \\
 & \mid & \lambda x : X.\, A \mid A\ e \mid \lambda \alpha : \kappa.\, A \mid A\ B \mid \alpha & \\
e & ::= & x \mid \lambda x : X.\, e \mid e\ e \mid \lambda \alpha : \kappa.\, e \mid e\ A \mid \mathsf{refl} & \text{Terms} \\
v & ::= & \lambda x : X.\, e \mid \lambda \alpha : \kappa.\, e \mid \mathsf{refl} & \text{Values} \\
\Gamma & ::= & \cdot \mid \Gamma, x : X \mid \Gamma, \alpha : \kappa & \text{Contexts}
\end{array}
$$

Figure 1 Syntax.

case. Using quasi-PERs, we give a single relational model which supports symmetry and transitivity of equality *as well as* the relating of terms with differing type representations.

To illustrate this point, we show how to state parametricity axioms for some types familiar from System F such as Church numerals and existentials. Exploiting the presence of type dependency, we also show how parametricity can be used to recover strong dependent pairs (*i.e.,* $\Sigma x : X.\, Y$ with π_1 and π_2 projections) from a Church encoding, as well as showing the soundness of quotient types in our model. Proof details can be found in the accompanying extended technical appendix, available online at `http://www.mpi-sws.org/~dreyer/`.

2 Syntax, Typing and Operational Semantics

Our overall system is an explicitly-typed version of the calculus of constructions, extended with an identity type and an elimination rule for equality based on equality reflection. We use extensional type theory to make equality axioms (e.g., η for Church encodings of pairs) behave well. In an intensional system, equality axioms make subject reduction fail, since the eliminator for the equality type gets stuck. Extensional type theory makes equality elimination implicit, and so has better computational behavior. In Figure 1, we give the syntactic categories of our type system, and in Figure 2, we list the judgements in our system. Most of the kinding and typing rules are standard, as is the standard call-by-name evaluation semantics, and so we omit them for space (they can be found in the appendix).

We present our system with distinct syntactic categories for kinds (ranged over by metavariables κ), types (ranged over by metavariables X, Y, A, B, C, and type variables α, β) and terms (ranged over by metavariables e, and term variables x, y, z). We typically adopt the convention of using A, B, and C for type constructors of arbitrary kind, and X and Y for type constructors of base kind. Almost all of the typing rules are standard for the calculus of constructions, and we discuss only our variations in detail.

Our treatment of equality follows extensional type theory. We introduce identities with refl when two terms are equal, and so the definitional equality for identity types satisfies the uniqueness of identity proofs property (a.k.a. Axiom K [17]). We also support equality reflection: if $\Gamma \vdash e_p : e =_X e'$, then $\Gamma \vdash e \equiv e' : X$. This rule makes typechecking undecidable, as typing derivations may need to invent equality proofs. To summarize, our definitional equality is just the $\beta\eta$-theory of the lambda calculus plus rules for equality types. For expository reasons, we have not included any parametricity properties in definitional equality, saving these for Section 5.

3 Semantics

Our overall semantics is a realizability model, in which types are interpreted as relations between closed terms e. However, since the syntactic types appearing within terms are

Γ ok	Context Well-formedness	$\Gamma \vdash \kappa : \mathsf{kind}$	Kind Well-formedness
$\Gamma \vdash A : \kappa$	Kinding of Type Constructors	$\Gamma \vdash e : X$	Typing of Terms
$\Gamma \vdash \kappa \equiv \kappa' : \mathsf{kind}$	Kind Equality	$\Gamma \vdash A \equiv A' : \kappa$	Type Equality
$\Gamma \vdash e \equiv e' : X$	Term Equality	$e \mapsto e'$	Operational Semantics

Figure 2 Summary of Judgments.

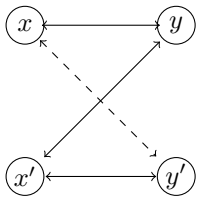
computationally irrelevant, we simplify matters by working with relations over **Exp**, the set of equivalence classes of closed terms modulo differences in syntactic types. That is, in the model, we consider $\lambda x : X.\ e = \lambda x : Y.\ e$, $\lambda\alpha : \kappa.\ e = \lambda\alpha : \kappa'.\ e$, and $e\ A = e\ B$, for *arbitrary* X, Y, κ, κ', A, B. This is analogous to building the model with type-erased terms, and we will sometimes write _ in place of (irrelevant) type annotations and arguments.

3.1 Quasi-PERs

The primary technical innovation in our work is to switch from a PER semantics of types to an interpretation based on *quasi-PERs*, which generalize PERs to the asymmetric case.

▶ **Definition 1.** (Quasi-PER) A quasi-PER between two sets $\hat{X}$ and $\hat{Y}$ is a *zigzag-complete* relation $R \subseteq \hat{X} \times \hat{Y}$: if $(x, y) \in R$, and $(x', y') \in R$ and $(x', y) \in R$, then $(x, y') \in R$.

The zigzag condition is best visualized pictorially:



So if R tells us that two elements of $\hat{X}$ are related to a given y, then they are related to all the same elements of $\hat{Y}$. Indeed, given a QPER $R \subseteq \hat{X} \times \hat{Y}$, both $R \circ R^{-1}$ is a PER on $\hat{X}$, and $R^{-1} \circ R$ is a PER on $\hat{Y}$. (All QPERs arise in this way, and so this could equivalently be taken as a definition of quasi-PERs.)

Like PERs, QPERs form a complete lattice. The meet of two QPERs is the intersection, and indeed they are closed under arbitrary intersections. As a result, they also have arbitrary joins, with the join $\bigsqcup \mathcal{R}$ defined as the intersection of every QPER containing $\bigcup \mathcal{R}$. The join will, in general, have more elements that the union, best illustrated by the direct construction of the join:

$$\begin{array}{lcl} \bigsqcup_0 \mathcal{R} & = & \bigcup \mathcal{R} \\ \bigsqcup_{k+1} \mathcal{R} & = & \bigsqcup_k \mathcal{R} \cup \{(x, y') \mid (x, y) \in \bigsqcup_k \mathcal{R} \ \wedge\ (x', y') \in \bigsqcup_k \mathcal{R} \ \wedge\ (x', y) \in \bigsqcup_k \mathcal{R}\} \\ \bigsqcup \mathcal{R} & = & \bigcup_{k \in \mathbb{N}} \bigsqcup_k \mathcal{R} \end{array}$$

So the join takes the union and fills in all the missing zigzags.

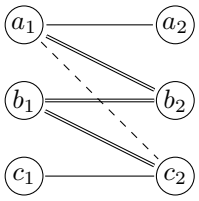
Our reason for using QPERs in our semantics of types is quite simple. When giving a relational type interpretation, we are pulled in two contrary directions. First, we want to use the relation to model representation independence: we want to be able to say that two different implementations of the same interface are equal. For this, we need to consider relations between different sets. Second, we also want to use our logical relation to *define* equality at each type. For this, we (seemingly) need symmetry and transitivity properties

on the relations we use to interpret types. These apparently conflicting demands can be reconciled by interpreting types as QPERs. While QPERs are capable of relating terms of different types, they also induce a canonical equivalence relation, which we can use in turn to model equality. The trick is that this equivalence relation is not a relation on terms, but on *pairs of related terms*.

To see how this works, first suppose we have two well-typed terms e and t at some type X. The fundamental property of logical relations will tell us that $(e_1, e_2) \in [\![X]\!]$ and $(t_1, t_2) \in [\![X]\!]$, where (e_1, e_2) and (t_1, t_2) are essentially e and t under different, but related, environments. The model of $e \equiv t : X$ will then be the proposition that $(e_1, t_2) \in [\![X]\!]$.

For the relation to model symmetry, *i.e.,* that $e \equiv t$ implies $t \equiv e$ for $e, t : X$, we will need to show that $(e_1, t_2) \in [\![X]\!]$ (together with the knowledge that (e_1, e_2) and (t_1, t_2) are in $[\![X]\!]$, thanks to the fundamental property) implies $(t_1, e_2) \in [\![X]\!]$. But this is *precisely* the definition of zigzag closure!

Similarly, we can show that transitivity holds for well-typed terms. Consider the diagram:



Here, $(a_1, a_2), (b_1, b_2)$, and (c_1, c_2) can again be thought of as programs a, b, c under different (but related) environments. Given that (a_1, b_2) and (b_1, c_2) are in $[\![X]\!]$, the zigzag closure lets us derive $(a_1, c_2) \in [\![X]\!]$, as required for transitivity.

Consequently, each QPER $Q \subseteq R \times S$ may also be viewed as a PER on the set $R \times S$:

▶ **Definition 2.** (Canonically induced PER) Every QPER $Q \subseteq R \times S$ induces an equivalence relation $\sim_Q \subseteq Q \times Q$ (and hence a PER on $R \times S$), defined as $(a_1, a_2) \sim_Q (b_1, b_2)$ iff the zigzag $\{(a_1, a_2), (b_1, b_2), (a_1, b_2), (b_1, a_2)\} \subseteq Q$. (We write $\sim$ as shorthand for $\sim_Q$ if Q is evident from context.)

Hence, a QPER provides a way of telling when pairs of related terms are equivalent. In this sense, our approach reverses the usual method of building parametric models. Instead of building a PER model of types *as well as* a relational model between such PERs, we use QPERs to directly define a relational model, from which a canonical PER model can then be derived after the fact.

3.2 A Comparison with PER Models

Readers familiar with traditional parametric models may be surprised with the QPER structure: since the composition of a QPER with itself is a PER, shouldn't QPERs have merely the same expressive power as PERs? In fact, QPERs give rise to a significantly coarser notion of program equivalence than PERs.

Traditional PER models are symmetric, and therefore cannot relate different implementations of the same type (e.g., PER models cannot show that Peano and binary representations of the natural numbers are equivalent). This deficiency is then rectified by giving a second relational model. However, the asymmetry inherent in QPERs allows us to relate different implementations, *without* having to give two models — indeed, our model seems to validate all of the program equivalences provable in Plotkin-Abadi logic.

$$\textsc{Cand} \triangleq \left\{ R \in \textsc{Qper}(\text{Exp}, \text{Exp}) \;\middle|\; \begin{array}{l} \forall (e_1, e_2) \in R.\ e_1 \downarrow \wedge\ e_2 \downarrow\ \wedge \\ \forall (e_1, e_2) \in R, (e_1', e_2') \in \text{Exp}^2. \\ \qquad e_1 \leftrightarrow^* e_1' \wedge e_2 \leftrightarrow^* e_2' \implies (e_1', e_2') \in R \end{array} \right\}$$

Figure 3 Candidate Relations.

3.3 The Semantic Interpretation

With these preliminaries in place, we can move on to a description of the model.

3.3.1 Contexts

The interpretation of the Γ ok judgment is the *set of grounding environments* γ that satisfy it. We give the interpretation in Figure 4.

An environment γ is in the interpretation of the empty context iff it is the empty environment. It is in the interpretation of the context $\Gamma, x : X$ ok iff it is an element of $[\![\Gamma \text{ ok}]\!]$, together with a pair (e, e') of closed terms from the interpretation of $\Gamma \vdash X : *$. Finally, γ is in the interpretation of the context $\Gamma, \alpha : \kappa$ ok iff it is an element of $[\![\Gamma \text{ ok}]\!]$, together with a tuple $((A, A'), R)$. Here, A and A' are closed syntactic types, and R is the semantic interpretation of the type. Note that there are no well-formedness constraints on the *syntactic* types A and A': we do not need them, since the operational semantics never examines a type constructor, and the relation R carries all the necessary semantic constraints.

In Figure 4, we also define a notion of equivalence $\gamma \sim_{\Gamma \text{ ok}} \gamma'$ on environments. This relation says that the relations R and R' to which γ and γ' map the same type variable α must be equal, and that pairs of terms $(e_1, e_1')/x$ and $(e_2, e_2')/x$ must lie in the same equivalence class of the relation. The $\sim_{\Gamma \text{ ok}}$ relation is indeed a PER, but actually proving that fact can only be done after the proof of soundness of the interpretation of types and kinds. This is due to the fact that the definition is "biased"—the second line asymmetrically uses $[\![\Gamma \vdash X : *]\!]\ \gamma_1$ on the right-hand side.

In Figure 5, we give notation concerning environments that we will use in the sequel. γ contains left- and right-bindings for each of the variables in its domain; γ_1 is the left projection of the environment, and γ_2 is the right projection. We write $\gamma(e)$ to indicate the pair of terms we get from the left and right projections of γ applied to e.

3.3.2 Kinds

We give the semantics of kinds in Figure 7. We begin by giving a "pre-interpretation" function $\|\cdot\|$ (defined in Figure 6), which gives an approximate interpretation of kinds, without reference to term or type arguments. This interpretation is less precise than we want, but is a useful device to simplify the argument that our main interpretation function $[\![\cdot]\!]$ is well-defined. That main interpretation of kinds, $[\![\Gamma \vdash \kappa : \mathsf{kind}]\!]$, on the other hand, *is* relative to a context γ. The interpretation of the base kind $\Gamma \vdash * : \mathsf{kind}$, given in Figure 3, is a slight restriction of the set of quasi-PERs on terms. Namely, we restrict ourselves to quasi-PERs of *terminating* terms, closed under expansion and reduction.

The interpretation of the higher kind $\Gamma \vdash \Pi\alpha : \kappa.\ \kappa' : \mathsf{kind}$ is morally a currying of the interpretation $\Gamma, \alpha : \kappa \vdash \kappa' : \mathsf{kind}$. In particular, it is the subset of functions $\|\kappa\| \to \|\kappa'\|$, such that (1) we ignore the syntactic part of an argument triple $((X_1, X_2), R)$ (the condition in the first line of the interpretation in Figure 7), and (2) on any argument $R \in [\![\Gamma \vdash \kappa : \mathsf{kind}]\!]\ \gamma$, the result is in $[\![\Gamma, \alpha : \kappa \vdash \kappa' : \mathsf{kind}]\!]\ (\gamma, ((A_1, A_2), R)/\alpha)$ (the second and third lines of the

$$
\begin{array}{lcl}
[\![\Gamma\ \mathsf{ok}]\!] & \in & \mathcal{P}(\|\Gamma\|) \\
[\![\cdot\ \mathsf{ok}]\!] & = & \{\langle\rangle\} \\
[\![\Gamma, x : X\ \mathsf{ok}]\!] & = & \{(\gamma, (e, e')/x) \mid \gamma \in [\![\Gamma\ \mathsf{ok}]\!] \wedge (e, e') \in [\![\Gamma \vdash X : *]\!]\ \gamma\} \\
[\![\Gamma, \alpha : \kappa\ \mathsf{ok}]\!] & = & \left\{(\gamma, ((X, X'), R)/\alpha) \;\middle|\; \gamma \in [\![\Gamma\ \mathsf{ok}]\!] \wedge ((X, X'), R) \in \mathrm{Type}^2 \times [\![\Gamma \vdash \kappa : \mathsf{kind}]\!]\ \gamma\right\}
\end{array}
$$

$$
\begin{array}{lcl}
\langle\rangle \sim_{\cdot\ \mathsf{ok}} \langle\rangle & \iff & \text{always} \\
(\gamma_1, (e_1, e_1')/x) \sim_{(\Gamma, x:X\ \mathsf{ok})} (\gamma_2, (e_2, e_2')/x) & \iff & \gamma_1 \sim_{\Gamma\ \mathsf{ok}} \gamma_2 \wedge (e_1, e_1') \sim_{[\![\Gamma \vdash X:*]\!]\ \gamma_1} (e_2, e_2') \\
(\gamma_1, (\overline{A}, R_1)/\alpha) \sim_{(\Gamma, \alpha:\kappa\ \mathsf{ok})} (\gamma_2, (\overline{B}, R_2)/\alpha) & \iff & \gamma_1 \sim_{\Gamma\ \mathsf{ok}} \gamma_2 \wedge R_1 = R_2
\end{array}
$$

Figure 4 Interpretations of Contexts and Environment Equivalence.

$$
\begin{array}{lcl}
(\langle\rangle)_i & = & \langle\rangle \\
(\gamma, (e_1, e_2)/x)_i & = & \gamma_i, e_i/x \\
(\gamma, ((A_1, A_2), R)/\alpha)_i & = & \gamma_i, A_i/\alpha
\end{array}
\qquad
\begin{array}{lcl}
\gamma(e) & = & (\gamma_1(e), \gamma_2(e)) \\
\gamma(A) & = & (\gamma_1(A), \gamma_2(A)) \\
\gamma(\kappa) & = & (\gamma_1(\kappa), \gamma_2(\kappa))
\end{array}
$$

Figure 5 Notation.

interpreting clause in Figure 7). On anything outside the dependent domain, we force the result to be a fixed ("dummy") element $!_{\kappa'} \in \|\kappa'\|$ (the fourth line). While not technically necessary, this last condition simplifies the proof, by freeing us of the need to quotient by the irrelevant possible values of a kind outside the domain of the context. Similarly, elements of $\Gamma \vdash \Pi x : X.\ \kappa' : \mathsf{kind}$ are the currying of the interpretation $\Gamma, x : X \vdash \kappa' : \mathsf{kind}$, with the condition that elements return the same result for all equivalent pairs $(e_1, e_2) \sim_{\hat{X}} (e_1', e_2')$, where $\hat{X} = [\![\Gamma \vdash X : *]\!]\ \gamma$ is the relational interpretation of X.

3.3.3 Type Constructors

In Figure 8, we give the interpretation of the type constructors of our language, as a function that takes a kinding derivation and returns an element of the appropriate semantic kind. The first line of the definition says that the interpretation of $\Gamma \vdash \alpha : \kappa$ proceeds by looking up α in the environment argument γ, and returning the relation component of the triple. (Here and elsewhere, we use overbar notation to denote pairs, *e.g.,* $\overline{A}$ denotes (A, A').)

The interpretation of a lambda-abstraction $\lambda\alpha : \kappa.\ A$ is just a function that takes an argument in κ, and returns the result of interpreting A in an extended environment. Likewise, a type constructor application $A\ B$ takes the meaning of A, and passes it the syntax and semantics of B. Similarly, a term abstraction $\lambda x : X.\ A$ just returns a function which takes a pair (e, e'), and returns the interpretation of A in an extended environment, and application $A\ e$ passes $\gamma(e)$, the pair of related instantiations of e, to the interpretation of A.

When the kind conversion rule is used to replace the kind of the constructor with an equivalent one, we simply interpret the subderivation and return that as our answer. As a result, however, we need an easy-to-prove coherence property for our semantic interpretations, stating that the interpretation of A is the same at any kind it inhabits (see the appendix).

Next, we give the interpretations of types of base kind. The kinding interpretation requires that such types be interpreted as relations (specifically, QPERs) between terms. The interpretation of the function type $\Pi x : X.\ Y$ is the set of (terminating) terms that take related arguments in X to related results in Y, in the context extended by the argument pair. This is essentially the usual rule for function types in logical relations, adjusted to support dependency. The interpretation of the polymorphic type $\Pi\alpha : \kappa.\ X$ puts a pair of

$$
\begin{array}{lcl}
\|*\| & = & \mathrm{Rel}(\mathrm{Exp}, \mathrm{Exp}) \\
\|\Pi x : X.\ \kappa\| & = & \mathrm{Exp}^2 \to \|\kappa\| \\
\|\Pi \alpha : \kappa.\ \kappa'\| & = & (\mathrm{Type}^2 \times \|\kappa\|) \to \|\kappa'\|
\end{array}
\qquad
\begin{array}{lcl}
\|\cdot\| & = & \{\langle\rangle\} \\
\|\Gamma, x : X\| & = & \left\{(\gamma, \bar{e}/x) \;\middle|\; \gamma \in \|\Gamma\| \wedge \bar{e} \in \mathrm{Exp}^2\right\} \\
\|\Gamma, \alpha : \kappa\| & = & \left\{(\gamma, (\overline{A}, R)/\alpha) \;\middle|\; \begin{array}{l} \gamma \in \|\Gamma\| \ \wedge \\ \overline{A} \in \mathrm{Type}^2 \ \wedge \\ R \in \|\kappa\| \end{array}\right\}
\end{array}
$$

$$
\begin{array}{lcl}
!_{*} & = & \emptyset \\
!_{\Pi x:X.\ \kappa} & = & \lambda(e, e') \in \mathrm{Exp}^2.\ !_{\kappa} \\
!_{\Pi \alpha:\kappa.\ \kappa'} & = & \lambda(\overline{A}, R) \in \mathrm{Type}^2 \times \|\kappa\|.\ !_{\kappa'}
\end{array}
$$

Figure 6 Pre-Interpretations of Kinds and Contexts.

$$[\![\Gamma \vdash \kappa : \mathsf{kind}]\!] \in \|\Gamma\| \to \mathcal{P}(\|\kappa\|)$$

$$
\begin{array}{lcl}
[\![\Gamma \vdash * : \mathsf{kind}]\!]\ \gamma & = & \textsc{Cand} \\
[\![\Gamma \vdash \Pi \alpha : \kappa.\ \kappa' : \mathsf{kind}]\!]\ \gamma & = & \left\{ T \in \|\Pi \alpha : \kappa.\ \kappa'\| \;\middle|\; \begin{array}{l} \forall \overline{A}, \overline{B}, R \in \|\kappa\|\,.\, T(\overline{A}, R) = T(\overline{B}, R) \wedge \\ \forall \overline{A}, R \in [\![\Gamma \vdash \kappa : \mathsf{kind}]\!]\ \gamma. \\ \quad T(\overline{A}, R) \in [\![\Gamma, \alpha : \kappa \vdash \kappa' : \mathsf{kind}]\!]\ (\gamma, (\overline{A}, R)/\alpha) \\ \wedge\ \forall \overline{A}, R \notin [\![\Gamma \vdash \kappa : \mathsf{kind}]\!]\ \gamma.\ T(\overline{A}, R) = !_{\kappa'} \end{array} \right\} \\
[\![\Gamma \vdash \Pi x : X.\ \kappa : \mathsf{kind}]\!]\ \gamma & = & \mathsf{let}\ \hat{X} = [\![\Gamma \vdash X : *]\!]\ \gamma\ \mathsf{in} \\
 & & \left\{ R \in \|\Pi x : X.\ \kappa\| \;\middle|\; \begin{array}{l} \forall \bar{e}, \overline{e'} \in \hat{X}.\ \bar{e} \sim_{\hat{X}} \overline{e'} \implies R\ \bar{e} = R\ \overline{e'} \wedge \\ \forall \bar{e} \in \hat{X}.\ R\ \bar{e} \in [\![\Gamma, x : X \vdash \kappa : \mathsf{kind}]\!]\ (\gamma, \bar{e}/x) \wedge \\ \forall \bar{e} \notin \hat{X}. R\ \bar{e} = !_{\kappa} \end{array} \right\}
\end{array}
$$

Figure 7 Interpretations of Kinds.

type abstractions in the relation, if for each relation R in kind κ, their bodies are related at the term relation for X, in the environment augmented with R for α.

A pair of terms reducing to $(\mathsf{refl}, \mathsf{refl})$ inhabit the identity type $e_1 =_X e_2$ only when $\gamma_1(e_1)$ and $\gamma_2(e_2)$ are related at X (alternatively, $\gamma(e_1) \sim_{\hat{X}} \gamma(e_2)$). Since the identity type is interpreted by a relation containing at most one pair of values, we validate axiom K.

4 Soundness

Our main theorem is a proof of Reynolds' fundamental property for our language. By induction over derivations, we can show that every well-typed term is related to itself by the relational interpretation of its type. Our proof proceeds in two stages.

1. Using the pre-interpretation of contexts and kinds given in Figure 6, which are clearly well-defined, we first show basic structural properties of the main semantic interpretation—namely, that it is well-defined, that it is coherent, and that it satisfies semantic weakening and substitution properties. (For space reasons, we do not state the exact lemmas in this extended abstract, but the lemmas and their proofs can be found in the appendix.)
2. Then, we prove the fundamental property. This is a large structural induction over the syntax of kind, type, and term derivations, as well as equality derivations. We state the fundamental property below, and give the complete proof in the appendix.

4.1 The Pre-Interpretation and Structural Properties

The pre-interpretation of kinds $\|\kappa\|$ (Figure 6) interprets κ as a set, by induction on the syntax of κ, ignoring term and type indices. The pre-interpretation $\|\Gamma\|$ merely characterizes the *shape* of environments that semantically realize Γ, without placing any interesting invariants

$$[\![\Gamma \vdash A : \kappa]\!] \in \|\Gamma\| \to \|\kappa\|$$

$$\begin{array}{lcl}
[\![\Gamma, \alpha : \kappa, \Gamma' \vdash \alpha : \kappa]\!]\ \gamma & = & R \qquad \text{if } \gamma(\alpha) = (\overline{A}, R) \\
[\![\Gamma \vdash \lambda\alpha : \kappa.\ A : \Pi\alpha : \kappa.\ \kappa']\!]\ \gamma & = & \lambda(\overline{B}, R).\begin{cases} [\![\Gamma, \alpha : \kappa \vdash A : \kappa']\!]\ (\gamma, (\overline{B}, R)/\alpha) & \text{if } R \in [\![\Gamma \vdash \kappa : \mathsf{kind}]\!]\ \gamma \\ !_{\kappa'} & \text{otherwise} \end{cases} \\
[\![\Gamma \vdash A\ B : [B/\alpha]\kappa']\!]\ \gamma & = & [\![\Gamma \vdash A : \Pi\alpha : \kappa.\ \kappa']\!]\ \gamma\ (\gamma(B), [\![\Gamma \vdash B : \kappa]\!]\ \gamma) \\
[\![\Gamma \vdash \lambda x : X.\ A : \Pi x : X.\ \kappa]\!]\ \gamma & = & \lambda\bar{e}.\begin{cases} [\![\Gamma, x : X \vdash A : \kappa]\!]\ (\gamma, \bar{e}/x) & \text{if } \bar{e} \in [\![\Gamma \vdash X : *]\!]\ \gamma \\ !_{\kappa} & \text{otherwise} \end{cases} \\
[\![\Gamma \vdash A\ e : [e/x]\kappa]\!]\ \gamma & = & [\![\Gamma \vdash A : \Pi x : X.\ \kappa]\!]\ \gamma\ \gamma(e) \\
[\![\Gamma \vdash A : \kappa]\!]\ \gamma & = & [\![\Gamma \vdash A : \kappa']\!]\ \gamma \text{ (when } \Gamma \vdash \kappa \equiv \kappa' : \mathsf{kind})
\end{array}$$

$$[\![\Gamma \vdash \Pi x : X.\ Y : *]\!]\ \gamma = \left\{ (e_1, e_1') \;\middle|\; \begin{array}{l} e_1 \downarrow \wedge\ e_1' \downarrow \wedge \\ \forall (e_2, e_2') \in [\![\Gamma \vdash X : *]\!]\ \gamma. \\ \qquad (e_1\ e_2, e_1'\ e_2') \in [\![\Gamma, x : X \vdash Y : *]\!]\ (\gamma, (e_2, e_2')/x) \end{array} \right\}$$

$$[\![\Gamma \vdash \Pi\alpha : \kappa.\ X : *]\!]\ \gamma = \left\{ (e, e') \;\middle|\; \begin{array}{l} e \downarrow \wedge\ e' \downarrow \wedge \\ \forall A, A', R \in [\![\Gamma \vdash \kappa : \mathsf{kind}]\!]\ \gamma. \\ \qquad (e\ A, e'\ A') \in [\![\Gamma, \alpha : \kappa \vdash X : *]\!]\ (\gamma, ((A, A'), R)/\alpha) \end{array} \right\}$$

$$[\![\Gamma \vdash e_1 =_X e_2 : *]\!]\ \gamma = \{(e, e') \mid e \mapsto^* \mathsf{refl} \wedge e' \mapsto^* \mathsf{refl} \wedge (\gamma_1(e_1), \gamma_2(e_2)) \in [\![\Gamma \vdash X : *]\!]\ \gamma\}$$

Figure 8 Interpretations of Type Constructors.

on them. This turns out to be sufficient for "bootstrapping" purposes (*i.e.,* for defining the main semantic interpretations $[\![\cdot]\!]$), as well as for proving various properties like semantic weakening and substitution. By virtue of their being hygienic and structural, these properties hold under the assumption that the environments they quantify over merely belong to the *pre*-interpretation of contexts rather than the main interpretation. As a result, we can establish these properties independently of the fundamental theorem, and thus rely on them freely (not just inductively) when proving the fundamental theorem.

4.2 Fundamental Property

Like the structural properties, we show the fundamental theorem by an inductive case analysis of the typing derivation. Unlike the structural properties, the fundamental theorem *does* require that environments γ be semantically well-formed (*i.e.,* are elements of $[\![\Gamma\ \mathsf{ok}]\!]$). Furthermore, we will need to show that the interpretations are appropriately invariant with respect to environment equivalence ($\gamma \sim_{\Gamma\ \mathsf{ok}} \gamma'$). Since there are many different judgment forms, we have to give clauses for each judgment form.

▶ **Theorem 3** (Fundamental Property)**.**
Suppose $\Gamma\ \mathsf{ok}$*, and* $\gamma, \gamma' \in [\![\Gamma\ \mathsf{ok}]\!]$ *such that* $\gamma \sim \gamma'$*. Then:*

1. *If* $D :: \Gamma \vdash \kappa : \mathsf{kind}$*, then* $[\![D]\!]\ \gamma = [\![D]\!]\ \gamma'$*.*
2. *If* $D :: \Gamma \vdash A : \kappa$*, then* $[\![D]\!]\ \gamma = [\![D]\!]\ \gamma'$*.*
3. *If* $D :: \Gamma \vdash e : X$ *then* $\gamma(e) \sim \gamma'(e) \in [\![\Gamma \vdash X : *]\!]\ \gamma$*.*
4. *If* $D :: \Gamma \vdash A : \kappa$*, then* $[\![D]\!]\ \gamma \in [\![\Gamma \vdash \kappa : \mathsf{kind}]\!]\ \gamma$*.*
5. *If* $\Gamma \vdash \kappa \equiv \kappa' : \mathsf{kind}$*, then* $[\![\Gamma \vdash \kappa : \mathsf{kind}]\!]\ \gamma = [\![\Gamma \vdash \kappa' : \mathsf{kind}]\!]\ \gamma'$*.*
6. *If* $\Gamma \vdash A \equiv A' : \kappa$*, then* $[\![\Gamma \vdash A : \kappa]\!]\ \gamma = [\![\Gamma \vdash A' : \kappa]\!]\ \gamma'$*.*
7. *If* $\Gamma \vdash e_1 \equiv e_2 : X$*, then* $\gamma(e_1) \sim \gamma'(e_2) \in [\![\Gamma \vdash X : *]\!]\ \gamma$*.*

This theorem justifies the use of parametricity reasoning about our language, since all well-typed terms are self-related by the corresponding relational interpretations of types. As a corollary, parametricity implies consistency: since the relational interpretation of the type $\Pi\alpha : *.\ \alpha$ is empty, it must also be a syntactically uninhabited type.

5 Examples

In all of the following proofs we assume Γ is well-formed, and that environment γ inhabits $[\![\Gamma\ \mathsf{ok}]\!]$ in order to appeal to the fundamental property.

5.1 Sums and Natural Numbers

Recall the Church encodings of some the basic data types in System F — the empty type 0 as $\Pi\alpha : *.\ \alpha$, the sum type $A+B$ as $\Pi\alpha : *.\ (A \to \alpha) \to (B \to \alpha) \to \alpha$, and the natural numbers $\mathbb{N}$ as $\Pi\alpha : *.\ \alpha \to (\alpha \to \alpha) \to \alpha$. Our model validates the expected β and η properties for these types. The proofs follow on the standard lines, and so we leave the details to the appendix. It is also possible to use parametricity to internalize the induction principles. This is more convenient to state with dependent records, so we will give that example next.

5.2 Dependent Records

Cartesian products can be defined in the usual way, and there are no surprises with them. More interesting is the fact that dependent records (Σ-types) are realizable in our model.

$$\Sigma x : X.\ Y \triangleq \Pi\alpha : *.\ (\Pi x : X.\ Y \to \alpha) \to \alpha$$

with the introduction form:

$$\mathsf{pair}\ x\ y \triangleq \lambda\alpha : *.\ \lambda k : \Pi x : X.\ Y \to \alpha.\ k\ x\ y$$

However, when it comes to eliminators, this type looks like a *weak* pair type, corresponding to a type with an eliminator $\mathsf{let}\ (x, y) = p\ \mathsf{in}\ e'$, rather than projective eliminators like $\pi_1(p)$ and $\pi_2(p)$. In the absence of parametricity, this is correct, but it is a remarkable fact [12] that in a parametric model, we can realize *strong* eliminators for this type, defined as follows:

- $\mathsf{fst} : (\Sigma x : X.\ Y) \to X = \lambda p.\ p\ X\ (\lambda x.\ \lambda y.\ x)$
- $\mathsf{snd} : \Pi p : (\Sigma x : X.\ Y).\ [\mathsf{fst}\, p/x]Y = \lambda p.\ p\ (\Sigma x : X.\ Y)\ \mathsf{pair}\ ([\mathsf{fst}\, p/x]Y)\ (\lambda x.\ \lambda y.\ y)$

Note that the projective eliminator snd is *not* syntactically well-typed. Instead, we will use our parametric model to show that it has the correct *semantic* type and equations, and so it *realizes* the projective eliminator. This means it is safe to add as an axiom to our system, and that it will have good computational behavior.

▶ **Lemma 4.** *(Normal forms for eta-expanded pairs)*
If $(p, p') \in [\![\Gamma \vdash \Sigma x : X.\ Y : *]\!]\ \gamma$, *then there exist terms* u, u', t, t' *such that* $(u, u') \in [\![\Gamma \vdash X : *]\!]\ \gamma$ *and* $(t, t') \in [\![\Gamma, x : X \vdash Y : *]\!]\ (\gamma, (u, u')/x)$ *such that* $p\ \_\ \mathsf{pair} \leftrightarrow^*\ \mathsf{pair}\ u\ t$ *and* $p'\ \_\ \mathsf{pair} \leftrightarrow^*\ \mathsf{pair}\ u'\ t'$.

▶ **Lemma 5.** *(Weak eta for pairs)*
If $(p, p') \in [\![\Gamma \vdash \Sigma x : X.\ Y : *]\!]\ \gamma$, *then* $(p, p') \sim (p\ \_\ \mathsf{pair}, p'\ \_\ \mathsf{pair})$.

The proofs of these two lemmas are essentially standard, and together imply that this type correctly encodes a weak pair. We use these facts to show snd has the correct semantic type:

▶ **Lemma 6.** *(Semantic well-typedness for* snd*)*
We have that $(\mathsf{snd}, \mathsf{snd}) \in [\![\Gamma \vdash \Pi q : (\Sigma x : X.\ Y).\ [\mathsf{fst}\, q/x]Y : *]\!]\ \gamma$.

This proof is direct, but it relies critically on the context and environment being well-formed, since it appeals to the fundamental property in several places.

▶ **Corollary 7.** *(Projective eta for* Σ*-types)*
If $(p, p') \in [\![\Gamma \vdash \Sigma x : X.\ Y : *]\!]\ \gamma$, *then* $(p, p') \sim (\mathsf{pair}\ (\mathsf{fst}\, p)\ (\mathsf{snd}\, p), \mathsf{pair}\ (\mathsf{fst}\, p')\ (\mathsf{snd}\, p'))$.

This follows easily with the previous three lemmas in hand.

5.3 Induction for the Natural Numbers

Though Church numerals are directly definable with polymorphism, their induction principle is not. That is, there is no syntactically typable term

$$ind : \Pi P : \mathbb{N} \to *.\ P(z) \to (\Pi n : \mathbb{N}.\ P(n) \to P(s\ n)) \to \Pi n : \mathbb{N}.\ P(n)$$

However, we can show that this type is realizable within our model. That is, we can show that the term

$$\begin{array}{ll} \lambda P, i, f, n. & \mathsf{let}\ o = \mathsf{pair}\ z\ i\ \mathsf{in} \\ & \mathsf{let}\ h = \lambda p.\ \mathsf{pair}\ (s\ (\mathsf{fst}\, p))\ (f\ (\mathsf{fst}\, p)\ (\mathsf{snd}\, p))\ \mathsf{in} \\ & \mathsf{snd}\, (n\ (\Sigma x : \mathbb{N}.\ P(x))\ o\ h) \end{array}$$

is related to itself at the type above. By using a dependent pair, we we can package up the two arguments of $\Pi n : \mathbb{N}.\ P(n) \to P(s\ n)$ into a single argument, which is what the step function in the Church encoding expects. We then use parametricity to prove that for all n, applying the iterator $n\ (\Sigma x : \mathbb{N}.\ P(x))\ o\ h$ gives us a record whose first component is n, and so whose second component must be of type $P(n)$. Details are given in the appendix.

5.4 Existential Types

Our model supports the standard encoding of existential types:

$$\begin{array}{lcl} \exists \alpha : \kappa.\ X(\alpha) & \triangleq & \Pi \beta : *.\ (\Pi \alpha : \kappa.\ X(\alpha) \to \beta) \to \beta \\ \mathsf{pack} & : & \Pi \alpha : \kappa.\ X(\alpha) \to \exists \alpha : \kappa.\ X(\alpha) \\ \mathsf{pack} & = & \lambda \alpha : \kappa.\ \lambda x.\ (\lambda \beta : *.\ \lambda k.\ k\ \alpha\ x) \end{array}$$

We can easily validate the expected β and η laws for existentials, as well as the representation independence principle, which allows existential packages with different but related implementations to be proven equivalent. Again, the proofs are standard (e.g., see [4]), and we leave them for the appendix. More interestingly, and perhaps surprisingly, we can show the soundness of an existential equality principle similar to the one from Plotkin-Abadi logic (left-to-right direction of Theorem 7 of [22]):

▶ Proposition 8 (Existential equality). If $(e, e') \in [\![\Gamma \vdash \exists \alpha : \kappa.\ X : *]\!]\ \gamma$, then there exist

1. $A, A' \in \mathrm{Type}$,
2. $R \in [\![\Gamma \vdash \kappa : \mathsf{kind}]\!]\ \gamma$
3. $(t, t') \in [\![\Gamma, \alpha : \kappa \vdash X : *]\!]\ (\gamma, ((A, A'), R)/\alpha)$

such that $(e, e') \sim_{[\![\Gamma \vdash \exists \alpha : \kappa.\ X : *]\!]\ \gamma} (\mathsf{pack}\ A\ t, \mathsf{pack}\ A'\ t')$.

This says that any two terms (e, e') related at existential type must be equivalent to *some* packages $(\mathsf{pack}\ A\ t, \mathsf{pack}\ A'\ t')$ that are related by a representation independence argument.

Proof. First consider the pair (e, e'), and the application $(e\ \_,\ e'\ \_)$. Starting from $(e, e') \in [\![\Gamma \vdash \exists \alpha : \kappa.\ X : *]\!]\ \gamma$, we instantiate the type abstraction on both sides and choose the relational interpretation of the abstract type to be the following, defined by a QPER join:

$$S \triangleq \bigsqcup_{R \in [\![\kappa]\!]\ \gamma} \left\{ (\mathsf{pack}\ A\ e, \mathsf{pack}\ A'\ e') \;\middle|\; \begin{array}{l} (A, A') \in \mathrm{Type}^2 \wedge \\ (e, e') \in [\![\Gamma, \alpha : \kappa \vdash X : *]\!]\ (\gamma, ((A, A'), R)/\alpha) \end{array} \right\}^{\dagger}$$

We write $T^{\dagger}$ to indicate the reduction and expansion-closure of a relation on terms T. Next, we verify that $(\mathsf{pack}, \mathsf{pack}) \in [\![\Gamma, \beta : * \vdash \Pi \alpha : \kappa.\ X \to \beta : *]\!]\ (\gamma, (\_, S)/\beta)$. From this we get:

$$(e\ \_\ \mathsf{pack},\ e'\ \_\ \mathsf{pack}) \in S$$

Note that these terms are eta-expansions of (e, e'), which must therefore be equivalent to $(\mathsf{pack}\ \_\ s, \mathsf{pack}\ \_\ s')$ for some s and s'.

Ideally, we would like to use the fact that $(\mathsf{pack}\ \_\ s, \mathsf{pack}\ \_\ s') \in S$ to conclude there is a QPER R such that $(s, s') \in [\![\Gamma, \alpha : \kappa \vdash X : *]\!]\ (\gamma, (\_, R)/\alpha)$. However, the QPER-join adds elements that are not in the union, so this does not immediately follow. Instead, we do induction on the join to show that there is some $(\mathsf{pack}\ \_\ t, \mathsf{pack}\ \_\ t') \sim (\mathsf{pack}\ \_\ s, \mathsf{pack}\ \_\ s')$ such that $(t, t') \in [\![\Gamma, \alpha : \kappa \vdash X : *]\!]\ (\gamma, (\_, R)/\alpha)$. Then, by the eta-rule for existentials, and transitivity of $\sim_{[\![\Gamma \vdash \exists \alpha : \kappa.\ X : *]\!]\ \gamma}$, we can conclude that $(e, e') \sim (\mathsf{pack}\ \_\ t, \mathsf{pack}\ \_\ t')$. ◀

That is, the two terms that witness the relation are not necessarily the *exact* terms that e and e' evaluate to, but rather are equivalent to them. (The same caveat holds for the existential equality principle in Plotkin-Abadi logic.) Because of this issue, we cannot give a direct realizer internalizing this reasoning principle in our type theory, as we need to extend the type theory with a form of proof-irrelevance first: knowing two existential packages are equal does not tell us which relation witnesses that equality! This we leave for future work.

However, we can still of course add equality axioms for particular instances of representation independence. For example, consider the following two existential packages:

$$\begin{array}{lcl} X & \triangleq & \exists \alpha : *.\ \alpha \times (\alpha \to \alpha) \times (\alpha \to \mathsf{bool}) \\ M : X & = & \mathsf{pack}\, \mathbb{N}\ (z, s, (\lambda n.\ n\ \mathsf{bool}\ \mathsf{true}\ (\lambda k.\ \mathsf{false}))) \\ O : X & = & \mathsf{pack}\, \mathsf{bool}\ (\mathsf{true}, (\lambda b.\ \mathsf{false}), (\lambda b.\ b)) \end{array}$$

This package exports a seed value, an operation on it, and a test that says whether the argument is the seed or not. Since M and O behave the same, we can relate them at existential type, and add $\mathsf{refl} : M =_X N$ as an axiom to our system.

5.5 Quotient Types

While not an application of parametricity in the sense of theorems for free [28], we can also show the realizability of quotient types [15] in our semantics. Quotient types, give a way to define new types by taking an existing type, and quotienting it by an equivalence relation.

To do this, we first define the auxiliary predicate Eq_X, which formalizes the notion of an equivalence relation. They are relations $R : X \to X \to *$, satisfying:

$$\begin{array}{lcl} \mathrm{Eq}_X(R) & \triangleq & \Pi x : X.\ R\ x\ x \times \\ & & \Pi x : X, y : X.\ R\ x\ y \leftrightarrow R\ y\ x \times \\ & & \Pi x : X, y : X, z : X.\ R\ x\ y \to R\ y\ z \to R\ x\ z \end{array}$$

Next, we can show the realizability of the following datatype:

$$\begin{array}{lcl} X/R & \triangleq & \exists \beta : *, \\ & & \Sigma \mathit{inj} : X \to \beta. \\ & & \Sigma \mathit{app} : \Pi \gamma : *.\ \Pi f : X \to \gamma. \\ & & \qquad (\Pi a : X, a' : X.\ R\ a\ a' \to f\ a =_\gamma f\ a') \to (\beta \to \gamma). \\ & & \Pi a : X, a' : X.\ R\ a\ a' \to \mathit{inj}(a) =_\beta \mathit{inj}(a') \times \\ & & \Pi \gamma.\ \Pi f, pf, x.\ \mathit{app}\ \gamma\ f\ pf\ (\mathit{inj}\ x) =_\gamma f\ x \end{array}$$

What we are doing is defining an existential type, such that if X is a type and R is an equivalence relation on it, we return a new type β and two operations *inj* and *app*.

The *inj* is the injection into the quotient type. It takes an X, and returns a β, with the property that if a and a' are related by R, then $\mathit{inj}\ a = \mathit{inj}\ a'$. The *app* function then lifts any function f from $X \to \gamma$ into one on $\beta \to \gamma$, provided that f respects the equivalence

relation R. The last two lines give the equational theory of the quotient type. First, if a and a' are related by R, then $inj\ a = inj\ a'$. Second, if we lift a function f to operate on quotients, and we pass it the argument $inj\ x$, then the application of the lifted function should equal $f\ x$.

Proof. (Sketch) The proof of the soundness of the axiom proceeds quite directly. First, we define the following relation:

$$S = \{(e_1, e_2') \mid \exists e_1', e_2, \bar{q}.\ (e_1, e_1') \in [\![X]\!] \ \wedge\ (e_2, e_2') \in [\![X]\!] \ \wedge\ \bar{q} \in [\![R]\!]\ (e_1, e_1')\ (e_2, e_2')\}$$

As an abuse of notation, we suppress most of the context and environment arguments from the definition. By making use of the fact that we have a proof of $\mathrm{Eq}_X(R)$, we can show that S is a QPER, and then use it as our witness for showing the existential type is inhabited. We can define *inj* and *app* as:

- $\mathsf{inj} = \lambda x : X.\ x$
- $\mathsf{app} = \lambda \gamma : *.\ \lambda f : X \to \gamma, pf : \ldots, x : X.\ f\ x$

and give two dummy realizers for the proofs:

- $\mathsf{equiv} = \lambda a, a', r.\ \mathsf{refl}$
- $\mathsf{appok} = \lambda \gamma.\ \lambda f, pf, x.\ \mathsf{refl}$

With these, we can then show the semantic well-typedness of the term

$$\mathsf{pack}\ X\ \mathsf{pair\ inj}\ (\mathsf{pair\ app}\ (\mathsf{pair\ equiv\ appok}))$$

by showing it is related to itself at the witness relation S. Note that this term is not well-typed in the syntactic system, but that it does inhabit the appropriate semantic type. In terms of the operational semantics of the underlying realizers, quotienting is a no-op: no representation changes are needed to protect the quotient type's invariant: data abstraction is enough. ◀

5.6 A Note on the Constructivity of the Axioms

In this section, we have frequently introduced axioms into the type theory, justified by reasoning about the model construction. Readers may worry that these axioms may ruin the computational properties of the language, by blocking reduction. Fortunately, all of the axioms we introduce *have computational content*. Since our model is a realizability-style construction, we show that an axiom is sound by giving untyped terms inhabiting the semantic interpretation of that axiom's type. As a result, all the axioms we add are constructive, since they necessarily have realizers equipping them with computational content.

6 Discussion and Related Work

6.1 Quasi-PERs

The earliest use of quasi-PERs to to model data abstraction we have found is by Tennent and Takeyama [26], who were studying program equivalences in the context of data refinement. In this e-mail, they sketched a logical relations model of the simply-typed lambda calculus in terms of quasi-PERs (which they called "zigzag-complete relations"). The term "quasi-PER" was coined by Hofmann [16], who gave a logical relations model of a simply-typed functional language, augmented with first-order state (*i.e.,* references to integers). Though this language had no polymorphism, it did have effect annotations, and so Hofmann needed

to prove representation independence to show the equivalence of programs with possibly-differing sets of effects (*i.e.,* "effect masking" [19]). Our work greatly extends the reach of quasi-PERs to support polymorphism, higher kinds, and type dependency, showing that the simple idea of quasi-PERs scales up even to a wide array of type-theoretic features.

Hutton and Voermans [18] studied a relational version of Squiggol [9], in which the category of sets and relations was replaced with the category of PERs and saturated relations In this setting, they observed that the functional relations were precisely the difunctionals. Many useful properties of quasi-PERs are worked out in this paper, even though their application of them is rather different from our own.

6.2 Semantic Models for Parametricity

The standard approach to building parametric models is to begin with a non-parametric model of the language, and then give a second interpretation of types as relations over the non-parametric semantic types. This approach is used in Bainbridge *et al.* [5], and an abstract characterization of it was given by Rosolini [24] using categories of *reflexive graphs*, which both Dunphy and Reddy [13] and Birkedal *et al.* [10] have developed further.

Our model does not immediately fit into this framework, since we give a relational semantics directly, *without* first building a non-parametric model. This represents a considerable simplification of the metatheory: we only need one semantics, rather than two.

Both Vytiniotis and Weirich [27] and Atkey [4] give parametric models of F_ω, which is equivalent to the calculus of constructions minus dependency. The work in [27] also gives a term model, and as such they need to prove a coherence theorem for the interpretation of kinds. In contrast, Atkey [4] does not need to prove such a coherence theorem, since he defines his relation over extensional semantic objects. He does, however, need to prove the identity extension lemma, to connect his base semantics with his relational semantics.

The way we set things up means our proofs are overall a bit easier than either of these two approaches (modulo the additional overhead imposed by type dependency). In [27], great care is taken to ensure that their relations only mention well-formed type constructors. We do not bother maintaining this invariant: since the operational semantics never examines a type argument, there is no need for the model to worry if a type argument is well formed or not. By moving to a realizability-based view, we also make it possible to add *realizable axioms*: we can add any axioms we want, as long as those axioms have (possibly syntactically ill-typed) lambda terms as realizers for their computational behavior.

Even stating the identity extension property for higher kinds requires equipping each kind with a distinguished notion of identity relation, to connect the base and relational interpretations. We do not need to do this, since we only have a relational semantics. We still need to ensure that our interpretation of higher kinds respects the equalities (quasi-PER) on types, but found this easier to work with than identity extension, since we only need to consider the base kind.

Concurrently with our work, Atkey, Ghani and Johann [3] have extended the reflexive graph model of parametricity to a model of dependent types using the families fibration over Set. This approach naturally handles parametricity properties for indexed data types, something we have not yet looked at in our model. A natural next step would be to redo their construction using relations over terms rather than sets, which would permit a direct comparison of quasi-PERs with the standard model of parametricity.

6.3 Internalizations of Parametricity

In recent work, Bernardy *et al.* [6] demonstrate how to generalize Reynolds' relational interpretation to systems of dependent types, and show that for sufficiently rich type theories, the image of the relational interpretation lies within the original type theory. This gives a syntax-directed embedding of a parametric interpretation of type theory into itself.

The translational approach is wholly syntactic, as opposed to our more semantic approach. One benefit of their method is that it yields concrete proof terms for parametricity properties. The two principal limitations of the translational approach are that (1) parametricity properties only apply to closed terms, and (2) there is no way to use parametricity to internalize program equivalences as equalities.

In [7], Bernardy and Moulin relax the first restriction by extending the syntax of type theory with *operators* to represent appeals to parametricity. Though the syntactic modifications they make to type theory are quite complex, the fundamental idea is quite simple: they are using indices to indicate the "color" of different subterms [8], and parametricity lets them show that different colors do not interfere with one another. However, the second restriction remains, and intentionally so. They support inductive definitions in the style of Coq and Agda, which permits internalizing the conversion relation by defining the Martin-Löf identity type as an inductive type. This inherently limits what equality can contain: the Church booleans cannot be shown to be equal to true or false, unless the conversion relation contains it. This approach might be described as "Strachey-style" [25], where the *uniformity* of parametric computations allows deriving powerful and elegant erasure properties.

The strengths and weaknesses of our approach are reversed. We do not give full proof terms, due to our use of equality reflection, but we can support parametricity arguments for open terms, and can internalize parametric program equivalences as equalities. Our approach can be viewed as "Reynolds-style" parametricity, where the emphasis is on the *relational* character of parametricity, leading to a focus on representation independence and eta-laws. A natural question is whether it is possible to combine the strengths of these two approaches, and gain the decidability advantages of their approach while retaining the simple interface to parametricity we can support.

References

1 S.F. Allen, M. Bickford, R.L. Constable, R. Eaton, C. Kreitz, L. Lorigo, and E. Moran. Innovations in computational type theory using Nuprl. *Journal of Applied Logic*, 4(4), 2006.

2 Thorsten Altenkirch. *Constructions, Inductive Types and Strong Normalization.* PhD thesis, University of Edinburgh, November 1993.

3 Bob Atkey, Neil Ghani, and Patricia Johann. A relationally parametric model of dependent type theory. Unpublished draft, 2013.

4 Robert Atkey. Relational parametricity for higher kinds. In *CSL*, 2012.

5 E. S. Bainbridge, Peter J. Freyd, Andre Scedrov, and Philip J. Scott. Functorial polymorphism. *Theor. Comput. Sci.*, 70(1), 1990.

6 Jean-Philippe Bernardy, Patrik Jansson, and Ross Paterson. Parametricity and dependent types. In *ICFP*, 2010.

7 Jean-Philippe Bernardy and Guilhem Moulin. A computational interpretation of parametricity. In *LICS*, 2012.

8 Jean-Philippe Bernardy and Guilhem Moulin. Type-theory in color. In *ICFP*, 2013.

9 Richard Bird and Oege de Moor. *Algebra of Programming.* International Series in Computing Science, Vol. 100. Prentice Hall, 1997.

10 Lars Birkedal, Rasmus Ejlers Møgelberg, and Rasmus Lerchedahl Petersen. Domain-theoretical models of parametric polymorphism. *Theor. Comput. Sci.*, 388(1-3), 2007.

11 Thierry Coquand and Gerard Huet. The calculus of constructions. *Inf. Comput.*, 76(2-3), February 1988.

12 Dan Doel. Proving induction principles via free theorems from parametricity. Agda code at `http://code.haskell.org/~dolio/agda-share/html/ParamInduction.html`, 4 April 2012.

13 Brian P. Dunphy and Uday S. Reddy. Parametric limits. In *LICS*, 2004.

14 Robert Harper. Constructing type systems over an operational semantics. *Journal of Symbolic Computation*, 14(1):71–84, 1992.

15 Martin Hofmann. A simple model for quotient types. In *TLCA*, 1995.

16 Martin Hofmann. Correctness of effect-based program transformations. In O. Grumberg, T. Nipkow, and C. Pfaller, editors, *Formal Logical Methods for System Security and Correctness*, volume 14. IOS Press, 2008.

17 Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *Twenty-five Years of Constructive Type Theory*. Oxford University Press, 1998.

18 Graham Hutton and Ed Voermans. Making Functionality More General. In *Glasgow Workshop on Functional Programming*, Skye, Scotland, 1992.

19 J. M. Lucassen and D. K. Gifford. Polymorphic effect systems. In *POPL*, 1988.

20 Per Martin-Löf. *Intuitionistic type theory*. Bibliopolis Naples, Italy, 1984.

21 James McKinna. Why dependent types matter. In *POPL*, 2006.

22 Gordon D. Plotkin and Martín Abadi. A logic for parametric polymorphism. In *TLCA*, 1993.

23 John C. Reynolds. Types, abstraction and parametric polymorphism. In *IFIP Congress*, 1983.

24 E. P. Robinson and Giuseppe Rosolini. Reflexive graphs and parametric polymorphism. In *LICS*, 1994.

25 C. Strachey. Fundamental concepts in programming languages. *HOSC*, 13(1), 2000 (original lectures in 1967).

26 Robert Tennent and Makoto Takeyama. What is a data refinement relation? E-mail to `data-refinement@etl.gp.jp`, January 1996.

27 Dimitrios Vytiniotis and Stephanie Weirich. Parametricity, type equality, and higher-order polymorphism. *J. Functional Programming*, 20(2), March 2010.

28 Philip Wadler. Theorems for free! In *FPCA*, 1989.

A Appendix

A.1 Typing Rules

In this section, we give the full typing rules for the language we consider. In Figures 9 and 10, we give the well-formedness conditions for contexts, kinds, types, and terms. In Figures 11 and 12, we give the equality rules for kinds, types, and terms.

A.2 Metatheory

In this section, we give the full statements of the omitted lemmas needed to prove the fundamental theorem (Section 4.2). Their full proofs are given in the online extended appendix. These lemmas build up to proving the well-definedness, coherence, weakening, and substitution properties of context, kind, and type interpretations, starting with their pre-interpretations.

Since the pre-interpretation is defined solely on syntax, we can prove the following two lemmas about it:

▶ **Lemma 9** (Kind Pre-interpretations Ignore Term Substitutions).
For all kinds κ and terms e, $\|\kappa\| = \|[e/x]\kappa\|$.

▶ **Lemma 10** (Kind Pre-interpretations Ignore Type Substitutions).
For all kinds κ and types A, $\|\kappa\| = \|[A/\alpha]\kappa\|$.

This implies the following trivial coherence property.

▶ **Theorem 11** (Kind Coherence). *If $\Gamma \vdash \kappa \equiv \kappa' : \mathsf{kind}$, then $\|\kappa\| = \|\kappa'\|$.*

Once we have this property in place, we can prove the following well-definedness conditions for the interpretations of the context, kind, and type judgements.

▶ **Theorem 12** (Well-Definedness).
1. *If $D :: \Gamma\ \mathsf{ok}$, then $[\![D]\!] \in \mathcal{P}(\|\Gamma\|)$.*
2. *If $D :: \Gamma \vdash \kappa : \mathsf{kind}$, then $[\![D]\!] \in \|\Gamma\| \to \mathcal{P}(\|\kappa\|)$.*
3. *If $D :: \Gamma \vdash A : \kappa$, then $[\![D]\!] \in \|\Gamma\| \to \|\kappa\|$.*

Now that we know that we have a well-formed definition, we can prove coherence property for the kind and type interpretations.

▶ **Theorem 13** (Coherence for Kind and Type Interpretations).
1. *If $D :: \Gamma \vdash \kappa : \mathsf{kind}$ and $D' :: \Gamma \vdash \kappa : \mathsf{kind}$ and $\gamma \in \|\Gamma\|$, then $[\![D]\!]\ \gamma = [\![D']\!]\ \gamma$.*
2. *If $D :: \Gamma \vdash A : \kappa$ and $D' :: \Gamma \vdash A : \kappa'$ and $\gamma \in \|\Gamma\|$, then $[\![D]\!]\ \gamma = [\![D']\!]\ \gamma$.*

This immediately implies the following corollary:

▶ **Corollary 14** (Coherence for Context Interpretation).
If $D :: \Gamma\ \mathsf{ok}$ and $D' :: \Gamma\ \mathsf{ok}$, then $[\![D :: \Gamma\ \mathsf{ok}]\!] = [\![D' :: \Gamma\ \mathsf{ok}]\!]$.

Now we can prove weakening and substitution.

▶ **Theorem 15** (Weakening of Kinding and Typing).
1. *If $D :: \Gamma_0, \Gamma_2 \vdash \kappa : \mathsf{kind}$ then there exists $D' :: \Gamma_0, \Gamma_1, \Gamma_2 \vdash \kappa : \mathsf{kind}$ such that for all $(\gamma_0, \gamma_1, \gamma_2) \in \|\Gamma_0, \Gamma_1, \Gamma_2\|$, we have $[\![D]\!]\ (\gamma_0, \gamma_2) = [\![D']\!]\ (\gamma_0, \gamma_1, \gamma_2)$.*
2. *If $D :: \Gamma_0, \Gamma_2 \vdash A : \kappa$ then there exists $D' :: \Gamma_0, \Gamma_1, \Gamma_2 \vdash A : \kappa$ such that for all $(\gamma_0, \gamma_1, \gamma_2) \in \|\Gamma_0, \Gamma_1, \Gamma_2\|$, we have $[\![D]\!]\ (\gamma_0, \gamma_2) = [\![D']\!]\ (\gamma_0, \gamma_1, \gamma_2)$.*

▶ **Theorem 16** (Substitution in Pre-Contexts).
1. *If $\Gamma \vdash e : X$, and $(\gamma, \gamma(e)/x, \gamma') \in \|\Gamma, x : X, \Gamma'\|$, then $(\gamma, \gamma') \in \|\Gamma, [e/x]\Gamma'\|$.*
2. *If $\Gamma \vdash A : \kappa$, and $(\gamma, (\gamma(A), R)/\alpha, \gamma') \in \|\Gamma, \alpha : \kappa, \Gamma'\|$, then $(\gamma, \gamma') \in \|\Gamma, [A/\alpha]\Gamma'\|$.*

▶ **Theorem 17** (Substitution of Terms).
Suppose that $\Gamma \vdash e : X$ and $(\gamma, \gamma(e)/x, \gamma') \in \|\Gamma, x : X, \Gamma'\|$. Then:
1. *For all $D :: \Gamma, x : X, \Gamma' \vdash \kappa_0 : \mathsf{kind}$, there exists $D' :: \Gamma, [e/x]\Gamma' \vdash [e/x]\kappa_0 : \mathsf{kind}$ such that $[\![D]\!]\ (\gamma, \gamma(e)/x, \gamma') = [\![D']\!]\ (\gamma, \gamma')$.*
2. *For all $D :: \Gamma, x : X, \Gamma' \vdash C : \kappa_0$, there exists $D' :: \Gamma, [e/x]\Gamma' \vdash [e/x]C : [e/x]\kappa_0$ such that $[\![D]\!]\ (\gamma, \gamma(e)/x, \gamma') = [\![D']\!]\ (\gamma, \gamma')$.*

▶ **Theorem 18** (Substitution of Types).
Suppose that $D_1 :: \Gamma \vdash A : \kappa$ and $(\gamma, (\gamma(A), [\![D_1]\!]\ \gamma)/\alpha, \gamma') \in \|\Gamma, \alpha : \kappa, \Gamma'\|$. Then:
1. *For all $D :: \Gamma, \alpha : \kappa, \Gamma' \vdash \kappa_0 : \mathsf{kind}$, there exists $D' :: \Gamma, [A/\alpha]\Gamma' \vdash [A/\alpha]\kappa_0 : \mathsf{kind}$ such that $[\![D]\!]\ (\gamma, (\gamma(A), [\![D_1]\!]\ \gamma)/\alpha, \gamma') = [\![D']\!]\ (\gamma, \gamma')$.*
2. *For all $D :: \Gamma, \alpha : \kappa, \Gamma' \vdash C : \kappa_0$, there exists $D' :: \Gamma, [A/\alpha]\Gamma' \vdash [A/\alpha]C : [A/\alpha]\kappa_0$ such that $[\![D]\!]\ (\gamma, (\gamma(A), [\![D_1]\!]\ \gamma)/\alpha, \gamma') = [\![D']\!]\ (\gamma, \gamma')$.*

$$\boxed{\Gamma \text{ ok}}$$

$$\frac{}{\cdot \text{ ok}} \qquad \frac{\Gamma \text{ ok} \qquad \Gamma \vdash X : *}{\Gamma, x : X \text{ ok}} \qquad \frac{\Gamma \text{ ok} \qquad \Gamma \vdash \kappa : \mathsf{kind}}{\Gamma, \alpha : \kappa \text{ ok}}$$

$$\boxed{\Gamma \vdash \kappa : \mathsf{kind}}$$

$$\frac{}{\Gamma \vdash * : \mathsf{kind}} \qquad \frac{\Gamma \vdash X : * \qquad \Gamma, x : X \vdash \kappa : \mathsf{kind}}{\Gamma \vdash \Pi x : X.\ \kappa : \mathsf{kind}} \qquad \frac{\Gamma \vdash \kappa : \mathsf{kind} \qquad \Gamma, \alpha : \kappa \vdash \kappa' : \mathsf{kind}}{\Gamma \vdash \Pi \alpha : \kappa.\ \kappa' : \mathsf{kind}}$$

Figure 9 Context and Kind Well-formedness.

$$\boxed{\Gamma \vdash A : \kappa}$$

$$\frac{\Gamma \vdash \kappa : \mathsf{kind} \qquad \Gamma, \alpha : \kappa \vdash Y : *}{\Gamma \vdash \Pi \alpha : \kappa.\ Y : *}$$

$$\frac{\Gamma \vdash X : * \qquad \Gamma, x : X \vdash Y : *}{\Gamma \vdash \Pi x : X.\ Y : *} \qquad \frac{\Gamma \vdash e : X \qquad \Gamma \vdash e' : X}{\Gamma \vdash e =_X e' : *}$$

$$\frac{\alpha : \kappa \in \Gamma}{\Gamma \vdash \alpha : \kappa} \qquad \frac{\Gamma \vdash X : * \qquad \Gamma, x : X \vdash A : \kappa}{\Gamma \vdash \lambda x : X.\ A : \Pi x : X.\ \kappa} \qquad \frac{\Gamma \vdash \kappa : \mathsf{kind} \qquad \Gamma, \alpha : \kappa \vdash A : \kappa'}{\Gamma \vdash \lambda \alpha : \kappa.\ A : \Pi \alpha : \kappa.\ \kappa'}$$

$$\frac{\Gamma \vdash A : \Pi x : X.\ \kappa \qquad \Gamma \vdash e : X}{\Gamma \vdash A\ e : [e/x]\kappa} \qquad \frac{\Gamma \vdash A : \Pi \alpha : \kappa.\ \kappa' \qquad \Gamma \vdash A' : \kappa}{\Gamma \vdash A\ A' : [A'/\alpha]\kappa'}$$

$$\frac{\Gamma \vdash A : \kappa' \qquad \Gamma \vdash \kappa \equiv \kappa' : \mathsf{kind}}{\Gamma \vdash A : \kappa}$$

$$\boxed{\Gamma \vdash e : X}$$

$$\frac{x : X \in \Gamma}{\Gamma \vdash x : X} \qquad \frac{\Gamma \vdash e : Y \qquad \Gamma \vdash X \equiv Y : *}{\Gamma \vdash e : X}$$

$$\frac{\Gamma \vdash \kappa : \mathsf{kind} \qquad \Gamma, \alpha : \kappa \vdash e : Y}{\Gamma \vdash \lambda \alpha : \kappa.\ e : \Pi \alpha : \kappa.\ Y} \qquad \frac{\Gamma \vdash e : \Pi \alpha : \kappa.\ Y \qquad \Gamma \vdash A : \kappa}{\Gamma \vdash e\ A : [A/\alpha]Y}$$

$$\frac{\Gamma, x : X \vdash e : Y}{\Gamma \vdash \lambda x : X.\ e : \Pi x : X.\ Y} \qquad \frac{\Gamma \vdash e : \Pi x : X.\ Y \qquad \Gamma \vdash e' : X}{\Gamma \vdash e\ e' : [e'/x]Y}$$

$$\frac{\Gamma \vdash e_1 \equiv e_2 : X}{\Gamma \vdash \mathsf{refl} : e_1 =_X e_2}$$

Figure 10 Type and Term Well-formedness.

$$\boxed{\Gamma \vdash \kappa \equiv \kappa' : \mathsf{kind}}$$

$$\frac{\Gamma \vdash e \equiv e' : X \qquad \Gamma, x : X \vdash \kappa : \mathsf{kind}}{\Gamma \vdash [e/x]\kappa \equiv [e'/x]\kappa : \mathsf{kind}} \qquad \frac{\Gamma \vdash A \equiv A' : \kappa \qquad \Gamma, \alpha : \kappa \vdash \kappa' : \mathsf{kind}}{\Gamma \vdash [A/\alpha]\kappa' \equiv [A'/\alpha]\kappa' : \mathsf{kind}}$$

$$\frac{\Gamma \vdash \kappa_1 \equiv \kappa_1' : \mathsf{kind} \qquad \Gamma, \alpha : \kappa_1 \vdash \kappa_2 \equiv \kappa_2' : \mathsf{kind}}{\Gamma \vdash \Pi\alpha : \kappa_1.\ \kappa_2 \equiv \Pi\alpha : \kappa_1'.\ \kappa_2' : \mathsf{kind}}$$

$$\frac{\Gamma \vdash X \equiv X' : * \qquad \Gamma, x : X \vdash \kappa \equiv \kappa' : \mathsf{kind}}{\Gamma \vdash \Pi x : X.\ \kappa \equiv \Pi x : X'.\ \kappa' : \mathsf{kind}}$$

$$\frac{\Gamma \vdash \kappa : \mathsf{kind}}{\Gamma \vdash \kappa \equiv \kappa : \mathsf{kind}} \qquad \frac{\Gamma \vdash \kappa \equiv \kappa' : \mathsf{kind}}{\Gamma \vdash \kappa' \equiv \kappa : \mathsf{kind}} \qquad \frac{\Gamma \vdash \kappa_1 \equiv \kappa_2 : \mathsf{kind} \qquad \Gamma \vdash \kappa_2 \equiv \kappa_3 : \mathsf{kind}}{\Gamma \vdash \kappa_1 \equiv \kappa_3 : \mathsf{kind}}$$

$$\boxed{\Gamma \vdash A \equiv A' : \kappa}$$

$$\frac{\Gamma \vdash e \equiv e' : X \qquad \Gamma, x : X \vdash A : \kappa}{\Gamma \vdash [e/x]A \equiv [e'/x]A : [e/x]\kappa} \qquad \frac{\Gamma \vdash A \equiv A' : \kappa \qquad \Gamma, \alpha : \kappa \vdash B : \kappa'}{\Gamma \vdash [A/\alpha]B \equiv [A'/\alpha]B : [A/\alpha]\kappa'}$$

$$\frac{\Gamma \vdash A \equiv A' : \kappa' \qquad \Gamma \vdash \kappa \equiv \kappa' : \mathsf{kind}}{\Gamma \vdash A \equiv A' : \kappa}$$

$$\frac{\Gamma \vdash A : \kappa}{\Gamma \vdash A \equiv A : \kappa} \qquad \frac{\Gamma \vdash A \equiv A' : \kappa}{\Gamma \vdash A' \equiv A : \kappa} \qquad \frac{\Gamma \vdash A_1 \equiv A_2 : \kappa \qquad \Gamma \vdash A_2 \equiv A_3 : \kappa}{\Gamma \vdash A_1 \equiv A_3 : \kappa}$$

$$\frac{\Gamma \vdash \kappa \equiv \kappa' : \mathsf{kind} \qquad \Gamma, \alpha : \kappa \vdash X \equiv X' : *}{\Gamma \vdash \Pi\alpha : \kappa.\ X \equiv \Pi\alpha : \kappa'.\ X' : *} \qquad \frac{\Gamma \vdash X \equiv X' : \mathsf{kind} \qquad \Gamma, x : X \vdash Y \equiv Y' : *}{\Gamma \vdash \Pi x : X.\ Y \equiv \Pi x : X'.\ Y' : *}$$

$$\frac{\Gamma \vdash \kappa \equiv \kappa' : \mathsf{kind} \qquad \Gamma, \alpha : \kappa \vdash B \equiv B' : \kappa''}{\Gamma \vdash \lambda\alpha : \kappa.\ B \equiv \lambda\alpha : \kappa'.\ B' : \Pi\alpha : \kappa.\ \kappa''} \qquad \frac{\Gamma \vdash X \equiv X' : * \qquad \Gamma, x : X \vdash B \equiv B' : \kappa}{\Gamma \vdash \lambda x : X.\ B \equiv \lambda x : X'.\ B' : \Pi x : X.\ \kappa}$$

$$\frac{\Gamma \vdash C \equiv C' : \Pi\alpha : \kappa.\ \kappa' \qquad \Gamma \vdash A \equiv A' : \kappa}{\Gamma \vdash C\ A \equiv C'\ A' : [A/\alpha]C} \qquad \frac{\Gamma \vdash C \equiv C' : \Pi x : X.\ \kappa \qquad \Gamma \vdash e \equiv e' : X}{\Gamma \vdash C\ e \equiv C'\ e' : [e/x]\kappa}$$

$$\frac{\Gamma \vdash \lambda x : X.\ A : \Pi x : X.\ \kappa \qquad \Gamma \vdash e : X}{\Gamma \vdash (\lambda x : X.\ A)\ e \equiv [e/x]A : [e/x]\kappa} \qquad \frac{\Gamma \vdash \lambda\alpha : \kappa.\ A : \Pi\alpha : \kappa.\ \kappa' \qquad \Gamma \vdash A' : \kappa'}{\Gamma \vdash (\lambda\alpha : \kappa.\ A)\ A' \equiv [A'/\alpha]A : [A'/\alpha]\kappa'}$$

$$\frac{\Gamma, x : X \vdash A\ x \equiv A'\ x : \kappa \qquad \Gamma \vdash A : \Pi x : X.\ \kappa \qquad \Gamma \vdash A' : \Pi x : X.\ \kappa}{\Gamma \vdash A \equiv A' : \Pi x : X.\ \kappa}$$

$$\frac{\Gamma, \alpha : \kappa \vdash A\ \alpha \equiv A'\ \alpha : \kappa' \qquad \Gamma \vdash A : \Pi\alpha : \kappa.\ \kappa' \qquad \Gamma \vdash A' : \Pi\alpha : \kappa.\ \kappa'}{\Gamma \vdash A \equiv A' : \Pi\alpha : \kappa.\ \kappa'}$$

Figure 11 Kind and Type Equality.

$$\boxed{\Gamma \vdash e_1 \equiv e_2 : X}$$

$$\frac{\Gamma \vdash e_p : e =_X e'}{\Gamma \vdash e \equiv e' : X} \qquad \frac{\Gamma \vdash e \equiv e' : Y \qquad \Gamma \vdash X \equiv Y : *}{\Gamma \vdash e \equiv e' : X}$$

$$\frac{\Gamma \vdash e_0 \equiv e_0' : Y \qquad \Gamma, x : Y \vdash e : X}{\Gamma \vdash [e_0/x]e \equiv [e_0'/x]e : [e_0/x]X} \qquad \frac{\Gamma \vdash A \equiv A' : \kappa \qquad \Gamma, \alpha : \kappa \vdash e : X}{\Gamma \vdash [A/\alpha]e \equiv [A'/\alpha]e : [A/\alpha]X}$$

$$\frac{\Gamma \vdash e : X}{\Gamma \vdash e \equiv e : X} \qquad \frac{\Gamma \vdash e \equiv e' : X}{\Gamma \vdash e' \equiv e : X} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : X \qquad \Gamma \vdash e_2 \equiv e_3 : X}{\Gamma \vdash e_1 \equiv e_3 : X}$$

$$\frac{\Gamma \vdash \lambda\alpha : \kappa.\ e : \Pi\alpha : \kappa.\ X \qquad \Gamma \vdash A : \kappa}{\Gamma \vdash (\lambda\alpha : \kappa.\ e)A \equiv [A/\alpha]e : [A/\alpha]X}$$

$$\frac{\Gamma, \alpha : \kappa \vdash e\ \alpha \equiv e'\ \alpha : Y \qquad \Gamma \vdash e : \Pi\alpha : \kappa.\ Y \qquad \Gamma \vdash e' : \Pi\alpha : \kappa.\ Y}{\Gamma \vdash e \equiv e' : \Pi\alpha : \kappa.\ Y}$$

$$\frac{\Gamma \vdash \lambda x : X.\ e : \Pi x : X.\ Y \qquad \Gamma \vdash e' : X}{\Gamma \vdash (\lambda x : X.\ e)\ e' \equiv [e'/x]e : [e'/x]Y}$$

$$\frac{\Gamma, x : X \vdash e\ x \equiv e'\ x : Y \qquad \Gamma \vdash e : \Pi x : X.\ Y \qquad \Gamma \vdash e' : \Pi x : X.\ Y}{\Gamma \vdash e \equiv e' : \Pi x : X.\ Y}$$

$$\frac{\Gamma \vdash e : e_1 =_X e_2 \qquad \Gamma \vdash e' : e_1 =_X e_2}{\Gamma \vdash e \equiv e' : e_1 =_X e_2}$$

$$\frac{\Gamma \vdash \kappa \equiv \kappa' : \mathsf{kind} \qquad \Gamma, \alpha : \kappa \vdash e \equiv e' : Y}{\Gamma \vdash \lambda\alpha : \kappa.\ e \equiv \lambda\alpha : \kappa'.\ e' : \Pi\alpha : \kappa.\ \Pi\alpha : \kappa.\ Y} \qquad \frac{\Gamma \vdash X \equiv X' : * \qquad \Gamma, x : X \vdash e \equiv e' : Y}{\Gamma \vdash \lambda x : X.\ e \equiv \lambda x : X'.\ e' : \Pi x : X.\ Y}$$

$$\frac{\Gamma \vdash e \equiv e' : \Pi\alpha : \kappa.\ Y \qquad \Gamma \vdash A \equiv A' : \kappa}{\Gamma \vdash e\ A \equiv e'\ A' : [A/\alpha]Y} \qquad \frac{\Gamma \vdash t \equiv t' : \Pi x : X.\ Y \qquad \Gamma \vdash e \equiv e' : X}{\Gamma \vdash t\ e \equiv t'\ e' : [e/x]Y}$$

Figure 12 Term Equality.

Modal Logic and Distributed Message Passing Automata

Antti Kuusisto

Institute of Computer Science
University of Wrocław
antti.j.kuusisto@gmail.com

Abstract

In a recent article, Lauri Hella and co-authors identify a canonical connection between modal logic and deterministic distributed constant-time algorithms. The paper reports a variety of highly natural logical characterizations of classes of distributed message passing automata that run in constant time. The article leaves open the question of identifying related logical characterizations when the constant running time limitation is lifted. We obtain such a characterization for a class of finite message passing automata in terms of a recursive bisimulation invariant logic which we call *modal substitution calculus* (MSC). We also give a logical characterization of the related class $\mathcal{A}$ of infinite message passing automata by showing that classes of labelled directed graphs recognizable by automata in $\mathcal{A}$ are exactly the classes co-definable by a modal theory. A class $\mathcal{C}$ is co-definable by a modal theory if the complement of $\mathcal{C}$ is definable by a possibly infinite set of modal formulae. We also briefly discuss expressivity and decidability issues concerning MSC. We establish that MSC contains the Σ_1^μ fragment of the modal μ-calculus in the finite. We also observe that the single variable fragment MSC^1 of MSC is not contained in MSO, and that the SAT and FINSAT problems of MSC^1 are complete for PSPACE.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.1 Mathematical Logic, C.2.4 Distributed Systems

Keywords and phrases Modal logic, message passing automata, descriptive characterizations, distributed computing

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.452

1 Introduction

Distributed computing concerns itself with the investigation of computation processes carried out by computer networks. In addition to performing local computation tasks, computers or processors in the network communicate with each other by sending messages back and forth. A distributed system can be modelled by a graph, where the nodes correspond to individual computers and the edges are communication channels through which messages can be sent, see [10]. For example, a distributed system can easily determine the sets of nodes that are directly linked to another node that has a local property P: each node with the property P simply sends a message "I have property P" to each of of its neighbours. Much of the theory of distributed computing abstracts away details related to local computation, concentrating on investigations concerning the network topology.

In the recent article [7], Hella and co-authors identify a highly natural connection between modal logic [2] and local distributed algorithms. While modal logic has been successfully applied in the distributed computing context before, the perspective in [7] is a radical departure from most of the traditional approaches, where the domain elements of a Kripke model correspond to possible states of a distributed computation process. In the framework

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 452–468

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of [7], a distributed system ***is*** a Kripke model, where the domain elements are individual computers and the arrows of the accessibility relation are communication channels. While such an interpretation is of course always possible, it turns out to be particularly helpful in the study of *weak models of distributed computing* (see [7, 11]). The article [7] identifies *descriptive characterizations* for a comprehensive collection of *complexity classes of distributed computing* in terms of modal logics. For example, it is shown that the class SB(1) is *captured*— in the sense of descriptive complexity theory [6, 4, 8]— by ordinary modal logic ML. A graph property is in SB(1) iff it can be defined by a formula of ML. Various other characterizations are also obtained. For example the class MB(1) is captured by *graded modal logic*, i.e., a modal logic which can count the number of accessible nodes. Furthermore, the logical characterizations enable the use of logical tools in the investigation of distributed complexity classes. The article [7] provides a *complete classification* of the investigated complexity classes with respect to their computational capacities. The proofs behind the related separation results make significant use of logical methods. In particular, the notion of bisimulation turns out to be very useful in this context.

While there are various characterization results in classical descriptive complexity theory, separation results are rare, and related questions have proved very difficult. Therefore the separation results in [7] are rather *delightful*, since they nicely demonstrate the potential of the *descriptive complexity approach in the framework of non-classical computing.*

A *local algorithm* [11] is a distributed constant-time algorithm that distributed systems carry out by executing a fixed finite number of synchronized *communication rounds*. Our example above concerning the property P is an example of a trivial local algorithm. The characterizations in [7] concern local algorithms carried out by *message passing automata* that run in constant time. The article leaves open the question of identifying related logical characterizations when the constant running time limitation is lifted. We obtain such a characterization for a class of *finite message passing automata* in terms of a recursive bisimulation invariant logic which we call *modal substitution calculus* (MSC). The characterization extends directly to multimodal contexts and to systems with graded modalities, and thereby provides a nice characterization of cellular automata. We also give a logical characterization of the related class $\mathcal{A}$ of general (possibly infinite) message passing automata by showing that classes of labelled directed graphs recognizable by automata in $\mathcal{A}$ are exactly the classes co-definable by a modal theory. A class $\mathcal{C}$ is is co-definable by a modal theory if the complement of $\mathcal{C}$ is definable by a possibly infinite set of modal formulae. In distributed computing attention is often directed towards understanding issues concerning network topologies of distributed systems, and therefore it is often convenient to study infinite message passing automata with even non-recursive local computation capacities. See [5] for further elaborations on related matters.

In addition to logical characterizations, we briefly discuss expressivity and decidability issues concerning MSC. We establish that MSC contains the Σ_1^μ fragment of the modal μ-calculus in the finite. We also observe that the single-variable fragment MSC^1 of MSC is not contained in MSO, and that the SAT and FINSAT problems of MSC^1 are complete for PSPACE.

The aim of this article is two-fold. On one hand, we wish to investigate further the intimate link between distributed computing and modal logic identified in [7]. Advancing the understanding of this link can ideally be beneficial to both research on distributed computing and research on (modal) logic. Bringing together these two seemingly unrelated research fields could turn out to be a fruitful and refreshing research programme. For example, it seems that the *local model* [10, 9] of distributed computing is intimately related to *hybrid*

logic [1]. On the other hand, we aim to promote the potential of the descriptive complexity approach in the framework on non-classical computing.

2 Preliminaries

Let S be an arbitrary set. We let $\bigcup S$ denote the set of elements x such that $x \in L$ for some $L \in S$. We let $Pow(S)$ denote the power set of S.

Let Π be an arbitrary set of *proposition symbols* $p \in \Pi$. The language ML(Π) of ordinary modal logic is generated by the grammar

$$\varphi \ ::= \ \top \mid p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \Diamond\varphi,$$

where $p \in \Pi$ and $\top$ is a logical constant symbol. Formulae in ML(Π) are called Π-*formulae*. We define the abbreviations $\bot = \neg\top$ and $\Box = \neg\Diamond\neg$. We also use the symbols $\vee$, $\rightarrow$ and $\leftrightarrow$ in the usual way.

A *Kripke model of the vocabulary* Π (Π-model) is a structure $M = (W, R, V)$, where W is a nonempty set, called the *domain* of the model, $R \subseteq W \times W$ is a binary relation, and $V : \Pi \rightarrow Pow(W)$ is a *valuation function*. The semantics of ML(Π) is defined with respect to *pointed* Π-*models* (M, w), where $M = (W, R, V)$ is a *Kripke model* of the vocabulary Π and $w \in W$ a *point* or a *node* in the domain W of the Kripke model. For $p \in \Pi$, we define $(M, w) \models p$ iff $w \in V(p)$. We also define $(M, w) \models \top$. For the connectives, we define

$$\begin{array}{lcl} (M,w) \models \neg\,\varphi & \Leftrightarrow & (M,w) \not\models \varphi, \\ (M,w) \models (\varphi \wedge \psi) & \Leftrightarrow & \big((M,w) \models \varphi \text{ and } (M,w) \models \psi\big), \\ (M,w) \models \Diamond\varphi & \Leftrightarrow & \exists v \in W\big(wRv \text{ and } (M,v) \models \varphi\big). \end{array}$$

The set of *subformulae* of a formula φ is defined in the standard way and denoted by SUBF(φ). The *modal depth* $md(\varphi)$ of a formula is defined recursively such that $md(\top) = md(p) = 0$, $md(\neg\,\psi) = md(\psi)$, $md(\psi \wedge \chi) = max\{md(\psi), md(\chi)\}$, and $md(\Diamond\psi) = md(\psi)+1$. In a Kripke model $\big((W, R, V), w\big)$, the set $succ(w)$ of *successors* of w is the set $\{\ u \in W \mid wRu\ \}$. The set $\{\ u \in W \mid uRw\ \}$ is the set of *predecessors* of w. If φ is a modal formula and M a Kripke model, we let $||\varphi||^M$ to the the set of points w such that $(M, w) \models \varphi$.

Let Π be a set of proposition symbols. Define the set $\mathcal{S} := \{\ X_i \mid i \in \mathbb{N}\ \}$ of *schema variable symbols*. Let $\mathcal{K} \subseteq \mathcal{S}$. The set of $(\Pi, \mathcal{K})$-*schemata* of *modal substitution calculus* (MSC) is the set generated by the grammar

$$\varphi \ \ ::= \ \ \top \mid p \mid X_i \mid \neg\,\varphi \mid (\varphi \wedge \varphi) \mid \Diamond\varphi,$$

where $p \in \Pi$, $X_i \in \mathcal{K}$, and $\top$ is a logical constant symbol. A *terminal clause* of the vocabulary Π of modal substitution calculus is a string of the form $X_i(0) \ :- \ \varphi$, where $X_i \in \mathcal{S}$ is a schema variable and φ a formula of ML(Π). An *iteration clause* of the vocabulary Π of modal substitution calculus is a string of the form $X_i \ :- \ \psi$, where $X_i \in \mathcal{S}$ is a schema variable and ψ is a $(\Pi, \mathcal{K})$-schema for some set $\mathcal{K} \subseteq \mathcal{S}$ of schema variable symbols. The symbol X_i of a terminal clause $X_i(0) \ :- \ \varphi$ or an iteration clause $X_i \ :- \ \psi$ is called the *head predicate* of the clause, and the formulae φ and ψ are the *bodies* of the clauses. Let $\mathcal{K} = \{\ Y_1, ..., Y_n\ \} \subseteq \mathcal{S}$ be a finite nonempty set of n distinct schema variable symbols. A $(\Pi, \mathcal{K})$-*program* Λ of MSC consists of a pair

$$\begin{array}{lll} Y_1(0) & :- & \varphi_1 \\ . & & \\ . & & \\ . & & \\ Y_n(0) & :- & \varphi_n \end{array} \qquad \begin{array}{lll} Y_1 & :- & \psi_1 \\ . & & \\ . & & \\ . & & \\ Y_n & :- & \psi_n \end{array}$$

of lists of clauses, where the first list contains n terminal clauses $Y_i(0) \ :- \ \varphi_i$ of the vocabulary Π, and the other list contains n iteration clauses $Y_i \ :- \ \psi_i$ such that each ψ_i is a $(\Pi, \mathcal{K})$-schema. Furthermore, the $(\Pi, \mathcal{K})$-program Λ specifies a set $\mathcal{A} \subseteq \mathcal{K}$ of *appointed predicates*, so formally Λ is a triple $(\mathcal{G}, \mathcal{I}, \mathcal{A})$, where $\mathcal{G}$ and $\mathcal{I}$ are the lists of terminal clauses and iteration clauses, respectively, and $\mathcal{A}$ is an arbitrary subset of $\mathcal{K}$ specifying the appointed predicates of Λ. A program Λ is a Π-*program* if Λ is a $(\Pi, \mathcal{K})$-program for some $\mathcal{K} \subseteq \mathcal{S}$.

We let $\mathrm{ATOM}(\Lambda)$ be the set of symbols $s \in \Pi \cup \{\top\}$ that occur in the clauses of Λ. The set $\mathrm{HEAD}(\Lambda)$ is the set of schema variable symbols that occur in the clauses of Λ. The set $\mathrm{SUBS}(\varphi)$ of *subschemata* of a schema φ is defined in the obvious way. The set $\mathrm{SUBS}(\Lambda)$ of subschemata of Λ is defined to be the smallest set such that $\mathrm{HEAD}(\Lambda) \subseteq \mathrm{SUBS}(\Lambda)$ and $\mathrm{SUBS}(\varphi) \subseteq \mathrm{SUBS}(\Lambda)$ for each φ that occurs as a body of any clause (terminal or iteration) of Λ. We define $\mathrm{SUBF}(\Lambda)$ to be the set of all schemata $\varphi \in \mathrm{SUBS}(\Lambda)$ that do not contain any schema variable symbols, i.e., $\mathrm{SUBF}(\Lambda)$ is the set of sub*formulae* of Λ.

For each variable $Y_i \in \mathrm{HEAD}(\Lambda)$ of Λ, we let Y_i^0 denote the right hand side of the terminal clause $Y_i(0) :- \ \varphi_i$. Recursively, assume we have defined an $\mathrm{ML}(\Pi)$-formula Y_i^n for each $Y_i \in \mathrm{HEAD}(\Lambda)$. Let $Y_j :- \ \varphi_j$ be the iteration clause corresponding to the variable Y_j. We define Y_j^{n+1} to be the $\mathrm{ML}(\Pi)$-formula obtained by simultaneously replacing each variable Y_i of the schema φ_j by the formula Y_i^n. Let φ be an arbitrary schema in $\mathrm{SUBS}(\Lambda)$. We let φ^n denote the $\mathrm{ML}(\Pi)$-formula obtained from φ by simultaneously replacing each variable $Y_i \in \mathrm{HEAD}(\Lambda)$ in φ by the formula Y_i^n.

Let (M, w) be a pointed Π-model and Λ a Π-program of MSC. We define that $(M, w) \models \Lambda$ if there is an appointed variable Y of Λ such that for some $n \in \mathbb{N}$, we have $(M, w) \models Y^n$. We say that Λ is true in (M, w), or that (M, w) satisfies Λ.

Let Π be a finite set of proposition symbols. A *message passing automaton* A of the vocabulary Π is a tuple $(Q, M, \pi, \delta, \mu, F)$. The object Q is a nonempty set of *states*. The set Q can be finite or countably infinite. The object M is a nonempty set of *messages*. The set M can be finite or countably infinite. The object $\pi : \mathit{Pow}(\Pi) \longrightarrow Q$ is an *initial transition function* that determines the beginning state of A. The object $\delta : \mathit{Pow}(M) \times Q \longrightarrow Q$ is a *transition function* that constructs a new state in Q based on a set $N \in \mathit{Pow}(M)$ of messages received and a previous state in Q. The object $\mu : Q \longrightarrow M$ is a *message construction function* that constructs a message for the automaton to send forward based on the state of the automaton. The object $F \subseteq Q$ is a set of *accepting states* of the automaton. A message passing automaton such that the sets Q and M are finite, is a *finite message passing automaton* FMPA. (MPA stands for a message passing automaton.)

A message passing automaton A of a vocabulary Π is *run* on a Kripke model $\big(W, R, V\big)$ of the vocabulary Π, considered to be a distributed system. Intuitively, we put a copy (A, w) of the automaton to each node $w \in W$. Then, each automaton (A, w) first scans the propositional information of the node w, and then makes a transition to a beginning state based on this. Then, the automata (A, u), where $u \in W$, begin running in *synchronized steps*. During each step, each automaton first broadcasts a message to each of its neighbours with respect to R, and then updates its state based on the set of messages it receives from its neighbours. More formally, the automaton A and Kripke model (W, R, V) define a *synchronized distributed*

system which executes an omega-sequence of *communication rounds* defined as follows. Each round $n \in \mathbb{N}$ defines a *global configuration* $f_n : W \longrightarrow Q$. The configuration of the zeroth round is the function f_0 such that $f_0(w) = \pi(\{\ p \in \Pi \mid w \in V(p)\ \})$. Recursively, assume that we have defined f_n, and call $N\ = \{\ m \in M \mid m = \mu(f_n(v)),\ v \in succ(w)\ \}$. Then $f_{n+1}(w)\ =\ \delta\big(N,\ f_n(w)\big)$.

Notice that the automaton A at node w receives messages from its *successors*, so messages flow in the direction opposite to the arrows (or pairs) of the relation R. This may seem strange at first, and indeed a more natural definition would stipulate that messages flow in the direction of the arrows. The reason behind the choice here is mainly technical, and related to the technical relationship between message passing automata and modal logic. An alternative approach would be to consider modal logics with only backwards looking diamonds, or to define a Kripke structure M corresponding to a distributed system S such that M would be obtained from S by reversing the arrows of S.

When we talk about *the state of an automaton A at the node w in round n*, we mean the state $f_n(w)$. We define that an automaton A *accepts* a pointed model (M, w) if there exists some $n \in \mathbb{N}$ such that $f_n(w) \in F$, in other words, if the automaton A at w visits an accepting state during the execution of the distributed system. Note that the automaton A at w does not stop passing messages even if it has visited an accepting state. Therefore this model of computing can be regarded as a kind of a *semidecision framework for distributed computation*: an accepting node will eventually know it has accepted, but a nonaccepting node can keep running forever without knowledge of acceptance. These kinds of asymmetric acceptance conditions are common in distributed computing (see for example [7]). It would be natural to consider even more complex acceptance conditions, for example we could define a subset $G \subseteq Q$ of *rejecting* states. For the sake of space limitations, we shall not consider such possibilities here. However, the considerations below can easily be adapted to deal with various more complex acceptance scenarios.

3 MSC captures FMPA-recognizability

3.1 Specifying FMPAs in MSC

Let $(M, w) = ((W, R, V), w)$ be a pointed model and A an automaton of the same vocabulary as M. Let Q be the set of states of A. For each $u \in W$, let $A\big((M, u), n\big)$ denote the state of A at u in round n. The set $\{\ q \in Q \mid q = A\big((M, u), n\big)$ for some $u \in succ(w)\ \}$ is called the *set of states defined by the successors of w in round n*.

▶ **Theorem 1.** *Let Π be a finite set of proposition symbols. Let A be a finite message passing automaton of the vocabulary Π. There exists a Π-program Λ_A of* MSC *such that for all pointed Π-models (M, w), the automaton A accepts (M, w) iff $(M, w) \models \Lambda_A$.*

Proof. Let $A = (Q, M, \pi, \delta, \mu, F)$. Define a formula variable X_q for each state $q \in Q$. For each $q \in Q$, define the terminal clause

$$X_q(0)\ :-\ \bigvee_{P \subseteq \Pi,\ \pi(P) = q} \Big(\bigwedge_{p \in P} p\ \wedge\ \bigwedge_{p \in \Pi \setminus P} \neg p\ \Big). \tag{1}$$

(Note that $\bigvee \emptyset = \bot$ and $\bigwedge \emptyset = \top$.) Let $S \subseteq Q$ be a set of states. Define the schema

$$\varphi_S\ :=\ \bigwedge_{q \in S} \Diamond X_q\ \wedge\ \bigwedge_{q \notin S} \neg \Diamond X_q.$$

If $S \subseteq Q$ is a set of states, we denote the set $\{\ \mu(q) \mid q \in S\ \}$ by $\mu(S)$. We define $\mathcal{M}(q, q')$ to be the set of exactly all sets $S \subseteq Q$ such that $\delta(\mu(S), q) = q'$. For each state $q' \in Q$, define the iteration clause

$$X_{q'} \ :- \ \bigwedge_{q \in Q} (\ X_q \ \rightarrow \bigvee_{S \in \mathcal{M}(q,q')} \varphi_S \ ). \tag{2}$$

The program Λ_A is the Π-program defined by the terminal clauses given by Equation 1 above and the iteration clauses given by Equation 2. The set of appointed predicates is the set of symbols X_q such that $q \in F$.

Let $M = (W, R)$ be a Kripke model of the vocabulary Π. We will show that for each node $v \in W$, each state $q \in Q$, and each round $n \in \mathbb{N}$, the state of the automaton A at node v in round n is q if and only if $(M, v) \models X_q^n$. This is shown by an induction on n. The case for $n = 0$ follows immediately by the definition of the initial transition function π and the definition of $X_q(0)$.

Assume that $(M, w) \models X_{q'}^{n+1}$. Thus

$$(M, w) \models \bigwedge_{q \in Q} (\ X_q^n \ \rightarrow \bigvee_{S \in \mathcal{M}(q,q')} \varphi_S^n \ ).$$

Let $r \in Q$ be the state of A at w in round n. By the induction hypothesis, we have $(M, w) \models X_r^n$, and therefore

$$(M, w) \models \bigvee_{S \in \mathcal{M}(r,q')} \varphi_S^n.$$

Thus $(M, w) \models \varphi_S^n$ for some $S \in \mathcal{M}(r, q')$. By the definition of schema φ_S, each formula X_q^n such that $q \in S$ is satisfied by some successor of w, and there exists no successor of w that satisfies a formula X_q^n such that $q \notin S$. Therefore, by the induction hypothesis, the set of states defined by $succ(w)$ in round n is S. Since $S \in \mathcal{M}(r, q')$, we conclude that the state of the automaton at w in round $n + 1$ is q'.

For the converse, assume that the state of A at w in round $n + 1$ is q'. Let r be the state of A at w in round n. Let S be the set of states defined by $succ(w)$ in round n. Hence, by the induction hypothesis, we have $(M, w) \models \varphi_S^n$. We also have $S \in \mathcal{M}(r, q')$ by the definition of r, q' and S. Therefore

$$(M, w) \models \bigvee_{S \in \mathcal{M}(r,q')} \varphi_S^n.$$

We also know, by the induction hypothesis, that for all $q \in Q$, $(M, w) \models X_q^n$ iff $q = r$. Therefore

$$(M, w) \models \bigwedge_{q \in Q} (\ X_q^n \ \rightarrow \bigvee_{S \in \mathcal{M}(q,q')} \varphi_S \ ),$$

and thus $(M, w) \models X_{q'}^{n+1}$, as desired. ◀

3.2 Simulating MSC programs by FMPAs

Let Λ be a program of MSC, and let $\mathrm{HEAD}(\Lambda) = \{ Y_1, ..., Y_m \}$. For each $n \in \mathbb{N}$, we define $md(\Lambda, n) = max\{\ md(Y_1^n), ..., md(Y_m^n)\ \}$. We let $mdt(\Lambda)$ denote the maximum modal depth of the body formulae in the terminal clauses of Λ. Similarly, we let $mdi(\Lambda)$ denote the maximum modal depth of the body schemata of the iteration clauses of Λ.

Define $scope(\Lambda, 0) = md(\Lambda, 0)$ and $scope(\Lambda, n + 1) = scope(\Lambda, n) + max\{1, mdi(\Lambda)\}$. If $(M, w) \models \Lambda$, then the *scope of* Λ *at* w is the number $scope(\Lambda, n)$, where n is the smallest

number $k \in \mathbb{N}$ such that we have $(M, w) \models Y_i^k$ for some appointed predicate Y_i. If $(M, w) \not\models \Lambda$, the scope of Λ at w is ω. Scope is a relatively natural spatio-temporal complexity measure for the execution of an MSC program, when the execution is done by first evaluating each formula Y_i^0, then each formula Y_i^1, and so on. Notice that even if $mdi(\Lambda) = 0$, scope is increased after each iteration step. It is of course possible to define other natural complexity measures for MSC programs.

Let A be an automaton and (M, w) a pointed model. If A accepts (M, w), then the *decision time of A at w* is the smallest number k such that the state of A at w is an accepting state in round k. If A does not accept (M, w), the decision time of A at w is ω.

Next we show how to define, when given a program Λ of MSC, a corresponding automaton A_Λ that accepts exactly the pointed models (M, w) such that $(M, w) \models \Lambda$. Furthermore, the decision time of A_Λ at each node w will be equal to the scope of Λ at w. Roughly, the states of A_Λ will encode finite sets of formulae satisfied by nodes of the underlying model. For more of the intuition behind the definition of A_Λ, see the proof of Theorem 2.

Let Π be a finite set of proposition symbols and fix a Π-program Λ of MSC. We assume that $mdi(\Lambda) \geq 1$. The pathological case where $mdi(\Lambda) = 0$ is discussed separately.

The set Q_Λ of states of A_Λ contains all pairs (S, m), where $m \leq mdi(\Lambda)-1$ is a nonnegative integer and $S \subseteq \mathrm{SUBS}(\Lambda)$ a set of schemata φ such that $md(\varphi) \leq m$. The set Q_Λ also contains all triples (S, m, f), where $m \leq mdt(\Lambda) - 1$ is a nonnegative integer, $S \subseteq \mathrm{SUBF}(\Lambda)$ is a set of *formulae* φ such that $md(\varphi) \leq m$, and f is simply a symbol indicating that this state encodes sets of *formulae* in $\mathrm{SUBF}(\Lambda)$. There are no other states in Q_Λ. The set of messages M_Λ is $Pow\big(\mathrm{SUBS}(\Lambda)\big)$. (Some states and some messages may turn out to be irrelevant for the computation of A_Λ.)

We then define the transition function π of A_Λ. Assume first that $mdt(\Lambda) \geq 1$. Let $P \subseteq \Pi$ be a set of proposition symbols. Define a set $U \subseteq \mathrm{SUBF}(\Lambda)$ to be the smallest set such that the following conditions hold.

1. $\big(P \cap \mathrm{SUBF}(\Lambda)\big) \cup \big(\{\top\} \cap \mathrm{SUBF}(\Lambda)\big) \subseteq U$.
2. For each $\neg\varphi \in \mathrm{SUBF}(\Lambda)$ of the modal depth 0, $\neg\varphi \in U$ iff $\varphi \notin U$.
3. For each $(\varphi \wedge \psi) \in \mathrm{SUBF}(\Lambda)$ of the modal depth 0, $(\varphi \wedge \psi) \in U$ iff both $\varphi \in U$ and $\psi \in U$.

We define $\pi(P) = (U, 0, f)$. If $mdt(\Lambda) = 0$, we define $\pi(P)$ for the set $P \subseteq \Pi$ of proposition symbols differently. First define a set $T \subseteq \mathrm{SUBF}(\Lambda)$ to be the smallest set such that the following conditions hold.

1. $\big(P \cap \mathrm{SUBF}(\Lambda)\big) \cup \big(\{\top\} \cap \mathrm{SUBF}(\Lambda)\big) \subseteq T$.
2. For each formula $\neg\varphi \in \mathrm{SUBF}(\Lambda)$ of the modal depth 0, we have $\neg\varphi \in T$ iff $\varphi \notin T$.
3. For each formula $(\varphi \wedge \psi) \in \mathrm{SUBF}(\Lambda)$ of the modal depth 0, we have $(\varphi \wedge \psi) \in T$ iff both $\varphi \in T$ and $\psi \in T$.

Now let T' be the set of symbols in $\mathrm{ATOM}(\Lambda) \cup \mathrm{HEAD}(\Lambda)$ of the modal depth 0 such that the following conditions hold.

1. For each $X \in \mathrm{HEAD}(\Lambda)$, we have $X \in T'$ iff $X^0 \in T$.
2. For each $\varphi \in \mathrm{ATOM}(\Lambda)$, we have $\varphi \in T'$ iff $\varphi \in T$.

Define U to be the set of schemata in $\mathrm{SUBS}(\Lambda)$ of the modal depth 0 such that the following conditions hold.

1. For each $\varphi \in \mathrm{ATOM}(\Lambda) \cup \mathrm{HEAD}(\Lambda)$, $\varphi \in U$ iff $\varphi \in T'$.
2. For each schema $\neg\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0, $\neg\varphi \in U$ iff $\varphi \notin U$.
3. For each schema $(\varphi \wedge \psi) \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0, $(\varphi \wedge \psi) \in U$ iff both $\varphi \in U$ and $\psi \in U$.

We define $\pi(P) = (U, 0)$.

We then define the transition function δ of A_Λ. Let (S, m) be a state of A_Λ. Let $N \subseteq M_\Lambda$ be a set of messages. Assume that $m < mdi(\Lambda) - 1$. Assume there exists a smallest set U such that the following conditions hold.

1. For each schema $\varphi \in \mathrm{SUBS}(\Lambda)$ such that $md(\varphi) < m + 1$, we have $\varphi \in U$ iff $\varphi \in S$.
2. For each schema $\Diamond\varphi \in \mathrm{SUBS}(\Lambda)$ such that $md(\Diamond\varphi) \leq m+1$, we have $\Diamond\varphi \in U$ iff $\varphi \in \bigcup N$.
3. For each schema $(\varphi \wedge \psi) \in \mathrm{SUBS}(\Lambda)$ such that $md(\varphi \wedge \psi) \leq m + 1$, we have $(\varphi \wedge \psi) \in U$ iff both $\varphi \in U$ and $\psi \in U$.
4. For each schema $\neg\varphi \in \mathrm{SUBS}(\Lambda)$ such that $md(\neg\varphi) \leq m + 1$, we have $\neg\varphi \in U$ iff $\varphi \notin U$.

We then define $\delta\big(N, (S, m)\big)$ to be the state $(U, m + 1)$. If no set U satifying the above conditions exists, we define $\delta\big(N, (S, m)\big)$ arbitrarily.

If $m = mdi(\Lambda) - 1$, we define $\delta\big(N, (S, m)\big)$ differently. Assume there exists a smallest set $T \subseteq \mathrm{SUBS}(\Lambda)$ such that the following conditions hold. (If no such set T exists, $\delta((S, m), N)$ is defined arbitrarily.)

1. For each schema $\varphi \in \mathrm{SUBS}(\Lambda)$ such that $md(\varphi) < m + 1$, we have $\varphi \in T$ iff $\varphi \in S$.
2. For each schema $\Diamond\varphi \in \mathrm{SUBS}(\Lambda)$ such that $md(\Diamond\varphi) \leq m+1$, we have $\Diamond\varphi \in T$ iff $\varphi \in \bigcup N$.
3. For each schema $(\varphi \wedge \psi) \in \mathrm{SUBS}(\Lambda)$ such that $md(\varphi \wedge \psi) \leq m + 1$, we have $(\varphi \wedge \psi) \in T$ iff both $\varphi \in T$ and $\psi \in T$.
4. For each schema $\neg\varphi \in \mathrm{SUBS}(\Lambda)$ such that $md(\neg\varphi) \leq m + 1$, we have $\neg\varphi \in T$ iff $\varphi \notin T$.

Now define a set $T' \subseteq \mathrm{HEAD}(\Lambda) \cup \mathrm{ATOM}(\Lambda)$ such that the following conditions hold.

1. For each $X \in \mathrm{HEAD}(\Lambda)$, we have $X \in T'$ iff $\varphi \in T$, where φ is the body of the iteration clause for X.
2. For each $\varphi \in \mathrm{ATOM}(\Lambda)$, we have $\varphi \in T'$ iff $\varphi \in T$.

Define U to be the set of schemata of the modal depth 0 in $\mathrm{SUBS}(\Lambda)$ such that the following conditions hold.

1. For each $\varphi \in \mathrm{ATOM}(\Lambda) \cup \mathrm{HEAD}(\Lambda)$, $\varphi \in U$ iff $\varphi \in T'$.
2. For each $\neg\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0, $\neg\varphi \in U$ iff $\varphi \notin U$.
3. For $(\varphi \wedge \psi) \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0, $(\varphi \wedge \psi) \in U$ iff both $\varphi \in U$ and $\psi \in U$.

Then $\delta(N, (S, m))$ is defined to be the state $(U, 0)$.

Let (S, m, f) be state of A_Λ. Let $N \subseteq M_\Lambda$ be a set of messages. Assume that $m < mdt(\Lambda) - 1$. Assume there exists a smallest set U such that the following conditions hold.

1. For each formula $\varphi \in \mathrm{SUBF}(\Lambda)$ such that $md(\varphi) < m + 1$, we have $\varphi \in U$ iff $\varphi \in S$.
2. For each formula $\Diamond\varphi \in \mathrm{SUBF}(\Lambda)$ such that $md(\Diamond\varphi) \leq m + 1$, we have $\Diamond\varphi \in U$ iff $\varphi \in \bigcup N$.
3. For each formula $(\varphi \wedge \psi) \in \mathrm{SUBF}(\Lambda)$ such that $md(\varphi \wedge \psi) \leq m+1$, we have $(\varphi \wedge \psi) \in U$ iff both $\varphi \in U$ and $\psi \in U$.
4. For each formula $\neg\varphi \in \mathrm{SUBF}(\Lambda)$ such that $md(\neg\varphi) \leq m + 1$, we have $\neg\varphi \in U$ iff $\varphi \notin U$.

We then define $\delta\big(N, (S, m, f)\big)$ to be the state $(U, m + 1, f)$. If no set U satifying the above conditions exists, we define $\delta\big(N, (S, m, f)\big)$ arbitrarily.

If $m = mdt(\Lambda)-1$, we define $\delta\big(N, (S, m, f)\big)$ differently. Assume there exists a smallest set $T \subseteq \mathrm{SUBS}(\Lambda)$ such that the following conditions hold. (If no such set T exists, $\delta(N, (S, m, f))$ is defined arbitrarily.)

1. For each formula $\varphi \in \mathrm{SUBF}(\Lambda)$ such that $md(\varphi) < m + 1$, we have $\varphi \in T$ iff $\varphi \in S$.
2. For each formula $\Diamond\varphi \in \mathrm{SUBF}(\Lambda)$ such that $md(\Diamond\varphi) \leq m+1$, we have $\Diamond\varphi \in T$ iff $\varphi \in \bigcup N$.
3. For each formula $(\varphi \wedge \psi) \in \mathrm{SUBF}(\Lambda)$ such that $md(\varphi \wedge \psi) \leq m + 1$, we have $(\varphi \wedge \psi) \in T$ iff both $\varphi \in T$ and $\psi \in T$.
4. For each formula $\neg\varphi \in \mathrm{SUBF}(\Lambda)$ such that $md(\neg\varphi) \leq m + 1$, we have $\neg\varphi \in T$ iff $\varphi \notin T$.

Now define a set $T' \subseteq \mathrm{HEAD}(\Lambda) \cup \mathrm{ATOM}(\Lambda)$ such that the following conditions hold.

1. For each $X \in \mathrm{HEAD}(\Lambda)$, we have $X \in T'$ iff $X^0 \in T$.
2. For each $\varphi \in \mathrm{ATOM}(\Lambda)$, we have $\varphi \in T'$ iff $\varphi \in T$.

Define U to be the set of schemata in $\mathrm{SUBS}(\Lambda)$ of the modal depth 0 such that the following conditions hold.

1. For each $\varphi \in \mathrm{ATOM}(\Lambda) \cup \mathrm{HEAD}(\Lambda)$, $\varphi \in U$ iff $\varphi \in T'$.
2. For each $\neg\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0, $\neg\varphi \in U$ iff $\varphi \notin U$.
3. For all $(\varphi \wedge \psi) \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0, $(\varphi \wedge \psi) \in U$ iff both $\varphi \in U$ and $\psi \in U$.

Then $\delta(N,(S,m,f))$ is defined to be the state $(U,0)$.

The message construction function μ of A_Λ is defined such that $\mu\big((S,m)\big) = S$ and $\mu\big((S,m,f)\big) = S$. The set F of accepting states of A_Λ is the set of states $(S,0)$ such that we have $Y \in S$ for some appointed head predicate of Λ.

We have now defined the automaton A_Λ, assuming that $mdi(\Lambda) \neq 0$. The definition of A_Λ in the pathological case where $mdi(\Lambda) = 0$ is discussed in the proof of Theorem 2.

▶ **Theorem 2.** *Let Π be a finite set of proposition symbols. Let Λ be a Π-program of* MSC. *Let (M,w) be a pointed Π-model. We have $(M,w) \models \Lambda$ if and only if A_Λ accepts (M,w). Furthermore, the scope of Λ at w equals the decision time of A_Λ at w.*

Proof. We begin by describing the idea of the proof. Let W be the domain of M. The automata A_Λ at the nodes $u \in W$ first compute the extensions $||X^0||^M$ of formulae X^0 for each $X \in \mathrm{HEAD}(\Lambda)$. The automata then operate in *cycles* of communication rounds. During a cycle, the automata compute the extensions of formulae X^{n+1} based on the extensions of formulae X^n computed during the previous cycle. The communication steps during the cycle contribute to the information about extensions of formulae of greater and greater modal depths. The proof will proceed by induction on the iteration number n, and each step of the induction will be a subinduction on modal depth of schemata. We assume that $mdi(\Lambda) \neq 0$. The case where $mdi(\Lambda) = 0$ will be briefly discussed at the end of the proof.

Define a set C_0 such that $C_0 = \{-1\} \times \{0,...,mdt(\Lambda)-1\}$ if $mdt(\Lambda) \neq 0$, and $C_0 = \emptyset$ if $mdt(\Lambda) = 0$. Define also $C_1 = \mathbb{N} \times \{0,...,mdi(\Lambda)-1\}$. Let $C = C_0 \cup C_1$. Order the pairs in C lexicographically, i.e., $(i,j) < (i',j') \Leftrightarrow (i < i' \vee (i = i' \wedge j < j'))$. Let $<_C$ denote this order. Let g be the isomorphism from $(C,<_C)$ to $(\mathbb{N},<)$. We let $Q_v(i,j)$ denote the set of schemata φ occurring in the state (S,m) or (S,m,f) of the automaton A_Λ at node v in the round $g((i,j))$. Observe that $Q_v(i,j)$ contains schemata of the modal depth up to j.

We will show by induction on n that the equivalence

$$(M,v) \models \varphi^n \text{ iff } \varphi \in Q_v\big(n,0\big)$$

holds for all $v \in W$, all $n \in \mathbb{N}$, and all schemata $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0. Each step of the induction is a subinduction on the modal depth of schemata.

Let $n = 0$. Some of the details of the case $n = 0$ are straightforward or rather similar to corresponding details of the case $n > 0$, and therefore omitted here. (See the appendix for the omitted cases.) The case $n > 0$ is discussed in detail, and the omitted details for the case $n = 0$ can be easily constructed from the corresponding arguments for the case $n > 0$.

Assume that $mdt(\Lambda) \neq 0$. For the case $mdt(\Lambda) = 0$, see the appendix. Call $\Phi = \mathrm{SUBF}(\Lambda) \cap \Pi$. By the definition of the transition function π, we have $(M,v) \models p \Leftrightarrow p \in \pi\big(\{\, p \in \Pi \mid v \in V(p) \,\}\big)$ for each $p \in \Phi$. Therefore, for each atomic formula $\varphi \in \mathrm{ATOM}(\Lambda)$, we have $(M,v) \models \varphi \Leftrightarrow \varphi \in Q_v(-1,0)$. Hence, since every formula $\varphi \in \mathrm{SUBF}(\Lambda)$ of the modal depth 0 is a Boolean combination of formulae in $\varphi \in \mathrm{ATOM}(\Lambda)$, we have $(M,v) \models \varphi \Leftrightarrow \varphi \in Q_v(-1,0)$ for all $\varphi \in \mathrm{SUBF}(\Lambda)$ of the modal depth 0.

We then need to establish that the equivalence $(M,v) \models \psi \;\Leftrightarrow\; \psi \in Q_v\big(-1, mdt(\Lambda)-1\big)$ holds for each $v \in W$ and each $\psi \in \mathrm{SUBF}(\Lambda)$ such that $md(\psi) \leq mdt(\Lambda)-1$. If $mdt(\Lambda) = 1$, we are done. If not, the equivalence can be proved by induction on the modal depth of formulae. We shall omit the details here (see the appendix). Once we have established that $(M,v) \models \psi \;\Leftrightarrow\; \psi \in Q_v(-1, mdt(\Lambda)-1)$, we can show that therefore $(M,v) \models \varphi^0 \;\Leftrightarrow\; \varphi \in Q_v\big(0,0\big)$ for all *schemata* $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0 and all $v \in W$, thereby concluding the argument for the case $n = 0$. We omit the details here (see the appendix).

Now assume the claim of the main induction holds for $n \in \mathbb{N}$, and consider the case for $n+1$. By the induction hypothesis, we have $(M,v) \models \varphi^n \;\Leftrightarrow\; \varphi \in Q_v(n,0)$ for all $v \in W$ and all $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0. We need to prove that

$$(M,v) \models \varphi^{n+1} \;\Leftrightarrow\; \varphi \in Q_v(n+1,0)$$

for all $v \in W$ and all schemata in $\mathrm{SUBS}(\Lambda)$ of the modal depth 0. In order to show this, we shall first establish that $(M,v) \models \varphi^n \;\Leftrightarrow\; \varphi \in Q_v(n,k)$ for all $v \in W$ and all $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth k such that $0 \leq k \leq mdi(\Lambda)-1$. This is proved by by induction on the modal depth k of schemata.

Since we already know that $(M,v) \models \varphi^n \;\Leftrightarrow\; \varphi \in Q_v(n,0)$ for all $v \in W$ and all $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0, the basis of the subinduction on modal depth is clear. In the case $mdi(\Lambda) = 1$, this suffices, and no subinduction is actually needed. Therefore assume that $mdi(\Lambda) > 1$ and let $k \in \{\,0\,,\ldots,\; mdi(\Lambda)-2\,\}$. Assume that $(M,v) \models \varphi^n \;\Leftrightarrow\; \varphi \in Q_v(n,k)$ for all schemata in $\mathrm{SUBF}(\Lambda)$ of the modal depth up to k and all $v \in W$. Let $\varphi \in \mathrm{SUBS}(\Lambda)$ be a schema of the modal depth $k+1$. The schema φ is a Boolean combination of schemata $\Diamond\psi$, where $md(\psi) \leq k$. It therefore suffices to show that for each such schema $\Diamond\psi$, we have $(M,v) \models \Diamond\psi^n \;\Leftrightarrow\; \Diamond\psi \in Q_v(n,k+1)$.

Assume first that $(M,v) \models \Diamond\psi^n$. Therefore some successor u of v satisfies $(M,u) \models \psi^n$. By the induction hypothesis, $\psi \in Q_u(n,k)$. Therefore the automaton A_Λ at u sends a message L such that $\psi \in L$ to its predecessors in round $g((n,k+1))$. Thus $\Diamond\psi \in Q_v(n,k+1)$.

Conversely, assume that $\Diamond\psi \in Q_v(n,k+1)$. Therefore v receives a message L such that $\psi \in L$ from some successor u in round $g((n,k+1)) = g((n,k))+1$. Hence $\psi \in Q_u(n,k)$. By the induction hypothesis, $(M,u) \models \psi^n$. Therefore $(M,v) \models \Diamond\psi^n$.

We have now established that $(M,v) \models \varphi^n \;\Leftrightarrow\; \varphi \in Q_v(n,k)$ for all $v \in W$ and all $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth k such that $0 \leq k \leq mdi(\Lambda)-1$. We shall next show that therefore $(M,v) \models \varphi^{n+1} \;\Leftrightarrow\; \varphi \in Q_v(n+1,0)$ for all $v \in W$ and all $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0.

Recall the definition of the sets T, T' and U in the definition of δ on input states (S,m) in the case where $m = mdi(\Lambda)-1$. We shall first show that $(M,w) \models \varphi^n \;\Leftrightarrow\; \varphi \in T$ holds for each $\varphi \in \mathrm{SUBS}(\Lambda)$ such that $md(\varphi) \leq mdi(\Lambda)$.

Let $\varphi \in \mathrm{SUBS}(\Lambda)$ be a schema such that $md(\varphi) \leq mdi(\Lambda)$. The schema φ is a Boolean combination of schemata $\Diamond\psi$, where $md(\psi) < mdi(\Lambda)$. It therefore suffices to show that for each such schema $\Diamond\psi$, we have $(M,v) \models \Diamond\psi^n \;\Leftrightarrow\; \Diamond\psi \in T$. This is shown by an argument analogous to the corresponding argument discussed above. (See the appendix.)

We can now conclude that $(M,v) \models X^{n+1} \;\Leftrightarrow\; X \in T'$ for all head symbols $X \in \mathrm{HEAD}(\Lambda)$, and also $(M,v) \models \varphi \;\Leftrightarrow\; \varphi \in T'$ for all atomic formulae $\varphi \in \mathrm{ATOM}(\Lambda)$. Therefore, since every schema $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0 is a Boolean combination of formulae in $\mathrm{ATOM}(\Lambda)$ and head symbols in $\mathrm{HEAD}(\Lambda)$, we have $(M,v) \models \varphi^{n+1} \;\Leftrightarrow\; \varphi \in Q_v(n+1,0)$ for all $v \in W$ and all $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0, as required.

Finally, if $mdi(\Lambda) = 0$, it is easy to define an automaton Λ that satisfies the requirements of the Theorem. The number $|\mathrm{HEAD}(\Lambda)|$ of different head predicates is finite, so there

are finitely many different truth distributions that the set of head predicates can obtain. Therefore, once we have computed the extensions of the formulae X^0, we can directly check at each node, without further communication with neighbouring automata, whether *any* iteration X^n of any appointed head predicate X of Λ is true. This holds because the Boolean combinations obtained by the head predicate set must begin repeating periodically after sufficiently many iterations. ◀

4 Modal theories capture complements of MPA-recognizable classes

Let Π be a finite set of proposition symbols. Let $\mathcal{C}$ be the class of pointed Π-models. A class $\mathcal{K} \subseteq \mathcal{C}$ of pointed Π-models is said to be *definable by a modal theory* if there exists a set Φ of modal Π-formulae (Π-theory) such that for all $(M, w) \in \mathcal{C}$, we have $(M, w) \models \Phi$ iff $(M, w) \in \mathcal{K}$. By $(M, w) \models \Phi$ we mean that $(M, w) \models \varphi$ for all $\varphi \in \Phi$. A class $\mathcal{K}' \subseteq \mathcal{C}$ is said to be *co-definable by a modal theory* Φ if $\mathcal{C} \setminus \mathcal{K}'$ is definable by the modal theory Φ.

Let Π be a finite set of proposition symbols. The set T_0 of Π-*types* of the modal depth 0 is defined to be the set containing a conjunction

$$\bigwedge_{p \in S} p \;\wedge\; \bigwedge_{p \notin S} \neg p$$

for each set $S \subseteq \Pi$, and no other formulae. We assume some canonical bracketing and ordering of conjuncts, so that there is exactly one conjunction for each set S in T_0. Note also that $\bigwedge \emptyset \;=\; \top$. The type $\tau_{(M,w),0}$ of a pointed Π-model (M, w) is the unique formula φ in T_0 such that $(M, w) \models \varphi$.

Assume then that we have defined the set T_n of Π-types of the modal depth n. Assume that T_n is finite, and assume also that each pointed Π-model (M, w) satisfies exactly one type $\tau_{(M,w),n}$ of the modal depth n. Define

$$\tau_{(M,w),n+1} \;:=\; \tau_{(M,w),n} \;\wedge\; \bigwedge\{\, \Diamond\tau \mid \tau \in T_n,\ (M, w) \models \Diamond\tau \,\} \;\wedge\; \bigwedge\{\, \neg\Diamond\tau \mid \tau \in T_n,\ (M, w) \not\models \Diamond\tau \,\}.$$

The formula $\tau_{(M,w),n+1}$ is the Π-type of the modal depth $n + 1$ of (M, w). We assume some standard ordering of conjuncts and bracketing, so that if two types $\tau_{(M,w),n+1}$ and $\tau_{(N,v),n+1}$ are equivalent, they are actually the same formula. We define T_{n+1} to be the set $\{\, \tau_{(M,w),n+1} \mid (M, w)$ is a pointed Π-model $\}$. We observe that the set T_{n+1} is finite, and that for each pointed Π-model (M, w), there is exactly one type $\tau \in T_{n+1}$ such that $(M, w) \models \tau$.

It is easy to show that each Π-formula φ of modal logic is equivalent to the disjunction of exactly all Π-types τ of the modal depth $md(\varphi)$ such that $\tau \models \varphi$. By $\tau \models \varphi$ we mean that for all pointed Π-models (M, w), we have $(M, w) \models \tau \;\Rightarrow\; (M, w) \models \varphi$. (Note that $\bigvee \emptyset = \bot$).

Define a *type automaton* A for Π to be message passing automaton whose set of states is exactly the set $\mathcal{T}$ of all Π-types. The set of messages is also the set $\mathcal{T}$. Furthermore, the initial transition function π is defined such that the state of A at (M, w) in round $n = 0$ is the type $\tau_{(M,w),0}$. Let N be a set of types. If all types in N are types of the same modal depth n, and if τ is a type of the modal depth n, we define $\delta(N, \tau)$ to be the type

$$\tau_{n+1} \;=\; \tau \wedge \bigwedge_{\sigma \in N} \Diamond\sigma \;\wedge\; \bigwedge_{\sigma \in T_n \setminus N} \neg\Diamond\sigma.$$

On other inputs, δ is defined arbitrarily. The message construction function μ is defined such that $\mu(\tau) = \tau$. The set of accepting states can be defined differently for different type

automata A of the vocabulary Π. It is easy to see that the state of any type automaton A at (M, v) in round n is τ iff the type of the modal depth n of (M, v) is τ.

▶ **Theorem 3.** *Let Π be a finite set of proposition symbols. Each class of pointed Π-models co-definable by a modal Π-theory is recognizable by a message passing automaton.*

Proof. Let $\mathcal{K}$ be a class of Π-models co-definable by a modal Π-theory Φ. Let φ be an arbitrary formula in Φ. The formula $\neg\varphi$ is equivalent to the disjunction of Π-types τ of the modal depth $md(\varphi)$ such that $\tau \models \neg\varphi$. Let $D(\neg\varphi)$ denote the disjunction. We write $\tau \in D(\neg\varphi)$ in order to indicate that τ is one of the disjuncts of $D(\neg\varphi)$.

Let $\mathcal{T}$ denote the set of exactly all Π-types. Define a Π-type automaton A such that the set of accepting states is the set $\{\, \tau \in \mathcal{T} \mid \tau \in D(\neg\varphi) \text{ for some } \varphi \in \Phi \,\}$. It is straightforward to show that the automaton accepts exactly the class $\mathcal{K}$ of pointed Π-models. ◀

▶ **Theorem 4.** *Let Π be a finite set of proposition symbols. Each class of pointed Π-models recognizable by a message passing automaton is co-definable by a modal theory.*

Proof. Let (M, w) be a pointed Π-model. Let A be a message passing automaton whose set of proposition symbols is Π. Let $n \in \mathbb{N}$. We let $A\big((M, w), n\big)$ denote the state of the automaton A at the node w in round n. We shall begin the proof by showing that the following statements are equivalent for all pointed Π-models (M, w) and (N, v) and all $n \in \mathbb{N}$.

1. The models (M, w) and (N, v) satisfy exactly the same Π-type of the depth n.
2. $A\big((M, w), k\big) = A\big((N, v), k\big)$ for each $k \leq n$ and each message passing automaton A whose set of proposition symbols is Π.

We prove the claim by induction on n. For $n = 0$, the claim holds trivially by definition of the transition function π.

Let (M, w) and (N, v) be pointed Π-models that satisfy exactly the same Π-types of the modal depth $n + 1$. Let A be an automaton and δ the transition function of A. Call $q_n = A\big((M, w), n\big)$ and $q_{n+1} = A\big((M, w), n + 1\big)$. Let $\sigma_1, ..., \sigma_k$ enumerate the Π-types of the modal depth n and assume that

$$\tau_{(M,w),n+1} \;=\; \tau_{(M,w),n} \;\wedge \bigwedge_{i \in \{1,\dots,m\}} \Diamond\, \sigma_i \;\wedge \bigwedge_{i \in \{m+1,\dots,k\}} \neg\Diamond\, \sigma_i$$

Since (M, w) and (N, v) satisfy the same Π-type $\tau_{(M,w),n+1}$ of the depth $n + 1$, they also satisfy the same Π-type $\tau_{(M,w),n}$ of the depth n. By the induction hypothesis, we therefore conclude that $A\big((N, v), n\big) = q_n$. Also, since (M, w) and (N, v) satisfy the same type of the depth $n + 1$, the set of types of the depth n satisfied by the successors of w is the same as the set satisfied by the successors of v. That set is $\{\, \sigma_1, ..., \sigma_m \,\}$ in both cases. Therefore, by the induction hypothesis, the set of states defined by $succ(w)$ in round n is exactly the same as the set of states defined by $succ(v)$ in round n. Therefore the set of messages received by w in round $n + 1$ is exactly the same as the set of messages received by v in round $n + 1$. Therefore, since $A\big((N, v), n\big) = q_n$, we conclude that $A\big((N, v), n + 1\big) = q_{n+1}$, as required.

Let (M, w) and (N, v) be pointed Π-models and assume that $A\big((M, w), k\big) = A\big((N, v), k\big)$ for each $k \leq n + 1$ and each message passing automaton A whose set of proposition symbols is Π. Since this is true for an arbitrary automaton A of the vocabulary Π, this holds for any *type automaton* of the vocabulary Π. Hence (M, w) and (N, v) satisfy exactly the same Π-types of the depth $n + 1$.

We have now established equivalence of the conditions 1 and 2 above. We are ready to show that each class of pointed Π-models recognizable by an automaton can also be co-defined by a modal theory.

Let A be an arbitrary Π-automaton. Let $\mathcal{C}$ be the class of exactly all pointed Π-models accepted by A. Define $\mathcal{T}$ to be the collection of exactly all Π-types. Let Φ denote the set of exactly all Π-types $\tau \in \mathcal{T}$ such that for some n, the type τ is the Π-type of the depth n of some pointed Π-model (M, w), and furthermore, the automaton A accepts (M, w) in round n. Define the infinite disjunction $\bigvee \Phi$. We shall establish that for all pointed Π-models (M, w), we have $(M, w) \models \bigvee \Phi$ iff A accepts (M, w).

Assume that $(M, w) \models \bigvee \Phi$. Thus $(M, w) \models \tau_n$ for some type τ_n of the depth n of some pointed model (M', w') accepted by A in round n. Now (M, w) and (M', w') satisfy the same type τ_n, so by the equivalence of the conditions 1 and 2 above, (M, w) and (M', w') must both be accepted by A in round n.

Assume that (M, w) is accepted by the automaton A. The pointed model (M, w) is accepted in some round n, and thus the type of the depth n of (M, w) is one of the disjuncts of Φ. Therefore $(M, w) \models \bigvee \Phi$. The modal theory $\{ \neg \tau \mid \tau \in \Phi \}$ co-defines the class $\mathcal{C}$ of pointed Π-models accepted by A. ◀

5 Expressivity and Decidability

In this section we very briefly investigate expressivity and decidability issues concerning MSC. We first investigate the *single variable fragment* MSC^1 of MSC. This fragment contains the programs Λ such that $|\mathrm{HEAD}(\Lambda)| = 1$. In the finite, the single variable fragment MSC^1 can simulate formulae of the μ-calculus of the type $\mu X.\varphi$, where φ is free of fixed point operators (see the proof of Proposition 7). Also, MSC^1 is not contained in MSO (proof of Proposition 6). It turns out that decidability and PSPACE-completeness of the satisfiability and finite satisfiability problems of MSC^1 follow rather trivially by the following delightful argument.

▶ **Proposition 5.** *The* SAT *and* FINSAT *problems for* MSC^1 *are* PSPACE*-complete.*

Proof. Let Λ be a program of MSC^1. Let X be the appointed head predicate symbol of Λ. (If Λ has no appointed symbol, Λ is not satisfiable.) We first check whether the formula X^0 is satisfiable by using a decision algorithm for ordinary modal logic. If not, we check whether the formula X^1 is satisfiable, again using a decision algorithm for ordinary modal logic. If not, we know that Λ is not satisfiable, for the following simple reason.

Let $(M, w) = ((W, R, V), w)$ be an arbitrary model of the same vocabulary as Λ. Let φ be the schema such that the iteration clause of Λ for X is $X :- \varphi$. Define the function $F : Pow(W) \to Pow(W)$ such that $F(U) = \{ u \in W \mid \big((W, R, V[X \mapsto U]), u\big) \models \varphi \}$. Since $||X^0||^M = ||X^1||^M = \emptyset$, we observe that $F(\emptyset) = \emptyset$. Since $||X^{n+1}||^M = F(||X^n||^M)$ for all $n \in \mathbb{N}$, we conclude that no formula X^k is satisfied by any node of M.

The claim of the current proposition now follows from the PSPACE-completeness of ordinary modal logic. ◀

We leave the question of decidability of MSC open at this stage, and sketch some proofs concerning expressivity instead. The μ-calculus (μML) is a bisimulation invariant logic that expands modal logic with a recursion mechanism based on least and greatest fixed point operators μX and νX. For the semantics and basic properties of μML, see [3].

▶ **Proposition 6.** $\mathrm{MSC}^1 \not\leq \mu\mathrm{ML}$. *This holds already in the finite.*

Proof Sketch. (Note that we only sketch a proof of this proposition.) Define a program Λ of MSC^1 which is true in (M, w) iff the following conditions hold.

1. There exists some $n \in \mathbb{N}$ such that there is a directed path of the length n from w to a point v without successors. We call v a dead-end.

2. There are no directed paths shorter than n from w to a dead-end, and each directed path of the length n originating from w ends in a dead-end.

If a pointed model (M, w) satisfies the above property, with n being the unique distance to a dead-end, we say that (M, w) has the n-path property.

Define $X^0 :- \Box\bot$ and $X :- \Diamond X \wedge \Box X$. It is easy to show by induction on n that for all pointed models (M, w), the model (M, w) satisfies the n-path property iff $(M, w) \models X^n$.

If a pointed model has the n-path property for some $n \in \mathbb{N}$, we say that (M, w) has the centre point property. The class of pointed models with the centre point property is not definable by any formula of μML. This is shown by establishing that there exists no formula $\varphi(x)$ of MSO such that $M \models \varphi(w)$ iff (M, w) has the centre point property. The claim that $\mathrm{MSC}^1 \not\leq \mu\mathrm{ML}$ in the finite then follows, as it is well known that $\mu\mathrm{ML} < \mathrm{MSO}$.

Assume, for the sake of contradiction, that there exists a formula $\varphi(x)$ of MSO that defines the centre point property. Therefore MSO can define the corresponding property in restriction to the class of rooted finite ranked trees with two successor relations. By the pumping lemma for tree languages it is then trivial to establish that this is a contradiction. ◀

Alternation of μ and ν-operators is a tricky issue in μ-calculus, and alternation hierarchies have been defined in various ways. We define Σ_1^μ to be the fragment of μ-calculus without ν-operators and with negations on the atomic level, i.e., the language built from literals with $\wedge, \vee, \Diamond$ and $\Box$, and μX when X occurs only positively in the scope of μX. We define Π_1^μ analogously.

▶ **Proposition 7.** $\Sigma_1^\mu < \mathrm{MSC}$ *in the finite.*

Proof Sketch. (Note that we only sketch the proof of this proposition.) It is folklore that μ-calculus can be defined with or without the capacity of using simultaneous fixed points, without change in expressive power. There are translations both ways, from standard μ-calculus into one with simultaneous fixed points and back. It is also folklore that μ-calculus can be defined in terms of modal equation systems (see [3]). For instance, a formula $\mu X.\psi\big(X, \mu Y.\varphi(X, Y)\big)$ translates to the *equation block*

$$\begin{array}{lll} X & :- & \psi(X, Y) \\ Y & :- & \varphi(X, Y), \end{array}$$

where $\psi(X, Y)$ is the formula obtained from the formula $\psi\big(X, \mu Y.\varphi(X, Y)\big)$ by replacing the subformula $\mu Y.\varphi(X, Y)$ by the variable Y. For a more concrete example, the formula $\mu X.\big(\Box X \vee \mu Y.(p \vee \Diamond(Y \vee X))\big)$ translates to the block

$$\begin{array}{lll} X & :- & \Box X \vee Y \\ Y & :- & p \vee \Diamond(Y \vee X). \end{array}$$

If $M = (W, R)$ is a model, the block

$$\begin{array}{lll} X & :- & \psi(X, Y) \\ Y & :- & \varphi(X, Y) \end{array}$$

defines a monotone function $F : (Pow(W))^2 \longrightarrow (Pow(W))^2$ such that

$$F(U, V) = \big(||\psi(U, V)||^M, ||\varphi(U, V)||^M\big).$$

The least fixed point $F^\infty(\emptyset, \emptyset)$ of this monotone operator is a pair (X^∞, Y^∞) such that $X^\infty = ||\mu X.\, \psi\big(X, \mu Y.\varphi(X, Y)\big)||^M$.

An arbitrary formula φ of Σ_1^μ translates into an equation block with a finite number of equations. We may assume that φ is of the form $\mu X.\psi$. (If not, we may use a dummy variable X.) The set X^∞ is then exactly the set $||\varphi||^M$. The very same equation block also defines a program Λ_φ of MSC, with the terminal clause corresponding to each variable Z being $Z(0) :- \bot$, and the set of appointed variables being $\{X\}$. Now, Λ_φ is true in a finite model M exactly in the nodes belonging to X^∞. This follows immediately, since there is a finite number $n \in \mathbb{N}$ such that $F_M^n(\emptyset, ..., \emptyset) = F_M^{n+1}(\emptyset, ..., \emptyset)$, i.e., the *closure ordinal* of F is finite. Therefore, for all $w \in W$, we have $w \in X^\infty$ iff there is some $n \in \mathbb{N}$ such that we have $(M, w) \models X^n$ for the appointed variable X of Λ.

The strictness of the inclusion $\Sigma_1^\mu <$ MSC in the finite follows by Proposition 6. ◀

▶ **Proposition 8.** $\Pi_1^\mu \not\leq$ MSC. *This holds already in the finite.*

Proof. MSC cannot define non-reachability: there exists no program of MSC true in exactly those pointed models (M, w) where there is no directed path to, say, a point v without successors, i.e., a dead-end. Assume that such a program exists. Run it in a *directed successor ring*, i.e, a connected finite model (W, R), where R is a binary relation and where both the out-degree and in-degree of each node is one. Let w be a node of the ring. If non-reachability is definable, there is an automaton A such that if we run it on the ring, it accepts w in some finite number n of rounds. However, let (N, S) be a finite model, where S is a successor ordering of N and $|N| \geq n + 10$. Let u be the least element of N with respect to S. It is straightforward to show that in the n-th round of running A, the state of A at w is exactly the same as at u. Therefore A accepts $((N, S), u)$, which is a contradiction.

The formula $\nu X.(\Diamond\top \wedge \Box X)$ states that a deadend cannot be reached from the point of observation. ◀

Finally, it is worth noting that the model checking problem for MSC is clearly decidable, as the sequence of global configurations defined by an FMPA and a finite model must eventually loop. With an MPA it is possible to recognize, with respect to the class of finite pointed Kripke models, even undecidable classes of models.

References

1 C. Areces and B. ten Cate. Hybrid Logics. In P. Blackburn, F. Wolter and J. van Benthem (eds.), *Handbook of Modal Logic*, Elsevier, 2006.

2 P. Blackburn, M. de Rijke and Y. Venema. *Modal Logic*. Cambridge University Press, 2011.

3 J. Bradfield and C. Stirling. Modal mu-calculi. In P. Blackburn, J. van Benthem and F. Wolter (eds.), *The Handbook of Modal Logic*, Elsevier, 2006.

4 H. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 2nd edition, 2005.

5 P. Fraigniaud, M. Göös, A. Korman and Jukka Suomela. What can be decided locally without identifiers? arXiv:1302.2570, 2013.

6 N. Immerman. *Descriptive Complexity*. Springer, 1999.

7 L. Hella, M. Järvisalo, A. Kuusisto, J. Laurinharju, T. Lempiäinen, K. Luosto, J. Suomela and J. Virtema. Weak models of distributed computing, with connections to modal logic. In *Proc. 31st ACM Symposium on Principles of Distributed Computing* (PODC), 2012.

8 L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

9 N. Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193-201, 1992.

10 D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

11 J. Suomela. Survey of local algorithms. *ACM Computing Surveys* 45, 2013.

A Appendix

A.1 Addenda to the proof of Theorem 2

Argument for the special case where $n = 0$ and $mdt(\Lambda) = 0$

Recall the definition of the sets T, T' and U in the definition of π on initial inputs $P \subseteq \Pi$ for the case where $mdt(\Lambda) = 0$. Let V be the valuation of M. Call $\Phi = \mathrm{SUBF}(\Lambda) \cap \Pi$. By the definition π, we have $(M, v) \models p$ iff $p \in \pi\big(\{\ p \in \Pi \mid v \in V(p)\ \}\big)$ for each $p \in \Phi$, and therefore, the equivalence $(M, v) \models \varphi$ iff $\varphi \in T$ holds for each *atomic formula* $\varphi \in \mathrm{ATOM}(\Lambda)$. Hence, since every formula $\varphi \in \mathrm{SUBF}(\Lambda)$ of the modal depth 0 is a Boolean combination of formulae in $\mathrm{ATOM}(\Lambda)$, we have $(M, v) \models \varphi$ iff $\varphi \in T$ for all *formulae* $\varphi \in \mathrm{SUBF}(\Lambda)$ of the modal depth 0. Hence we have $(M, v) \models X^0$ iff $X \in T'$ for all schema variable symbols $X \in \mathrm{HEAD}(\Lambda)$, and also $(M, v) \models \varphi$ iff $\varphi \in T'$ for all atomic formulae $\varphi \in \mathrm{ATOM}(\Lambda)$. Therefore, since every *schema* $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0 is a Boolean combination of formulae in $\mathrm{ATOM}(\Lambda)$ and head predicate symbols in $\mathrm{HEAD}(\Lambda)$, the equivalence $(M, v) \models \varphi^0$ iff $\varphi \in Q_v(0, 0)$ holds for all schemata $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0, as required.

$(M, v) \models \psi \ \Leftrightarrow\ \psi \in Q_v(-1, mdt(\Lambda) - 1)$

We have $(M, v) \models \varphi$ iff $\varphi \in Q_v(-1, 0)$ for all $v \in W$ and all $\varphi \in \mathrm{SUBF}(\Lambda)$ of the modal depth 0, so the base step of the induction is clear. Let $k \in \mathbb{N}$ such that $k \leq mdt(\Lambda) - 2$, and assume that the equivalence $(M, v) \models \psi$ iff $\psi \in Q_v(-1, k)$ holds for each $v \in W$ and each $\psi \in \mathrm{SUBF}(\Lambda)$ such that $md(\psi) \leq k$. Let $v \in W$ and let $\varphi \in \mathrm{SUBF}(\Lambda)$ be a formula of the modal depth $k + 1$. We must show that $(M, v) \models \varphi$ iff $\varphi \in Q_v(-1, k + 1)$. The formula φ is a Boolean combination of formulae $\Diamond\psi$, where $md(\psi) \leq k$. It therefore suffices to show that for each such formula $\Diamond\psi$, we have $(M, v) \models \Diamond\psi$ iff $\Diamond\psi \in Q_v(-1, k + 1)$.

Assume first that $(M, v) \models \Diamond\psi$. Therefore some successor u of v satisfies $(M, u) \models \psi$. By the induction hypothesis, $\psi \in Q_u(-1, k)$. Hence, in round $k + 1$, the automaton A at u sends a message L such that $\psi \in L$ to the predecessors of u. Thus $\Diamond\psi \in Q_v(-1, k + 1)$.

Assume then that $\Diamond\psi \in Q_v(-1, k + 1)$. Therefore the automaton A_Λ at node v receives a message L such that $\psi \in L$ from some successor u in round $k + 1$. Therefore $\psi \in Q_u(-1, k)$. By the induction hypothesis, $(M, u) \models \psi$. Therefore $(M, v) \models \Diamond\psi$.

$(M, v) \models \varphi^0 \ \Leftrightarrow\ \varphi \in Q_v\big(0, 0\big)$

Recall the definition of the sets T, T' and U in the definition of δ on input states (S, m, f) in the case where $m = mdt(\Lambda) - 1$. We shall first show that $(M, v) \models \varphi$ iff $\varphi \in T$ holds for each $v \in W$ and each formula $\varphi \in \mathrm{SUBF}(\Lambda)$ such that $md(\varphi) \leq mdt(\Lambda)$.

Let $v \in W$. Let $\varphi \in \mathrm{SUBF}(\Lambda)$ be a formula such that $md(\varphi) \leq mdt(\Lambda)$. The formula φ is a Boolean combination of formulae $\Diamond\psi$, where $md(\psi) < mdt(\Lambda)$. By the definition of T, it suffices to show that for each such formula $\Diamond\psi$, we have $(M, v) \models \Diamond\psi$ iff $\Diamond\psi \in T$.

Assume first that $(M, v) \models \Diamond\psi$. Therefore some successor u of v satisfies $(M, u) \models \psi$. Therefore, since $md(\psi) < mdt(\Lambda)$, we know that $\psi \in Q_u\big(-1, mdt(\Lambda) - 1\big)$. Thus the automaton A_Λ at u sends a message L such that $\psi \in L$ to its predecessors in round $mdt(\Lambda)$. Thus $\Diamond\psi \in T$.

Conversely, assume that $\Diamond\psi \in T$. Therefore v receives a message L such that $\psi \in L$ from some successor u in round $mdt(\Lambda)$. Hence we have $\psi \in Q_u\big(-1, mdi(\Lambda) - 1\big)$. Therefore we know that $(M, u) \models \psi$. Thus $(M, v) \models \Diamond\psi$.

We have now established that $(M, v) \models \varphi \;\Leftrightarrow\; \varphi \in T$ for all $v \in W$ and all formulae $\varphi \in \mathrm{SUBF}(\Lambda)$ of the modal depth up to $mdt(\Lambda)$. Thus $(M, v) \models X^0 \;\Leftrightarrow\; X \in T'$ for all head predicate symbols $X \in \mathrm{HEAD}(\Lambda)$, and also $(M, v) \models \varphi$ iff $\varphi \in T'$ for all atomic formulae $\varphi \in \mathrm{ATOM}(\Lambda)$. Therefore, since every *schema* $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0 is a Boolean combination of formulae in $\varphi \in \mathrm{ATOM}(\Lambda)$ and head predicate symbols in $\mathrm{HEAD}(\Lambda)$, we have $(M, v) \models \varphi^0$ iff $\varphi \in Q_v\big(0, 0\big)$ for all $v \in W$ and all schemata in $\varphi \in \mathrm{SUBS}(\Lambda)$ of the modal depth 0, as required. This concludes the base case of our argument by induction on n.

$(M, v) \models \Diamond\psi^n \;\Leftrightarrow\; \Diamond\psi \in T$

Assume first that $(M, v) \models \Diamond\psi^n$. Therefore some successor u of v satisfies $(M, u) \models \psi^n$. Hence, since $md(\psi) < mdi(\Lambda)$, we know that $\psi \in Q_u\big(n, mdi(\Lambda) - 1\big)$. Therefore the automaton A_Λ at u sends a message L such that $\psi \in L$ to its predecessors in round $g(n + 1, 0)$. Thus $\Diamond\psi \in T$.

Conversely, assume that $\Diamond\psi \in T$. Therefore v receives a message L such that $\psi \in L$ from some successor u in round $g(n + 1, 0)$. Hence, $\psi \in Q_u\big(n, mdi(\Lambda) - 1\big)$. Therefore we know that $(M, u) \models \psi^n$. Therefore $(M, v) \models \Diamond\psi^n$.

Global semantic typing for inductive and coinductive computing

Daniel Leivant

Indiana University Bloomington
leivant@indiana.edu

Abstract

Common data-types, such as $\mathbb{N}$, can be identified with term algebras. Thus each type can be construed as a global set; e.g. for $\mathbb{N}$ this global set is instantiated in each structure $\mathcal{S}$ to the denotations in $\mathcal{S}$ of the unary numerals. We can then consider each declarative program as an axiomatic theory, and assigns to it a semantic (Curry-style) type in each structure. This leads to the *intrinsic theories* of [18], which provide a purely logical framework for reasoning about programs and their types. The framework is of interest because of its close fit with syntactic, semantic, and proof theoretic fundamentals of formal logic.

This paper extends the framework to data given by coinductive as well as inductive declarations. We prove a Canonicity Theorem, stating that the denotational semantics of an equational program P, understood operationally, has type τ over the canonical model iff P, understood as a formula has type τ in every "data-correct" structure. In addition we show that every intrinsic theory is interpretable in a conservative extension of first-order arithmetic.

1998 ACM Subject Classification F.3 Logics and meanings of programs

Keywords and phrases Inductive and coinductive types, equational programs, intrinsic theories, global model theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.469

1 Introduction

The notion of program typing, first introduced by Curry [5, 32], views types as semantic properties of pre-existing untyped objects. A function f has type $\tau \to \sigma$ if it maps objects of type τ to objects of type σ; f may well be defined for input values that are not of type τ. In contrast, Church [4] considered types as inherent properties of objects: a function has type $\tau \to \sigma$ when its domain consists of the objects of τ, and its codomain consists of objects of type σ. The difference between semantic and inherent typing is thus ontologically significant in a way that phrases such as "explicit" and "implicit" do not convey.

A distinction between inherent and semantic typing can also be made for inductive data types T, such as the booleans, natural numbers, and strings. While each such data-type has a canonical intended meaning, it is isomorphic to the term algebra over some set C of constructors. That syntactic representation suggests a *global semantics* for such types. Namely, T is a global predicate, that is a mapping that to each structure $\mathcal{S}$ (for a vocabulary V containing C) assigns the set of denotations of closed C-terms.

(Recall that global semantics is an organizing principle for descriptive and computational devices over a class of structures, such as all finite graphs [6, 9]. An example is Fagin's celebrated result that a global relation over finite structures is NP iff it is definable by an existential second-order formula [7].)

The global viewpoint is of particular interest with respect to *programs* over inductive data. Each such program P may be of type $T \to T$ in one V-structure and not in another;

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 469–483

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

e.g. if P is non-total on $\mathbb{N}$, then it is of type $\mathtt{N} \to \mathtt{N}$ in the flat-domain structure with $\mathtt{N}$ interpreted as $\mathbb{N}_\perp$, but not when $\mathtt{N}$ is interpreted as $\mathbb{N}$. Already this simple observation resolves the status of an incorrect proposal by Herbrand, by which a set of equations P is said to compute a function $f : \mathbb{N} \to \mathbb{N}$ iff f is the unique solution of P.[1]

This proposal was corrected by Gödel [8], who replaced the Tarskian semantics of a set of equations (i.e. true or false in a given structure) by an operational semantics. But in fact Herbrand was right to equate operational semantics with Tarskian semantics, with one caveat: P computes f in the standard structure iff f has, by Tarskian semantics, type $N \to N$ in *every* model of P. We elaborate on this below.

Within the global framework, it makes sense to consider formal V-theories for proving global typing properties of equational programs. We adopt as programming model equational programs, since these mesh directly with formal reasoning: a program's equations can be construed as axioms, computations as derivations in equational logic, and types as formulas. Moreover, equational programs are amenable to term-model constructions, which turn out to be a useful meta-mathematical tool. Theories for reasoning directly about equational programs were developed in [18], where they were dubbed *intrinsic theories*. Among other benefits, they support attractive proof-theoretic characterizations of major function classes, such as the provable functions of Peano Arithmetic and the primitive recursive functions [18, 19].

In this paper we generalize the global semantics approach of [18] to *data-systems,* that is collections of data-types generated by both inductive and coinductive definitions. To do so we shall start by describing a syntactic framework, in analogy to the term algebras, in which a syntactic representation of the intended data-types is possible. We shall continue by giving an operational semantics for equational programs over data-systems, i.e. when data-objects may be infinite. We then prove a generalized Canonicity Theorem, which states that a program over a data-system is correctly typed in the standard structure (e.g. terminating for inductive output and fair for coinductive output) just in case such typing is correct by Tarskian semantics in all structures. Finally we show that the obvious first-order theories for proving correct-typing of equational programs are no stronger than Peano Arithmetic.

2 Data systems

2.1 Symbolic data

A *constructor-vocabulary* is a finite set $\mathcal{C}$ of function identifiers, referred to as *constructors,* each assigned an *arity* ≥ 0; as usual, constructors of arity 0 are *object-identifiers*. We'll posit the presence of a master constructor-vocabulary, and consider its sub-vocabularies. Given a constructor-vocabulary $\mathcal{C}$, the *replete term-set* for $\mathcal{C}$ is the set $R_\mathcal{C}$ consisting of all finite or infinite ordered trees of constructors, where each node with constructor $\mathbf{c}$ of arity r has exactly r children. Obviously $R_\mathcal{C}$ is definable coinductively, but we will be interested primarily in its subsets, defined both inductively and coinductively.

The *replete $\mathcal{C}$-structure* is the structure $\mathcal{R}_\mathcal{C}$ with[2]

[1] Actually, Herbrand's proposal also called $<$ for a constructive proof that a solution exists and is unique. But that $<$ additional condition is ill-formed, and cannot be replaced by provability in $<$ some intuitionistic theory, since that would imply that the computable total functions $<$ form a semi-decidable collection.

[2] We use typewriter font for actual identifiers, boldface for meta-level variables ranging over syntactic objects, and italics for other meta-level variables.

1. $\mathcal{C}$ as vocabulary (i.e. similarity-type);
2. $R_{\mathcal{C}}$ as universe; and
3. a syntactic interpretation of the constructors: for an r-ary $\mathbf{c} \in \mathcal{C}$ $\quad [\![\mathbf{c}]\!](a_1 \dots a_r)$ $\quad$ is the tree with $\mathbf{c}$ at the root and $a_1 \dots a_r$ as immediate sub-trees.

2.2 Equational programs

In addition to the set $\mathcal{C}$ of constructors we posit an infinite set $\mathcal{X}$ of *variables,* and an infinite set $\mathcal{F}$ of function-identifiers, dubbed *program-functions,* and assigned arities ≥ 0 as well. The sets $\mathcal{C}$, $\mathcal{X}$ and $\mathcal{F}$ are, of course, disjoint. If $\mathcal{E}$ is a set consisting of function-identifiers and (possibly) variables, we write $\bar{\mathcal{E}}$ for the set of terms generated from $\mathcal{E}$ by application: if $\mathbf{g} \in \mathcal{E}$ is a function-identifier of arity r, and $\mathbf{t}_1 \dots \mathbf{t}_r$ are terms, then so is $\mathbf{g}\,\mathbf{t}_1 \cdots \mathbf{t}_r$. We use informally the parenthesized notation $\mathbf{g}(\mathbf{t}_1, \dots, \mathbf{t}_r)$, when convenient.[3] We refer to elements of $\overline{\mathcal{C}}$, $\overline{\mathcal{C} \cup \mathcal{X}}$ and $\overline{\mathcal{C} \cup \mathcal{X} \cup \mathcal{F}}$ as *data-terms, base-terms,* and *program-terms,* respectively.[4]

We adopt equational programs, in the style of Herbrand-Gödel, as computation model. There are easy inter-translations between equational programs and program-terms such as those of $\mathbf{FLR}_0$ [21]. We prefer however to focus on equational programs because they integrate easily into logical calculi, and are naturally construed as axioms. Codifying equations by terms is a conceptual detour, since the computational behavior of such terms is itself spelled out using equations or rewrite-rules.

A *program-equation* is an equation of the form $\mathbf{f}(\mathbf{t}_1, \dots, \mathbf{t}_k) = \mathbf{q}$, where $\mathbf{f}$ is a program-function of arity $k \geq 0$, $\mathbf{t}_1 \dots \mathbf{t}_k$ are base-terms, and $\mathbf{q}$ is a program-term. Two program-equations are *compatible* if their left-hand sides cannot be unified. A *program-body* is a finite set of pairwise-compatible program-equations. A *program* $(P, \mathbf{f})$ (of arity k) consists of a program-body P and a program-function $\mathbf{f}$ (of arity k) dubbed the program's *principal-function.* We identify each program with its program-body when in no danger of confusion.

Programs of arity 0 can be used to define objects. For example, the singleton program T consisting of the equation $\mathtt{t} = \mathtt{sss0}$ defines 3, in the sense that in every model $\mathcal{S}$ of T the interpretation of the identifier $\mathtt{t}$ is the same as that of the numeral for 3. Consider instead a 0-ary program defining an infinite term such as singleton program I consisting of $\mathtt{ind} = \mathtt{s}(\mathtt{ind})$. This does not have any solution in the free algebra of the unary numerals, that is: the free algebra cannot be expanded into the richer vocabulary with $\mathtt{ind}$ as a new identifier, so as to satisfy the equation I.[5] But I is modeled, for example, in any structure where $\mathtt{s}$ is interpreted as identity, and $\mathtt{ind}$ as any structure element. Also, I is modeled over any ordinal $\alpha \succ \omega$, with $\mathtt{s}$ interpreted as the function $x \mapsto 1 + x$ and I as any infinite $\beta \in \alpha$.

2.3 Operational semantics of programs

If $(P, \mathbf{f})$ is a program over the set $\bar{\mathcal{C}}$ of data-terms (which are all finite) then we can say that it computes the partial function $\quad g : \bar{\mathcal{C}} \rightharpoonup \bar{\mathcal{C}} \quad$ when $g(p) = q$ iff the equation $\mathbf{f}(p) = q$ is derivable from P in equational logic. But non-trivial replete term-structures have infinite terms, so the output of a program over $R_{\mathcal{C}}$ must be computed piecemeal from finite information about the input values.

To express piecemeal computation of infinite data, without using extra constants or tools, we posit that each program over $\mathcal{C}$ has defining equations for destructors and a discriminator.

---

[3] In particular, when $\mathbf{g}$ is of arity 0, it is itself a term, whereas with parentheses we'd have $\mathbf{g}()$.
[4] Data-terms are often referred to as *values*, and base-terms as *patterns.*
[5] As usual, when a structure is an expansion of another they have the same universe.

That is, if the given vocabulary's k constructors are $\mathbf{c}_1 \dots \mathbf{c}_k$, with m the maximal arity, then the program-functions include the unary identifiers $\pi_{i,m}$ $(i = 1..m)$ and δ, and all programs contain the equations (for $\mathbf{c}$ an r-ary constructor)

$$\begin{array}{lcll} \pi_{i,m}(\mathbf{c}(x_1,\dots,x_r)) & = & x_i & (i = 1..r) \\ \pi_{i,m}(\mathbf{c}(x_1,\dots,x_r)) & = & \mathbf{c}(x_1,\dots,x_r) & (i = r{+}1..m) \\ \delta(\mathbf{c}_i(\vec{\mathbf{t}}), x_1,\dots,x_k) & = & x_i & (i = 1..k) \end{array}$$

We call a repeated composition of destructors a *deep destructor*. For $a \in R_{\mathcal{C}}$ and variable $\mathtt{v}$ let $\Delta_{a/\mathbf{v}}$ consist of all statements $\delta(\Pi(\mathtt{v}), x_1, \dots, x_k) = x_i$ where Π is a deep destructor, that are true when $\mathtt{v} = a$. That is, $\Delta_{a/\mathbf{v}}$ conveys, node by node, the structure of the syntactic tree a, using $\mathtt{v}$ as a name for a.

▶ **Definition 1.** We say that a unary program $(P, \mathbf{f})$ *computes* the partial function $g : R_{\mathcal{C}} \rightharpoonup R_{\mathcal{C}}$ when for every $a, b \in R_{\mathcal{C}}$ we have $g(a) = b$ iff for each deep-destructor Π the equation $\delta(\Pi(\mathbf{f}(\mathtt{v})), \vec{x}) = x_i$ is derivable in equational logic from P and $\Delta_{a/\mathbf{v}}$, where $\mathbf{c}_i$ is the main constructor of $\Pi(b)$.

That is, one can establish in equational logic the equality of $f(a)$ and of b at each "address" Π, given unbounded information about the structure of a.

The definition for programs of arity > 1 is similar.

Note that the piecemeal definition of computability is made necessary only by the presence of infinite data:

▶ Proposition 2. If, in definition (1) above, the terms a, b are finite, then $g(a) = b$ just in case $\mathbf{f}(a) = b$ is derivable from P in equational logic. ◀

The proof is straightforward, and omitted here since Proposition 2 is not used in the sequel.

2.4 Inductive Data systems

To motivate the general definition, let us consider first purely inductive types. One defines a single type by its closure rules: the natural numbers are given by the two rules $\mathtt{N}(\mathtt{0})$ and $\mathtt{N}(x) \to \mathtt{N}(\mathtt{s}(x))$. Similarly, words in $\{0, 1\}$ can be construed as terms using unary constructors $\mathtt{0}$ and $\mathtt{1}$, as well as nullary constructor $\mathtt{e}$, and generated by the rules $\mathtt{W}(\mathtt{e})$, $\mathtt{W}(x) \to \mathtt{W}(\mathtt{0}(x))$ and $\mathtt{W}(x) \to \mathtt{W}(\mathtt{1}(x))$. If $\mathtt{G}$ names a given type G, then the type of binary trees with leaves in G is generated by the rules $\mathtt{G}(x) \to \mathtt{T}(x)$, and $\mathtt{T}(x) \wedge \mathtt{T}(y) \to \mathtt{T}(\mathtt{p}(x, y))$.

We can similarly generate types jointly (i.e. simultaneously). For example, the following rules generate the 01-strings with no adjacent $\mathtt{1}$'s, by defining jointly the set (denoted by $\mathtt{E}$) of such strings that start with $\mathtt{1}$, and the set (denoted by $\mathtt{Z}$) of those that don't: $\mathtt{Z}(\mathtt{e})$, $\mathtt{Z}(x) \to \mathtt{Z}(\mathtt{0}(x))$, $\mathtt{Z}(x) \to \mathtt{E}(\mathtt{1}(x))$, and $\mathtt{E}(x) \to \mathtt{Z}(\mathtt{0}(x))$.

Generally, a definition of inductive types from given types $\vec{\mathbf{G}}$ consists of:

1. A list $\vec{\mathbf{D}}$ of unary relation-identifiers, dubbed *type identifiers*;
2. a set of *composition rules,* of the form

$$\frac{\mathbf{Q}_1(x_1) \quad \cdots \quad \mathbf{Q}_r(x_r)}{\mathbf{D}_i(\mathbf{c}\, x_1 \cdots x_r)}$$

where $\mathbf{c}$ is a constructor, and each $\mathbf{Q}_\ell$ is one of the data-predicates in $\vec{\mathbf{G}}, \vec{\mathbf{D}}$.

These rules delineate the intended meaning of inductive $\vec{\mathbf{D}}$ from below. Namely, elements of $\mathbf{D}_i$ are built up by the composition rules. Thus the data-predicates in $\vec{\mathbf{D}}$ are defined jointly, potentially using also the given types $\vec{\mathbf{G}}$.

Conjuncting the composition rules, we obtain a single rule, consisting of the the universal closure of conjunctions of implications into the data-type being defined.

2.5 Coinductive decomposition rules

Inductive composition rules state sufficient reasons to assert that a term has a given type, implied by the types of its immediate sub-terms. The intended semantics of an inductive type $\mathbf{D}$ is thus the smallest set of terms closed under those rules. Coinductive decomposition rules state necessary conditions for a term to have a given type, by implying the types of its immediate sub-terms. The intended semantics is the largest set of terms satisfying those conditions.

For instance, the type of ω-words over $0/1$ is given by the decomposition rule

$$\mathtt{W}^\omega(x) \to (\exists y\ \mathtt{W}^\omega(y) \wedge x = \mathtt{0}y) \vee (\exists y\ \mathtt{W}^\omega(y) \wedge x = \mathtt{1}y) \tag{1}$$

This is not quite captured by the implications $\mathtt{W}^\omega(\mathtt{0}x) \to \mathtt{W}^\omega(x)$ and $\mathtt{W}^\omega(\mathtt{1}x) \to \mathtt{W}^\omega(x)$, since these do not guarantee that every element of $\mathtt{W}^\omega$ is of the right form.

The implication $\mathtt{W}^\omega(x) \to \mathtt{W}^\omega(\pi_{1,1}(x))$ also fails to capture the rules (1), as shown by the following example. In analogy to the inductive definition above of the words with no adjacent $\mathtt{1}$'s, the ω-words over $0/1$ with no adjacent $\mathtt{1}$'s are delineated jointly by the two decomposition rules

$$\mathtt{Z}(x) \ \to\ (\exists y\ \mathtt{Z}(y) \wedge x = \mathtt{0}(y)) \ \vee\ (\exists y\ \mathtt{E}(y) \wedge x = \mathtt{0}(y))$$

and

$$\mathtt{E}(x) \ \to\ \exists y\ \mathtt{Z}(y) \wedge x = \mathtt{1}(y)$$

These rules cannot be captured using destructors, since those do not differentiate between cases for the input's main constructor.

These observations justify the following definition.

▶ **Definition 3.** A *decomposition definition of coinductive types from given types* $\vec{\mathbf{G}}$ consists of:

1. A list $\vec{\mathbf{D}}$ of *type identifiers*;
2. for each of the types $\mathbf{D}_i$ in $\vec{\mathbf{D}}$ a *decomposition rule,* of the form

$$\mathbf{D}_i(x) \ \to\ \psi_1 \vee \cdots \vee \psi_k$$

where each ψ_i is of the form

$$\exists y_1 \ldots y_r\ x = \mathbf{c}(\vec{y}) \ \wedge\ \mathbf{Q}_1(y_1) \ \wedge \cdots \wedge \mathbf{Q}_r(y_r)$$

Here $\mathbf{c}$ is a constructor of arity r, and each $\mathbf{Q}_\ell$ is one of the data-predicates in $\vec{\mathbf{G}}, \vec{\mathbf{D}}$. Thus, the single decomposition rule for each $\mathbf{D}_i$ is an implication from $\mathbf{D}_i$ to the disjunction of existential statements.

2.6 Data systems

We now define data-systems, where data-types can be defined by any combination of induction and coinduction. Descriptive and deductive tools for such definitions were studied extensively, e.g. referring to typed lambda calculi, with operators μ for smallest fixpoint and

ν for greatest fixpoint. For instance, the Common Algebraic Specification Language CASL has been used as a unifying standard in the algebraic specification community, and extended to coalgebraic data [27, 29, 22, 30]. Several frameworks combining inductive and coinductive data, such as [24], strive to be comprehensive, including various syntactic distinctions and categories, in contrast to our minimalistic approach.

▶ **Definition 4.** A *data-system* $\mathcal{D}$ over a set $\mathcal{C}$ of constructors consists of:

1. A double-list $\vec{\mathbf{D}}_1 \dots \vec{\mathbf{D}}_k$ (the order matters) of unary relation-identifiers, dubbed *type-identifiers*, where each $\vec{\mathbf{D}}_i$ is dubbed a *data-bundle,* and designated as either *inductive* or *coinductive.*
2. For each inductive data-bundle $\vec{\mathbf{D}}_i$ an *inductive definition* given as a (finite) set of *data-composition* rules of the form

$$(\bigwedge_{\ell=1..r} \mathbf{Q}_\ell(x_\ell)) \to \mathbf{D}_{ij}(\mathbf{c}\, x_1 \cdots x_r)$$

   where each $\mathbf{Q}_\ell$ is one of the data-predicates in $\vec{\mathbf{D}}_1 \dots \vec{\mathbf{D}}_i$. Note that r may be 0.
   These rules delineate the intended meaning of inductive $\vec{\mathbf{D}}_i$ from below. Namely, elements of $\mathbf{D}_{ij}$ are built up by the data-introduction rules. Thus the data-predicates in each $\vec{\mathbf{D}}_i$ are defined simultaneously, potentially using also previously defined predicates among $\vec{\mathbf{D}}_1 \dots \vec{\mathbf{D}}_{i-1}$.
3. For each coinductive data-predicate $\mathbf{D}_{ij}$ a *coinductive definition* consisting of a *data-decomposition* rule, of the form

$$\mathbf{D}_{ij}(x) \;\to\; \psi_1 \vee \cdots \vee \psi_k$$

   where each ψ_ℓ is of the form

$$\exists y_1 \dots y_r \; x = \mathbf{c}(\vec{y}) \;\wedge\; \mathbf{Q}_1(y_1) \;\wedge\; \cdots \;\wedge\; \mathbf{Q}_r(y_r)$$

   with $\mathbf{c}$ a constructor, and each $\mathbf{Q}_\ell$ in $\vec{\mathbf{D}}_j$, $j \le i$.

2.7 Examples of data-systems

1. Let $\mathcal{C}$ consist of the identifiers `0`, `1`, `e`, `s`, and `c`, of arities 0,0,0,1, and 2, respectively. Consider the following data-system, for the double list $((\mathtt{B}), (\mathtt{N}), (\mathtt{F}, \mathtt{S}), (\mathtt{L}))$ with inductive `B` and `N` (booleans and natural numbers), coinductive `F` and `S` (streams with alternating boolean and numeral entries starting with booleans (respectively, with natural numbers)), and finally an inductive `L` for lists of such streams.

$$\mathtt{B}(\mathtt{0}) \quad \mathtt{B}(\mathtt{1})$$

$$\mathtt{N}(\mathtt{0}) \quad \mathtt{N}(x) \to \mathtt{N}(\mathtt{s}x)$$

$$\mathtt{F}(x) \to \exists y, z \; (x = \mathtt{c}yz) \;\wedge\; \mathtt{B}(y) \;\wedge\; \mathtt{S}(z)$$
$$\mathtt{S}(x) \to \exists y, z \; (x = \mathtt{c}yz) \;\wedge\; \mathtt{N}(y) \;\wedge\; \mathtt{F}(z)$$

$$\mathtt{L}(\mathtt{0}) \quad \mathtt{L}(\mathtt{e}) \quad \mathtt{F}(x) \wedge \mathtt{L}(y) \to \mathtt{L}(\mathbf{c}xy) \quad \mathtt{S}(x) \wedge \mathtt{L}(y) \to \mathtt{L}(\mathbf{c}xy)$$

   Note that constructors are reused for different data-types. This is in agreement with our untyped, generic approach, where the intended type information is conveyed by the data-predicates, and the data-objects are untyped. In other words, data-types are semantic (Curry style) rather than ontological (Church style).

2. Here is a data system with a type for infinite binary trees with nodes decorated by finite/infinite binary trees with booleans as leaves. The constructors are `0`, `1`, `p`, and `d` of arities 0,0,2 and 3 respectively. The data-predicates are an inductive `B`, and two coinductive `D` (for trees) and `T` (for trees of trees). The composition rules for `B` are

$$\mathtt{B}(\mathtt{0}) \qquad \text{and} \qquad \mathtt{B}(\mathtt{1})$$

The decomposition rules for `T` and `D` (as a single bundle) are

$$\mathtt{D}(x) \to \mathtt{B}(x) \vee \exists y_1, y_2 \;\; x = \mathtt{p}(y_1, y_2) \quad \wedge \; \mathtt{D}(y_1) \; \wedge \; \mathtt{D}(y_2)$$

and

$$\mathtt{T}(x) \to \exists u, y_1, y_2 \;\; x = \mathtt{d}(u, y_1, y_2) \quad \wedge \; \mathtt{D}(u) \; \wedge \; \mathtt{T}(y_1) \; \wedge \; \mathtt{T}(y_2)$$

2.8 Computational completeness of equational programs

The equivalence of equational programs over $\mathbb{N}$ with the μ-recursive functions was implicit already in [8], and explicit in [13]. Their equivalence with λ-definability [3, 14] and hence with Turing computability [35] followed quickly. When equational programs are used over infinite data, a match with Turing machines must be based on an adequate representation of infinite data by functions over inductive data. For instance, each infinite 0/1 word w can be identified with the function $\hat{w} : \mathbb{N} \to \mathbb{B}$ defined by $\hat{w}(k) =$ the k'th constructor of w. Similarly, infinite binary trees with node decorated with 0/1 can be identified with functions from $\mathbb{W} = \{\mathtt{0}, \mathtt{1}\}^*$ to $\{\mathtt{0}, \mathtt{1}\}$. Conversely, a function $f : \mathbb{N} \to \mathbb{B}$ can be identified with the ω-word $\check{f}$ whose n'th entry is $f(n)$.

It follows that a functional $g : (\mathbb{N} \to \mathbb{B}) \to (\mathbb{N} \to \mathbb{B})$ can be identified with the function $\check{g} : \mathbb{B}^\omega \to \mathbb{B}^\omega$, defined by $\check{g}(w) = (g(\hat{w}))^\vee$. Conversely, a function $h : \mathbb{B}^\omega \to \mathbb{B}^\omega$ can be identified with the functional $\hat{h} : (\mathbb{N} \to \mathbb{B}) \to (\mathbb{N} \to \mathbb{B})$ defined by $\hat{h}(f) = (h(\check{f}))^\wedge$.

We state without proof the straightforward, albeit tedious, observation that the two notions are equivalent. (The Theorem and its proof are unrelated to other parts of the present paper.)

▶ **Theorem 5.** *A partial function $h : \mathbb{B}^\omega \to \mathbb{B}^\omega$ is computable by an equational program iff the functional $\hat{h}$ is computable by some oracle Turing machine.*

Dually, a functional $g : (\mathbb{N} \to \mathbb{B}) \to (\mathbb{N} \to \mathbb{B})$ is computable by an oracle Turing machine iff the function $\check{g}$ is computable by an equational program.

3 A Canonicity Theorem: operational semantic is equivalent to Tarskian semantic

3.1 Data-correct expansions and the canonical structure

Let $\mathcal{D}$ be a data-system with $\mathcal{C}$ as constructor-set, and $\mathcal{S}$ a structure over a vocabulary that includes all identifiers in $\mathcal{C}$, but not the type-identifiers of $\mathcal{D}$. The *data-correct expansion*[6] of $\mathcal{S}$ is the expansion to the full vocabulary of $\mathcal{D}$, with the data-predicates $\mathbf{D}_{ij}$ interpreted as follows. (Recall that the interpretation of the constructors is already given in $\mathcal{S}$.)

[6] As usual, we say that a structure $\mathcal{S}$ is an expansion of a structure $\mathcal{Q}$ if $\mathcal{S}$ differs from $\mathcal{Q}$ only in interpreting additional vocabulary identifiers. E.g. $\mathbb{N}$ with addition and multiplication is an expansion of $\mathbb{N}$ with addition only.

1. If $\vec{\mathbf{D}}_i$ is inductive, then the sets $[\![\mathbf{D}_{ij}]\!]$ are obtained from $[\![\vec{\mathbf{D}}_1]\!]$ ... $[\![\vec{\mathbf{D}}_{i-1}]\!]$ by the data composition rules for $\vec{\mathbf{D}}_i$. That is, each inductive bundle is interpreted as the minimal subsets of the $R_{\mathcal{C}}$ closed under the bundle's composition rules.
2. Dually, if $\vec{\mathbf{D}}_i$ is coinductive, then $[\![\mathbf{D}_{ij}]\!]$ are the sets of finite and infinite terms obtained from $[\![\vec{\mathbf{D}}_1]\!]$ ... $[\![\vec{\mathbf{D}}_{i-1}]\!]$ by the decomposition rules for $\vec{\mathbf{D}}_i$. That is, each coinductive data-bundle is interpreted as the maximal vector of subsets of $R_{\mathcal{C}}$ for which the decomposition rules are applicable (i.e. every element is subject to a decomposition rule) and true.

The *canonical model* $\mathcal{A} \equiv \mathcal{A}_{\mathcal{D}} = [\![\mathcal{D}]\!]$ of a data-system $\mathcal{D}$ is the data-correct expansion of the replete structure $\mathcal{R}_{\mathcal{C}}$.

3.2 Typing statements

Suppose $(P, \mathbf{f})$ is a program (unary, say) over $\mathcal{C}$. The program computes a partial function $g : R_{\mathcal{C}} \rightharpoonup R_{\mathcal{C}}$.

▶ **Definition 6.** Given a data-system $\mathcal{D}$ over $\mathcal{C}$, with $\mathbf{D}$ and $\mathbf{E}$ among its data-predicates, we say that g is *of type* $\mathbf{D} \to \mathbf{E}$ if for every $a \in [\![\mathbf{D}]\!]_{\mathcal{A}}$ the function g is defined for input a, and $g(a) \in [\![\mathbf{E}]\!]_{\mathcal{A}}$. We also say in that case that P *is of type* $\mathbf{D} \to \mathbf{E}$. The definition for $(P, \mathbf{f})$ of arity > 1 is similar.

Note that each function, including the constructors, can have multiple types. Also, a program may compute a non-total mapping over $R_{\mathcal{C}}$, and still be of type $\mathbf{D} \to \mathbf{E}$, i.e. compute a total function from type $\mathbf{D}$ to type $\mathbf{E}$. To adequately capture the computational behavior of equational programs, multiple representations of divergence might be necessary; see [18] for examples and discussion.

The partiality of computable functions is commonly addressed either by allowing partial structures [16, 1, 23], or by considering semantic domains, with an object $\perp$ denoting divergence. The approach here is based instead on the "global" behavior of programs in all structures.

When a function $f : R_{\mathcal{C}} \to R_{\mathcal{C}}$ fails to be of a type $\mathbf{D} \to \mathbf{E}$ then the restriction of f to $\mathbf{D}$ is a partial function. That is, values $f(a) \in R_{\mathcal{C}} - [\![\mathbf{E}]\!]$ correspond to the divergence of the program for input $a \in R_{\mathcal{C}}$.

3.3 Canonicity for inductive data

Definition 1 provides the computational semantics of a program $(P, \mathbf{f})$. But as a set of equations a program can be construed simply as a first-order formula, namely the conjunction of the universal closure of those equations. As such, a program has its Tarskian semantic, referring to arbitrary structures for the vocabulary in hand, that is the constructors and the program-functions used in P. A model of P is then just a structure that satisfies each equation in P.

Herbrand proposed to define the computable functions (over $\mathbb{N}$) as those that are unique solutions of equational programs.[7] It is rather easy to show that every computable function is indeed the unique solution of a set of equations. But the converse fails, as illustrated by the following example.[8] Let $G[x] \equiv \exists y.\, G_0(x, y)$ be undecidable, with G_0 decidable. Clearly

[7] This proposal was made to Gödel in personal communication, and reported in [8]. A modified proposal, incorporating an operational-semantics ingredient, was made in [11].

[8] The first counter-example to Herbrand's proposal is probably due to Kalmar [12]. The example given here is a simplification of an example of Kreisel, quoted in [28].

there is a program for the function f defined by

$$f(x,v) = \begin{cases} 1 & \text{if} \quad \exists y < v.\ G_0(x,y) \\ 2 \cdot f(x, v+1) & \text{otherwise} \end{cases}$$

If, for a given x, $\exists y.\ G_0(x,y)$, then $\lambda v.f(x,v)$ has a unique solution, with $f(x,0) > 0$. Otherwise $f(x,v) = 0$ is the unique solution. So if f were computable, then G would be decidable.

In fact, Herbrand's definition yields precisely the hyper-arithmetical functions [28]. But Herbrand was not far off: he only needed to refer collectively to all data-correct structures:

▶ **Theorem 7.** (Canonicity Theorem for $\mathbb{N}$) [18] *An equational program $(P, \mathbf{f})$ over $\mathbb{N}$ computes a total function iff the formula $\mathtt{N}(x) \to \mathtt{N}(\mathbf{f}(x))$ is true in every data-correct model of P.*

3.4 Canonicity Theorem for Data Systems

We generalize Theorem 7 to all data-systems. Let $\mathcal{D}$ be a data-system for a constructor set $\mathcal{C}$.

▶ **Theorem 8.** *(*Canonicity Theorem for Data Systems) *An equational program $(P, \mathbf{f})$ over $\mathcal{C}$ computes a function $f : \mathbf{D} \to \mathbf{E}$ (using the operational semantics of equational logic) iff the formula $\mathbf{D}(x) \to \mathbf{E}(\mathbf{f}(x))$ is true (in the sense of Tarskian semantic of first-order formulas) in every data-correct model of P.*

We present the proof in the rest of the present subsection. Given a program $(P, \mathbf{f})$ over a data-system $\mathcal{D}$, we construct a canonical model $\mathcal{M}(P)$ to serve as a "test-structure" for the program. Let $\mathcal{C}'$ consist of the program-functions in P, and $\mathbf{T}(P)$ be $\mathcal{R}_{\mathcal{B}}$ for the vocabulary $\mathcal{B} = \mathcal{C} \cup \mathcal{C}'$. Thus the elements of $\mathbf{T}(P)$ are finite and infinite terms built using both the constructors and the program-functions used in P. (Using only terms with a finite number of program-functions along each branch would suffice, but this restriction, albeit natural, is immaterial here.) Let $\approx_P$ be the binary relation over $\mathbf{T}(P)$ that holds between two terms $\mathbf{t}, \mathbf{t}' \in T(P)$ iff $P \vdash^{=} \Pi\mathbf{t} = \Pi\mathbf{t}'$ for every deep destructor Π; that is, the pointwise equality of $\mathbf{t}$ and $\mathbf{t}'$ can be proved from P in equational logic. This is trivially an equivalence relation.

Now define $\mathcal{B}(P)$ to be the structure for the vocabulary $\mathcal{C} \cup \mathcal{C}'$ whose universe is the quotient $\mathbf{T}(P)/\approx_P$, and where each function-identifier is interpreted as symbolic application: a function-identifier $\mathbf{g}$, unary say, maps each equivalence class $[\mathbf{t}]_\approx$ to the equivalence class $[\mathbf{g}(\mathbf{t})]_\approx$; and similarly for arities > 1.

Let $\mathcal{M}(P)$ be the data-correct expansion of $\mathcal{B}(P)$. The following observation implies an alternative, more direct, definition of $\mathcal{M}(P)$.

▶ **Lemma 9.** *Each data-predicate $\mathbf{D}$ is interpreted in $\mathcal{M}(P)$ as* $\{[a]_\approx \mid a \in [\![\mathbf{D}]\!]_{\mathcal{A}}\}$.

Proof. The closure conditions defining the sets $[\![\mathbf{D}]\!]_{\mathcal{M}(P)}$ for data-predicates $\mathbf{D}$ of $\mathcal{D}$ are the same as for the canonical model $\mathcal{A} = \mathcal{R}_{\mathcal{C}}$. ◀

Theorem 8 now follows from the following Lemma.

▶ **Lemma 10.** *The following are equivalent.*
1. *The program $(P, \mathbf{f})$ computes a function $g : \mathbf{D} \to \mathbf{E}$.*
2. *$\mathcal{M}(P) \models \forall x\ \mathbf{D}(x) \to \mathbf{E}(\mathbf{f}(x))$.*
3. *$\mathcal{S} \models \forall x\ \mathbf{D}(x) \to \mathbf{E}(\mathbf{f}(x))$ for every data-correct structure $\mathcal{S}$.*

The equivalence above lifts to arities > 1.

Proof. (1) implies (3) since the equational computation of P over $[\![\mathcal{D}]\!]$ remains correct in every data-correct model of P.

(3) implies (2), since $\mathcal{M}(P)$ is data-correct by definition.

Finally, towards proving that (2) implies (1), assume (2). Let $g : R_{\mathcal{C}} \to R_{\mathcal{C}}$ be the function computed by $(P, \mathbf{f})$, unary say. Taking an input $a \in R_{\mathcal{C}}$ such that $a \in [\![\mathbf{D}]\!]_{\mathcal{A}}$, we have $[a]_{\approx} \in [\![\mathbf{D}]\!]_{\mathcal{M}(P)}$, by Lemma 9. This implies by (2) that $[\mathbf{f}(a)] \in [\![\mathbf{E}]\!]_{\mathcal{M}(P)}$. But by Lemma 9 again, this implies that $\mathbf{f}(a) \approx b$ for some $b \in [\![\mathbf{E}]\!]_{\mathcal{S}}$, establishing (1). ◀

4 Intrinsic theories

4.1 Intrinsic theories for inductive data

Intrinsic theories for inductive data-types were introduced in [18]. They support unobstructed reference to partial functions and to non-denoting terms, common in functional and equational programming. Each intrinsic theory is intended to be a framework for reasoning about the typing properties of programs, including their termination and fairness. In particular, declarative programs are considered as formal theories. This contrasts with two longstanding approaches to reasoning about programs and their termination, namely programs as modal operators [31, 25, 10], and programs (and their computation traces) as explicit mathematical objects [15, 16].

Let $\mathcal{D}$ be a data-system consisting of a single inductive bundle $\vec{\mathbf{D}}$. The *intrinsic theory for $\mathcal{D}$,* is a first order theory over the vocabulary of $\mathcal{D}$, whose axioms are

- The closure rules of $\mathcal{D}$.
- **Inductive delineation (data-elimination, Induction)**, which are the dual of the closure rules. Namely, if a vector $\vec{\varphi}[x]$ of first order formulas satisfies the composition rules for $\vec{\mathbf{D}}$, then it contains $\vec{\mathbf{D}}$:

$$\mathit{Comp}[\vec{\varphi}] \;\to\; \wedge_i \forall x\, \mathbf{D}_i(x) \to \varphi_i[x]$$

  where $\mathit{Comp}[\vec{\varphi}]$ is the conjunction of the composition rules for the bundle, with each $\mathbf{D}_i(\mathbf{t})$ replaced by $\varphi_i[\mathbf{t}]$.
- **Separation Axioms,** stating that every constructor is injective, and $\mathbf{c}(\vec{x}) \neq \mathbf{d}(\vec{y})$ for all distinct constructors $\mathbf{c}$ and $\mathbf{d}$. These imply that all data-terms are distinct. The Separation Axioms are superfluous for

Examples: $\mathbb{N}$, i.e. $\mathcal{A}(\mathtt{0}, \mathtt{s})$.

The Intrinsic theory for $\mathbb{N}$ has for vocabulary the constructors $\mathtt{0}$ and $\mathtt{s}$, and a unary relation identifier N. The axioms, given as natural deduction rules, are

- Data-introduction:

$$\frac{}{N(\mathtt{0})} \qquad \frac{N(x)}{N(\mathtt{s}x)}$$

- Data-elimination:

$$\frac{N(\mathbf{t}) \quad \varphi[\mathtt{0}] \quad \begin{array}{c}\{\varphi[z]\}\\ \cdots \\ \varphi[\mathtt{s}(z)]\end{array}}{\varphi(\mathbf{t})}$$

Identifying $\mathbb{W} = \{\mathtt{0}, \mathtt{1}\}^*$ with the free algebra generated from the nullary constructor ε and the unary $\mathtt{0}$ and $\mathtt{1}$, the intrinsic theory $\mathbf{IT}(\mathbb{W})$ has as vocabulary these constructors and a unary data-predicate W. The deductive rules are:

- Data-introduction:

$$\frac{}{W(\varepsilon)} \qquad \frac{W(\mathbf{t})}{W(\mathtt{0}(\mathbf{t}))} \qquad \frac{W(\mathbf{t})}{W(\mathtt{1}(\mathbf{t}))}$$

- Data-elimination:

$$\frac{W(\mathbf{t}) \quad \varphi[\varepsilon] \quad \begin{matrix}\{\varphi[z]\}\\ \cdots\\ \varphi[\mathtt{0}(z)]\end{matrix} \quad \begin{matrix}\{\varphi[z]\}\\ \cdots\\ \varphi[\mathtt{1}(z)]\end{matrix}}{\varphi(\mathbf{t})}$$

4.2 Provable typing in intrinsic theories

▶ **Definition 11.** A unary program $(P, \mathbf{f})$ is *provably of type* $\mathbf{D} \to \mathbf{E}$ in a theory $\mathbf{T}$ if $\mathbf{D}(x) \to \mathbf{E}(\mathbf{f}(x)$ is provable in $\mathbf{T}$ from the universal closure of the equations in P.

For example, consider the doubling function $\mathtt{dbl}$ over $\mathbb{N}$ defined by the program $\mathtt{dbl}(\mathtt{0}) = \mathtt{0}$, $\mathtt{dbl}(\mathtt{s}(x)) = \mathtt{s}(\mathtt{s}(\mathtt{dbl}(x)))$. The following is a proof of $\mathtt{N}(x) \to \mathtt{N}(\mathtt{dbl}(x))$, using induction on the predicate $\varphi[z] \equiv \mathtt{N}(\mathtt{dbl}(z))$. The double-bars indicate the omission of trivial steps.

$$\dfrac{\dfrac{\mathtt{N}(\mathtt{0})}{\mathtt{N}(\mathtt{dbl}(\mathtt{0}))} \qquad \dfrac{\dfrac{\forall x\ \mathtt{dbl}(\mathtt{s}(x)) = \mathtt{s}(\mathtt{s}(\mathtt{dbl}(x)))}{\mathtt{dbl}(\mathtt{s}(z)) = \mathtt{s}(\mathtt{s}(\mathtt{dbl}(z)))} \qquad \dfrac{\mathtt{N}(\mathtt{dbl}(z))}{\mathtt{N}(\mathtt{s}(\mathtt{s}(\mathtt{dbl}(z))))}}{\mathtt{N}(\mathtt{dbl}(\mathtt{s}(z)))}}{\mathtt{N}(\mathtt{dbl}(x))}$$

In fact, we have:

▶ **Theorem 12.** [18]. *The provable programs of the intrinsic theory* $\mathbf{IT}(\mathbb{N})$ *for the natural numbers are precisely the provably-recursive programs of Peano's Arithmetic.*

Note that Theorem 12 gives a characterization of the provable functions of PA without involving any particular choice of base functions (such as additional and multiplication).

4.3 Intrinsic theories for arbitrary data-systems

Let $\mathcal{D}$ be a data-system. The *intrinsic theory for* $\mathcal{D}$, denoted $\mathbf{IT}(\mathcal{D})$, is a first order theory over the vocabulary of $\mathcal{D}$, whose axioms are the inductive composition rule and coinductive decomposition rules of $\mathcal{D}$, as well as their duals:

- **Inductive delineation (data-elimination, Induction)**: If a vector $\vec{\varphi}[x]$ of first order formulas satisfies the composition rules for an inductive bundle $\vec{\mathbf{D}}$, then it contains $\vec{\mathbf{D}}$:

$$\mathit{Comp}[\vec{\varphi}] \ \to\ \wedge_i \forall x\ \mathbf{D}_i(x) \to \varphi_i[x]$$

  where $\mathit{Comp}[\vec{\varphi}]$ is the conjunction of the composition rules for the bundle, with each $\mathbf{D}_i(\mathbf{t})$ replaced by $\varphi_i[\mathbf{t}]$.
- **Coinductive delineation (data-introduction, Coinduction)**: If a vector $\vec{\varphi}[x]$ of first order formulas satisfies the decomposition rule for a coinductive bundle $\vec{\mathbf{D}}$, then it is contained in $\vec{\mathbf{D}}$:

$$\mathit{Dec}[\vec{\varphi}] \ \to\ \wedge_i \forall x\ \varphi_i[x] \to \mathbf{D}_i(x)$$

  where $\mathit{Dec}[\vec{\varphi}]$ is the conjunction of the decomposition rules for the bundle, with each $\mathbf{D}_i(\mathbf{t})$ replaced by $\varphi_i[\mathbf{t}]$.
- *Separation* axioms, stating that every constructor is injective, and $\mathbf{c}(\vec{x}) \neq \mathbf{d}(\vec{y})$ for all distinct constructors $\mathbf{c}$ and $\mathbf{d}$.

5 Proof theoretic strength

Our general intrinsic theories refer to infinite basic objects (coinductive data), in contrast to intrinsic theories for inductive data only, as well as traditional arithmetical theories. However, the deductive machinery itself does not imply the existence of any particular coinductive object, as would be the case, for example, in the presence of some form of the Axiom of Choice or of a comprehension principle. As a consequence, any intrinsic theory, merging inductive and coinductive constructions in any way, is interpretable in a formal theory which proof theoretically is no stronger than Peano Arithmetic.

Consider the formalism **PRA** of Primitive Recursive Arithmetic, with function identifiers for all primitive recursive functions, and their defining equations as axioms. In addition, we have the Separation Axioms for $\mathbb{N}$ (as above), and the schema of Induction for all formulas.[9] Let $\mathbf{PRA}^*$ be **PRA** augmented with function variables and quantifiers over them. The schema of Induction applies now to all formulas in the extended language, but otherwise there are no axioms stipulating the existence of additional functions. It is well known that **PRA** is interpretable in Peano's Arithmetic (where only addition and multiplication are given as functions with their defining equations).

▶ **Lemma 13.** *The theory* $\mathbf{PRA}^*$ *is conservative over* **PRA**. *That is, if a formula in the language of* **PRA** *is provable in* $\mathbf{PRA}^*$*, then it is provable already in* **PRA**.

The proof is a simplified version of the proof in [34, §2.4.8] that the hereditarily recursive operators form a model of arithmetic in all finite types. Here it suffices to observe that the function quantifiers in $\mathbf{PRA}^*$ can be interpreted as ranging over the computable (or even the PR) functions.

Lemma 13 implies, in particular, that $\mathbf{PRA}^*$ is not proof-theoretically stronger than **PA**.

▶ **Theorem 14.** (Arithmetic interpretability) *Every intrinsic theory is interpretable in* $\mathbf{PRA}^*$.

Proof. Each $t \in R_{\mathcal{C}}$ can be represented by a function $f_t : \mathbb{N} \to \mathbb{N}$, that maps addresses $a = \langle b_1 \cdots b_k \rangle \in \mathbb{N}$ to the code $\mathbf{c}^\sharp$ of the constructor $\mathbf{c}$ at address $\langle b_1 \dots b_k \rangle$ of t, if such a constructor exists, and to a reserved code (0 say) if t has no constructor at address a. For instance, if $t = \mathtt{p}(\mathtt{e}, \mathtt{0}(\mathtt{e}))$ then $f_t\langle\rangle = \mathtt{p}^\sharp$, $f_t\langle 0\rangle = \mathtt{e}^\sharp$, $f_t\langle 1\rangle = \mathtt{0}^\sharp$, $f_t\langle 1, 0\rangle = \mathtt{e}^\sharp$, and $f_t(a) = 0$ for every other address a.

Suppose $\mathcal{D}$ is a data-system over $R_{\mathcal{C}}$, with successive bundles $\vec{\mathbf{D}}_1$, ..., $\vec{\mathbf{D}}_k$. If $\vec{\mathbf{D}}_1 = \langle \mathbf{D}_{11} \dots \mathbf{D}_{1\ell} \rangle$ is inductive, then we can define Σ^0_1 formulas $D_1 \dots D_\ell$, with a single free unary-function variable f, such that $D_i[f]$ is true just in case f codes a tree $t \in R_{\mathcal{C}}$ which is in $\mathbf{D}_i$.

If $\vec{\mathbf{D}}_1$ is coinductive, then the same holds with the formulas D_i taken to be Π^0_1. Next, we can define formulas $\vec{D}_2$ in the second level of the arithmetical hierarchy, with a free unary-function variable f, which are true of f iff it codes some $t \in R_{\mathcal{C}}$ in the bundle $\vec{\mathbf{D}}_2$. Thus, the entire data-system can be interpreted in $\mathbf{PRA}^*$ by formulas of some level $\leq k$ in the arithmetical hierarchy.

We can then define, for each (first-order) formula φ of an intrinsic theory $\mathbf{T}$, a formula $\varphi^\star$ of $\mathbf{PRA}^*$, such that φ is true in the canonical model of the data-system iff $\varphi^\star$ is true in the standard model of **PRA**.

[9] See e.g. [33] for details and related discussions.

Next we show that if φ is provable in the intrinsic theory, then $\varphi^\star$ is provable in $\mathbf{PRA}^*$. Indeed, it is easy to observe that induction for inductive data can be proved, for the $\star$-interpreted formulas, by induction. More interestingly, coinduction for coinductive data is also provable, for the interpretable formulas, by induction. For example, consider coinduction for the binary ω-words, represented by the data-predicate $\mathtt{W}$:

$$\frac{\varphi[\mathbf{t}] \qquad \forall x\ \varphi[x] \to \exists y\ \varphi[y] \wedge (x = \mathtt{0}(y) \vee x = \mathtt{1}(y))}{\mathtt{W}(\mathbf{t})}$$

For the Π^0_1 formula W interpreting $\mathtt{W}$, we need to prove $W[\mathbf{t}^\star]$ from the formula

$$\forall f\ (\ \varphi^\star[f] \to \exists g\ \varphi^\star[g] \wedge (f = \langle \mathtt{0}\rangle * g\ \vee\ f = \langle \mathtt{1}\rangle * g)\ ) \tag{2}$$

(Here $\langle u\rangle * g$ is the function that maps the root to u and an address $0^{\ell+1}$ to $g(0^\ell)$.) But recall that $W[\mathbf{t}^\star]$ states for every address a that the function denoted by $\mathbf{t}^\star$, has at each address a a certain (trivial) local property. This can now be proved by induction on the height n of a, using (2). The induction basis needs only the value of the function $\mathbf{t}^\star$ at the root, which is given by (2). The induction's step is similar.

Note that although we refer here to the iterated tails of $\mathbf{t}^\star$, thus seemingly to infinitely many functions, any function h among these can be referred to indirectly via $\exists u\ h = u * \mathbf{t}^\star$. ◀

6 Applications and further developments

Intrinsic theories provide a minimalist framework for reasoning about data and computation. The benefits were already evident when dealing with inductive data only, including a characterization of the provable functions of Peano's Arithmetic without singling out any functions beyond the constructors, a particularly simple proof of Kreisel's Theorem that classical arithmetic is Π^0_2-conservative over intuitionistic arithmetic [18], and a particularly simple characterization of the primitive-recursive functions [19]. The latter application guided a dual characterization of the primitive corecursive functions in terms of intrinsic theories with positive coinduction [20].

Intrinsic theories are also related to type theories, via Curry-Howard morphisms, providing an attractive framework for extraction of computational contents from proofs, using functional interpretations and realizability methods. The natural extension of the framework to coinductive methods, described here, suggests new directions in extracting such methods for coinductive data as well.

Finally, intrinsic theories are naturally amenable to ramification, leading to a transparent Curry-Howard link with ramified recurrence [2, 17] as well as ramified corecurrence [26].

References

1 Egidio Astesiano, Michel Bidoit, Hélène Kirchner, Bernd Krieg-Brückner, Peter D. Mosses, Donald Sannella, and Andrzej Tarlecki. CASL: the common algebraic specification language. *Theor. Comput. Sci.*, 286(2):153–196, 2002.

2 Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the poly-time functions, 1992.

3 Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

4 Alonzo Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5:56—68, 1940.

5 Haskell Curry. First properties of functionality in combinatory logic. *Tohoku Mathematical Journal*, 41:371–401, 1936.

6 H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, Berlin, 1995.

7 Ronald Fagin. Generalized first order spectra and polynomial time recognizable sets. In R. Karp, editor, *Complexity of Computation*, pages 43–73. SIAM-AMS, 1974.

8 Kurt Gödel. On undecidable propositions of formal mathematical systems. In Martin Davis, editor, *The Undecidable*. Raven, New York, 1965. Lecture notes taken by Kleene and Rosser at the Institute for Advanced Study, 1934.

9 Yuri Gurevich. Logic and the challenge of computer science. In *trends in theoretical computer science*, pages 1–57. Computer Science Press, Rockville, MD, 1988.

10 David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, Cabridge, MA, 2000.

11 Jacque Herbrand. Sur la non-contradiction de l'arithmétique. *Journal für die reine und angewandte Mathematik*, 1932:1—–8, 1932. English translation in [36] 618–628.

12 László Kalmár. Über ein Problem betreffend die Definition des Begriffes des allgemeine-rekursiven Funktion. *Zeit. mathematische Logik u Grund. der Mathematik*, 1:93–96, 1955.

13 Stephen C. Kleene. General recursive functions of natural numbers. *Mathematische annalen*, 112:727–742, 1936.

14 Stephen C. Kleene. Lambda definability and recursiveness. *Duke Mathematical Journal*, 2:340–353, 1936.

15 Stephen C. Kleene. *Introduction to Metamathematics*. Wolters-Noordhof, Groningen, 1952.

16 Stephen C. Kleene. *Formalized Recursive Functions and Formalized Realizability*, volume 89 of *Memoirs of the AMS*. American Mathematical Society, Providence, 1969.

17 Daniel Leivant. Ramified recurrence and computational complexity I: Word recurrence and poly-time. In Peter Clote and Jeffrey Remmel, editors, *Feasible Mathematics II*, Perspectives in Computer Science, pages 320–343. Birkhauser-Boston, New York, 1994. www.cs.indiana.edu/∼leivant/papers.

18 Daniel Leivant. Intrinsic reasoning about functional programs I: First order theories. *Annals of Pure and Applied Logic*, 114:117–153, 2002.

19 Daniel Leivant. Intrinsic reasoning about functional programs II: Unipolar induction and primitive-recursion. *Theor. Comput. Sci.*, 318(1-2):181–196, 2004.

20 Daniel Leivant and Ramyaa Ramyaa. Implicit complexity for coinductive data: a characterization of corecurrence. In Jean-Yves Marion, editor, *DICE*, volume 75 of *EPTCS*, pages 1–14, 2011.

21 Yiannis N. Moschovakis. The formal language of recursion. *J. Symb. Log.*, 54(4):1216–1252, 1989.

22 Till Mossakowski, Lutz Schröder, Markus Roggenbach, and Horst Reichel. Algebraic-coalgebraic specification in CoCasl. *J. Log. Algebr. Program.*, 67(1-2):146–197, 2006.

23 Peter D. Mosses. *CASL Reference Manual, The Complete Documentation of the Common Algebraic Specification Language*, volume 2960 of *Lecture Notes in Computer Science*. Springer, 2004.

24 Peter Padawitz. Swinging types=functions+relations+transition systems. *Theor. Comput. Sci.*, 243(1-2):93–165, 2000.

25 Vaughan R. Pratt. Semantical considerations on floyd-hoare logic. In *FOCS*, pages 109–121. IEEE Computer Society, 1976.

26 Ramyaa Ramyaa and Daniel Leivant. Ramified corecurrence and logspace. *Electr. Notes Theor. Comput. Sci.*, 276:247–261, 2011.

27 Horst Reichel. A uniform model theory for the specification of data and process types. In Didier Bert, Christine Choppy, and Peter D. Mosses, editors, *WADT*, volume 1827 of *Lecture Notes in Computer Science*, pages 348–365. Springer, 1999.

28 H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967.

29 Jan Rothe, Hendrik Tews, and Bart Jacobs. The coalgebraic class specification language CCSL. *J. UCS*, 7(2):175–193, 2001.

30 Lutz Schröder. Bootstrapping inductive and coinductive types in HasCASL. *Logical Methods in Computer Science*, 4(4), 2008.

31 Krister Segerberg. A completeness theorem in the modal logic of programs (preliminary report). *Notices American matheamtical society*, 24:A–552, 1977.

32 Jonathan Seldin. Curry's anticipation of the types used in programming languages. In *Proceedings of the Annual Meeting of the Canadian Society for History and Philosophy of Mathematics*, pages 143–163, Toronto, 2002.

33 S. Simpson. *Subsystems of Second-Order Arithmetic*. Springer-Verlag, Berlin, 1999.

34 A. S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*. Volume 344 of LNM. Springer-Verlag, Berlin, 1973.

35 Alan M. Turing. Computability and lambda-definability. *Journal of Symbolic Logic*, 2:153–163, 1937.

36 J. van Heijenoort. *From Frege to Gödel, A Source Book in Mathematical Logic*. Harvard University Press, Cambridge, MA, 1967.

Two-Variable Logic on 2-Dimensional Structures

Amaldev Manuel[1] and Thomas Zeume[2]

1 **LIAFA, Université Paris Diderot**
amal@liafa.univ-paris-diderot.fr
2 **TU Dortmund University**
thomas.zeume@cs.tu-dortmund.de

Abstract

This paper continues the study of the two-variable fragment of first-order logic (FO^2) over two-dimensional structures, more precisely structures with two orders, their induced successor relations and arbitrarily many unary relations. Our main focus is on ordered data words which are finite sequences from the set $\Sigma \times \mathcal{D}$ where Σ is a finite alphabet and $\mathcal{D}$ is an ordered domain. These are naturally represented as labelled finite sets with a linear order $\leq_l$ and a total preorder $\leq_p$.

We introduce ordered data automata, an automaton model for ordered data words. An ordered data automaton is a composition of a finite state transducer and a finite state automaton over the product Boolean algebra of finite and cofinite subsets of $\mathbb{N}$. We show that ordered data automata are equivalent to the closure of $FO^2(+1_l, \leq_p, +1_p)$ under existential quantification of unary relations. Using this automaton model we prove that the finite satisfiability problem for this logic is decidable on structures where the $\leq_p$-equivalence classes are of bounded size. As a corollary, we obtain that finite satisfiability of FO^2 is decidable (and it is equivalent to the reachability problem of vector addition systems) on structures with two linear order successors and a linear order corresponding to one of the successors. Further we prove undecidability of FO^2 on several other two-dimensional structures.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases FO^2, Data words, Satisfiability, Decidability, Automata

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.484

1 Introduction

The undecidability of the satisfiability and finite satisfiability problem for first-order logic [6, 32, 31] lead to a quest for decidable yet expressive fragments (see for example [3, 15]).

Here we continue the study of the two-variable fragment of first order logic (two-variable logic or FO^2 for short). This fragment is known to be reasonably expressive and its satisfiability and finite satisfiability problems are decidable [25], in fact they are complete for NExpTime [11]. Unfortunately many important properties as for example transitivity cannot be expressed in two-variable logic. This shortcoming led to an examination of extensions of two-variable logic by special relation symbols that are interpreted as equivalence relations or orders [26, 1, 19, 20, 18, 28, 30].

In this paper we are interested in extensions of two-variable logics by two orders and their induced successors. This can be seen as two-variable logic on 2-dimensional structures. We restrict our attention to linear orders and preorders[1]. This setting yields some interesting applications.

[1] Informally, a *preorder* is an equivalence relation whose equivalence classes are ordered by a linear order.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 484–499

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Data words, introduced in [4], extend usual words by assigning data values to every position. Applications of data words arise for example in verification, where they can be used for modeling runs of infinite state systems, and in database theory, where XML trees can be modeled by data trees. Data words with a linearly ordered data domain can be seen as finite structures with a linear order on the positions and a preorder on the positions induced by the linear order of the data domain. Those relations, as well as their induced successor relations, can then be referred to by two-variable logic on data words [1].

Two other logics closely related to two-dimensional two-variable logic are *compass logic* and *interval temporal logic*. In compass logic two-dimensional temporal operators allow for moving north, south, east and west along a grid [33]. In interval temporal logic operators like 'after', 'during' and 'begins' allow for moving along intervals [14]. The connection of intervals to the two-dimensional setting becomes clear when one interprets an interval $[a, b]$ as point (a, b). In [27] decidability results for two-variable logic in the two-dimensional setting have been transferred to those two logics.

Those applications motivate working towards a thorough understanding of 2-dimensional two-variable logic in general, and the decidability frontier for the finite satisfiability problem in this setting in particular. Next we discuss the state-of-the-art in this area and how our results fit in. All those results are summarized in Figure 3.

The frontier for decidability of the finite satisfiability problem for the extension of two-variable logic by two linear order relations and their induced successor relations is well-understood. It is undecidable when all those relations can be accessed by the logic. It is decidable when only the two successor relations can be accessed [23]. This paper contains a gap (the reduction to Presburger automata is wrong) which can, however, be fixed using the same technique. In [10] an optimal decision procedure is given that uses a different approach; and more recently the result has been generalized to two-variable logic with counting on structures with two trees using yet another approach [5]. When two linear orders and one of their successors can be accessed the problem is decidable as well [27]. We prove that the remaining open case of two successors and one corresponding linear order is decidable.

The addition of two preorders to two-variable logic yields an undecidable finite satisfiability problem [27]. We prove that also the other cases, that is (1) adding two preorder successor relations and (2) adding one preorder relation and one (possibly non-corresponding) preorder successor yield an undecidable finite satisfiability problems.

For the extension of two-variable logic with one linear order, one preorder and their induced successors the picture is not that clear. However, many of the results from above translate immediately, because in two-variable logic one can express that a preorder relation is a linear order. Besides those inherited results the following is known for the finite satisfiability problem. If the access is restricted to one linear order as well as a preorder and its successor, then it is decidable in ExpSpace [27]. Access to a linear order with its successor and either preorder or preorder successor yields undecidability. The former is proved in [2], the latter is an easy adaption. The only remaining open case is when one linear successor, one preorder successor and (possibly) the corresponding preorder can be accessed. We attack this case, and show that when the preorder is restricted to have equivalence classes of bounded size, then the finite satisfiability problem is decidable. The general case was shown to be undecidable after the submission of this work, see Section 7.

Contributions. Besides the above mentioned results, we contribute as follows:

- We introduce *ordered data automata*, an automaton model for structures with one successor relation (of an underlying linear order) and a preorder and its accompanying successor

relation. This model is an adaption of data automata, introduced in [2], to data words with an ordered data domain.
- Ordered data automata are shown to be equivalent to the existential two-variable fragment of monadic second order logic (EMSO^2) over such structures.
- We prove that the emptiness problem for this automaton model is decidable, when the equivalence classes of the preorder contain a bounded number of elements. The decidability of the finite satisfiability problem of two-variable logic over structures with two linear successor relations and one of their corresponding orders is a corollary.

Organization. After some basic definitions in Section 2, we introduce ordered data automata in Section 3 and prove that they are expressively equivalent to $\mathrm{EMSO}^2(+1_l, +1_p, \leq_p)$ in Section 4. Section 5 is devoted to proving decidability of the emptiness problem for ordered data automata when the equivalence classes of $\leq_p$ are bounded. In Section 6 lower bounds for several variants are proved. We conclude with a discussion of recent developments as well as open problems in Section 7. Due to the space limit, most proofs will only be available in the full version of the paper.

Acknowledgements. We thank Thomas Schwentick for introducing us to two-variable logic and for many helpful discussions. The first author was supported by funding from European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 259454. The second author acknowledges the financial support by the German DFG under grant SCHW 678/6-1.

2 Preliminaries

We denote the set $\{0, 1, \ldots\}$ of natural numbers by $\mathbb{N}$ and $\{1, \ldots, n\}$, for $n \in \mathbb{N}$ by $[n]$.

A binary relation $\leq_p$ over a finite set A is a *preorder*[2] if it is reflexive, transitive and total, that is, if for all elements u,v and w from A (i) $u \leq_p u$ (ii) $u \leq_p v$ and $v \leq_p w$ implies $u \leq_p w$ and (iii) $u \leq_p v$ or $v \leq_p u$ holds. A *linear order* $\leq_l$ on A is an antisymmetric total preorder, that is, if $u \leq_l v$ and $v \leq_l u$ then $u = v$. Thus, the essential difference between a total preorder and a linear order is that the former allows for two distinct elements u and v that both $u \leq_p v$ and $v \leq_p u$ hold. We call two such elements *equivalent with respect to* $\leq_p$ and denote this by $u \sim_p v$. Hence, a total preorder can be seen as an equivalence relation $\sim_p$ whose equivalence classes are linearly ordered by a linear order. Clearly, every linear order is a total preorder with equivalence classes of size one. We write $u <_l v$ if $u \leq_l v$ but not $v \leq_l u$, analogously for a preorder order $\leq_p$. Further, if C and C' are the equivalence classes of u and v, respectively, then we write $C \leq_p C'$ if $u \leq_p v$.

For a linear order $\leq_l$ an induced *successor relation* $+1_l$ can be defined in the usual way, namely by letting $+1_l(u, v)$ if and only if $u <_l v$ and there is no w with $u <_l w <_l v$. Similarly a preorder $\leq_p$ induces a successor relation $+1_p$ based on the linear order on its equivalence classes, i.e. $+1_p(u, v)$ if and only if $u <_p v$ and there is no w with $u <_p w <_p v$. Thus an element can have several successor elements in $+1_p$.

Two elements u and v are called $\leq_p$*-close* (alternatively $+1_p$*-close*), if either $+1_p(u, v)$ or $u \sim_p v$ or $+1_p(v, u)$. They are called $\leq_p$*-adjacent* (alternatively $+1_p$*-adjacent*) if they are $\leq_p$-close but $u \sim_p v$ does not hold. Analogously for $+1_l$-close, $\leq_l$-close, $+1_l$-adjacent and $\leq_l$-adjacent. The elements u and v are far away with respect to $\leq_p$ if they are not $\leq_p$-close etc. By $u \ll_p v$ we denote that u and v are $\leq_p$-far away and $u \leq_p v$.

[2] In this paper all preorders are total.

In this paper, linear orders and their induced successor relations will be denoted by $\leq_l, \leq_{l_1}, \leq_{l_2}, \ldots$ and $+1_l, +1_{l_1}, +1_{l_2}, \ldots$. Analogously preorders and their induced successor relations will be denoted by $\leq_p, \leq_{p_1}, \leq_{p_2}, \ldots$ and $+1_p, +1_{p_1}, +1_{p_2}, \ldots$.

Ordered Structures, Words and Preorder Words. In this article, an *ordered structure* is a finite structure with non-empty universe and some linear orders, some total preorders, some successor relations and some unary relations. An O-structure is a structure with some unary relations and some binary relations indicated by O. For example, a $(+1_l, +1_p, \leq_p)$-structure has some unary relations and a linear order, a preorder successor and its corresponding preorder. An O-structure is a structure from $\mathrm{FinOrd}(O)$.

A *word* w over an alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ is a finite sequence $\tau_1 \ldots \tau_n$ of letters from Σ. One can think of w as a linear order over $[n]$ where each element i is labeled by letter τ_i from Σ. Thus there is a natural correspondence between words and $\leq_l$-structures (or, alternatively, $+1_l$-structures or $(+1_l, \leq_l)$-structures). Also every $+1_l$-structure naturally corresponds to some word.

Note that words over alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ correspond to $+1_l$-structures with unary relations $\mathcal{P} = (P_{\sigma_1}, \ldots, P_{\sigma_k}\}$. On the other hand, $+1_l$-structures with unary relations $\mathcal{P}$ correspond to words over alphabet $2^{\mathcal{P}}$. Here, and in the following, we will ignore this and assume that appropriate alphabets and unary relations are chosen when necessary.

A *preorder word* w is a sequence $\vec{v}_1 \ldots \vec{v}_l$ of tuples from $\mathbb{N}^\Sigma$. A preorder word w can be identified with a preorder $\leq_p$ with Σ-labeled elements where each $\vec{v}_i = (n_{\sigma_1}, \ldots, n_{\sigma_k})$ is identified with one equivalence class C_i of $\leq_p$. The class C_i contains $\sum_j n_{\sigma_j}$ many elements and n_{σ_j} of those elements are labeled σ_j. Thus a preorder word can be thought of as a word where every position can contain several elements (as opposed to one element in usual words). The identification of tuples with equivalence classes allows for reusing notions for preorders in the context of preorder words, by thinking of $\vec{v}_i$ as an equivalence class. For example, we will say say that $\vec{v}_i$ contains a σ_i-labeled element u, if $n_{\sigma_i} > 0$. Note that there is a natural correspondence between preorder words and ordered $+1_p$-structures (or, alternatively, $\leq_p$-structures or $(+1_p, \leq_p)$-structures).

Ordered Data Words. Fix a finite alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ and an infinite set $\mathbb{D}$ of *data values* (the *data domain*) which is totally ordered by a linear order $\leq_l^{\mathbb{D}}$. For the purpose of this paper, it is sufficient to think of $\mathbb{D}$ as the set $\mathbb{N}$ of natural numbers and of $\leq_l^{\mathbb{D}}$ as the natural order on $\mathbb{N}$.

An *ordered data word* w is a sequence of pairs from $\Sigma \times \mathbb{D}$. We introduce some important notions for ordered data words. In the following fix an ordered data word $w = (\sigma_1, d_1) \ldots (\sigma_n, d_n)$. A preorder $\leq_p$ on $[n]$ is induced by the data values of w by $i \leq_p j$ if $d_i \leq_l^{\mathbb{D}} d_j$. A *class* of w is an equivalence class of $\leq_p$, i.e. a maximal subset $C \subseteq [n]$ of positions of w such that $d_i = d_j$ for all $i, j \in C$. Let, in the following, $C_1 \leq_p \ldots \leq_p C_l$ be the classes of w. The *string projection* of w is the word $\sigma_1 \ldots \sigma_n$ over Σ and is denoted by $sp(w)$. The *preorder projection* $pp(w)$ is the preorder word that corresponds to $\leq_p$, that is $pp(w) = \vec{c}_1 \ldots \vec{c}_l$ where each $\vec{c}_i = (n_{\sigma_1}, \ldots, n_{\sigma_k})$ with n_{σ_j} is the number of σ_j-labeled elements in C_i. Ordered data words naturally correspond to $(+1_l, +1_p, \leq_p)$-structures (again with many alternative representations). See Figure 1 for an example.

Two-Variable Logic on Ordered Structures. Existential monadic second order logic EMSO extends predicate logic by existential quantification of unary relations. The two-variable fragment of EMSO, denoted by EMSO^2, contains all EMSO-formulas whose first-order part uses at most two distinct variables x and y. Two-variable logic FO^2 is the restriction of first order logic to formulas with at most two distinct variable x and y.

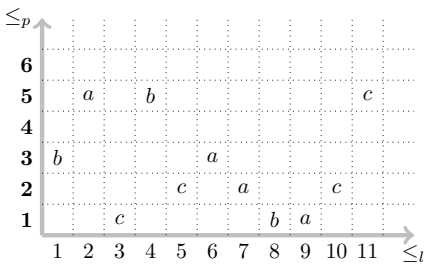


Figure 1 The ordered structure representing the ordered data word $(b,\mathbf{3})(a,\mathbf{5})(c,\mathbf{1})(b,\mathbf{5})(c,\mathbf{2})(a,\mathbf{3})(a,\mathbf{2})(b,\mathbf{1})(a,\mathbf{1})(c,\mathbf{2})(c,\mathbf{5})$. The classes are $\{3,8,9\} \leq_p \{5,7,10\} \leq_p \{1,6\} \leq_p \{2,4,11\}$, the string projection is *bacbcaabacca*, and the preorder projection is $(1,1,1)(1,0,2)(1,1,0)(1,1,1)$ where, e.g., $(1,0,2)$ indicates that in class $\{5,7,10\}$ there is one a-labeled element, no b-labeled element and two c-labeled elements.

Denote by $\mathrm{EMSO}(O)$ existential monadic second order logic over a vocabulary that contains some unary relation symbols and binary relation symbols from O which have to be interpreted by O-structures. For example, formulas in $\mathrm{EMSO}(+1_l)$ can use some unary relation symbols and the binary relation symbol $+1_l$, and $+1_l$ has to be interpreted as a linear successor. Similar notation will be used for FO^2.

Words, that is $+1_l$-structures, can be seen as interpretations for $\mathrm{EMSO}(+1_l)$-formulas. Similarly preorder words and ordered data words are interpretations for $\mathrm{EMSO}(+1_p, \leq_p)$- and $\mathrm{EMSO}(+1_l, +1_p, \leq_p)$-formulas, respectively.

The language $L(\varphi)$ of $\varphi \in \mathrm{EMSO}^2(+1_l)$ is the set of words, more precisely their corresponding $+1_l$-structures, that satisfy φ. Similarly for other sets of relations. The classical theorem of Büchi, Elgot and Trakhtenbrot states that $\mathrm{EMSO}(+1_l, \leq_l)$ is equivalent to finite state automata. This holds even for $\mathrm{EMSO}^2(+1_l)$. In the next section we introduce an automaton model which is equivalent to $\mathrm{EMSO}^2(+1_l, +1_p, \leq_p)$.

▶ **Example 1.** Let L_1 be the language that contains all data words w over $\Sigma = \{a,b\}$ such that the data value of every a-labeled position in w is smaller than the data values of all b-labeled positions. Let L_2 be the language that contains all data words w such that the a-labeled elements with the largest data value are immediately to the left of a b-labeled element. Then the following $\mathrm{EMSO}^2(+1_l, +1_p, \leq_p)$-formulas φ_1 and φ_2 define L_1 and L_2:

$$\varphi_1 = \forall x \forall y \big((a(x) \wedge b(y)) \rightarrow (x \leq_p y \wedge \neg y \leq_p x) \big)$$
$$\varphi_2 = \forall x \Big(\big(a(x) \wedge \neg \exists y (a(y) \wedge (x \leq_p y \wedge \neg y \leq_p x)) \big) \rightarrow \exists y \big(b(y) \wedge +1_l(x,y) \big) \Big)$$

3 An Automaton Model for Ordered Data Words

In this section we introduce ordered data automata, an automaton model for structures with one linear successor relation $+1_l$ (of an underlying linear order $\leq_l$) and one preorder relation $\leq_p$ accompanied by its successor relation $+1_p$. This automaton model is an adaption of data automata as introduced in [2]. In the next section ordered data automata are shown to be equivalent to $\mathrm{EMSO}^2(+1_l, +1_p, \leq_p)$.

Very roughly, ordered data automata process a $(+1_l, \leq_p, +1_p)$-structure by reading it once in linear-order-direction and once in preorder-direction. Therefore an essential part of an ordered data automaton is an automaton capable of reading preorder words. We introduce an automaton model for preorder words first.

Preorder Automata. Roughly speaking, preorder automata are finite state automata that read preorder words $w = \vec{w}_1 \dots \vec{w}_n$. When reading some $\vec{w}_i$, a transition of such an automaton can be applied if the transition matches the current state and the components of $\vec{w}_i$ satisfy interval constraints specified by the transition. We formalize this.

An *interval* $I = (l, r)$ where $l \in \mathbb{N}$ and $r \in \mathbb{N} \cup \{\infty\}$ contains all $i \in \mathbb{N}$ with $l \leq i < r$. A Σ-*constraint* $\vec{c}$ assigns an interval to every $\sigma \in \Sigma$, i.e. it is a tuple from $(\mathbb{N}, \mathbb{N} \cup \{\infty\})^\Sigma$. A tuple $\vec{w} \in \mathbb{N}^\Sigma$ *satisfies* a Σ-constraint $\vec{c}$, if every component n_σ of $\vec{w}$ is in the interval (l, r) asigned to σ by $\vec{c}$.

A *preorder automaton* $\mathcal{A}$ is a tuple $(Q, \Sigma, \Delta, q_I, F)$, where the states Q, the input alphabet Σ, the initial state $q_I \in Q$ and the final states $F \subseteq Q$ are as in usual finite state automata. The transition relation Δ is a finite subset of $Q \times C \times Q$ where C is a set of Σ-constraints.

The semantics is as follows. When p is a state of $\mathcal{A}$ and $\vec{w}$ is a letter from $\mathbb{N}^\Sigma$, then a transition $(p, \vec{c}, q) \in \Delta$ can be applied if $\vec{w}$ satisfies $\vec{c}$. A run of the automaton $\mathcal{A}$ over a word $\vec{w}_1 \dots \vec{w}_n$ is a sequence of transitions $\delta_1 \dots \delta_n$ with $\delta_i = (p_{i-1}, \vec{c}_i, p_i)$ such that δ_i is applicable to $\vec{w}_i$. The run is accepting if $p_0 = q_I$ and $p_n \in F$. The language $L(\mathcal{A})$ accepted by $\mathcal{A}$ is the set of all preorder words with an accepting run of $\mathcal{A}$.

▶ **Example 2.** Let L be the language of preorder words w over $\Sigma = \{a, b\}$ where every letter $\vec{w}_i$ of w contains an a-labeled element and at most two b-labeled elements. The preorder automaton $\mathcal{A}$ with two states s and e, transitions $\{(s, ((1, \infty), (0, 3)), s), (s, ((0, 1), (0, \infty)), e), (s, ((0, \infty), (3, \infty)), e)\}$, initial state s and single finite state s accepts L.

Preorder automata can be seen as a normal form of finite state automata over the product Boolean algebra of finite and cofinite subsets of $\mathbb{N}$ This observation yields immediately:

▶ **Lemma 3.** *Preorder automata are closed under union, intersection, complementation and letter-to-letter projection.*

The Theorem of Büchi, Elgot and Trakhtenbrot translates to preorder automata. The proof is along similar lines.

▶ **Theorem 4.** *For a language L of preorder words, the following statements are equivalent:*

- *There is a preorder automaton that accepts L.*
- *There is an* $\mathrm{EMSO}^2(+1_p)$*-formula that defines L.*

Ordered Data Automata. The *marked string projection* of an ordered data word is its string projection annotated by information about the relationship of data values of adjacent positions. Formally, let $w = (\sigma_1, d_1) \dots (\sigma_n, d_n)$ be an ordered data word. Then the *marking* $m(i) = (m, m')$ of position i is a tuple from $\Sigma_M = \{-\infty, -1, 0, 1, \infty, -\}^2$ and is defined as follows. If $i = 1$ (or $i = n$) then $m = -$ (or $m' = -$). Otherwise let $C_1 \leq_p \dots \leq_p C_r$ be the classes of w. If C_k, C_l and C_s are the classes of d_{i-1}, d_i and d_{i+1}, respectively, then

$$m(i) = \begin{cases} -\infty & \text{if} \quad l > k+1 \\ -1 & \text{if} \quad l = k+1 \\ 0 & \text{if} \quad l = k \\ 1 & \text{if} \quad l = k-1 \\ \infty & \text{if} \quad l < k-1 \end{cases} \qquad m'(i) = \begin{cases} -\infty & \text{if} \quad l > s+1 \\ -1 & \text{if} \quad l = s+1 \\ 0 & \text{if} \quad l = s \\ 1 & \text{if} \quad l = s-1 \\ \infty & \text{if} \quad l < s-1 \end{cases}$$

The marked string projection of w is the string $(\sigma_1, m(1)) \dots (\sigma_n, m(n))$ over $\Sigma \times \Sigma_M$ and is denoted by $msp(w)$.

An *ordered data automata* (short: ODA) $\mathcal{A} = (\mathcal{B}, \mathcal{C})$ over Σ consists of a non-deterministic letter-to-letter finite state transducer (short: string transducer) $\mathcal{B}$ with input alphabet $\Sigma \times \Sigma_M$ and output alphabet Σ', and a preorder automaton $\mathcal{C}$ with input alphabet Σ'.

An ODA $\mathcal{A} = (\mathcal{B}, \mathcal{C})$ works as follows. First, for a given ordered data word w, the transducer $\mathcal{B}$ reads the marked string projection of w. A run ρ_B of the transducer defines a

unique (for each run) new labelling of each position. Let w' be the ordered data word thus obtained from w. Second, the preorder automaton $\mathcal{C}$ runs over the preorder projection of w' yielding a run ρ_C. The run $\rho_{\mathcal{A}} = (\rho_B, \rho_C)$ of $\mathcal{A}$ is accepting, if both ρ_B and ρ_C are accepting. The automaton $\mathcal{A}$ accepts w if there is an accepting run of $\mathcal{A}$ on w. The set of ordered data words accepted by $\mathcal{A}$ is denoted by $\mathcal{L}(\mathcal{A})$.

▶ **Example 5.** The language L_1 from Example 1 can be decided by an ODA $\mathcal{A} = (\mathcal{B}, \mathcal{C})$ with $\Sigma = \Sigma' = \{a, b\}$ as follows. Let w be an ordered data word. The string transducer $\mathcal{B}$ does not relabel any position. Thus the input preorder word of the preorder automaton $\mathcal{C}$ is the preorder projection $\vec{w}_1 \dots \vec{w}_m$ of w. The preorder automaton $\mathcal{C}$ verifies that after the first $\vec{w}_i$ containing an b-labeled element, no a-labeled element occurs in any $\vec{w}_j$ with $j \geq i$.

The language L_2 can be decided by an ODA $\mathcal{A} = (\mathcal{B}, \mathcal{C})$ with $\Sigma = \{a, b\}$ and $\Sigma' = \{a, b\} \times \{0, 1\}$ as follows. Let $w = (\sigma_1, d_1) \dots (\sigma_n, d_n)$ be an ordered data word. The automaton $\mathcal{A}$ processes w as follows. The string transducer $\mathcal{B}$ guesses the a-labeled positions with the largest data value, relabels them with $(a, 1)$ and checks that the following position is b-labeled. All other letters σ are relabeled by $(\sigma, 0)$. Let w' be the ordered data word thus obtained. The input of $\mathcal{C}$ is the preorder projection $\vec{w}'_1 \dots \vec{w}'_m$ of w', and $\mathcal{C}$ verifies that $(a, 1)$-labeled elements occur only in $\vec{w}'_m$.

▶ **Lemma 6.** *Languages accepted by ODA are closed under union, intersection and letter-to-letter projection.*

The following proposition can be proved like Lemma 3 in [23].

▶ **Proposition 1.** *Languages accepted by ODA are not closed under complementation.*

4 Ordered Data Automata and $\mathrm{EMSO}^2(+1_l, +1_p, \leq_p)$ are equivalent

In this section we prove

▶ **Theorem 7.** *For a language L of ordered data words, the following statements are equivalent:*

- *L is accepted by an ordered data automaton.*
- *L is definable in $\mathrm{EMSO}^2(+1_l, +1_p, \leq_p)$.*

This equivalence transfers to the case where the preorder is a linear order (i.e. every equivalence class of the preorder is of size one).

The construction of a formula from an automaton is straightforward. The other direction proceeds by translating a given EMSO^2-formula φ into an equivalent formula in Scott Normal Form, i.e. into a formula of the form $\exists X_1 \dots X_n(\forall x \forall y\ \psi \wedge \bigwedge_i \forall x \exists y\ \chi_i)$ where ψ and χ_i are quantifier-free formulas (see e.g. [12] for the translation). Since ODA are closed under union, intersection and renaming it is sufficient to show that for every formula of the form $\forall x \forall y\ \psi$ and $\forall x \exists y\ \chi$ there is an equivalent ODA.

The proofs of the following lemmas use the abbreviations

$$\begin{aligned} \Delta_= &= \{x = y, x \neq y\}, \\ \Delta_l &= \{+1_l(x, y), \neg{+1_l}(x, y), +1_l(y, x), \neg{+1_l}(y, x)\}, \\ \Delta_p &= \{+1_p(x, y), +1_p(y, x), x \sim_p y, x \ll_p y, y \ll_p x\}. \end{aligned}$$

▶ **Lemma 8.** *For every formula of the form $\forall x \forall y\ \psi$ with quantifier-free ψ there is an equivalent ODA.*

Proof. We first write ψ in conjunctive normal form and distribute the universal quantifier over the conjunction. Therefore, again due to the closure of ODA under intersection, we can restrict our attention to formulas of the form

$$\varphi = \forall x \forall y (\alpha(x) \vee \beta(y) \vee \delta_{=}(x,y) \vee \delta_l(x,y) \vee \delta_p(x,y))$$

where α, β are unary formulas and $\delta_{=}(x,y)$, $\delta_l(x,y)$ and $\delta_p(x,y)$ are as follows. Denote by $\mathsf{Disj}(\Phi)$ the set of disjunctive formulas over a set of formulas Φ. The formulas $\delta_{=}(x,y)$, $\delta_l(x,y)$ and $\delta_p(x,y)$ are in $\mathsf{Disj}(\Delta_{=})$, $\mathsf{Disj}(\Delta_l)$ and $\mathsf{Disj}(\Delta_p)$, respectively. Note that Δ_p contains only positive formulas since negation of any formula in Δ_p can be replaced by a disjunction of formulas from Δ_p.

Without loss of generality we assume that neither $\delta_{=}(x,y)$, $\delta_l(x,y)$ nor $\delta_p(x,y)$ are the empty disjunction. (Assume that $\delta_{=}(x,y) = \bot$, then $\delta_{=}(x,y) \equiv x = y \wedge x \neq y$. Distributing $x = y \wedge x \neq y$ yields two formulas of the required form.)

In the following we do an exhaustive case analysis. If φ is a tautology, then there is an equivalent ODA. Therefore we assume from now on that φ is not a tautology.

When φ is not a tautology then $\delta_{=}$ is either $x \neq y$ or $x = y$. If $\delta_{=}$ is $x \neq y$ then we can write φ as $\forall x \forall y\big((\alpha'(x) \wedge \beta'(y) \wedge x = y) \to \gamma(x,y)\big)$ where α' and β' are the negations of the unary formulas α and β and $\gamma(x,y) = \delta_l(x,y) \vee \delta_p(x,y)$. Substituting $x = y$ in γ yields a formula that is equivalent to True or to False. Thus the property expressed by φ can be checked by the string transducer of an ODA. Hence from now on we assume that $\delta_{=}$ is the formula $x = y$.

The formula δ_l can either contain a negative formula from Δ_l or it does not contain any negative formula. If δ_l contains a negative formula from Δ_l we rewrite φ as

$$\forall x \forall y\big((\alpha'(x) \wedge \beta'(y) \wedge x \neq y \wedge \delta_l'(x,y)) \to \delta_p(x,y)\big)$$

where δ_l' is the negation of δ_l. Since δ_l' is a conjunction that contains a positive formula from Δ_l it is logically equivalent to a positive formula from Δ_l, that is, it is equivalent either to $+1_l(y,x)$ or to $+1_l(x,y)$. In this case the formula φ expresses a regular property over the marked string projection of the structure. Hence it can be seen immediately that the property expressed by φ can be checked by the string transducer of an ODA. Hence from now on we assume that δ_l contains no negative formula from Δ_l.

Then δ_l is either $+1_l(x,y) \vee +1_l(y,x)$ or $+1_l(y,x)$ or $+1_l(y,x)$. In this case we rewrite φ as $\forall x \forall y\big((\alpha'(x) \wedge \beta'(y) \wedge x \neq y \wedge \delta_p'(x,y)) \to \delta_l(x,y)\big)$ where δ_p' is the negation of $\delta_p(x,y)$. As noted before, the conjunction $\delta_p'(x,y)$ can be expressed as a disjunction of formulas from Δ_p. Hence φ is equivalent to $\forall x \forall y\big((\alpha'(x) \wedge \beta'(y) \wedge x \neq y \wedge \delta_p''(x,y)) \to \delta_l(x,y)\big)$ where $\delta_p''(x,y)$ is a disjunction of formulas in Δ_p. Distributing this disjunction yields a formula of the form $\forall x \forall y \bigwedge \big((\alpha'(x) \wedge \beta'(y) \wedge x \neq y \wedge \delta_p'''(x,y)) \to \delta_l(x,y)\big)$ where $\delta_p'''(x,y)$ is a formula from Δ_p.

By distributing the conjunction over the $\forall$-quantifiers and by using the closure of ODA under intersection, it is sufficient to show that there is an equivalent ODA for formulas of the form $\chi = \forall x \forall y\big((\alpha'(x) \wedge \beta'(y) \wedge x \neq y \wedge \delta_p(x,y)) \to \delta_l(x,y)\big)$ where $\delta_p(x,y)$ is a formula from Δ_p and δ_l is positive.

For the following, we assume that δ_l is the formula $+1_l(x,y) \vee +1_l(y,x)$. The cases $\delta_l = +1_l(x,y)$ and $\delta_l = +1_l(y,x)$ are similar. We do a case analysis for $\delta_p(x,y)$.

Let $\delta_p = +1_p(x,y)$. Assume that C_i and C_{i+1} are two adjacent $\leq_p$-classes. Then the formula χ states that whenever C_i contains an α'-labeled element u and C_{i+1} contains a β'-labeled element v, then u and v are adjacent with respect to $\leq_l$. This implies that the number of α'-labeled elements in C_i and β'-labeled elements in C_{i+1} is at most three. Moreover those elements are adjacent in the linear order.

Thus, an ODA verifying this property can be constructed as follows. The string transducer annotates every α'-labeled element u by the number of β'-labeled elements v with $+1_p(u,v)$ and either $+1_l(u,v)$ or $+1_l(v,u)$. Analogously the string transducer annotates every β'-labeled element by the number of adjacent α'-labeled elements in the preceding $\leq_p$-class.

Then the preorder automaton verifies for each $\leq_p$-class C_i and its successor $\leq_p$-class C_{i+1} that either

- C_i contains no α'-labeled elements or C_{i+1} contains no β'-labeled elements, or
- C_i contains an α'-labeled element and C_{i+1} contains a β'-labeled element and
  - C_i and C_{i+1} contain more than three of those elements (then the preorder automaton rejects)
  - C_i and C_{i+1} contain less than three of those elements. Then it checks that those three are adjacent by using the annotation given by the transducer (and accepts or rejects accordingly).

The cases $\delta_p = x \sim_p y$ and $\delta_p = x \ll_p y$ are very similar. ◀

▶ **Lemma 9.** *For every formula of the form $\forall x \exists y\ \chi$ with quantifier-free χ there is an equivalent ODA.*

The proof of Lemma 9 will be presented in the full version of the paper. This completes the proof of Theorem 7.

5 Deciding Emptiness for Ordered Data Automata on k-bounded Ordered Data Words

An ordered data word w is k-bounded if each class of w contains at most k elements. In this case the preorder projection of w is a k-bounded preorder word and can be seen as a word over the finite alphabet $\{0,\dots,k\}^{|\Sigma|}$. Hence an ODA restricted to k-bounded ordered data words can be seen as a composition of a finite state transducer and a finite state automaton. We call such automata *k-bounded ODA*.

Since k-boundedness can be expressed in $\mathrm{EMSO}^2(+1_l, +1_p, \leq_p)$ we can conclude that the result from the previous section carry over to the case of k-bounded ordered data words, i.e. a language $\mathcal{L}$ of k-bounded ordered data words is accepted by a k-bounded ODA if and only if it can be defined by an $\mathrm{EMSO}^2(+1_l, +1_p, \leq_p)$ formula φ.

The rest of this section is devoted to the proof of the following theorem.

▶ **Theorem 10.** *The finite satisfiability problem for $\mathrm{EMSO}^2(+1_l, +1_p, \leq_p)$ on k-bounded data words is decidable.*

▶ **Corollary 11.** *The finite satisfiability problem for $\mathrm{EMSO}^2(+1_{l_1}, +1_{l_2}, \leq_{l_2})$ is decidable.*

This generalizes Theorem 3 from [23], where the finite satisfiability problem of $\mathrm{FO}^2(+1_{l_1}, +1_{l_2})$ was shown to be decidable. We sketch the proof of Theorem 10; a detailed proof will appear in the full paper. By the above remarks it is sufficient to show that the emptiness problem of k-bounded ODA is decidable. We reduce the emptiness problem for k-bounded ODA to the emptiness problem for multicounter automata. The latter is known to be decidable [24, 21]. The idea is as follows. From a k-bounded ODA $\mathcal{A} = (\mathcal{B}, \mathcal{C})$ we will construct a multicounter automaton $\mathcal{M}$ such that $L(\mathcal{A})$ is non-empty if and only if $L(\mathcal{M})$ is non-empty. Intuitively, $\mathcal{M}$ will be constructed such that if $\mathcal{A}$ accepts a k-bounded ordered data word w then $\mathcal{M}$ accepts a word w' which is the preorder projection of w annotated

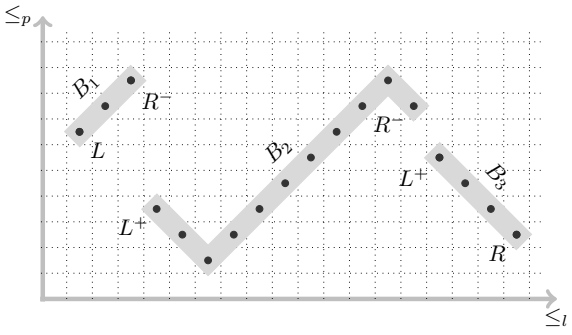


Figure 2 Blocks in the ordered-structure-representation of a 3-bounded ordered data word w. Each $\bullet$ represents one element of w, the $\leq_l$-axis represents positions whereas the $\leq_p$-axis represents data values. Labels are omitted for clarity.

by lots of extra information[3]. On the other hand if $\mathcal{M}$ accepts an annotated word w' then an ordered data word w and an accepting run of $\mathcal{A}$ on w can be reconstructed from the information encoded in w'. Therefore $\mathcal{M}$ reads a k-bounded preorder word $w' = \vec{w}'_1 \dots \vec{w}'_m$ and simultaneously verifies

- that the extra information in w' encodes an accepting run of $\mathcal{C}$ on w'.
- that the elements occuring in w' can be dynamically (that is while reading $\vec{w}'_1, \vec{w}'_2, \dots$) arranged to a word x such that x encodes
  - a marked string y whose marking is consistent with w' (and therefore allows for the construction of an ordered data word w from w' and y), and
  - an accepting run of $\mathcal{B}$ on y.

We will need the following notions for ordered data words. A *block* B of an ordered data word w is a maximal subword of w such that all successive positions in B are $\leq_p$-close in w. See Figure 2 for an example of blocks.

Since w is k-bounded, every class of w intersects with at most k many blocks. It is easy to see, that one can color each block B of w with a number $N(B)$ from $\{1, \dots, 2k\}$ such that $N(B) \neq N(B')$ if B and B' are $\leq_l$-adjacent blocks or $\leq_p$-adjacent blocks. Even more, such a coloring can be uniquely obtained from w (for example by coloring lexicographically).

In the following we describe how to annotate every element of an ordered data word w with extra information. A *block label* (N, X) with *block number* N and *block position* X is a letter from $\Sigma_B = \{1, \dots, 2k\} \times (\{L, L^+, L^-, C\} \times \{R, R^+, R^-, C\})$. Let $\mathcal{A} = (\mathcal{B}, \mathcal{C})$ be a k-bounded ODA with input alphabet Σ, intermediate alphabet Σ' and let $Q_\mathcal{B}$ and $Q_\mathcal{C}$ be the states of $\mathcal{B}$ and $\mathcal{C}$ respectively. A *run label* $(\sigma', r_\mathcal{B}, r_\mathcal{C}, r_B)$ is a letter from $\Sigma_R = \Sigma' \times Q_\mathcal{B}^2 \times Q_\mathcal{C}^2 \times Q_\mathcal{B}^2$ where $r_\mathcal{B}$, $r_\mathcal{C}$ and b_B are called *$\mathcal{B}$-label*, *$\mathcal{C}$-label* and *$\mathcal{B}$-block label*, respectively.

An *annotated ordered data word* is an ordered data word over $\Sigma \times \Sigma_M \times \Sigma_B \times \Sigma_R$ where Σ_M is the alphabet $\{-\infty, -1, 0, 1, \infty, -\}^2$ of markings. Likewise an *annotated preorder word* is a preorder word over $\Sigma \times \Sigma_M \times \Sigma_B \times \Sigma_R$. The preorder projection of an annotated data word is a preorder word over $\Sigma \times \Sigma_M \times \Sigma_B \times \Sigma_R$.

The *annotation* $\mathsf{ann}(w, \rho)$ of an ordered data word $w = w_1 \dots w_n$ with respect to a run $\rho = (\rho_\mathcal{B}, \rho_\mathcal{C})$ of an ODA $\mathcal{A} = (\mathcal{B}, \mathcal{C})$ on w is an annotated ordered data word that labels every element w_i with its marking m; a block label τ according to the position of w_i in its block; and a run label π describing the output of $\mathcal{B}$ on run ρ when reading w_i, the states of $\mathcal{B}$ and $\mathcal{C}$ according to run ρ, and the states where $\mathcal{B}$ enters and leaves the block of w_i in run ρ. The preorder projection of the annotation of an ordered data word w is denoted by $\mathsf{annpp}(w, \rho)$.

[3] Recall that the preorder projection of a k-bounded ordered data word is a k-bounded preorder word, i.e. a word over $\{0, \dots, k\}^{|\Sigma|}$.

Intuitively maximal contiguous subwords of $\mathsf{annpp}(w, \rho)$ with the same block number N correspond to a block in w. Therefore such contiguous subwords of annotated preorder words are called *symbolic N-blocks*.

We now state the proof idea of Theorem 10 more precisely. From an ordered data automaton $\mathcal{A} = (\mathcal{B}, \mathcal{C})$ we construct a multicounter automaton $\mathcal{M}$ that reads annotated k-bounded preorder words such that

- If $\mathcal{A}$ accepts a k-bounded ordered data word w via run ρ then $\mathcal{M}$ accepts $\mathsf{annpp}(w, \rho)$.
- If $\mathcal{M}$ accepts an annotated k-bounded preorder word w' then a k-bounded ordered data word w can be constructed from w' which is accepted by $\mathcal{A}$.

Given an annotated k-bounded preorder word $w' = \vec{w}'_1 \dots \vec{w}'_n$, the multicounter automaton $\mathcal{M}$ tries to reconstruct a k-ordered data word w from w' such that w is accepted by $\mathcal{A}$. Every symbolic block B' in w' will represent a block B in w. We will prove that such a reconstruction is possible whenever the following conditions (C0) – (C3) are satisfied:

(C0) a) The block position label and the label from Σ_M are consistent for every element of w'.
b) Every symbolic block B' of w' contains exactly one $\{L, L^-, L^+\}$-labeled element and one $\{R, R^-, R^+\}$-labeled element.
c) All elements of a letter $\vec{w}'_i$ have the same $\mathcal{C}$-label..
d) The $\mathcal{B}$- and $\mathcal{C}$-labels are consistent with the Σ- and Σ'-labels for every element u of w'.

(C1) The $\mathcal{C}$-labels in w' encode an accepting run of $\mathcal{C}$.

(C2) For every symbolic block $B' = \vec{w}'_l \dots \vec{w}'_m$ of w' there is an annotated ordered data word B with data values from the set $\{l, \dots, m\} \subset \mathbb{N}$ such that
a) B is a single block and $pp(B) = B'$. Further, the data value of an element u of B is d when u corresponds to an element contained in $\vec{w}'_d$ in B'.
b) The first position of B carries block position label L, L^+ or L^-. The last position of B carries block position label R, R^+ or R^-. All other positions carry block position label C.
c) All elements of B' carry the same $\mathcal{B}$-block label (p, q).
d) There is a run of $\mathcal{B}$ on B that starts in p, ends in q and is consistent with the $\mathcal{B}$-labels of B.

(C3) Let $B'_1, \dots, B'_m$ be the symbolic blocks of w'. Further let $\vec{w}'_{s_i}$ be the position of B'_i, that contains[4] the $\{L, L^-, L^+\}$-labeled element l_i of B'_i. Analogously let $\vec{w}'_{t_i}$ be the position of B'_i, that contains the $\{R, R^-, R^+\}$-labeled element r_i of B'_i. There is a permutation π of $\{1, \dots, m\}$ such that
a) If (p, q) is the $\mathcal{B}$-block label of $l_{\pi(1)}$, then p is the start state of $\mathcal{B}$. Further the block position label of $l_{\pi(1)}$ is L.
b) If (p, q) is the $\mathcal{B}$-block label of $r_{\pi(m)}$ then q is a final state of $\mathcal{B}$. Further the block position label of $r_{\pi(m)}$ is R.
c) If (p, q) and (p', q') are the $\mathcal{B}$-block labels of $B'_{\pi(i)}$ and $B'_{\pi(i+1)}$, respectively, then $q = p'$.
d) If r_i is labeled with R^+, then l_{i+1} is labeled with L^-. Further $t_i \ll_p s_{i+1}$.
e) Likewise if r_i is labeled with R^-, then l_{i+1} is labeled with L^+. Further $s_{i+1} \ll_p t_i$.

[4] Recall that $\vec{w}'_{s_i}$ can be identified with the equivalence class of the preorder corresponding to B'_i.

Intuitively, the Conditions (C2) help to reconstruct runs from $\mathcal{C}$. Runs of $\mathcal{B}$ are reconstructed with the help of Conditions (C2) and (C3), where (C2) helps reconstructing runs of $\mathcal{B}$ on blocks whereas (C3) helps reconstructing the order of blocks.

Recall that k-bounded preorder words over Σ can be seen as a word over the finite alphabet $\{0,\dots,k\}^{|\Sigma|}$.

▶ **Lemma 12.** *For every k-bounded ODA $\mathcal{A}$ there is a finite state automaton $\mathcal{M}$ that accepts exactly the annotated k-bounded preorder words that satisfy conditions (C0) and (C1) from above.*

▶ **Lemma 13.** *For every k-bounded ODA $\mathcal{A} = (\mathcal{B},\mathcal{C})$ there is a finite state automaton $\mathcal{M}$ that accepts exactly the annotated k-bounded preorder words that satisfy condition (C2).*

▶ **Lemma 14.** *For every k-bounded ODA $\mathcal{A}$ there is a multicounter automaton $\mathcal{M}$ that accepts exactly the annotated k-bounded preorder words that satisfy conditions (C3).*

Using the previous lemmata we can now complete the proof of Theorem 10.

Proof of Theorem 10. For a given k-bounded ODA $\mathcal{A} = (\mathcal{B},\mathcal{C})$ let $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ be the multicounter automata from Lemmata 12, 13 and 14, respectively. Let M be the intersection multicounter automaton for those three automata.

We prove that $L(\mathcal{A})$ is empty if and only if $L(\mathcal{M})$ is empty. The statement of Theorem 10 follows from this. First, let w be a k-bounded ordered data word accepted by $\mathcal{A}$. Then there is an accepting run $\rho = (\rho_\mathcal{B}, \rho_\mathcal{C})$ of $\mathcal{A}$ on w. The word $w' = \mathsf{annpp}(w,\rho)$ satisfies conditions (C0) – (C3) and is therefore accepted by $\mathcal{M}$ due to Lemmata 12, 13 and 14.

Second, let $w' = \vec{w}'_1 \dots \vec{w}'_m$ be a k-bounded preorder word accepted by $\mathcal{M}$. We construct a k-bounded data word $w \in L(\mathcal{A})$ and an accepting run $\rho = (\rho_\mathcal{B}, \rho_\mathcal{C})$ of $\mathcal{A}$ on w with $\mathsf{annpp}(w,\rho) = w'$. Therefor let $B'_1,\dots,B'_l$ be the symbolic blocks of w'. Condition (C2) guarantees the existence of annotated data words $B_1,\dots,B_l$ with $pp(B_i) = B'_i$ and data values from $\{l_i,\dots,r_i\}$ when $B'_i = w'_{l_i}\dots w'_{r_i}$. By Condition (C2d) there is a run ρ_i for each B_i starting in p_i and ending in q_i where (p_i,q_i) is the $\mathcal{B}$-label of B'_i.

Now let π the permutation from Condition (C3). We define the ordered data word $w = D_{\pi(1)}\dots D_{\pi(l)}$ where $D_{\pi(i)}$ is obtained from $B_{\pi(i)}$ by removing the annotations. Note that the $D_{\pi(i)}$ are blocks by Conditions (C2a), (C2b), (C3d) and (C3e). The concatenation $\rho_\mathcal{B}$ of the runs $\rho_{\pi(1)},\dots,\rho_{\pi(l)}$ is an accepting run of $\mathcal{B}$ on w by Conditions (C3a), (C3b) and (C3c). An accepting run of $\mathcal{C}$ on the output of $\rho_\mathcal{B}$ exists by Condition (C1). ◀

6 Hardness Results for Two-Dimensional Ordered Structures

This section aims at filling the remaining gaps for finite satisfiability of two-variable logic on two-dimensional ordered structures. We refer the reader to Figure 3 for a summary of the results obtained in the literature and here.

We start with a matching lower bound for the finite satisfiability problem of $\mathrm{EMSO}^2(+1_l,+1_p,\leq_p)$ over k-bounded structures. This bound already holds for $\mathrm{FO}^2(+1_{l_1},+1_{l_2},\leq_{l_2})$.

▶ **Theorem 15.** *Finite satisfiability of $\mathrm{FO}^2(+1_{l_1},+1_{l_2},\leq_{l_2})$ is at least as hard as the emptiness problem for multicounter automata.*

▶ **Corollary 16.** *Finite satisfiability of $\mathrm{FO}^2(+1_l,+1_p,\leq_p)$ over k-bounded ordered data words is at least as hard as the emptiness problem for multicounter automata.*

It is not surprising that the finite satisfiability problem of FO^2 with two additional preorder successor relations is undecidable, as those allow for encoding a grid. A minor technical difficulty arises when the corresponding equivalence relations are not available. Undecidability even holds for 2-bounded preorder successor relations.

▶ **Theorem 17.** *Finite satisfiability of two-variable logic with two additional* 2*-bounded preorder successor relations is undecidable.*

We denote the relation $+1_l{}^2$ by $+2_l$. The following slightly improves Theorem 4 in [23].

▶ **Corollary 18.** *Finite satisfiability of* $\mathrm{FO}^2(+1_{l_1}, +2_{l_1}, +1_{l_2}, +2_{l_2})$ *is undecidable.*

The following theorems complement results from [2] and [28]. The proofs use similar methods as used in those works.

▶ **Theorem 19.** *Finite satisfiability of* $\mathrm{FO}^2(+1_l, \leq_l, +1_p)$ *is undecidable.*

▶ **Theorem 20.** *Finite satisfiability of* $\mathrm{FO}^2(+1_{p_1}, \leq_{p_2})$*, i.e. two-variable logic with one additional preorder successor relation and one additional preorder relation, is undecidable.*

7 Discussion

The current status of research on two-variable logic with additional successor and order relations is summarized in Figure 3.

We saw that EMSO^2 with a linear order successor, a k-bounded preorder relation and its induced successor relation is decidable.

After submission of this work, the finite satisfiability problem of $\mathrm{FO}^2(+1_l, +1_p)$ has been shown to be undecidable by Thomas Schwentick and the authors of this work [22], but has not been peer reviewed yet. We strongly conjecture that finite satisfiability for the other remaining open case, namely $\mathrm{FO}^2(+1_l, \leq_p)$, is decidable. We are actually working on the details of the proof and plan to include both results into the full version of this paper.

It remains open whether there is some m such that $\mathrm{FO}^2(+1_{l_1}, \ldots, +1_{l_m})$ is undecidable. A method for proving undecidability of $\mathrm{FO}^2(+1_{l_1}, \ldots, +1_{l_m})$ should not extend to $\mathrm{FO}^2(F_1, \ldots, F_m)$ where $F_1, \ldots, F_m$ are binary predicates that are interpreted as permutations. A successor relation $+1_l$ can be seen as a permutation with only one cycle and one label that marks the first element. Finite satisfiability of $\mathrm{FO}^2(F_1, \ldots, F_m)$ is decidable since one can express that some arbitrary interpreted binary predicate R is a permutation by using two-variable logic with counting quantifiers which in turn is decidable by [13]. This is an observation by Juha Kontinen.

▶ **Open Question 1.** *Is there an* m *such that* $\mathrm{FO}^2(+1_{l_1}, \ldots, +1_{l_m})$ *is undecidable?*

Temporal logics on data words have seen much research recently [7, 8, 16]. However, to the best of our knowledge, most of those logics have been restricted in the sense that comparison of data values was only allowed with respect to equality. In [29] a temporal logic that allows for comparing ordered data values was introduced. The authors intend to use the techniques and results obtained for two-variable logic with additional successors and orders to investigate temporal logics on data values that allow more structure on the data value side.

▶ **Open Question 2.** *Are there expressive but still decidable temporal logics on data words with successor and order relations on the data values?*

[5] Under elementary reductions.

Logic	Complexity (lower/upper)	Comments
One linear order		
$\mathrm{FO}^2(+1_l)$	NExpTime-complete	[9]
$\mathrm{FO}^2(\leq_l)$	NExpTime-complete	[26, 9]
$\mathrm{FO}^2(+1_l, \leq_l)$	NExpTime-complete	[9]
One total preorder		
$\mathrm{FO}^2(+1_p)$	ExpSpace-complete	ExpCorridorTiling
$\mathrm{FO}^2(\leq_p)$	NExpTime/ExpSpace	
$\mathrm{FO}^2(+1_p, \leq_p)$	ExpSpace-complete	[28]
Two linear orders		
$\mathrm{FO}^2(+1_{l_1}, +1_{l_2})$	NExpTime-complete	[23, 10, 5]
$\mathrm{FO}^2(+1_{l_1}, \leq_{l_2})$	NExpTime/ExpSpace	[28]
$\mathrm{FO}^2(+1_{l_1}, +1_{l_2}, \leq_{l_2})$	Multicounter-Emptiness[5]	★, Corollary 11 and Theorem 15
$\mathrm{FO}^2(+1_{l_1}, \leq_{l_1}, \leq_{l_2})$	NExpTime/ExpSpace	[28]
$\mathrm{FO}^2(+1_{l_1}, \leq_{l_1}, +1_{l_2}, \leq_{l_2})$	Undecidable	[23]
Two total preorders		
$\mathrm{FO}^2(+1_{p_1}, +1_{p_2})$	Undecidable	★, Theorem 17
$\mathrm{FO}^2(+1_{p_1}, \leq_{p_2})$	Undecidable	★, Theorem 20
$\mathrm{FO}^2(\leq_{p_1}, \leq_{p_2})$	Undecidable	[27]
One linear order and one total preorder		
$\mathrm{FO}^2(+1_l, +1_p)$	? (see discussion)	★ Special case: Theorem 10
$\mathrm{FO}^2(+1_l, \leq_p)$	? (see discussion)	★ Special case: Theorem 10
$\mathrm{FO}^2(+1_l, \leq_l, +1_p)$	Undecidable	★, Theorem 19
$\mathrm{FO}^2(+1_l, \leq_l, \leq_p)$	Undecidable	[2]
$\mathrm{FO}^2(+1_l, +1_p, \leq_p)$	? (see discussion)	★, Special case: Theorem 10
$\mathrm{FO}^2(\leq_l, +1_p, \leq_p)$	ExpSpace-complete	[28]
Many orders		
$\mathrm{FO}^2(\leq_{l_1}, \leq_{l_2}, \leq_{l_3})$	Undecidable	[17]
$\mathrm{FO}^2(+1_{l_1}, +1_{l_2}, +1_{l_3})$	?	
$\mathrm{FO}^2(+1_{l_1}, +1_{l_2}, +1_{l_3}, \ldots)$	?	

Figure 3 Summary of results on finite satisfiability of FO^2 with successor and order relations. Cases that are symmetric and where undecidability is implied are omitted. Results in this paper are marked by ★.

We conclude with highlighting a small difference in treating successor relations for data words. In this paper, the preorder successor is *complete* in the sense that every element (except for elements contained in the last preorder equivalence class) has a preorder successor. In many data domains, especially in those that are subject to change, it is sufficient to interpret the preorder successor relation with respect to those data values present in the structure. Such domains are for example the words in the English language, ISBN numbers etc.

However, for data words over the natural numbers it can be useful that some data values are not present in a data word, i.e. that the successor relation can be incomplete. As a complete successor relation can be axiomatized given an incomplete successor relation, this is a more general setting. This setting is used in [28].

References

1 Mikolaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Logic*, 12(4):27:1–27:26, July 2011.

2 Mikolaj Bojanczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, and Claire David. Two-variable logic on words with data. In *LICS*, pages 7–16. IEEE Computer Society, 2006.

3 Egon Börger, Erich Grädel, and Yuri Gurevich. *The classical decision problem.* Springer Verlag, 2001.

4 Patricia Bouyer. A logical characterization of data languages. *Inf. Process. Lett.*, 84(2):75–85, 2002.

5 Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and trees. In *LICS*, 2013 (To appear).

6 Alonzo Church. A note on the Entscheidungsproblem. *J. Symb. Log.*, 1(1):40–41, 1936.

7 Stéphane Demri, Deepak D'Souza, and Régis Gascon. A decidable temporal logic of repeating values. In *LFCS*, volume 4514 of *Lecture Notes in Computer Science*, pages 180–194. Springer, 2007.

8 Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.

9 Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Information and Computation*, 179(2):279 – 295, 2002.

10 Diego Figueira. Satisfiability for two-variable logic with two successor relations on finite linear orders. *CoRR*, abs/1204.2495, 2012.

11 Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

12 Erich Grädel and Martin Otto. On logics with two variables. *Theor. Comput. Sci.*, 224(1-2):73–113, 1999.

13 Erich Grädel, Martin Otto, and Eric Rosen. Two-variable logic with counting is decidable. In *LICS*, pages 306–317, 1997.

14 Joseph Y Halpern and Yoav Shoham. A propositional modal logic of time intervals. *Journal of the ACM (JACM)*, 38(4):935–962, 1991.

15 Ullrich Hustadt, Renate A Schmidt, and Lilia Georgieva. A survey of decidable first-order fragments and description logics. *Journal of Relational Methods in Computer Science*, 1(251-276):3, 2004.

16 Ahmet Kara, Thomas Schwentick, and Thomas Zeume. Temporal logics on words with multiple data values. In *FSTTCS*, volume 8 of *LIPIcs*, pages 481–492. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

17 Emanuel Kieronski. Decidability issues for two-variable logics with several linear orders. In Marc Bezem, editor, *CSL*, volume 12 of *LIPIcs*, pages 337–351. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

18 Emanuel Kieronski, Jakub Michaliszyn, Ian Pratt-Hartmann, and Lidia Tendera. Two-variable first-order logic with equivalence closure. In *LICS*, pages 431–440, 2012.

19 Emanuel Kieronski and Martin Otto. Small substructures and decidability issues for first-order logic with two variables. In *LICS*, pages 448–457, 2005.

20 Emanuel Kieronski and Lidia Tendera. On finite satisfiability of two-variable first-order logic with equivalence relations. In *LICS*, pages 123–132, 2009.

21 S Rao Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 267–281. ACM, 1982.

22 Amal Manuel, Thomas Schwentick, and Thomas Zeume. A Short Note on Two-Variable Logic with a Linear Order Successor and a Preorder Successor. *ArXiv e-prints*, June 2013.

23 Amaldev Manuel. Two orders and two variables. In *MFCS*, volume 6281 of *Lecture Notes in Computer Science*, pages 513–524, 2010.

24 Ernst W Mayr. An algorithm for the general petri net reachability problem. *SIAM Journal on computing*, 13(3):441–460, 1984.

25 Michael Mortimer. On languages with two variables. *Zeitschr. f. math. Logik u. Grundlagen d. Math.*, 21:135–140, 1975.

26 Martin Otto. Two variable first-order logic over ordered domains. *J. Symb. Log.*, 66(2):685–702, 2001.

27 Thomas Schwentick and Thomas Zeume. Two-variable logic with two order relations. In *CSL*, volume 6247 of *Lecture Notes in Computer Science*, pages 499–513, 2010.

28 Thomas Schwentick and Thomas Zeume. Two-variable logic with two order relations. *Logical Methods in Computer Science*, 8(1), 2012.

29 Luc Segoufin and Szymon Torunczyk. Automata based verification over linearly ordered data domains. In *STACS*, volume 9 of *LIPIcs*, pages 81–92. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

30 Wieslaw Szwast and Lidia Tendera. FO^2 with one transitive relation is decidable. pages 317–328, 2013.

31 Boris Trakhtenbrot. The impossibilty of an algorithm for the decision problem for finite models. *Doklady Akademii NaukSSR*, 70(2):569–572, 1950.

32 Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, 42(2):230–265, 1936.

33 Yde Venema. Expressiveness and completeness of an interval tense logic. *Notre Dame Journal of Formal Logic*, 31(4):529–547, 1990.

Categorical Duality Theory: With Applications to Domains, Convexity, and the Distribution Monad

Yoshihiro Maruyama*

Quantum Group, Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK
maruyama@cs.ox.ac.uk
http://researchmap.jp/ymaruyama/

Abstract

Utilising and expanding concepts from categorical topology and algebra, we contrive a moderately general theory of dualities between algebraic, point-free spaces and set-theoretical, point-set spaces, which encompasses infinitary Stone dualities, such as the well-known duality between frames (aka. locales) and topological spaces, and a duality between σ-complete Boolean algebras and measurable spaces, as well as the classic finitary Stone, Gelfand, and Pontryagin dualities. Among different applications of our theory, we focus upon domain-convexity duality in particular: from the theory we derive a duality between Scott's continuous lattices and convexity spaces, and exploit the resulting insights to identify intrinsically the dual equivalence part of a dual adjunction for algebras of the distribution monad; the dual adjunction was uncovered by Bart Jacobs, but with no characterisation of the induced equivalence, which we do give here. In the Appendix, we place categorical duality in a wider context, and elucidate philosophical underpinnings of duality.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages

Keywords and phrases duality, monad, categorical topology, domain theory, convex structure

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.500

1 Introduction

There are two conceptions of space: one comes from the ontic idea that the ultimate constituents of space are points with no extension; the other does not presuppose the concept of points in the first place, and starts with an epistemically more certain concept such as regions or observable properties. For instance, a topological space is an incarnation of the former idea of space, and a frame (or locale) is an embodiment of the latter. Duality often exists between point-free and point-set conceptions of space (to put it differently, between epistemology and ontology of space; see the Appendix as well), as exemplified by the well-known duality between frames and topological spaces (see, e.g., Johnstone [12]).

The most general duality theorem is this: any category $\mathbf{C}$ is dually equivalent to the opposite category $\mathbf{C}^{\mathrm{op}}$. It, of course, makes no substantial sense; however, note that it prescribes a generic form of duality in a non-obvious way (we say "non-obvious" because there may be different conceptions of a generic form of duality, some of which may not be based upon category theory at all). In this paper, we attempt to avoid such triviality by focusing upon a more specific context: we aim at developing a moderately general theory

* I am grateful to Samson Abramsky and Bob Coecke for discussions and encouragements. Special thanks to Jiri Velebil for telling me about categorical topology, which made my earlier, naïve concept of point-set space more sophisticated. I would like to express my gratitude to Hilary Priestley for discussions on Johnstone's dual adjunction theorem. This work was supported by the Nakajima Foundation.

of dualities between point-free and point-set spaces, whilst having in mind applications to domain-convexity duality, where domains are seen as point-free convex structures.

Our general theory of dualities between point-free and point-set spaces builds upon the celebrated idea of a duality induced by a Janusian (aka. schizophrenic) object: "a potential duality arises when a single object lives in two different categories" (Lawvere's words quoted in Barr et al. [3]). Note that in this paper we mean by dualities dual adjunctions in general; dual equivalences are understood as special cases. There are different theories of dualities based upon the same idea (see, e.g., [3, 5, 12, 17]); some of them use universal algebra, whilst others are categorical. Our theory is in between universal algebra and category theory (although we use categorical terminology, nevertheless, everything can be recasted in terms of universal algebra and of general point-set spaces introduced in Maruyama [15]). More detailed comparison with related work is given below.

Our duality theory allows us to derive a number of concrete dualities, including infinitary Stone dualities, such as the aforementioned duality in point-free topology, and a certain duality between σ-complete Boolean algebras and measurable spaces, as well as the classic finitary Stone, Gelfand, and Pontryagin dualities (since the Pontryagin duality is a self-duality, how to treat it is slightly different from how to do the others as noted below).

In the present paper we focus, *inter alia*, upon dualities between point-free and point-set convex structures. On the one hand, we consider Scott's continuous lattices to represent point-free convex structures for certain reasons explained later, in Subsection 3.1. On the other hand, there are two kinds of point-set convex structures: i.e., convexity spaces (see van de Vel [18] or Coppel [6]; the definition is given in Preliminaries in Section 2) and algebras of the distribution monad (aka. barycentric algebras; see Fritz [8]).

Our general theory tells us that there is a duality between continuous lattices and convexity spaces. In contrast, Jacobs [11] shows a dual adjunction between preframes and algebras of the distribution monad, which can be reformulated as a dual adjunction between continuous lattices and algebras of the distribution monad. Although Jacobs [11] left it open to identify intrinsically the induced dual equivalence, in this paper, we give an intrinsic characterisation of the dual equivalence part of the dual adjunction for algebras of the distribution monad.

Technical Summary. In our duality theory, we mainly rely upon concrete category theory as in Adámek et al. [1], especially concepts from categorical topology (see also [2, 4]) and categorical algebra (in particular the theory of monads).

We start with a category $\mathbf{C}$ monadic over $\mathbf{Set}$ (which is equivalent to possibly infinitary varieties in terms of universal algebra) and with a topological category $\mathbf{D}$ of certain type, and then assume that there is a Janusian object Ω living in both $\mathbf{C}$ and $\mathbf{D}$. In passing, we introduce the new concept of classical topological axiom, and use it to identify a certain class of those full subcategories of a functor-structured category that represent categories of point-set spaces. Under the assumption of what we call the harmony condition, which basically means that algebraic operations are continuous in a suitable sense, we finally show that there is a dual adjunction between $\mathbf{C}$ and $\mathbf{D}$, given by homming into Ω.

The dual adjunction formally restricts to a dual equivalence via the standard method of taking those objects of $\mathbf{C}$ and $\mathbf{D}$ that are fixed under the unit and counit of the adjunction; to put it intuitively, the objects whose double duals are isomorphic to themselves. At the same time, however, it is often highly non-trivial to identify intrinsically the dual equivalence part of a dual adjunction in a concrete situation, as Porst-Tholen [17] remark, "This can be a very hard problem, and this is where categorical guidance comes to an end."[1]

[1] To exemplify what is meant here, consider the dual adjunction between frames and spaces, which

Using specialised, context-dependent methods, rather than the generic one mentioned above, we give intrinsic characterisations of dual equivalences induced by the dual adjunction for convexity spaces, and by the dual adjunction for algebras of the distribution monad. The concept of polytopes plays a crucial role in the characterisations, and in understanding how semilattices involve convex structures.

Comparison with Related Work. Our general theory of dualities may be compared with other duality theories as follows. Clark-Davey's theory of natural dualities [5] is based upon the same idea of a duality induced by a Janusian object. However, our theory is more comprehensive than natural duality theory, in that whilst natural duality theory specialises in dualities for finitary algebras, our theory is intended to encompass infinitary algebras as well (e.g., frames, σ-complete Boolean algebras, and continuous lattices). Our theory thus encompasses both finitary and infinitary Stone-type dualities.

Johnstone's general concrete duality [12, VI.4] and Porst-Tholen's natural dual adjunction [17] are more akin to ours.[2] A crucial difference is, however, that we stick to the practice of Stone-type dualities as far as possible. In their theories, there is no concrete concern with how to equip the "spectrum" of an "algebra" with a "topology" or how to equip the (collection of) "functions" on a "space" with an "algebraic" structure.

We consider that the processes of algebraisation and topologisation are essential in the practice of Stone-type dualities. In particular, algebraisation and topologisation are strikingly different processes in the practice; in spite of this, the two processes are treated in their theories as being in parallel and symmetric at a level of abstraction, which looks like an excessive abstraction from our perspective of the practice of duality.

We put a strong emphasis on the asymmetry between the two processes of algebraisation and topologisation in the practice of Stone-type dualities, and thus aim at simulating the processes within our theory, thereby representing the practice of duality in an adequate manner. In order to achieve this goal, our theory cannot and should not be so general as to symmetrise the asymmetry; this is the reason why we call our theory "moderately" general.

In comparison with Maruyama [15], which discusses a theory of T_1-type dualities based upon Chu spaces and a generic concept of closure conditions, the present paper aims at a theory of sober-type dualities; an example of duality of T_1-type is a (not very well known) duality between T_1 spaces and coatomistic frames (a subtlety is frame morphisms must be "maximal" to dualise continuous maps; see [15]). Sober-type dualities are based upon prime spectra, whilst T_1-type dualities are based upon maximal spectra. Affine varieties (except singletons) in $\mathbb{C}^n$ with Zariski topologies are non-sober T_1 spaces; they are homeomorphic to the maximal spectra of their coordinate rings. Both the former and the latter theories can be applied to different sorts of spaces, yielding T_1-type and sober-type dualities respectively.

Jacobs [11] and Maruyama [14] independently unveiled (different) dualities for convexity (convexity algebras in [14] are replaced in this paper by continuous lattices), and the present paper is meant to elucidate a precise link between them, which remained unclear so far. The two dualities turn out to be essentially the same in spite of their rather different outlooks.

---

restricts to a dual equivalence between the frames and spaces whose double duals are isomorphic to themselves; this is trivial. Nevertheless, it is not trivial at all to notice that those frames are exactly the frames with enough points (i.e., spatial frames), and those spaces are precisely the spaces in which any non-empty irreducible closet set is the closure of a unique point (i.e., sober spaces).

[2] Johnstone's dual adjunction (Lemma VI.4.2) seems to be not very rigorous because he dares to say "we choose not to involve ourselves in giving a precise meaning to the word 'commute' in the last sentence" (p. 254), and the dual adjunction result actually relies upon the assumption of that commutativity. In this paper, we precisely formulate the concept of commutativity as what we call the harmony condition.

2 General Duality Theory

After preliminaries, we first review categorical topology, and then get into a general theory of dualities based upon the concepts of monad, functor-(co)structured category, and topological (co)axiom. Among other things, we introduce the new concept of classical topological axiom with the aim of treating different sorts of point-set spaces in a unified way.

Preliminaries. For a category $\mathbf{C}$ and a faithful functor $U : \mathbf{C} \to \mathbf{Set}$, a tuple $(\mathbf{C}, U)$ is called a concrete category, where $\mathbf{Set}$ denotes the category of sets and functions. U is called the underlying functor of the concrete category. For simplicity, we often omit and make implicit the functor U of a concrete category $(\mathbf{C}, U)$. We can also define the notion of a concrete category over a general category. For a category $\mathbf{C}$ and a faithful functor $U : \mathbf{C} \to \mathbf{D}$, $(\mathbf{C}, U)$ is called a concrete category over $\mathbf{D}$. A concrete category (over $\mathbf{Set}$) in this paper is called a construct in [1]. $\mathbf{Top}$ denotes the category of topological spaces and continuous functions. $\mathbf{Conv}$ denotes the category of convexity spaces and convexity preserving maps, where a convexity space is a tuple $(X, \mathcal{C})$ such that X is a set and $\mathcal{C}$ is a subset of the powerset of X that is closed under directed unions and arbitrary intersections; a convexity-preserving map is such that the inverse image of any convex set under it is again convex (see van de Vel [18] and Coppel [6], which develop substantial amount of convex geometry based upon this general concept of convexity space). $\mathbf{Meas}$ denotes the category of measurable spaces and measurable functions. $\mathbf{Frm}$ denotes the category of frames and their homomorphisms. $\mathbf{ContLat}$ denotes the category of continuous lattices and their homomorphisms (i.e., maps preserving directed joins and arbitrary meets). $\mathbf{BA}_\sigma$ denotes the category of σ-complete Boolean algebras with σ-distributivity and their homomorphisms, where σ-distributivity means that countable joins distribute over countable meets. $\mathcal{Q} : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Set}$ denotes the contravariant powerset functor.

2.1 A Categorical Conception of Point-Set Spaces

Here we introduce a general concept of space that encompasses topological spaces, convexity spaces, and measurable spaces. Our duality theory shall be developed based upon that concept of (generalised) space. For the fundamentals of functor-(co)structured category and topological (co)axiom, we refer to Adámek et al. [1]. We first review the notion of functor-structured category. Let $(\mathbf{C}, U : \mathbf{C} \to \mathbf{Set})$ be a concrete category in the following.

▶ **Definition 2.1** ([1]). A category $\mathbf{Spa}(U)$ is defined as follows.

1. An object of $\mathbf{Spa}(U)$ is a tuple $(C, \mathcal{O})$ where $C \in \mathbf{C}$ and $\mathcal{O} \subset U(C)$.
2. An arrow of $\mathbf{Spa}(U)$ from $(C, \mathcal{O})$ to $(C', \mathcal{O}')$ is an arrow $f : C \to C'$ of $\mathbf{C}$ such that $U(f)[\mathcal{O}] \subset \mathcal{O}'$.

A category of the form $\mathbf{Spa}(U)$ is called a functor-structured category. A category of the form $(\mathbf{Spa}(U))^{\mathrm{op}}$ is called a functor-costructured category.

We consider $\mathbf{Spa}(U)$ as a concrete category equipped with a faithful functor $U \circ F : \mathbf{Spa}(U) \to \mathbf{Set}$ where $F : \mathbf{Spa}(U) \to \mathbf{C}$ is the forgetful functor that maps $(C, \mathcal{O})$ to C.

Then we can show the following (for the definition of topological category, see [1]; although there are different notions of a topological category, we follow the terminology of [1]).

▶ Proposition 2.2 ([1]). Both a functor-structured category $\mathbf{Spa}(U)$ and a functor-costructured category $(\mathbf{Spa}(U))^{\mathrm{op}}$ are topological.

The concept of topological (co)axiom is defined as follows.

▶ **Definition 2.3** ([1]). A topological axiom in $(\mathbf{C}, U)$ is defined as an arrow $p : C \to C'$ of $\mathbf{C}$ such that

1. $U(C) = U(C')$;
2. $U(p) : U(C) \to U(C)$ is the identity morphism on $U(C)$.

An object C of $\mathbf{C}$ satisfies a topological axiom $p : D \to D'$ in $(\mathbf{C}, U)$ iff, for any arrow $f : D \to C$ of $\mathbf{C}$, there is an arrow $f' : D' \to C$ of $\mathbf{C}$ such that $U(f) = U(f')$.

A topological coaxiom is defined as a topological axiom with the following concept of satisfaction. An object C of $\mathbf{C}$ satisfies a topological coaxiom $p : D' \to D$ in $(\mathbf{C}, U)$ iff, for any arrow $f : C \to D$ of $\mathbf{C}$, there is an arrow $f : C \to D'$ of $\mathbf{C}$ such that $U(f) = U(f')$.

Topological axioms and coaxioms are the same, but the corresponding notions of satisfaction are dual to each other. For examples of topological (co)axiom, we refer to [1].

▶ **Definition 2.4** ([1])**.** Let X be a class of topological (co)axioms in a concrete category $\mathbf{C}$. A full subcategory $\mathbf{D}$ of $\mathbf{C}$ is definable by X in $\mathbf{C}$ iff the objects of $\mathbf{D}$ coincide with those objects of $\mathbf{C}$ that satisfy all the topological (co)axioms in X.

As in the following proposition, we can show a topological analogue of the Birkhoff theorem in universal algebra (for more details, see Theorem 22.3 and Corollary 22.4 in [1]).

▶ Proposition 2.5 ([1]). Let $\mathbf{C}$ be a concrete category. The following are equivalent:

1. $\mathbf{C}$ is fibre-small and topological;
2. $\mathbf{C}$ is isomorphic to a subcategory of a functor-structured category that is definable by a class of topological axioms in the functor-structured category.
3. $\mathbf{C}$ can be embedded into a functor-structured category as a full subcategory that is closed under the formation of products, initial subobjects, and indiscrete objects.

Now we introduce the new concept of classical topological (co)axiom, which shall play a crucial role in formulating our dual adjunction theorem.

▶ **Definition 2.6.** A classical topological axiom in $\mathbf{Spa}(U)$ is defined as a topological axiom $p : (C, \mathcal{O}) \to (C', \mathcal{O}')$ in $\mathbf{Spa}(U)$ such that

- Any element of $\mathcal{O}' \setminus \mathcal{O}$ can be expressed as a (possibly infinitary) Boolean combination of elements of $\mathcal{O}$.

A classical topological coaxiom in $(\mathbf{Spa}(U))^{\mathrm{op}}$ is defined as a topological coaxiom $p : (C, \mathcal{O}) \to (C', \mathcal{O}')$ in $(\mathbf{Spa}(U))^{\mathrm{op}}$ such that

- Any element of $\mathcal{O} \setminus \mathcal{O}'$ can be expressed as a (possibly infinitary) Boolean combination of elements of $\mathcal{O}'$.

Let $\mathcal{Q} : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Set}$ denote the contravariant powerset functor. Any of the category **Top** of topological spaces, the category **Conv** of convexity spaces, and the category **Meas** of measurable spaces is a full subcategory of $(\mathbf{Spa}(\mathcal{Q}))^{\mathrm{op}}$ that is definable by a class of classical topological coaxioms as in the following proposition, which can be shown just by spelling out the definitions involved.

▶ Proposition 2.7. **Top** is definable by the following class of classical topological coaxioms in $(\mathbf{Spa}(\mathcal{Q}))^{\mathrm{op}}$:

$$\begin{aligned} 1_S : (S, \{\emptyset, S\}) &\to (S, \emptyset) \\ 1_S : (S, \{X, Y, X \cap Y\}) &\to (S, \{X, Y\}) \\ 1_S : (S, \mathcal{O} \cup \{\textstyle\bigcup \mathcal{O}\}) &\to (S, \mathcal{O}) \end{aligned}$$

for all sets S, all subsets X, Y of S and all subsets $\mathcal{O}$ of the powerset of S.

Conv is definable by the following class of classical topological coaxioms in $(\mathbf{Spa}(\mathcal{Q}))^{\mathrm{op}}$: (i) $1_S : (S, \{\emptyset, S\}) \to (S, \emptyset)$; (ii) $1_S : (S, \mathcal{C} \cup \{\bigcap \mathcal{C}\}) \to (S, \mathcal{C})$; (iii) $1_S : (S, \mathcal{C}' \cup \{\bigcup \mathcal{C}'\}) \to (S, \mathcal{C}')$

for all sets S, all subsets $\mathcal{C}$ of the powerset of S, and all those subsets $\mathcal{C}'$ of the powerset of S that are directed with respect to inclusion.

Meas is definable by the following class of classical topological coaxioms in $(\mathbf{Spa}(\mathcal{Q}))^{\mathrm{op}}$: (i) $1_S : (S, \{\emptyset, S\}) \to (S, \emptyset)$; (ii) $1_S : (S, \{X, X^c\}) \to (S, \{X\})$; (iii) $1_S : (S, \mathcal{B} \cup \{\bigcup \mathcal{B}\}) \to (S, \mathcal{B})$ for all sets S, all subsets X of S, and all those subsets $\mathcal{B}$ of the powerset of S that are of cardinality $\leq \omega$.

In order to develop a general duality theory, thus, we shall focus upon a full subcategory **Spa** of $(\mathbf{Spa}(\mathcal{Q}))^{\mathrm{op}}$ that is definable by a class of classical topological coaxioms in $(\mathbf{Spa}(\mathcal{Q}))^{\mathrm{op}}$.

We call $(S, \mathcal{O}) \in \mathbf{Spa}$ a generalised space and $\mathcal{O}$ a generalised topology.

Given a subset $\mathcal{P}$ of the powerset of a set S, we can generate a topology on S from $\mathcal{P}$, which is the weakest topology containing $\mathcal{P}$. We can also do the same thing in the case of generalised topology.

▶ Proposition 2.8. For a set S, let $\mathcal{P}$ be a subset of the powerset of S. Then, there is a weakest generalised topology on S containing $\mathcal{P}$ in **Spa**, i.e., there is $(S, \mathcal{O}) \in \mathbf{Spa}$ such that, if $\mathcal{P} \subset \mathcal{O}'$ for $(S, \mathcal{O}') \in \mathbf{Spa}$, then $\mathcal{O} \subset \mathcal{O}'$. We then say that $\mathcal{O}$ is generated in **Spa** by $\mathcal{P}$.

Proof. Define $\mathcal{O} = \bigcap\{\mathcal{X} \ ; \ \mathcal{P} \subset \mathcal{X} \text{ and } (S, \mathcal{X}) \in \mathbf{Spa}\}$. It is sufficient to show that $\mathcal{O}$ is a generalised topology on S in **Spa**, i.e., $(S, \mathcal{O})$ satisfies the class of topological coaxioms that define **Spa**. Assume that $p : (X, \mathcal{B}') \to (X, \mathcal{B})$ is one of such coaxioms and that $f : (S, \mathcal{O}) \to (X, \mathcal{B})$ is an arrow in $(\mathbf{Spa}(\mathcal{Q}))^{\mathrm{op}}$. For $B \in \mathcal{B}' \setminus \mathcal{B}$, we have $f^{-1}(B) \in \mathcal{X}$ for any $\mathcal{X}$ with $\mathcal{P} \subset \mathcal{X}$ and $(S, \mathcal{X}) \in \mathbf{Spa}$, which implies that $f^{-1}(B) \in \mathcal{O}$. ◀

2.2 Dual Adjunction via Harmony Condition

Throughout this subsection, let

- **Alg** denote a full subcategory of the Eilenberg-Moore category of a monad T on **Set**;
- **Spa** denote a full subcategory of $(\mathbf{Spa}(\mathcal{Q}))^{\mathrm{op}}$ that is definable by a class of classical topological coaxioms in $(\mathbf{Spa}(\mathcal{Q}))^{\mathrm{op}}$.

We aim at establishing a dual adjunction between **Alg** and **Spa** under the two assumptions:

- there is an object Ω living in both **Alg** and **Spa**, i.e., there is $\Omega \in \mathbf{Set}$ both with a structure map $h_\Omega : T(\Omega) \to \Omega$ such that $(\Omega, h_\Omega) \in \mathbf{Alg}$ and with a generalised topology $\mathcal{O}_\Omega \subset \mathcal{Q}(\Omega)$ such that $(\Omega, \mathcal{O}_\Omega) \in \mathbf{Spa}$;
- $(\mathbf{Alg}, \mathbf{Spa}, \Omega)$ satisfies the harmony condition in Definition 2.9 below.

Ω is intuitively a set of truth values, and shall work as a so-called dualising object (we do not use the term "schizophrenic", since it has a different technical meaning in a certain context). We simply write Ω instead of (Ω, h_Ω) or $(\Omega, \mathcal{O}_\Omega)$ when there is no confusion.

The harmony condition intuitively means that the algebraic structure of **Alg** and the geometric structure of **Spa** are in harmony via Ω. The precise definition is given below.

▶ **Definition 2.9.** $(\mathbf{Alg}, \mathbf{Spa}, \Omega)$ is said to satisfy the harmony condition iff, for each $S \in \mathbf{Spa}$,

$$(\ \mathrm{Hom}_{\mathbf{Spa}}(S, \Omega), \ \ h_S : T(\mathrm{Hom}_{\mathbf{Spa}}(S, \Omega)) \to \mathrm{Hom}_{\mathbf{Spa}}(S, \Omega) \ )$$

is an object in **Alg** such that, for any $s \in S$ (let p_s be the corresponding projection from $\mathrm{Hom}_{\mathbf{Spa}}(S, \Omega)$ to Ω), the following diagram commutes:

$$\begin{array}{ccc} T(\mathrm{Hom}_{\mathbf{Spa}}(S, \Omega)) & \xrightarrow[h_S]{} & \mathrm{Hom}_{\mathbf{Spa}}(S, \Omega) \\ \Big\downarrow{\scriptstyle T(p_s)} & & \Big\downarrow{\scriptstyle p_s} \\ T(\Omega) & \xrightarrow[h_\Omega]{} & \Omega \end{array}$$

▶ Remark 2.10. The commutative diagram above means that the induced operations of $\mathrm{Hom}_{\mathbf{Spa}}(S,\Omega)$ are defined pointwise. The harmony condition then consists of the two parts:
(i) $\mathrm{Hom}_{\mathbf{Spa}}(S,\Omega)$ is closed under the pointwise operations;
(ii) $\mathrm{Hom}_{\mathbf{Spa}}(S,\Omega)$ with the pointwise operations is in **Alg**.
Here, (ii) is not so important for the reason that we can drop condition (ii) if **Alg** is the Eilenberg-Moore category of a monad on **Set**, rather than a full subcategory of it; this follows from the fact that, since **Alg** is then closed under products and subalgebras, we have a product Ω^S in **Alg**, and hence $\mathrm{Hom}_{\mathbf{Spa}}(S,\Omega)$ in **Alg** as a subalgebra of Ω^S (obviously, it actually suffices to assume that **Alg** is a quasi-variety or an implicational full subcategory of the Eilenberg-Moore category of a monad on **Set** in the sense of [1]). Regarding $\mathrm{Hom}_{\mathbf{Spa}}(S,\Omega)$ as the collection of generalised continuous functions on S, (i) above means that the continuous functions are closed under the algebraic operations defined pointwise, which is the most important part of the harmony condition, and after which the "harmony" condition is named.

We assume the harmony condition in the following part of this subsection.

The geometric structure of $\mathrm{Hom}_{\mathbf{Alg}}(A,\Omega)$ for $A \in \mathbf{Alg}$ can be provided as follows. By Proposition 2.8, equip $\mathrm{Hom}_{\mathbf{Alg}}(A,\Omega)$ with the generalised topology generated (in **Spa**) by

$$\{\langle a\rangle_O \ ; \ a \in A \text{ and } O \in \mathcal{O}_\Omega\}$$

where

$$\langle a\rangle_O := \{v \in \mathrm{Hom}_{\mathbf{Alg}}(A,\Omega) \ ; \ v(a) \in O\}.$$

The algebraic structure of $\mathrm{Hom}_{\mathbf{Spa}}(S,\Omega)$ is provided by h_S above.

The induced contravariant Hom-functors $\mathrm{Hom}_{\mathbf{Alg}}(\text{-},\Omega) : \mathbf{Alg} \to \mathbf{Spa}$ and $\mathrm{Hom}_{\mathbf{Spa}}(\text{-},\Omega) : \mathbf{Spa} \to \mathbf{Alg}$ can be shown to be well defined and form a dual adjunction between categories **Alg** and **Spa**, i.e., we have the following dual adjunction theorem:

▶ **Theorem 2.11.** There is a dual adjunction between **Alg** and **Spa**, given by contravariant functros $\mathrm{Hom}_{\mathbf{Alg}}(\text{-},\Omega)$ and $\mathrm{Hom}_{\mathbf{Spa}}(\text{-},\Omega)$. To be precise, $\mathrm{Hom}_{\mathbf{Alg}}(\text{-},\Omega)$ is left adjoint to $\mathrm{Hom}_{\mathbf{Spa}}(\text{-},\Omega)^{\mathrm{op}}$.

A proof of the theorem is given soon after the following remark.

▶ Remark 2.12. The theorem encompasses the well-known dual adjunction between frames and topological spaces; in this case, Ω is the two element frame with the Sierpinski topology, and the harmony condition boils down to the obvious fact that the collection of open sets is closed under the operations of arbitrary unions and finite intersections. The frame-space duality is thus an immediate corollary of the theorem above; this exhibits a sharp contrast to those general theories of dualities that require substantial work in deriving concrete results. Our theory is for duality in context, contrived to be effective in concrete situations.

In a similar way, we can derive a dual adjunction between σ-complete Boolean algebras and measurable spaces by letting Ω be the two element algebra with the discrete topology (in fact, any algebra with the discrete topology works as Ω), where σ-complete Boolean algebras may be seen as point-free measurable spaces. In Section 3, we discuss in detail a dual adjunction between continuous lattices and convexity spaces.

The most plain case is the dual adjunction between **Set** and **Set**, induced by the two element set as a dualising object Ω (any set actually works); the harmony condition is nothing in this case. The discrete Stone adjunction between Boolean algebras and **Set** is well known. The theorem above gives us a vast generalisation of it: there is a dual adjunction between any algebraic category (or variety in terms of universal algebra) and **Set**, induced by any

$\Omega \in \mathbf{Alg}$; the harmony condition is nothing in this case as well, thanks to the discrete nature of **Set** (i.e., the set of all functions $f : S \to \Omega$ are closed under arbitrary operations on it).

Furthermore, the theorem above encompasses the topological Stone adjunction between Boolean algebras and topological spaces, its diverse extensions for distributive lattices, MV-algebras ($[0,1]$ works as a dualising object in this case), and algebras of substructural logics, and the Gelfand adjunction between commutative C^*-algebras with units 1 and topological spaces; note that the category of commutative C^*-algebras with 1 is monadic over **Set** (see [16]). Any dual adjunction automatically cuts down to a dual equivalence as explained below, and the method can be applied to all the dual adjunctions mentioned above in order to obtain dual equivalences (still it often is not that easy to give intrinsic characterisations of the resulting dual equivalences as already discussed in Section 1).

Let us think of the Pontryagin self-duality for locally compact Abelian groups. Although for simplicity we did not assume a topological structure on **Alg** and an algebraic structure on **Spa** in our set-up, this gets relevant in order to treat the Pontryagin duality within our framework. It is indeed straightforward: we start with topological **Alg** and algebraic **Spa**, and assume two harmony conditions; and the following proof can easily be adapted to that situation (just repeat the same arguments for the additional structures on **Alg** and **Spa**).

2.2.1 Proof of Dual Adjunction Theorem

We first show that the two Hom-functors are well defined.

▶ **Lemma 2.13.** The contravariant functor $\mathrm{Hom}_{\mathbf{Alg}}(\text{-}, \Omega) : \mathbf{Alg} \to \mathbf{Spa}$ is well defined.

Proof. The object part is well defined by Proposition 2.8. We show that the arrow part is well defined. Let $f : A \to A'$ be an arrow in **Alg**. We prove that $\mathrm{Hom}_{\mathbf{Alg}}(f, \Omega) : \mathrm{Hom}_{\mathbf{Alg}}(A', \Omega) \to \mathrm{Hom}_{\mathbf{Alg}}(A, \Omega)$ is an arrow in **Spa**. For $a \in A$ and $O \in \mathcal{O}_\Omega$, we have:

$$\begin{aligned} \mathrm{Hom}_{\mathbf{Alg}}(f,\Omega)^{-1}(\langle a\rangle_O) &= \{v \in \mathrm{Hom}_{\mathbf{Alg}}(A',\Omega) \;;\; \mathrm{Hom}_{\mathbf{Alg}}(f,\Omega)(v) \in \langle a\rangle_O\} \\ &= \{v \in \mathrm{Hom}_{\mathbf{Alg}}(A',\Omega) \;;\; v \circ f(a) \in O\} \\ &= \langle f(a)\rangle_O. \end{aligned}$$

Since **Spa** is definable by a class of Boolean topological coaxioms and since Boolean set operations are preserved by the inverse image function f^{-1}, this implies that $\mathrm{Hom}_{\mathbf{Alg}}(f, \Omega)$ is an arrow in **Spa**. ◀

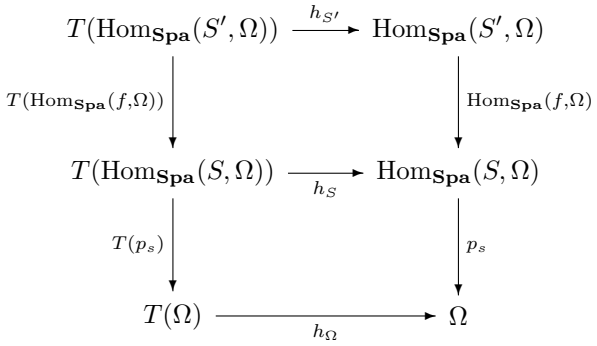
▶ **Lemma 2.14.** The contravariant functor $\mathrm{Hom}_{\mathbf{Spa}}(\text{-}, \Omega) : \mathbf{Spa} \to \mathbf{Alg}$ is well defined.

Proof. The object part is well defined by the harmony condition (or can be verified as in (i) or (ii) in Remark 2.10 if we employ either of the other two definitions of **Alg**).

We show that the arrow part is well defined. Let $f : S \to S'$ be an arrow in **Spa**. We prove that $\mathrm{Hom}_{\mathbf{Spa}}(f, \Omega) : \mathrm{Hom}_{\mathbf{Spa}}(S', \Omega) \to \mathrm{Hom}_{\mathbf{Spa}}(S, \Omega)$ is an arrow in **Alg**, i.e., the following diagram commutes:

$$\begin{array}{ccc} T(\mathrm{Hom}_{\mathbf{Spa}}(S',\Omega)) & \xrightarrow{h_{S'}} & \mathrm{Hom}_{\mathbf{Spa}}(S',\Omega) \\ \downarrow{\scriptstyle T(\mathrm{Hom}_{\mathbf{Spa}}(f,\Omega))} & & \downarrow{\scriptstyle \mathrm{Hom}_{\mathbf{Spa}}(f,\Omega)} \\ T(\mathrm{Hom}_{\mathbf{Spa}}(S,\Omega)) & \xrightarrow[h_S]{} & \mathrm{Hom}_{\mathbf{Spa}}(S,\Omega) \end{array}$$

By the harmony condition applied to S (or the commutativity of the lower square in the figure below), this is equivalent to the commutativity of the outermost square in the following diagram for any $s \in S$:



where recall that p_s denotes the corresponding projection. By the harmony condition applied to S', we have: for any $s' \in S'$, $h_\Omega \circ T(p_{s'}) = p_{s'} \circ h_{S'}$. By taking $s' = f(s)$ in this equation, we have $h_\Omega \circ T(p_{f(s)}) = p_{f(s)} \circ h_{S'}$. It is straightforward to verify that $p_{f(s)} = p_s \circ \mathrm{Hom}_{\mathbf{Spa}}(f,\Omega)$. Thus we obtain $h_\Omega \circ T(p_s \circ \mathrm{Hom}_{\mathbf{Spa}}(f,\Omega)) = p_s \circ \mathrm{Hom}_{\mathbf{Spa}}(f,\Omega) \circ h_{S'}$. Since T is a functor, this yields the commutativity of the outermost square above. Hence, the arrow part is well defined. ◀

Now we define two natural transformations in order to show the dual adjunction.

▶ **Definition 2.15.** Natural transformations

$$\Phi : 1_{\mathbf{Alg}} \to \mathrm{Hom}_{\mathbf{Spa}}(\mathrm{Hom}_{\mathbf{Alg}}(\text{-},\Omega),\Omega)$$

and

$$\Psi : 1_{\mathbf{Spa}} \to \mathrm{Hom}_{\mathbf{Alg}}(\mathrm{Hom}_{\mathbf{Spa}}(\text{-},\Omega),\Omega)$$

are defined as follows. For $A \in \mathbf{Alg}$, define Φ_A by $\Phi_A(a)(v) = v(a)$ where $a \in A$ and $v \in \mathrm{Hom}_{\mathbf{Alg}}(A,\Omega)$. For $S \in \mathbf{Spa}$, define Ψ_S by $\Psi_S(x)(f) = f(x)$ where $x \in S$ and $f \in \mathrm{Hom}_{\mathbf{Spa}}(S,\Omega)$.

We have to show that Φ and Ψ are well defined.

▶ **Lemma 2.16.** For $A \in \mathbf{Alg}$ and $a \in A$, $\Phi_A(a)$ is an arrow in $\mathbf{Spa}$.

Proof. For $O \in \mathcal{O}_\Omega$, we have

$$\Phi_A(a)^{-1}(O) = \{v \in \mathrm{Hom}_{\mathbf{Alg}}(A,\Omega)\ ;\ \Phi_A(a)(v) \in O\} = \langle a \rangle_O.$$

Thus, $\Phi_A(a)$ is an arrow in $\mathbf{Spa}$. ◀

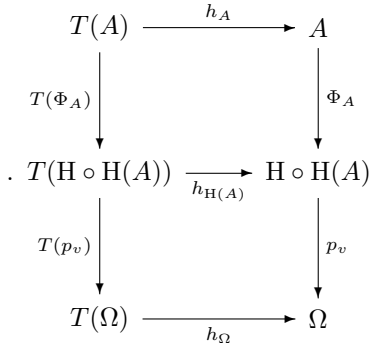
▶ **Lemma 2.17.** For $S \in \mathbf{Spa}$ and $x \in S$, $\Psi_S(x)$ is an arrow in $\mathbf{Alg}$.

Proof. This lemma follows immediately from the harmony condition applied to $\mathrm{Hom}_{\mathbf{Spa}}(S,\Omega)$ together with the fact that $p_x = \Psi_S(x)$. ◀

We also have to show that Φ_A is an arrow in $\mathbf{Alg}$ and that Ψ_S is an arrow in $\mathbf{Spa}$.

▶ **Lemma 2.18.** For $A \in \mathbf{Alg}$, Φ_A is an arrow in $\mathbf{Alg}$.

Proof. Let $h_A : T(A) \to A$ denote the structure map of A. For the simplicity of description, let $\mathrm{H}(A)$ denote $\mathrm{Hom}_{\mathbf{Alg}}(A, \Omega)$ and $\mathrm{H} \circ \mathrm{H}(A)$ denote $\mathrm{Hom}_{\mathbf{Spa}}(\mathrm{Hom}_{\mathbf{Alg}}(A, \Omega), \Omega)$. In order to show the commutativity of the upper square in the diagram below, it is sufficient to prove that the outermost square is commutative for any $v \in \mathrm{H}(A)$, since the lower square is commutative because of the harmony condition applied to $\mathrm{H} \circ \mathrm{H}(A)$.



It is straightforward to verify that $p_v \circ \Phi_A = v$. Then, it suffices to show that $v \circ h_A = h_\Omega \circ T(v)$. This is nothing but the fact that $v \in \mathrm{H}(A)$. ◀

▶ **Lemma 2.19.** For $S \in \mathbf{Spa}$, Ψ_S is an arrow in $\mathbf{Spa}$.

Proof. For $f \in \mathrm{Hom}_{\mathbf{Alg}}(\mathrm{Hom}_{\mathbf{Spa}}(\text{-}, \Omega), \Omega)$ and $O \in \mathcal{O}_\Omega$, we have

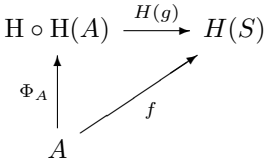
$$\Psi_S^{-1}(\langle f \rangle_O) = \{x \in S \ ; \ \Psi_S(x) \in \langle f \rangle_O\} = f^{-1}(O).$$

Since $\mathbf{Spa}$ is definable by a class of Boolean topological coaxioms and since Boolean set operations are preserved by the inverse image function Ψ_S^{-1}, this implies that Ψ_S is an arrow in $\mathbf{Spa}$. ◀

Now it is straightforward to verify that Φ and Ψ are actually natural transformations.

We finally give a proof of the dual adjunction theorem, Theorem 2.11: $\mathrm{Hom}_{\mathbf{Alg}}(\text{-}, \Omega)$ is left adjoint to $\mathrm{Hom}_{\mathbf{Spa}}(\text{-}, \Omega)^{\mathrm{op}}$ with Φ the unit and Ψ^{op} the counit of the adjunction.

Proof. Let $A \in \mathbf{Alg}$ and $S \in \mathbf{Spa}$. It is enough to show that, for any $f : A \to \mathrm{Hom}_{\mathbf{Spa}}(S, \Omega)$ in $\mathbf{Alg}$, there is a unique $g : S \to \mathrm{Hom}_{\mathbf{Alg}}(A, \Omega)$ in $\mathbf{Spa}$ such that the following diagram commutes:



where $\mathrm{H}(S)$ denotes $\mathrm{Hom}_{\mathbf{Spa}}(S, \Omega)$, $\mathrm{H}(g)$ denotes $\mathrm{Hom}_{\mathbf{Spa}}(g, \Omega)$, $\mathrm{H}(A)$ denotes $\mathrm{Hom}_{\mathbf{Alg}}(A, \Omega)$ and $\mathrm{H} \circ \mathrm{H}(A)$ denotes $\mathrm{Hom}_{\mathbf{Spa}}(\mathrm{Hom}_{\mathbf{Alg}}(A, \Omega), \Omega)$. We first show that such g exists. Define $g : S \to \mathrm{Hom}_{\mathbf{Alg}}(A, \Omega)$ by $g(x)(a) = \Psi_S(x)(f(a))$ where $x \in S$ and $a \in A$. Then we have

$$\begin{aligned}(\mathrm{Hom}_{\mathbf{Spa}}(g, \Omega) \circ \Phi_A(a))(x) &= (\Phi_A(a) \circ g)(x) = g(x)(a) \\ &= \Psi_S(x)(f(a)) = f(a)(x).\end{aligned}$$

Thus, the above diagram commutes for this g. It remains to show that g is an arrow in **Spa**. For $a \in A$ and $O \in \mathcal{O}_\Omega$, we have

$$\begin{aligned} g^{-1}(\langle a \rangle_O) &= \{x \in S \,;\, g(x) \in \langle a \rangle_O\} \\ &= \{x \in S \,;\, g(x)(a) \in O\} \\ &= \{x \in S \,;\, f(a)(x) \in O\} \\ &= f(a)^{-1}(O). \end{aligned}$$

Since $f(a) \in \mathrm{Hom}_{\mathbf{Spa}}(S, \Omega)$ and since **Spa** is definable by a class of Boolean topological coaxioms, this implies that g is an arrow in **Spa**.

Finally, in order to show the uniqueness of such g, we assume that $g' : S \to \mathrm{Hom}_{\mathbf{Alg}}(A, \Omega)$ in **Spa** makes the above diagram commute. Then we have

$$f(a)(x) = (\mathrm{Hom}_{\mathbf{Spa}}(g', \Omega) \circ \Phi_A(a))(x) = (\Phi_A(a) \circ g')(x) = g'(x)(a).$$

Since we also have $f(a)(x) = g(x)(a)$, it follows that $g = g'$. This completes the proof. ◀

2.2.2 Deriving Equivalence from Adjunction

We briefly review standard methods to derive a (dual) equivalence from a (dual) adjunction. Assume that $F : \mathbf{C} \to \mathbf{D}$ is left adjoint to $G : \mathbf{D} \to \mathbf{C}$ with Φ and Ψ the unit and the counit of the adjunction, respectively.

▶ **Definition 2.20.** $\mathrm{Fix}(\mathbf{C})$ is a full subcategory of $\mathbf{C}$ such that $C \in \mathrm{Fix}(\mathbf{C})$ iff Φ_C is an isomorphism in $\mathbf{C}$. $\mathrm{Fix}(\mathbf{D})$ is a full subcategory of $\mathbf{D}$ such that $D \in \mathrm{Fix}(\mathbf{D})$ iff Ψ_D is an isomorphism in $\mathbf{D}$.

▶ Proposition 2.21. $\mathrm{Fix}(\mathbf{C})$ and $\mathrm{Fix}(\mathbf{D})$ are categorically equivalent. Moreover, this equivalence is the maximal one that can be derived from the adjunction between $\mathbf{C}$ and $\mathbf{D}$.

If we require a condition about the original adjunction, we have another way to describe $\mathrm{Fix}(\mathbf{C})$ and $\mathrm{Fix}(\mathbf{D})$. We first introduce the following notations.

▶ **Definition 2.22.** $\mathrm{Img}(\mathbf{C})$ is a full subcategory of $\mathbf{C}$ such that $C \in \mathrm{Img}(\mathbf{C})$ iff $C \simeq G(D)$ for some $D \in \mathbf{D}$. $\mathrm{Img}(\mathbf{D})$ is a full subcategory of $\mathbf{D}$ such that $D \in \mathrm{Img}(\mathbf{D})$ iff $D \simeq F(C)$ for some $C \in \mathbf{C}$.

▶ Proposition 2.23. Assume that $F(C) \in \mathrm{Fix}(\mathbf{D})$ for any $C \in \mathbf{C}$ and that $G(D) \in \mathrm{Fix}(\mathbf{C})$ for any $D \in \mathbf{D}$. It then holds that $\mathrm{Img}(\mathbf{C}) = \mathrm{Fix}(\mathbf{C})$ and $\mathrm{Img}(\mathbf{D}) = \mathrm{Fix}(\mathbf{D})$. Hence, $\mathrm{Img}(\mathbf{C})$ and $\mathrm{Img}(\mathbf{D})$ are categorically equivalent.

Note that the above assumption is satisfied in the case of the duality between spatial frames and sober topological spaces, and also in the case of a duality between spatial continuous lattices and sober convexity spaces, which is presented below.

3 Domain-Convexity Duality

In this section, we apply the general theory to obtain a dual adjunction between continuous lattices and convexity spaces, and then refine the dual adjunction into a dual equivalence between algebraic lattices and sober convexity spaces, which allows us to characterise the dual equivalence part of Jacobs' dual adjunction for algebras of the distribution monad, with the help of the notion of idempotency for those algebras.

3.1 Convexity-Theoretical Duality for Scott's Continuous Lattices

The concept of a continuous lattice is usually defined in terms of way-below relations: i.e., a continuous lattice is a complete lattice in which any element can be expressed as the join of those elements that are way-below it. From our perspective of duality between point-free and point-set spaces, another characterisation of continuous lattices is helpful:

▶ Proposition 3.1 (Theorem I-2.7 in [7])). A poset is a continuous lattice iff it satisfies the following: (i) it has directed joins including 0; (ii) it has arbitrary meets including 1; (iii) arbitrary meets distribute over directed joins.

The proposition above suggests that continuous lattices may be considered to be point-free convexity spaces; recall that a convexity space is a tuple $(S, \mathcal{C})$ where S is a set, and $\mathcal{C}$ is a subset of $\mathcal{P}(S)$ that is closed under directed unions and arbitrary intersections; $\mathcal{C}$ is called the convexity of the space. Many theorems in convex geometry such as Helly-type theorems (see [9]) can be treated in terms of convexity spaces with suitable conditions (see [6, 18]).

Just as a frame is a point-free abstraction of a topological space, so a continuous lattice is a point-free abstraction of a convexity space; this is what the proposition above tells us. Note that item 1 above is mathematically redundant, but suggests the definition of a homomorphism, which preserves directed joins and arbitrary meets.

This idea in turn suggests that there is a duality between **ContLat** and **Conv**. To apply our duality theory, recall that the continuous lattices are the algebras of the filter monad on **Set** (see [7]), and that the convexity spaces can be expressed as a full subcategory of $(\mathbf{Spa}(\mathcal{Q}))^{\mathrm{op}}$ that is definable by a class of classical topological coaxioms.

We can see $\mathbf{2}$ (i.e., $\{0, 1\}$) as a continuous lattice by its natural ordering $0 < 1$ and also as a convexity space by equipping it with the Sierpinski convexity $\{\emptyset, \{1\}, \mathbf{2}\}$. In order to show that homming into $\mathbf{2}$ gives us a dual adjunction between continuous lattices and convexity spaces, it suffices to verify the harmony condition. It is immediate because the harmony condition in this case boils down to the fact that $\mathrm{Hom}_{\mathbf{Conv}}(S, \mathbf{2})$, which can be seen as the set of convex sets in S, forms a continuous lattice. We then obtain the following theorem.

▶ **Theorem 3.2.** There is a dual adjunction between **ContLat** and **Conv**, given by contravariant functors $\mathrm{Hom}_{\mathbf{Conv}}(\text{-}, \mathbf{2})$ and $\mathrm{Hom}_{\mathbf{ContLat}}(\text{-}, \mathbf{2})$.

We can formally refine the dual adjunction into a dual equivalence in the canonical way as already discussed. It is non-trivial, however, to find an intrinsic description of the induced dual equivalence. We shall achieve it in the following. Although we do not have space to give proofs in this subsection, relevant proofs can be found in Maruyama [14]; note that it causes no essential change in proofs to replace convexity algebras in [14] with continuous lattices.

$\mathrm{Hom}_{\mathbf{Conv}}(X, \mathbf{2})$ can be seen as the collection of convex sets in X, so we write $\mathrm{Conv}(\text{-})$ for $\mathrm{Hom}_{\mathbf{Conv}}(\text{-}, \mathbf{2})$. Likewise, we write $\mathrm{Spec}(\text{-})$ for $\mathrm{Hom}_{\mathbf{ContLat}}(\text{-}, \mathbf{2})$, for the reason that $\mathrm{Hom}_{\mathbf{ContLat}}(L, \mathbf{2})$ can be seen as the collection of Scott-open meet-complete filters of L where meet-completeness is defined as closedness under arbitrary meets.

▶ **Definition 3.3.** We denote by $\Phi : \mathrm{Id}_{\mathbf{ContLat}} \to \mathrm{Conv} \circ \mathrm{Spec}$ and $\Psi : \mathrm{Id}_{\mathbf{Conv}} \to \mathrm{Spec} \circ \mathrm{Conv}$ the unit and counit of the dual adjunction between **ContLat** and **Conv**, respectively.

The question is when the unit Φ and the counit Ψ give isomorphisms.

We define the notion of spatiality of continuous lattices as the existence of enough Scott-open meet-complete filters:

▶ **Definition 3.4.** A continuous lattice L is spatial iff, for any $a, b \in L$ with $a \not\leq b$, there is a Scott-open meet-complete filter P of L such that $a \in P$ and $b \notin P$.

The following proposition is crucial.

▶ Proposition 3.5. A continuous lattice L is spatial iff $\Phi_L : L \to \mathrm{Conv} \circ \mathrm{Spec}(L)$ is an isomorphism.

Spatiality is characterised as algebraicity.

▶ Proposition 3.6. Let L be a continuous lattice. Then, L is spatial iff L is algebraic (i.e., every element can be expressed as the join of a directed set of compact elements).

Sober convexity spaces are defined in terms of polytopes, which make sense in general convexity spaces as follows.

▶ **Definition 3.7.** The convex hull $\mathrm{ch}(Y)$ of a subset Y of a convexity space $(X, \mathcal{C})$ is defined as $\bigcap\{Z \mid Z \in \mathcal{C} \text{ and } Y \subset Z\}$. Then, a polytope in a convexity space is defined as the convex hull of a set of finitely many points in it.

A convex set C in $\mathcal{C}$ is said to be directed-irreducible iff if $C = \bigcup_{i \in I} C_i$ for a directed subset $\{C_i \,;\, i \in I\}$ of $\mathcal{C}$ then there exists $i \in I$ such that $C = C_i$.

▶ Proposition 3.8. A convex subset of a convexity space is directed-irreducible iff it is a polytope.

Polytopes form a canonical basis for any convexity (if we assume the axiom of choice):

▶ Proposition 3.9. Any convex set in a convexity space can be expressed as the union of a directed set of polytopes.

Sobriety is defined as follows (polytopes may be replaced with directed-irreducible sets).

▶ **Definition 3.10.** A convexity space is sober iff every polytope in it is the convex hull of a unique point.

In contrast to the first impression, sobriety is a natural concept. Let us see examples.

▶ **Definition 3.11.** Given a convexity space S, we can equip the set of polytopes in S with the ideal convexity: i.e., a convex set is an ideal of the lattice of polytopes.

The space of polytopes is then sober, and gives the soberification of the original space. The space of polytopes corresponds to the space of irreducible varieties in algebraic geometry; to put it differently, "a unique point" above plays, in convex geometry, the role of "a generic point" in terms of algebraic geometry.

In algebraic geometry, we soberify a variety by adding irreducible varieties as additional generic points (in other words, the prime spectrum of the coordinate ring of a variety gives the soberification). In convex geometry, we soberify a space by adding polytopes as generic points (in other words, the prime spectrum of the lattice of convex sets gives the soberification). Here recall that polytopes can be characterised by directed-irreducibility.

▶ Proposition 3.12. For a convexity space S, S is sober iff Ψ_S is an isomorphism in **Conv**.

Let **SobConv** denote the category of sober convexity spaces and convexity preserving maps, **AlgLat** the category of algebraic lattices and homomorphisms, and **SpaContLat** the category of spatial continuous lattices and homomorphisms. We finally obtain the following.

▶ **Theorem 3.13.** **AlgLat** (= **SpaContLat**) and **SobConv** are dually equivalent.

3.2 Jacobs Duality for Algebras of the Distribution Monad

Let $\mathcal{D} : \mathbf{Set} \to \mathbf{Set}$ be the distribution monad on **Set**. The object part is defined by:

$$\mathcal{D}(X) := \left\{ f : X \to [0,1] \mid \sum_{x \in X} f(x) = 1 \text{ and } f \text{ has a finite support} \right\}.$$

The arrow part is defined by:

$$\mathcal{D}(f : X \to Y)(g : X \to [0,1])(y) = \sum_{f(x)=y} g(x).$$

As in [8, 11], algebras of $\mathcal{D}$ can concretely be described as barycentric algebras, which are basically sets with convex combination operations; the precise definition is given below.

Jacobs [11] shows a dual adjunction between preframes and algebras of $\mathcal{D}$. We first observe that we can restrict the category of preframes into the category of continuous lattices, since the dual of an algebra of $\mathcal{D}$ (i.e., $\mathrm{PF}(X)$ below) is actually a continuous lattice. And then we characterise the induced dual equivalence via the concept of idempotent algebras of $\mathcal{D}$.

▶ **Definition 3.14.** A $\mathcal{D}$-algebra (aka. barycentric algebra) is a set X with a ternary function

$$\langle \text{-}, \text{-}, \text{-} \rangle : [0,1] \times X \times X \to X$$

such that

1. $\langle r, x, x \rangle = x$;
2. $\langle 0, x, y \rangle = y$;
3. $\langle r, x, y \rangle = \langle 1 - r, y, x \rangle$;
4. $\langle r, x, \langle s, y, z \rangle \rangle = \langle r + (1-r)s, \langle r/(r + (1-r)s), x, y \rangle, z \rangle$.

Morphisms of $\mathcal{D}$-algebras are affine maps, i.e., maps f preserving $\langle \text{-}, \text{-}, \text{-} \rangle$ in the following way:

$$f(\langle r, x, y \rangle) = \langle r, f(x), f(y) \rangle.$$

$\mathbf{Alg}(\mathcal{D})$ denotes the category of $\mathcal{D}$-algebras and affine maps.

$\mathbf{Alg}(\mathcal{D})$ in the sense above is equivalent to the Eilenberg-Moore category of the distribution monad (see [11, 8]).

Semilattices can be regarded as $\mathcal{D}$-algebras in a canonical way.

▶ Proposition 3.15. Any meet-semilattice L forms a $\mathcal{D}$-algebra: define $\langle \text{-}, \text{-}, \text{-} \rangle : [0,1] \times L \times L \to L$ by

$$\langle r, x, y \rangle = x \wedge y$$

if $r \in (0,1)$; otherwise, define $\langle r, x, y \rangle = x$ if $r = 1$, and $\langle r, x, y \rangle = y$ if $r = 0$. Similarly, any join-semilattice forms a $\mathcal{D}$-algebra (by replacing $\wedge$ above with $\vee$).

In the following, we suppose any semilattice is equipped with the convex structure defined in the proposition above. We review the following concepts from Jacobs [11].

▶ **Definition 3.16.** For a $\mathcal{D}$-algebra $(X, \langle \text{-}, \text{-}, \text{-} \rangle)$, a subset Y of X is defined as

- a subalgebra iff $y_1, y_2 \in Y$ implies that for any $r \in [0,1]$, $\langle r, y_1, y_2 \rangle \in Y$;
- a filter iff $\langle r, x_1, x_2 \rangle \in Y$ and $r \neq 0, 1$ together imply both $x_1 \in Y$ and $x_2 \in Y$;
- a prime filter iff it is both a subalgebra and a filter.

Let us define a contravariant functor

$$\mathrm{PF}(\text{-}) : \mathbf{Alg}(\mathcal{D})^{\mathrm{op}} \to \mathbf{ContLat}.$$

For a $\mathcal{D}$-algebra X, $\mathrm{PF}(X)$ is the lattice of prime filters of X. For an affine map f, we let $\mathrm{PF}(f) = f^{-1}$.

We define a contravariant functor

$$\mathrm{Sp}(\text{-}) : \mathbf{ContLat}^{\mathrm{op}} \to \mathbf{Alg}(\mathcal{D})$$

as follows. For a continuous lattice L, define $\mathrm{Sp}(L)$ as the set of Scott-open meet-complete filters of L, equipped with a meet-semilattice structure by finite intersections, and hence with a $\mathcal{D}$-algebra structure (see Proposition 3.15). For a homomorphism f, we let $\mathrm{Sp}(f) = f^{-1}$.

Since $\mathrm{PF}(X)$ always forms a continuous lattice, the methods of Jacobs [11] completely work in the present situation, thus yielding the following dual adjunction theorem.

▶ **Theorem 3.17.** There is a dual adjunction between **ContLat** and $\mathbf{Alg}(\mathcal{D})$, given by $\mathrm{Sp} : \mathbf{ContLat}^{\mathrm{op}} \to \mathbf{Alg}(\mathcal{D})$ and $\mathrm{PF} : \mathbf{Alg}(\mathcal{D})^{\mathrm{op}} \to \mathbf{ContLat}$.

In the following, we aim at identifying the dual equivalence induced by the dual adjunction above. Towards this end, we introduce the concept of idempotent $\mathcal{D}$-algebras.

▶ **Definition 3.18.** A $\mathcal{D}$-algebra $(X, \langle \text{-}, \text{-}, \text{-} \rangle)$ is idempotent iff for any $x, y \in X$, and for any $r, s \in (0, 1)$ (i.e., the open unit interval),

$$\langle r, x, y \rangle = \langle s, x, y \rangle.$$

It is straightforward to see the following.

▶ Proposition 3.19. Any meet-semilattice and join-semilattice form an idempotent $\mathcal{D}$-algebra. In particular, $\mathrm{Sp}(L)$ is an idempotent $\mathcal{D}$-algebra.

▶ Proposition 3.20. For a $\mathcal{D}$-algebra X, $\mathrm{PF}(X)$ is an algebraic lattice.

Proof. This follows from the fact that $\mathrm{PF}(X)$ is a subalgebra of the powerset algebraic lattice $\mathcal{P}(X)$ with respect to directed unions and arbitrary intersections, and that the class of all algebraic lattices is closed under subalgebras. ◀

▶ Proposition 3.21. If L is an algebraic lattice, then L is isomorphic to $\mathrm{PF} \circ \mathrm{Sp}(L)$.

Proof. Firstly, $\mathrm{Sp}(L)$ can be regarded as $\mathrm{Hom}_{\mathbf{ContLat}}(L, \mathbf{2})$ by identifying subsets with their characteristic functions. Likewise, $\mathrm{PF} \circ \mathrm{Sp}(L)$ can be thought of as $\mathrm{Hom}_{\mathbf{Conv}}(\mathrm{Sp}(L), \mathbf{2})$. Then, the previously obtained duality between algebraic lattices and sober convexity spaces immediately tells us that L is indeed isomorphic to $\mathrm{PF} \circ \mathrm{Sp}(L)$. ◀

▶ Proposition 3.22. If X is an idempotent $\mathcal{D}$-algebra, then X is isomorphic to $\mathrm{Sp} \circ \mathrm{PF}(X)$.

Proof. For $x, y \in X$, define $x \wedge y$ by $\langle 1/2, x, y \rangle$. By idempotency, X with $\wedge$ forms a meet-semilattice. Since $\mathrm{Sp} \circ \mathrm{PF}(X)$ is an idempotent $\mathcal{D}$-algebra by Proposition 3.19, $\mathrm{Sp} \circ \mathrm{PF}(X)$ also forms a meet-semilattice in the same way. It holds that if the meet-semilattices of two idempotent $\mathcal{D}$-algebras are isomorphic, then the original $\mathcal{D}$-algebras are isomorphic as well. Thus, it suffices to prove that X is isomorphic to $\mathrm{Sp} \circ \mathrm{PF}(X)$ as a meet-semilattice.

Now, X can in turn be equipped with the ideal convexity: the convex sets are defined as the ideals of X. Then, X is a sober convexity space (with the convex sets of X forming an algebraic lattice). The polytopes of X, denoted $\mathrm{Poly}(X)$, form a join-semilattice: for two polytopes $\mathrm{ch}(X)$ and $\mathrm{ch}(Y)$ with X, Y finite, their join is defined as $\mathrm{ch}(X \cup Y)$, where recall ch(-) denotes the convex hull operation. And then $\mathrm{Poly}(X)^{\mathrm{op}}$, the order dual of the polytope join-semilattice $\mathrm{Poly}(X)$, is actually isomorphic to X as a meet-semilattice; this holds for any meet-semilattice X by an equivalence between the categories of join-semilattices and of sober convexity spaces as remarked in Maruyama [14]. Since $\mathrm{PF}(X)$ is the lattice of ideals of X, it turns out that $\mathrm{Sp} \circ \mathrm{PF}(X)$ is the meet-semilattice of compact (aka. directed-irreducible) elements of the ideal lattice, which is precisely $\mathrm{Poly}(X)^{\mathrm{op}}$ (see Proposition 3.8); recall $\mathrm{Poly}(X)^{\mathrm{op}}$ is isomorphic to X, and the proof is done. ◀

Let $\mathbf{IdemAlg}(\mathcal{D})$ denote the category of idempotent $\mathcal{D}$-algebras. Propositions 3.21 and 3.22 above finally give us the following theorem identifying the dual equivalence part of the dual adjunction between $\mathbf{ContLat}$ and $\mathbf{Alg}(\mathcal{D})$.

▶ **Theorem 3.23.** The dual adjunction between $\mathbf{ContLat}$ and $\mathbf{Alg}(\mathcal{D})$ restricts to a dual equivalence between $\mathbf{AlgLat}$ and $\mathbf{IdemAlg}(\mathcal{D})$. This is the largest dual equivalence induced by the dual adjunction.

Since the converses of Propositions 3.21 and 3.22 hold by Propositions 3.19 and 3.20 respectively, the theorem above gives the maximal dual equivalence that can result from restricting the dual adjunction between $\mathbf{ContLat}$ and $\mathbf{Alg}(\mathcal{D})$. Although we do not have space to work out details, the duality above is closely related to the classic Hofmann-Mislove-Stralka duality [10]; indeed, the duality above reveals a convexity-theoretical aspect of the Hofmann-Mislove-Stralka duality.

Summing up, we have obtained the following dualities in this section:

$$\mathbf{IdemAlg}(\mathcal{D}) \simeq \mathbf{AlgLat}^{\mathrm{op}} \simeq \mathbf{SobConv}.$$

It thus follows that $\mathbf{IdemAlg}(\mathcal{D}) \simeq \mathbf{SobConv}$; behind the equivalence, we actually have an adjunction between $\mathbf{IdemAlg}(\mathcal{D})$ and $\mathbf{Conv}$. And the functor from $\mathbf{Conv}$ to $\mathbf{IdemAlg}(\mathcal{D})$ has clear meaning in terms of polytopes: it maps a convexity space S to the $\mathcal{D}$-algebra of $\mathrm{Poly}(S)^{\mathrm{op}}$, i.e., the order dual of the polytope join-semilattice of S. These domain-convexity dualities tell us that domain theory and convex geometry are naturally intertwined in the (sometimes beautiful, sometimes insane) universe of mathematics.

References

1 J. Adámek, H. Herrlich, and G. E. Strecker, *Abstract and Concrete Categories*, John Wiley and Sons, Inc., 1990.
2 J. Adámek and J. Reiterman, Topological categories presented by small sets of axioms, *Journal of Pure and Applied Algebra* 42 (1986) 1-14.
3 M. Barr, J. F. Kennison and R. Raphael, Isbell duality, *Theory and Applications of Categories* 20 (2008) 504-542.
4 G. C. L. Brümmer, Topological categories, *Topology and its Applications*, 18 (1984) 27-41.
5 D. M. Clark and B. A. Davey, *Natural Dualities for the Working Algebraist*, CUP, 1998. *Ann. Math. Artif. Intell.* 56 (2009) 339-360.
6 W. A. Coppel, *Foundations of Convex Geometry*, CUP, 1998.
7 G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. W. Mislove, and D. S. Scott, *Continuous Lattices and Domains*, CUP, 2003.
8 T. Fritz, Convex Spaces I: Definition and Examples, arXiv:0903.5522.
9 P. M. Gruber and J. M. Wills (eds.), *Handbook of Convex Geometry Vol. A. B*, North-Holland, 1993.
10 K. H. Hofmann, M. Mislove and A. Stralka, *The Pontryagin Duality of Compact* 0-*Dimensional Semilattices and its Applications*, Lecture Notes in Math. 394, Springer, 1974.
11 B. Jacobs, Convexity, Duality, and Effects, *Proc. of 6th IFIP International Conference on Theoretical Computer Science* (2010) 1-19.
12 P. T. Johnstone, *Stone Spaces*, CUP, 1986.
13 E. G. Manes, *Algebraic Theories*, Springer, 1976.
14 Y. Maruyama, Fundamental results for pointfree convex geometry, *Ann. Pure Appl. Logic* 161 (2010) 1486-1501.
15 Y. Maruyama, From operational duality to coalgebraic quantum symmetry, Proc. of CALCO'13, *Springer LNCS* 8089 (2013) 220-235.

16 J. W. Pelletier and J. Rosický, Generating the equational theory of C^*-algebras and related categories, *Proc. of Conference on Categorical Topology* (1989) 163-180.
17 H.-E. Porst and W. Tholen, Concrete dualities, *Category Theory at Work* (1991) 111-136.
18 M. L. J. van de Vel, *Theory of Convex Structures*, North-Holland, 1993.

A Categorical Duality as Philosophy of Space

We finally speculate on categorical duality in a broader context, and attempt to elucidate conceptual foundations of categorical duality, especially in relation to philosophy of space. Let us start with the following (rough) picture of categorical dualities in diverse disciplines, almost all of which may be conceived of as arising between the epistemological and the ontological. The concept of duality between ontology and epistemology, we think, yields a unifying perspective on categorical dualities in wide-ranging fields; it is like "duality between the conceptual and the formal" in Lawvere's terms in his seminal hyperdoctrine paper "Adjointness in Foundations" (more links with other thinkers shall be pursued afterwards).

	Ontological	Epistemological	Duality
Logic	Model	Theory	Stone
Algebraic Logic	Algebraic Semantics	Logical System	Tarski
Complex Geometry	Riemann Surface	Algebraic Function Field	Riemann
Classical Alg. Geom.	Variety over k	k-Algebra	Hilbert
Modern Alg. Geom.	Scheme	Ring	Grothendieck
Representation Theory	Group	Representation	Pontryagin
Topology	Point	Open Set	Isbell, Papert
Convex Geometry	Point	Convex Set	Maruyama
Galois Theory	(Profinite) G-set	Algebra Extension	Galois
Program Semantics	Denotation of Program	Observable Property	Abramsky
System Science	Computer System	Its Behaviour	Coalg|Alg
General Relativity	Spacetime Manifold	Field	Weyl
Quantum Physics	State	Observable	Gelfand

The aforementioned duality between denotations and observable properties of programs basically amounts to domain-theoretical variants of infinitary Stone dualities, such as the Isbell-Papert one. The duality theory of the present paper is relevant to finitary and infinitary Stone, Gelfand, Pontryagin, and even Hilbert dualities (because Hilbert and Stone dualities are closely related as discussed below). The above duality between computer systems and their behaviours boils down to algebra-coalgebra duality in mathematical terms. The most basic case is the Abramksy duality between modal algebras and coalgebras of the Vietoris endofunctor on the category of Stone spaces, which was later rediscovered and explicated by Kupke-Kurz-Venema. A universal-algebraic general theory of such algebra-coalgebra dualities is developed in the author's previous paper "Natural Duality, Modality, and Coalgebra" in *Journal of Pure and Applied Algebra*.

Some of the dualities above are tightly intertwined as a matter of fact. The Stone duality for classical logic is precisely equivalent to a Hilbert duality for geometry over $\mathbb{GF}(2)$ (i.e., the prime field of two elements). Furthermore, logical completeness for classical logic corresponds to Nullstellensatz for geometry over $\mathbb{GF}(2)$, in a mathematically rigorous manner; note that this is different from the model-theoretic correspondence between logic and algebraic geometry. Here it should be noted that logical completeness tells us a poset duality between models and theories, and Nullstellensatz a poset duality between affine varieties and radical ideals, which can be upgraded into the corresponding categorical dualities, namely the Stone

duality and the Hilbert duality, respectively (note that the Stone duality is a generalisation of completeness, the syntax-semantics equivalence, in a mathematically precise sense). In this sense, completeness and Nullstellensatz may be said to be "predualities." The correspondence between logic and algebraic geometry may be summarised as follows:

	Logic	Algebraic Geometry
Algebra	Formulae	Polnomials
Spectrum	Models	Variety
Poset Duality	Completeness	Nullstellensatz
Categorical Duality	Stone Duality	Hilbert Duality

The author's recent investigation shows that this correspondence between logic and algebraic geometry extends to $\mathbb{GF}(p^n)$-valued logic and geometry over $\mathbb{GF}(p^n)$ where p is a prime number, and n is an integer more than 1 (and $\mathbb{GF}(p^n)$ is the Galois field of order p^n).

The concept of space has undergone a revolution in the modernisation of mathematics, shifting the emphasis from underlying point-set spaces to algebraic structures upon them, to put it more concretely, from topological spaces to locales (or formal topology as predicate locales), toposes, schemes (i.e., sheaves of rings), and non-commutative point-free spaces (such as C^* and von Neumann algebras). Categorical duality has supported and eased this shift from point-set to point-free space, since it basically tells us the algebraic point-free structure on a point-set space keeps the same amount of information as the original point-set space, allowing us to recover the points as the spectrum of the algebraic structure.

Having seen different categorical dualities seemingly share certain conceptual essence, it would be natural to ask where the (mathematical) origin of those dualities lie, even though there may be no single origin, and the concept of origin *per se* may be misguided. Since duality allows us to regard algebra itself as (point-free) space, another relevant question is where the origin of the shift from point-set to point-free space is.

The first mathematician who elucidated the point could be Riemann, who proved (what is now called) a Riemann surface can be recovered from its function field. At the same time, however, we may think of several serious contenders, especially Kronecker and Dedekind-Weber on the one hand, who are considered (e.g., by Harold Edwards) to be precursors of arithmetic geometry, and Brouwer on the other. Whilst it seems Riemann did not take algebra itself to be space, Kronecker and Dedekind-Weber indeed algebraised complex geometry (e.g., the Riemann-Roch theorem), considering algebraic function fields *per se* to be (equivalents of) spaces (and uncovering a grand link with algebraic number theory, the crucial analogy between algebraic number fields and function fields). Brouwer, even though coming into the scene later than them, vigorously formulated and articulated, in terms of so-called spreads and choice sequences, the notion of continuums that does not presuppose point, transforming a bare, speculative idea into a full-fledged, mathematically substantial enterprise.

Comparable shifts seem to have been caused in philosophy as well. Whitehead's process philosophy puts more emphasis on dynamic *processes* than static *substances*. His philosophy of space is, in its spirit, very akin to the idea of point-free topology:

> Whitehead's basic thought was that we obtain the abstract idea of a spatial point by considering the limit of a real-life series of volumes extending over each other, for example in much the same way that we might consider a nested series of Russian dolls or a nested series of pots and pans. However, it would be a mistake to think of a spatial point as being anything more than an abstraction.

This is from Irvine's *Stanford Encyclopedia of Philosophy* article on Whitehead, and may indeed be read as a brilliant illustration of the idea of prime ideals (or filter) as points in

duality theory and algebraic geometry: recall that the open neigbourhoods of a point in a topological space form a completely prime filter of its open set locale, with the complement yielding a prime ideal. Although Whitehead's philosophy of space tends to be discussed in the context of mereology, which is sort of peculiar mathematics, nevertheless, it is indeed highly relevant to the core idea of modern geometry in mainstream mathematics; the idea of points as prime ideals is particularly important in algebraic and non-commutative geometry.

Whitehead's process philosophy would be relevant to category theory in general: for example, John Baez asserts "a category is the simplest framework where we can talk about systems (objects) and processes (morphisms)" in his paper "Physics, Topology, Logic and Computation: A Rosetta Stone." Abramsky-Coecke's categorical quantum mechanics follows a similar line of idea, regarding a †-compact category as a "universe" of quantum processes expressed in an intuitively meaningful graphical language; this is Bob Coecke's quantum picturalism. At the same time, however, we must be aware of the possibility that formalisation distorts or misses a crucial point of an original philosophical idea. Indeed, Whitehead's concept of process would ultimately be unformalisable by its nature. This remark is applicable throughout the whole discussion here, and we have to be cautious of distortion via formalisation, a common mistake the mathematician or logician tends to make.

Yet another point-free philosopher of space is Wittgenstein: "What makes it apparent that space is not a collection of points, but the realization of a law?" (*Philosophical Remarks*, p. 216). Wittgenstein's intensional view on space is a compelling consequence of his persistent disagreement with the set-theoretical extensional view of mathematics:

> Mathematics is ridden through and through with the pernicious idioms of set theory. One example of this is the way people speak of a line as composed of points. A line is a law and isn't composed of anything at all (*Philosophical Grammar*, p. 211).

What does he mean by "law"? Brouwer defined his concept of a spread as a certain law to approximate a "point", and this could possibly be a particular case of Brouwer's influence on Wittgenstein's philosophy. More detailed discussion is in my paper: "Wittgenstein's Conception of Space and the Modernist Transformation of Geometry via Duality", *Papers of 36th International Wittgenstein Symposium*, Austrian Wittgenstein Society, 2013.

Where is the philosophical origin of such a mode of thinking? Just as remarked in the case of the mathematical origin, there may be no single origin, and it might even be wrong to seek an origin at all. Certain postmodern philosophers assert that the idea of the original tends to be invented through a number of copies: after all, there may only be copies having no origin or essence in common. Anyway, we could just envisage a bunch of family-resemblant copies (possibly sharing no genuine feature in common at all) in the form of a series of dichotomies:

Cassirer Shift	Substance	Function
Whitehead Shift	Material	Process
Brouwer Shift	Point	Choice Sequence
Wittgenstein Shift	Tractatus	Investigations
Bohr Shift	Classical Realism	Complementarity
Gödel Shift	Right	Left
Lawvere Duality	Conceptual	Formal
Granger Duality	Object	Operation
Zeno Paradox	Continuous	Discrete
Aristotle	Matter	Form
Natural Philosophy	Newton	Leibniz
Kant	Thing Itself	Appearance
Phenomenology	Object	Subject
Theory of Meaning	Davidson	Dummett

"Shift" means that each thinker emphasises in his dichotomy the shift from a concept on the left-hand side to that on the right-hand side. Cassirer could possibly be a philosophical origin of the modernist shift discussed so far. In contrast with "shift", "duality" does not imply anything on which concept is prior to the other; rather, it does suggest equivalence between two views concerned. Finally, no uniform relationships are intended to hold between two concepts in each of the rest of dichotomies, which do not particularly focus upon shifting from one concept to the other. It would be of conceptual significance to reflect upon the table of categorical dualities in the light of these philosophical dichotomies.

Duality is more than dualism, just as categorical duality in point-free geometry starts with the dualism of space and then tells us that the two conceptions of space are equivalent via functors (in a sense reducing dualism to monism; or it could be called monism on the top of dualism). Category theory often goes beyond dualism. Other sorts of dualism include "geometry vs. algebra", and "model-theoretic vs. proof-theoretic semantics." For instance, the concept of algebras of monads even encompasses geometric structures such as topological spaces and convex structures. Categorical logic tells us model-theoretic semantics amounts to interpreting logic in set-based categories, and proof-theoretic semantics to interpreting logic in so-called syntactic categories. We may thus say category theory transcends dualism.

Gödel's shift from "right" to "left" would need to be explicated. In his "The modern development of the foundations of mathematics in the light of philosophy", Gödel says:

> [T]he development of philosophy since the Renaissance has by and large gone from right to left [...] Particularly in physics, this development has reached a peak in our own time, in that, to a large extent, the possibility of knowledge of the objectivisable states of affairs is denied, and it is asserted that we must be content to predict results of observations. This is really the end of all theoretical science in the usual sense [...]

In the physical context, thus, Gödel's "right" means the emphasis of reality, substance, and the like, and "left" something like observational phenomena. Turning into other contexts, Gödel says metaphysics is "right", and formal logic is "left" in his terminology.

We finally articulate three senses of foundations of mathematics, thereby arguing that philosophy of space counts as foundations mathematics in one of the three senses. A popular, prevailing conception of foundations of mathematics is what may be called a "Reductive Absolute Foundation", which reduces everything to one framework, giving an absolute, domain-independent context to work in. The most popular one is (currently) set-theoretical foundations, but category theory (e.g., Lawvere's ETCS and toposes) can do the job as well.

Category theory can give another sense of foundation. That is a "Structural Relative Foundation", which changes a framework according to our structural focus (and see what remains *invariant*, and what does not), and gives a relative, domain-specific context to work in: e.g., ribbon categories for foundations of knot theory and †-compact categories for foundations of quantum mechanics and information (in these two cases, certain monoidal or linear logical structures are shared and invariant). Recall Grothendieck's relative point of view, and that change of base is a fundamental idea of category theory. The reductive-structural distinction is taken from Prawitz' notions of reductive and structural proof theory.

Philosophy of space as discussed above, we believe, counts as a "Conceptual Foundation of Mathematics", which aims at elucidating the nature of fundamental concepts in mathematics, and, presumably, are compelling for the working mathematician as well (where we basically mean mathematical space rather than physical or intuitive space). In this strand, the author's Categorical Universal Logic proposes a logical universal concept of space to unify toposes and quantum space categories in terms of monad-relativised Lawvere hyperdoctrines, allowing us

to reconcile Abramsky-Coecke's categorical quantum mechanics and Birkhoff-von Neumann's traditional quantum logic, which have (slightly misleadingly) been claimed to be in conflict with each other (several papers on CUL are available on the author's webpage).

Axiomatizing Subtyped Delimited Continuations

Marek Materzok

University of Wrocław
Wrocław, Poland
marek.materzok@cs.uni.wroc.pl

Abstract

We present direct equational axiomatizations of the call-by-value lambda calculus with the control operators $shift_0$ and $reset_0$ that generalize Danvy and Filinski's *shift* and *reset* in that they allow for abstracting control beyond the top-most delimited continuation. We address an untyped version of the calculus as well as a typed version with effect subtyping. For each of the calculi we present a set of axioms that we prove sound and complete with respect to the corresponding CPS translation.

1998 ACM Subject Classification D.3.3 Language Constructs and Features

Keywords and phrases Delimited Continuations, Continuation Passing Style, Axiomatization

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.521

1 Introduction

Control operators for delimited continuations allow to alter the control flow of programs by capturing the current continuation as a first-class value, which can be activated later. The most well-known are *shift*/*reset*, introduced by Danvy and Filinski in [3]. They have many important applications, including representing monads, partial evaluation, mobile computing, linguistics and operating systems [8].

The $shift_0/reset_0$ control operators were first introduced besides the well-known *shift*/*reset* by Danvy and Filinski in [3]. The operators were recently found to have many desirable properties [8]. They can, like *shift*/*reset*, be described with a CPS translation. They have an interesting type system, which distinguishes between side-effect free and effectful terms. They can express the whole CPS hierarchy, in both typed and untyped settings [9]. And they recently helped to construct a theory of multiple prompts [4].

We are interested in the problem of reasoning directly about code using the $shift_0/reset_0$ control operators. Specifically, we look for a set of equational axioms which are sound and complete with respect to the CPS translation – and thus allow for the same reasoning which is possible on the CPS code. Previously, Sabry and Felleisen have given such axioms for call-by-value lambda calculus with *call/cc* [10][12]. Kameyama and Hasegawa solved the problem for *shift*/*reset* control operators [6]. Axioms for the CPS Hierarchy were given by Kameyama [5].

In this paper, we present the axiomatization for $shift_0/reset_0$ control operators, and prove soundness and completeness with respect to the CPS translation. We do this both in the untyped setting and in the typed setting with effect subtyping, where we use a type-directed selective CPS translation which takes subtyping into account. The proof method is a variation of the one presented by Sabry in [11]. Crucial for the proof is the \$ control operator, which was described and formalized in [9].

The paper is organized as follows. We introduce the $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$ languages and their CPS translations in Section 2. Our untyped axiomatizations for the two languages are given in

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 521–539

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$$
\begin{array}{rcll}
\mathcal{C}[\![x]\!] & = & \lambda k.\, k\, x & \\
\mathcal{C}[\![\lambda x.\, e]\!] & = & \lambda k.\, k\, (\lambda x.\, \mathcal{C}[\![e]\!]) & \\
\mathcal{C}[\![e_1\, e_2]\!] & = & \lambda k.\, \mathcal{C}[\![e_1]\!]\, (\lambda f.\, \mathcal{C}[\![e_2]\!]\, (\lambda x.\, f\, x\, k)) & \\
\mathcal{C}[\![\mathcal{S}_0 k.\, e]\!] & = & \lambda k.\, \mathcal{C}[\![e]\!] & \\
\mathcal{C}[\![\langle e \rangle]\!] & = & \mathcal{C}[\![e]\!]\, (\lambda x.\, \lambda k.\, k\, x) & \textit{for } \lambda_{\mathcal{S}_0} \\
\mathcal{C}[\![e_1\, \$\, e_2]\!] & = & \lambda k.\, \mathcal{C}[\![e_1]\!]\, (\lambda f.\, \mathcal{C}[\![e_2]\!]\, f\, k) & \textit{for } \lambda_{\$}
\end{array}
$$

Figure 1 CPS translations for $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$.

Section 3. We describe the typed variants of the two languages, $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$, in Section 4. We present the typed axiomatizations in Section 5. In Section 6 we discuss the axioms and relate them to axioms for different systems of delimited continuations. Finally, we conclude in Section 7.

2 The languages $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$

Before we present the main results of the paper, we need to introduce and formally describe the languages used.

2.1 The language $\lambda_{\mathcal{S}_0}$ (*shift*$_0$/*reset*$_0$)

First, we define syntactic categories for expressions, values and evaluation contexts in the language $\lambda_{\mathcal{S}_0}$:

$$
e \;::=\; v \mid \mathcal{S}_0 x.\, e \mid e\, e \mid \langle e \rangle \qquad v \;::=\; x \mid \lambda x.\, e \qquad E \;::=\; \bullet \mid E\, e \mid v\, E
$$

The evaluation contexts are represented inside-out, which is formalized by the following definition of plugging a term inside a context:

$$
\bullet[e] \;=\; e \qquad (E\, e_2)[e] \;=\; E[e\, e_2] \qquad (v\, E)[e] \;=\; E[v\, e]
$$

The *shift*$_0$ operator $\mathcal{S}_0 x.\, \cdot$ captures the surrounding context up to (and including) the nearest dynamically surrounding delimiter $\langle \cdot \rangle$, which is then removed. The delimiter can also be removed when the enclosed expression is a value. This is displayed by the following reduction rules:

$$
\langle E[\mathcal{S}_0 x.\, e] \rangle \;\to\; e[\lambda x.\, \langle E[x] \rangle / x] \qquad \langle v \rangle \;\to\; v
$$

The language has also the standard beta and eta reductions. We will not concern ourselves more about the reduction rules, because the subject of this paper is the CPS semantics.

The CPS translation for the language $\lambda_{\mathcal{S}_0}$ is shown in Figure 1. (Please ignore for the moment the line marked "for $\lambda_{\$}$".) This is the untyped CPS translation first defined in [8] and proven correct with respect to the reduction rules.

The idea behind the translation is that the successive lambda abstractions (introduced by the translation of *shift*$_0$) bind continuations delimited by successive *reset*$_0$'s. It can be thought of as an infinitely-iterated (on the final answer position, like in the CPS Hierarchy) CPS translation, but eta reduced. In other words, we have potentially infinite number of continuation „levels", and *shift*$_0$/*reset*$_0$ operators allow us to change the current level.

2.2 The language $\lambda_{\$}$ (shift$_0$/\$)

The language $\lambda_{\$}$ was first defined in [9]. It is a generalization of $\lambda_{\mathcal{S}_0}$, a variant of which was explored by Kiselyov and Shan in [7]. The language plays a vital role in proving completeness of the axioms for $\lambda_{\mathcal{S}_0}$, and it is very interesting on its own. We describe it in this subsection.

As with $\lambda_{\mathcal{S}_0}$, we begin with introducing syntactic categories for expressions, values and evaluation contexts:

$$e \quad ::= \quad v \mid \mathcal{S}_0 x.\, e \mid e\, e \mid e \,\$\, e \qquad v \quad ::= \quad x \mid \lambda x.\, e \qquad E \quad ::= \quad \bullet \mid E\, e \mid v\, E \mid E \,\$\, e$$

We see that the *reset*$_0$ operator $\langle\cdot\rangle$ from $\lambda_{\mathcal{S}_0}$ was replaced by the (right-associative) binary operator \$. It is a generalization of *reset*$_0$: while the expression $\langle e\rangle$ means "evaluate e inside a new, empty context", $e_1 \,\$\, e_2$ means "evaluate e_2 inside a context terminated with e_1". We can express *reset*$_0$ using \$ by writing $(\lambda x.\, x) \,\$\, e$ instead of $\langle e\rangle$.

The \$ operator allows to easily restore captured contexts: the expression $\mathcal{S}_0 k.\, k \,\$\, e$ (where $k \notin \mathrm{V}(e)$) means the same as e. (We will make this statement formal in the following section.)

We define plugging terms inside evaluation contexts as follows:

$$\begin{array}{rclcrcl} \bullet[e] & = & e & & (v\, E)[e] & = & E[v\, e] \\ (E\, e_2)[e] & = & E[e\, e_2] & & (E \,\$\, e_2)[e] & = & E[e \,\$\, e_2] \end{array}$$

We have the following reduction rules for *shift*$_0$/\$, which generalize the reduction rules for *shift*$_0$/*reset*$_0$:

$$v \,\$\, E[\mathcal{S}_0 x.\, e] \quad \to \quad e[\lambda x.\, v \,\$\, E[x]/x] \qquad\qquad v_1 \,\$\, v_2 \quad \to \quad v_1\, v_2$$

Again, as with $\lambda_{\mathcal{S}_0}$, we give the meaning of $\lambda_{\$}$ terms with a CPS translation. It is shown in Figure 1.

The translation differs from the one for $\lambda_{\mathcal{S}_0}$ (shown in the same figure) only on the rule for the delimiter: while in the translation for *reset*$_0$ the translated subexpression has the (CPS-translated) identity function applied to it, in the translation for \$ the translated left subexpression is evaluated and applied to the translated right subexpression.

3 Untyped axiomatization

In this section we present sound and complete (with respect to the untyped translations of Figure 1) equational axiomatizations of $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$.

3.1 The axioms for $\lambda_{\mathcal{S}_0}$

The axioms for $\lambda_{\mathcal{S}_0}$ are presented in Figure 2. The first two (β_v and η_v) are the standard beta-value conversions. The third (β_Ω) is a beta-conversion restricted to evaluation contexts. The fourth and fifth ($\langle\mathcal{S}_0\rangle$ and $\langle v\rangle$) are equational versions of the reductions from the reduction semantics. These five axioms are standard and expected, the last two are interesting.

The axiom $\eta_{\langle\cdot\rangle}$ says that it is always possible to capture a continuation and restore it without changing the meaning of the expression. The axiom implies the existence of a potentially infinite tower of *reset*$_0$'s outside any expression. This is expected – the untyped CPS translation of Figure 1 is related to infinitely-iterated standard CPS translation, as discussed before in Section 2.1. The axiom is similar to Kameyama and Hasegawa's $\mathcal{S}$-elim; we discuss the connection in Section 6.1.

The last axiom, $\langle\lambda\rangle$, asserts that in expressions of the form $\langle(\lambda x.\, e_1)\, e_2\rangle$ we know that the topmost continuation for e_1 must be empty, and we can always throw it away and replace it with a new empty continuation.

$$
\begin{array}{rcll}
(\lambda x.\, e)\, v & = & e[v/x] & (\beta_v) \\
\lambda x.\, v\, x & = & v \quad x \notin \mathrm{V}(v) & (\eta_v) \\
(\lambda x.\, E[x])\, e & = & E[e] \quad x \notin \mathrm{V}(E) & (\beta_\Omega) \\
\langle E[\mathcal{S}_0 x.\, e]\rangle & = & e[\lambda x.\, \langle E[x]\rangle/x] \quad x \notin \mathrm{V}(E) & (\langle \mathcal{S}_0 \rangle) \\
\langle v\rangle & = & v & (\langle v \rangle) \\
\mathcal{S}_0 k.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, k\, x)\, e\rangle & = & e \quad k \notin \mathrm{V}(e) & (\eta_{\langle\cdot\rangle}) \\
\langle (\lambda x.\, \mathcal{S}_0 k.\, \langle e_1\rangle)\, e_2\rangle & = & \langle (\lambda x.\, e_1)\, e_2\rangle \quad k \notin \mathrm{V}(e_1) & (\langle\lambda\rangle)
\end{array}
$$

Figure 2 Axioms for $\lambda_{\mathcal{S}_0}$.

$$
\begin{array}{rcll}
(\lambda x.\, e)\, v & = & e[v/x] & (\beta_v) \\
\lambda x.\, v\, x & = & v \quad x \notin \mathrm{V}(v) & (\eta_v) \\
v \,\$\, \mathcal{S}_0 x.\, e & = & e[v/x] & (\beta_\$)
\end{array}
\qquad
\begin{array}{rcll}
\mathcal{S}_0 x.\, x \,\$\, e & = & e \quad x \notin \mathrm{V}(e) & (\eta_\$) \\
v_1 \,\$\, v_2 & = & v_1\, v_2 & (\$_v) \\
v \,\$\, E[e] & = & (\lambda x.\, v \,\$\, E[x]) \,\$\, e & (\$_E)
\end{array}
$$

Figure 3 Axioms for $\lambda_\$$.

3.2 The axioms for $\lambda_\$$

We present the axioms for $\lambda_\$$ in Figure 3. They are conceptually very different than the axioms for $\lambda_{\mathcal{S}_0}$, which may be surprising. But they are very regular and reveal the conceptual elegance of the $\lambda_\$$ language.

The first two axioms (β_v and η_v) are, as before, the standard beta-value conversions. The third one, $\beta_\$$, says that when we capture an empty context terminated with v, we get v back. The fourth, $\eta_\$$, means that we can always capture the top context, and then put it back as the terminating value on the delimiter. The two axioms can be thought of as beta and eta conversion axioms for *shift*$_0$ and $\$$ operators.

The fifth axiom, $\$_v$, says that if the inside of the context is a value, we can just apply it to the terminating value on the delimiter. This is an equational version of the reduction rule $v_1 \,\$\, v_2 \to v_1\, v_2$.

The last axiom, $\$_E$, says that we can move a suffix of the evaluation context delimited with $\$$ to the terminating value.

Please take notice that there is no axiom corresponding directly to the reduction rule $v \,\$\, E[\mathcal{S}_0 x.\, e] \to e[\lambda x.\, v \,\$\, E[x]/x]$. But the corresponding equation is still valid:

$$\lambda_\$ \vdash v \,\$\, E[\mathcal{S}_0 x.\, e] = (\lambda x.\, v \,\$\, E[x]) \,\$\, \mathcal{S}_0 x.\, e = e[\lambda x.\, v \,\$\, E[x]/x]$$

The β_Ω axiom also turned out to be redundant.

3.3 Reducing $\lambda_{\mathcal{S}_0}$ to $\lambda_\$$

In this subsection we show that the axioms for $\lambda_{\mathcal{S}_0}$ are sound and complete if and only if the axioms for $\lambda_\$$ are sound and complete. To achieve this, let us define a pair of translations – $\mathcal{D}[\![\cdot]\!]$ from $\lambda_{\mathcal{S}_0}$ to $\lambda_\$$, and $\mathcal{D}^{-1}[\![\cdot]\!]$ in the other direction:

$$
\begin{array}{rcl}
\mathcal{D}[\![\langle e\rangle]\!] & = & (\lambda x.\, x) \,\$\, \mathcal{D}[\![e]\!] \\
\mathcal{D}^{-1}[\![e_1 \,\$\, e_2]\!] & = & (\lambda f.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, f\, x)\, \mathcal{D}^{-1}[\![e_2]\!]\rangle)\, \mathcal{D}^{-1}[\![e_1]\!]
\end{array}
$$

The remainder of the translations is defined homomorphically.

The translations have the following properties (λ consists of full β and η axioms):

$$
\begin{array}{rcl@{\qquad}rcl}
\mathcal{G}[\![x]\!] & = & \mathcal{S}_0 k.\, k\, x & \mathcal{P}[\![x]\!] & = & x \\
\mathcal{G}[\![\lambda x.\, e]\!] & = & \mathcal{S}_0 k.\, k\, (\lambda x.\, \mathcal{G}[\![e]\!]) & \mathcal{P}[\![\lambda x.\, e]\!] & = & \lambda x.\, \mathcal{P}[\![e]\!] \\
\mathcal{G}[\![e_1\, e_2]\!] & = & \mathcal{S}_0 k.\, (\lambda f.\, (\lambda x.\, k \,\$\, f\, x) \,\$\, \mathcal{G}[\![e_2]\!]) \,\$\, \mathcal{G}[\![e_1]\!] & \mathcal{P}[\![v_1\, v_2]\!] & = & \mathcal{P}[\![v_1]\!]\, \mathcal{P}[\![v_2]\!] \\
\mathcal{G}[\![\mathcal{S}_0 k.\, e]\!] & = & \mathcal{S}_0 k.\, \mathcal{G}[\![e]\!] & \mathcal{P}[\![\mathcal{S}_0 x.\, e]\!] & = & \lambda x.\, \mathcal{P}[\![e]\!] \\
\mathcal{G}[\![e_1 \,\$\, e_2]\!] & = & \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, f \,\$\, \mathcal{G}[\![e_2]\!]) \,\$\, \mathcal{G}[\![e_1]\!] & \mathcal{P}[\![v \,\$\, e]\!] & = & \mathcal{P}[\![e]\!]\, \mathcal{P}[\![v]\!]
\end{array}
$$

Figure 4 CGS translation of $\lambda_\$$ to $\lambda_\$^{\mathrm{G}}$; translation of $\lambda_\$^{\mathrm{G}}$ to λ.

▶ Property 1. We have the following:

1. For every $\lambda_{\mathcal{S}_0}$ term e we have $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![\mathcal{D}[\![e]\!]]\!] = e$.
2. For every $\lambda_\$$ term e we have $\lambda_\$ \vdash \mathcal{D}[\![\mathcal{D}^{-1}[\![e]\!]]\!] = e$.
3. For every $\lambda_{\mathcal{S}_0}$ term e we have $\lambda \vdash \mathcal{C}[\![e]\!] = \mathcal{C}[\![\mathcal{D}[\![e]\!]]\!]$.
4. For every $\lambda_\$$ term e we have $\lambda \vdash \mathcal{C}[\![e]\!] = \mathcal{C}[\![\mathcal{D}^{-1}[\![e]\!]]\!]$.
5. $\lambda_{\mathcal{S}_0} \vdash e_1 = e_2$ implies $\lambda_\$ \vdash \mathcal{D}[\![e_1]\!] = \mathcal{D}[\![e_2]\!]$.
6. $\lambda_\$ \vdash e_1 = e_2$ implies $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![e_1]\!] = \mathcal{D}^{-1}[\![e_2]\!]$.

▶ **Theorem 2.** *The axioms for $\lambda_{\mathcal{S}_0}$ are sound iff the axioms for $\lambda_\$$ are sound.*

Proof. Suppose that the axioms for $\lambda_{\mathcal{S}_0}$ are sound. Assume $\lambda_\$ \vdash e_1 = e_2$. By Property 1.6 we have $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![e_1]\!] = \mathcal{D}^{-1}[\![e_2]\!]$. Using the assumed soundness of $\lambda_{\mathcal{S}_0}$ axioms gives $\lambda \vdash \mathcal{C}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!] = \mathcal{C}[\![\mathcal{D}^{-1}[\![e_2]\!]]\!]$. From Property 1.4 we get $\lambda \vdash \mathcal{C}[\![e_1]\!] = \mathcal{C}[\![e_2]\!]$. The other direction is analogous. ◀

▶ **Theorem 3.** *The axioms for $\lambda_{\mathcal{S}_0}$ are complete iff the axioms for $\lambda_{\mathcal{S}_0}$ are complete.*

Proof. Suppose that the axioms for $\lambda_{\mathcal{S}_0}$ are complete. Assume $\lambda \vdash \mathcal{C}[\![e_1]\!] = \mathcal{C}[\![e_2]\!]$. From Property 1.4 we get $\lambda \vdash \mathcal{C}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!] = \mathcal{C}[\![\mathcal{D}^{-1}[\![e_2]\!]]\!]$. Using the assumed completeness of $\lambda_{\mathcal{S}_0}$ axioms we get $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![e_1]\!] = \mathcal{D}^{-1}[\![e_2]\!]$. By Property 1.6 we have $\lambda_\$ \vdash \mathcal{D}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!] = \mathcal{D}[\![\mathcal{D}^{-1}[\![e_2]\!]]\!]$. From Property 1.2 follows the thesis. The other direction is analogous. ◀

3.4 CGS translation

Following the approach of Sabry [11], we show soundness and completeness of $\lambda_\$$ axioms in two steps. We introduce a translation from $\lambda_\$$ targeting a certain syntactical subset of $\lambda_\$$, which we call $\lambda_\$^{\mathrm{G}}$. We first prove soundness and completeness of $\lambda_\$$ with respect to $\lambda_\$^{\mathrm{G}}$, and then of $\lambda_\$^{\mathrm{G}}$ with respect to λ.

The language $\lambda_\$^{\mathrm{G}}$ is defined as follows:

$$e \quad ::= \quad \mathcal{S}_0 x.\, e \mid v\, v \mid v \,\$\, e \qquad v \quad ::= \quad x \mid \lambda x.\, e$$

In other words, we only allow applications with values on both sides and \$ with a value on the left side. Please also notice that the syntactic categories of expressions and values are separate in $\lambda_\$^{\mathrm{G}}$.

The translation is described in Figure 4. It is very similar to the CPS translation shown in Figure 1. The difference is that in the translation introduced in this section we use *shift*$_0$ and \$ instead of function abstraction and application for passing the continuation. We call this translation continuation-grabbing style (CGS) translation, because (as in the Sabry's translation) the terms actively „grab" their surrounding continuation, instead of passively waiting for it using a lambda abstraction, as is the case in the CPS translation.

The translation has an important property that by replacing shift_0 by λ and \$ by function application in target terms (Figure 4), we obtain the CPS translation from Figure 1:

▶ Property 4 (CPS-translation). $\mathcal{C}[\![e]\!] = \mathcal{P}[\![\mathcal{G}[\![e]\!]]\!]$

CGS terms are closed on β, η, $\beta_\$$ and $\eta_\$$ reductions. The equalities generated by these four reductions, restricted so that one cannot obtain non-CGS terms by expansion, form an axiomatization of $\lambda^{\mathrm{G}}_\$$.

We can easily prove soundness of $\lambda_\$$ axioms with respect to $\lambda^{\mathrm{G}}_\$$:

▶ **Lemma 5.** *If* $\lambda_\$ \vdash e_1 = e_2$*, then* $\lambda^G_\$ \vdash \mathcal{G}[\![e_1]\!] = \mathcal{G}[\![e_2]\!]$.

In order to prove completeness, we need another important property of the CGS translation – that the target terms are equal in $\lambda_\$$ to the source terms:

▶ **Lemma 6.** *For every* $\lambda_\$$ *term* e *we have* $\lambda_\$ \vdash e = \mathcal{G}[\![e]\!]$.

We also make the observation that every pair of $\lambda^{\mathrm{G}}_\$$ terms equal in $\lambda^{\mathrm{G}}_\$$ is also equal in $\lambda_\$$:

▶ **Lemma 7.** *If* $\lambda^G_\$ \vdash e_1 = e_2$*, then* $\lambda_\$ \vdash e_1 = e_2$.

We can now easily prove completeness of $\lambda_\$$ axioms with respect to $\lambda^{\mathrm{G}}_\$$:

▶ **Lemma 8.** *If* $\lambda^G_\$ \vdash \mathcal{G}[\![e_1]\!] = \mathcal{G}[\![e_2]\!]$*, then* $\lambda_\$ \vdash e_1 = e_2$.

Proof. Assume that $\lambda^{\mathrm{G}}_\$ \vdash \mathcal{G}[\![e_1]\!] = \mathcal{G}[\![e_2]\!]$. By Lemma 7 we have $\lambda_\$ \vdash \mathcal{G}[\![e_1]\!] = \mathcal{G}[\![e_2]\!]$. The thesis follows from Lemma 6. ◀

3.5 From $\lambda^{\mathrm{G}}_\$$ to λ

Soundness of $\lambda^{\mathrm{G}}_\$$ axioms with respect to λ is trivial:

▶ **Lemma 9.** *If* $\lambda^G_\$ \vdash e_1 = e_2$*, then* $\lambda \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$.

We still need to prove completeness of $\lambda^{\mathrm{G}}_\$$ axioms with respect to λ. This seems to be an easy task, but there is one important complication. Take a look at the translation in Figure 4. The translation replaces the \$ operator with function applications and the shift_0 operator with lambda abstractions. This causes new redexes to appear in the image of $\mathcal{P}[\![\cdot]\!]$; for example,

$$\lambda \vdash \mathcal{P}[\![\lambda x.\, x \,\$\, e]\!] = \lambda x.\, \mathcal{P}[\![e]\!]\, x = \mathcal{P}[\![e]\!]$$

But the $\lambda^{\mathrm{G}}_\$$ terms $\lambda x.\, x \,\$\, e$ and e are in separate syntactical categories – the one is a value, the other is an expression, and in $\lambda^{\mathrm{G}}_\$$ values are not expressions.

We introduce an intermediate language, $\lambda^{\mathrm{I}}_\$$, defined as follows:

$$e \quad ::= \quad \mathcal{S}_0 x.\, e \mid v\, v \mid v \,\$\, e \mid i_v[v] \qquad v \quad ::= \quad x \mid \lambda x.\, e \mid i_e[e]$$

The language is a syntactic extension of $\lambda^{\mathrm{G}}_\$$, which additionally allows using an expression as a value (and vice versa) with an explicit injection. Then we define two translations – $\mathcal{P}_I[\![\cdot]\!]$ from $\lambda^{\mathrm{I}}_\$$ to λ, and $\mathcal{I}[\![\cdot]\!]$ from $\lambda^{\mathrm{I}}_\$$ to $\lambda^{\mathrm{G}}_\$$:

$$\begin{array}{lcllcl} \mathcal{P}_I[\![i_v[v]]\!] & = & \mathcal{P}_I[\![v]\!] & \mathcal{I}[\![i_v[v]]\!] & = & \mathcal{S}_0 k.\, \mathcal{I}[\![v]\!]\, k \\ \mathcal{P}_I[\![i_e[e]]\!] & = & \mathcal{P}_I[\![e]\!] & \mathcal{I}[\![i_e[e]]\!] & = & \lambda x.\, x \,\$\, \mathcal{I}[\![e]\!] \end{array}$$

The translation $\mathcal{P}_I[\![\cdot]\!]$ is based on $\mathcal{P}[\![\cdot]\!]$, but ignores the explicit injections. The other translation, $\mathcal{I}[\![\cdot]\!]$, leaves most of the term unchanged (the cases not mentioned are defined homomorphically), but expands the injections so that the result is a valid $\lambda^{\mathrm{G}}_\$$ term. The expansions have the property that their translations to λ can be eta-reduced. Thus we have the following:

$$\frac{\tau \le \tau' \qquad \sigma \le \sigma'}{\tau\ \sigma \le \tau'\ \sigma'} \qquad \frac{}{\alpha \le \alpha} \qquad \frac{\tau_1' \le \tau_1 \qquad \tau_2\ \sigma \le \tau_2'\ \sigma'}{\tau_1 \xrightarrow{\sigma} \tau_2 \le \tau_1' \xrightarrow{\sigma'} \tau_2'}$$

$$\frac{}{\epsilon \le \epsilon} \qquad \frac{\tau_1\ \sigma_1 \le \tau_2\ \sigma_2}{\epsilon \le [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2} \qquad \frac{\tau_1'\ \sigma_1' \le \tau_1\ \sigma_1 \qquad \tau_2\ \sigma_2 \le \tau_2'\ \sigma_2'}{[\tau_1\ \sigma_1]\ \tau_2\ \sigma_2 \le [\tau_1'\ \sigma_1']\ \tau_2'\ \sigma_2'}$$

$$\frac{}{\Gamma, x : \tau_1 \vdash x : \tau_1}\ \text{VAR} \qquad \frac{\Gamma \vdash e : \tau\ \sigma \qquad \tau\ \sigma \le \tau'\ \sigma'}{\Gamma \vdash e : \tau'\ \sigma'}\ \text{SUB} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2\ \sigma}{\Gamma \vdash \lambda x : \tau_1.\, e : \tau_1 \xrightarrow{\sigma} \tau_2}\ \text{ABS}$$

$$\frac{\Gamma, x : \tau_1 \xrightarrow{\sigma} \tau_2 \vdash e : \tau_3\ \sigma'}{\Gamma \vdash \mathcal{S}_0 x : \tau_1 \xrightarrow{\sigma} \tau_2.\, e : \tau_1\ [\tau_2\ \sigma]\ \tau_3\ \sigma'}\ \text{SFT} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \xrightarrow{\sigma} \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1\ e_2 : \tau_2\ \sigma}\ \text{PAPP}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \xrightarrow{[\tau_4'\ \sigma_4']\ \tau_3'\ \sigma_3'} \tau_2\ [\tau_2'\ \sigma_2']\ \tau_1'\ \sigma_1' \qquad \Gamma \vdash e_2 : \tau_1\ [\tau_3'\ \sigma_3']\ \tau_2'\ \sigma_2'}{\Gamma \vdash e_1\ e_2 : \tau_2\ [\tau_4'\ \sigma_4']\ \tau_1'\ \sigma_1'}\ \text{APP}$$

Rule for $\lambda_{\mathcal{S}_0}$:

$$\frac{\Gamma \vdash e : \tau'\ [\tau']\ \tau\ \sigma}{\Gamma \vdash \langle e \rangle : \tau\ \sigma}\ \text{RST}$$

Rules for $\lambda_{\$}$:

$$\frac{\Gamma \vdash e_1 : \tau_1 \xrightarrow{\sigma} \tau_2 \qquad \Gamma \vdash e_2 : \tau_1\ [\tau_2\ \sigma]\ \tau_3\ \sigma'}{\Gamma \vdash e_1\ \$\ e_2 : \tau_3\ \sigma'}\ \text{PDOL}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \xrightarrow{\sigma} \tau_2\ [\tau_2'\ \sigma_2']\ \tau_1'\ \sigma_1' \qquad \Gamma \vdash e_2 : \tau_1\ [\tau_2\ \sigma]\ \tau_3\ [\tau_3'\ \sigma_3']\ \tau_2'\ \sigma_2'}{\Gamma \vdash e_1\ \$\ e_2 : \tau_3\ [\tau_3'\ \sigma_3']\ \tau_1'\ \sigma_1'}\ \text{DOL}$$

Figure 5 The type systems $\lambda^{\le}_{\mathcal{S}_0}$ and $\lambda^{\le}_{\$}$ (with subtyping).

▶ Property 10. For any $\lambda^{\mathrm{I}}_{\$}$ term e we have $\lambda \vdash \mathcal{P}_I[\![e]\!] = \mathcal{P}[\![\mathcal{I}[\![e]\!]]\!]$.

▶ Property 11. For any $\lambda^{\mathrm{G}}_{\$}$ term e we have $\mathcal{P}_I[\![e]\!] = \mathcal{P}[\![e]\!]$ and $\mathcal{I}[\![e]\!] = e$.

The equational theory for $\lambda^{\mathrm{I}}_{\$}$ consists of β_v, η_v, $\beta_{\$}$, $\eta_{\$}$ and the following two equalities:

$$i_v[v] \quad = \quad \mathcal{S}_0 x.\, v\, x \qquad (i_v) \qquad i_e[e] \quad = \quad \lambda x.\, x\, \$\, e \qquad (i_e)$$

The following lemma is trivial:

▶ **Lemma 12.** *For every two $\lambda^I_{\$}$ terms e_1 and e_2, if $\lambda^I_{\$} \vdash e_1 = e_2$, then $\lambda^G_{\$} \vdash \mathcal{I}[\![e_1]\!] = \mathcal{I}[\![e_2]\!]$.*

Thanks to the injections, we can prove the completeness lemma for $\lambda^{\mathrm{I}}_{\$}$:

▶ **Lemma 13.** *For every two $\lambda^I_{\$}$ expressions e_1 and e_2, if $\lambda \vdash \mathcal{P}_I[\![e_1]\!] = \mathcal{P}_I[\![e_2]\!]$, then $\lambda^I_{\$} \vdash e_1 = e_2$. The same holds for values.*

We can use it to prove completeness for $\lambda^{\mathrm{G}}_{\$}$:

▶ **Lemma 14.** *For every two $\lambda^G_{\$}$ terms e_1 and e_2, if $\lambda \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$, then $\lambda^G_{\$} \vdash e_1 = e_2$.*

Proof. Suppose that $\lambda \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$. By Property 11, we have $\lambda \vdash \mathcal{P}_I[\![e_1]\!] = \mathcal{P}_I[\![e_2]\!]$. Using Lemma 13 we get $\lambda^{\mathrm{I}}_{\$} \vdash e_1 = e_2$, Lemma 12 gives us $\lambda^{\mathrm{G}}_{\$} \vdash \mathcal{I}[\![e_1]\!] = \mathcal{I}[\![e_2]\!]$. By Property 11, we have $\lambda^{\mathrm{G}}_{\$} \vdash e_1 = e_2$. ◀

We can now finally prove soundness and completeness of the $\lambda_{\$}$ axioms:

▶ **Theorem 15** (Soundness). *If $\lambda_{\$} \vdash e_1 = e_2$, then $\lambda \vdash \mathcal{C}[\![e_1]\!] = \mathcal{C}[\![e_2]\!]$.*

▶ **Theorem 16** (Completeness). *If $\lambda \vdash \mathcal{C}[\![e_1]\!] = \mathcal{C}[\![e_2]\!]$, then $\lambda_{\$} \vdash e_1 = e_2$.*

$$
\begin{array}{rcll}
\mathcal{C}[\![e]\!]_{\mathrm{SUB}(\tau\ \sigma \le \tau'\ \sigma', D)} & = & \mathcal{C}[\![\tau\ \sigma \le \tau'\ \sigma']\!][\mathcal{C}[\![e]\!]_D] & \\
\mathcal{C}[\![x]\!]_{\mathrm{VAR}} & = & x & \\
\mathcal{C}[\![\lambda x.\, e]\!]_{\mathrm{ABS}(D)} & = & \lambda x.\, \mathcal{C}[\![e]\!]_D & \\
\mathcal{C}[\![e_1\ e_2]\!]_{\mathrm{PAPP}(D_1, D_2)} & = & \mathcal{C}[\![e_1]\!]_{D_1}\ \mathcal{C}[\![e_2]\!]_{D_2} & \\
\mathcal{C}[\![e_1\ e_2]\!]_{\mathrm{APP}(D_1, D_2)} & = & \lambda k.\, \mathcal{C}[\![e_1]\!]_{D_1}\ (\lambda f.\, \mathcal{C}[\![e_2]\!]_{D_2}\ (\lambda x.\, f\ x\ k)) & \\
\mathcal{C}[\![\mathcal{S}_0 x.\, e]\!]_{\mathrm{SFT}(D)} & = & \lambda x.\, \mathcal{C}[\![e]\!]_D & \\
\mathcal{C}[\![e_1\ \$\ e_2]\!]_{\mathrm{PDOL}(D_1, D_2)} & = & \mathcal{C}[\![e_2]\!]_{D_2}\ \mathcal{C}[\![e_1]\!]_{D_1} & \text{for } \lambda^{\le}_{\$} \vee \lambda^{\le}_{\mathcal{S}_0} \\
\mathcal{C}[\![e_1\ \$\ e_2]\!]_{\mathrm{DOL}(D_1, D_2)} & = & \lambda k.\, \mathcal{C}[\![e_1]\!]_{D_1}\ (\lambda f.\, \mathcal{C}[\![e_2]\!]_{D_2}\ f\ k) & \text{for } \lambda^{\le}_{\$} \vee \lambda^{\le}_{\mathcal{S}_0} \\
\mathcal{C}[\![\langle e \rangle]\!]_{\mathrm{RST}(D)} & = & \mathcal{C}[\![e]\!]_D\ (\lambda x.\, x) & \text{for } \lambda^{\le}_{\mathcal{S}_0}
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{C}[\![\alpha \le \alpha]\!][e] & = & e \\
\mathcal{C}[\![\tau_1' \xrightarrow{\sigma_1} \tau_1 \le \tau_2' \xrightarrow{\sigma_2} \tau_2]\!][e] & = & \lambda x.\, \mathcal{C}[\![\tau_1\ \sigma_1 \le \tau_2\ \sigma_2]\!][e\, \mathcal{C}[\![\tau_2' \le \tau_1']\!][x]] \\
\mathcal{C}[\![\tau_1\ \epsilon \le \tau_2\ \epsilon]\!][e] & = & \mathcal{C}[\![\tau_1 \le \tau_2]\!][e] \\
\mathcal{C}[\![\tau\ \epsilon \le \tau'\ [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2]\!][e] & = & \lambda k.\, \mathcal{C}[\![\tau_1\ \sigma_1 \le \tau_2\ \sigma_2]\!][k\, \mathcal{C}[\![\tau \le \tau']\!][e]] \\
\mathcal{C}[\![\tau\ [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2 \le \tau'\ [\tau_1'\ \sigma_1']\ \tau_2'\ \sigma_2']\!][e] & = & \lambda k.\, \mathcal{C}[\![\tau_1'\ \sigma_1' \le \tau_2'\ \sigma_2']\!][e \\
 & & \quad (\lambda x.\, \mathcal{C}[\![\tau_1'\ \sigma_1' \le \tau_1\ \sigma_1]\!][k\, \mathcal{C}[\![\tau \le \tau']\!][x]])]
\end{array}
$$

Figure 6 Type-directed selective CPS translations for $\lambda^{\le}_{\mathcal{S}_0}$ to $\lambda^{\le}_{\$}$.

4 Typed languages $\lambda^{\le}_{\mathcal{S}_0}$ and $\lambda^{\le}_{\$}$

We now take a break from equational axiomatizations and describe typed versions of $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$, called $\lambda^{\le}_{\mathcal{S}_0}$ and $\lambda^{\le}_{\$}$. We present sound and complete axiomatizations for these languages in the next section.

In this work we use explicit type annotations on bound variables. This style of presentation of typed languages is called „de Bruijn style" by Barendregt [1]. Thus the syntax of the $\lambda^{\le}_{\mathcal{S}_0}$ and $\lambda^{\le}_{\$}$ languages is the same as for $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$, with the following changes (τ is the syntactic category of types):

$$e \quad ::= \quad v \mid \mathcal{S}_0 x : \tau.\, e \mid \ldots \qquad v \quad ::= \quad x \mid \lambda x : \tau.\, e$$

We often omit the type annotations for clarity, but they are still implicitly present.

4.1 Type systems

The description is shortened because of space limitations; for more details, see [8] and [9]. First let us define syntactic categories of types and effect annotations:

$$\tau \quad ::= \quad \alpha \mid \tau \xrightarrow{\sigma} \tau \qquad \sigma \quad ::= \quad \epsilon \mid [\tau\ \sigma]\ \tau\ \sigma$$

An effect annotation can only be given meaning together with the type it annotates. The typing judgment $\Gamma \vdash e : \tau_1'\ [\tau_1\ \sigma_1]\ \ldots \tau_n'\ [\tau_n\ \sigma_n]\ \tau\ \epsilon$ means "the expression e, when evaluated inside contexts of types $\tau_1' \xrightarrow{\sigma_1} \tau_1, \ldots, \tau_n' \xrightarrow{\sigma_n} \tau_n$, gives an answer of type τ. In particular, the judgment $\Gamma \vdash e : \tau\ \epsilon$ means "the expression e has no control effects and, when evaluated, yields a value of type τ". We will often omit ϵ where it leads to no confusion.

The type systems are shown in Figure 5. The type system for $\lambda^{\le}_{\$}$ is the one presented in [9]. The type system for $\lambda^{\le}_{\mathcal{S}_0}$ differs slightly only in the rule PAPP from the one from [8]. The modification does not change the expressiveness of the type system, but helps with the proofs.

$$
\begin{array}{rcll}
(\lambda x.\, e)\, p & = & e[p/x] & (\beta_p) \\
\lambda x.\, p\, x & = & p & x \notin \mathrm{V}(p) \quad (\eta_p) \\
(\lambda x.\, E[x])\, e & = & E[e] & x \notin \mathrm{V}(E) \quad (\beta_\Omega) \\
\langle E[\mathcal{S}_0 x.\, e] \rangle & = & e[\lambda x.\, \langle E[x] \rangle / x] & x \notin \mathrm{V}(E) \quad (\langle \mathcal{S}_0 \rangle) \\
\langle p \rangle & = & p & (\langle p \rangle) \\
\mathcal{S}_0 k.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, k\, x)\, e \rangle & = & e & k \notin \mathrm{V}(e) \quad (\eta_{\langle\cdot\rangle}) \\
\langle (\lambda x.\, \mathcal{S}_0 k.\, \langle e_1 \rangle)\, e_2 \rangle & = & \langle (\lambda x.\, e_1)\, e_2 \rangle & k \notin \mathrm{V}(e_1) \quad (\langle\lambda\rangle)
\end{array}
$$

Figure 7 Axioms for $\lambda^{\leq}_{\mathcal{S}_0}$.

$$
\begin{array}{rcll}
(\lambda x.\, e)\, p & = & e[p/x] & (\beta_p) \\
\lambda x.\, p\, x & = & p \quad x \notin \mathrm{V}(p) & (\eta_p) \\
p \,\$\, \mathcal{S}_0 x.\, e & = & e[p/x] & (\beta^p_\$)
\end{array}
\qquad
\begin{array}{rcll}
\mathcal{S}_0 x.\, x \,\$\, e & = & e \quad x \notin \mathrm{V}(e) & (\eta_\$) \\
p_1 \,\$\, p_2 & = & p_1\, p_2 & (\$_p) \\
p \,\$\, E[e] & = & (\lambda x.\, p \,\$\, E[x]) \,\$\, e & (\$_E)
\end{array}
$$

Figure 8 Axioms for $\lambda^{\leq}_{\$}$.

The type systems include three subtyping relations, defined on types, effect annotations and annotated types, which are also defined in Figure 5. The subtyping relations are partial orders: they are reflexive, weakly antisymmetric and transitive. We also have the following:

▶ Property 17. Every derivable subtyping judgment has only one derivation.

The property allows us to identify a subtyping judgment with its only derivation, which is important for our proofs.

4.2 Selective CPS translations

For the typed languages $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$ we can define different translations than these defined in Figure 1. The type information can be used to preserve pure (or control effect free) code without changes and CPS-translate only the impure parts. The translations are shown in Figure 6. The translation for $\lambda^{\leq}_{\mathcal{S}_0}$ is the one from [8]; the one for $\lambda^{\leq}_{\$}$ is derived from it.

These translations preserve types in the following sense. Let us define translations from $\lambda^{\leq}_{\mathcal{S}_0}$ types and typed annotations to simple types of $\lambda^{\to}$:

$$
\begin{array}{rcl}
\mathcal{C}[\![\alpha]\!] & = & \alpha \\
\mathcal{C}[\![\tau' \xrightarrow{\sigma} \tau]\!] & = & \mathcal{C}[\![\tau']\!] \to \mathcal{C}[\![\tau\ \sigma]\!]
\end{array}
\qquad
\begin{array}{rcl}
\mathcal{C}[\![\tau\ \epsilon]\!] & = & \mathcal{C}[\![\tau]\!] \\
\mathcal{C}[\![\tau\ [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2]\!] & = & (\mathcal{C}[\![\tau]\!] \to \mathcal{C}[\![\tau_1\ \sigma_1]\!]) \to \mathcal{C}[\![\tau_2\ \sigma_2]\!]
\end{array}
$$

We have the following:

▶ Property 18 (Type preservation). If D is a derivation of $\Gamma \vdash e : \tau\ \sigma$ in $\lambda^{\leq}_{\mathcal{S}_0}$, then $\mathcal{C}[\![\Gamma]\!] \vdash \mathcal{C}[\![e]\!]_D : \mathcal{C}[\![\tau\ \sigma]\!]$ in $\lambda^{\to}$. This also holds for $\lambda^{\leq}_{\$}$.

5 Typed axiomatization

In this section we present sound and complete (with respect to the type-directed selective translation of Figure 6) equational axiomatizations of $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$. The development mostly follows the untyped one, but there are a few surprises, starting with the axioms themselves.

5.1 The typed axioms for $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$

We present the axioms in Figure 7 and Figure 8. We use the letter p to denote pure expressions (the ones which can be typed with the empty effect annotation ϵ). The axioms seem similar to the untyped ones, but several things need to be noted.

First, because the axioms themselves are typed, they can only be used when the types match. For example, the typed $\eta_\$$ axiom cannot be typed pure, so it is only applicable on terms with an impure type. (Therefore the implicit infinite tower of resets, which is present in the untyped languages, disappears in typed ones.)

Second, the dependence on types makes it important to mention the typing context, which gives types to variables. Because the subtyping rule allows the same term to have different types, the concrete type which we consider needs also to be mentioned. Thus we use the notation $\lambda^{\leq}_{\mathcal{S}_0}; \Gamma \vdash e_1 = e_2 : \tau\ \sigma$ when talking about equality modulo the axioms.

Third, in the typed axioms the syntactical value restriction present in the untyped axioms β_v, η_v, $\beta_\$$, $\langle v \rangle$ and $\$_v$ is replaced by type-dependent purity restriction. This change is caused by the fact that the CPS translations considered are selective – they leave the pure terms unchanged. The typed axioms are more general, because every value has a pure typing.

Finally, we point out that the axioms give a call-by-name interpretation to pure subprograms. This is not problematic because the language considered is terminating and has no side effects other than capturing of delimited contexts by shift_0.

5.2 Reducing $\lambda^{\leq}_{\mathcal{S}_0}$ to $\lambda^{\leq}_{\$}$

Similar to the untyped case, the axioms for $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$ are related by the following theorem. We omit the proof because of space limitations.

▶ **Theorem 19.** *The axioms for $\lambda^{\leq}_{\mathcal{S}_0}$ are sound (complete) iff the axioms for $\lambda^{\leq}_{\$}$ are sound (complete).*

5.3 Typed CGS translation

Analogously to the untyped case, we present a translation from $\lambda^{\leq}_{\$}$ which targets a certain subset of it, called $\lambda^{\rightarrow}_{\$}$. Differently to the untyped case, we will define this subset not by restricting syntax, but by using a simpler type system, which consists of rules VAR, ABS, SFT, PAPP and PDOL (Figure 5).

It is worth notice that the restrictions imposed by the restricted type system for $\lambda^{\rightarrow}_{\$}$ are analogous to the syntactic restrictions on $\lambda^{\mathrm{G}}_{\$}$: the restriction of being a value in the untyped case corresponds to the restriction of having a pure typing in the typed case. Another interesting point is that there is no subtyping in the type system. The rules for impure application and impure \$ are also gone, and with good reason: without subtyping, these rules fail subject reduction.

The axioms β_p, η_p, $\beta^p_\$$ and $\eta_\$$ form an axiomatization of $\lambda^{\rightarrow}_{\$}$. Take notice that the type system of $\lambda^{\rightarrow}_{\$}$ makes the full beta reduction valid, because it forces the type of the function argument to be pure. Therefore, as in the untyped case, the typed CGS language is evaluation order independent.

We present the typed CGS translation in Figure 9. The translation is derived from the typed CPS translation in Figure 6 using the same principles as with the untyped one. As before, replacing the occurrences of shift_0 with lambda abstractions and occurrences of \$ with function applications (as in Figure 4, but extended to work on terms with type annotations). in the result terms of $\mathcal{G}[\![\cdot]\!]$. gives us the CPS translation:

$$
\begin{aligned}
\mathcal{G}[\![e]\!]_{\mathrm{SUB}(\tau\ \sigma\leq\tau'\ \sigma',D)} &= \mathcal{G}[\![\tau\ \sigma\leq\tau'\ \sigma']\!][\mathcal{G}[\![e]\!]_D]\\
\mathcal{G}[\![x]\!]_{\mathrm{VAR}} &= x\\
\mathcal{G}[\![\lambda x.\,e]\!]_{\mathrm{ABS}(D)} &= \lambda x.\,\mathcal{G}[\![e]\!]_D\\
\mathcal{G}[\![e_1\ e_2]\!]_{\mathrm{PAPP}(D_1,D_2)} &= \mathcal{G}[\![e_1]\!]_{D_1}\ \mathcal{G}[\![e_2]\!]_{D_2}\\
\mathcal{G}[\![e_1\ e_2]\!]_{\mathrm{APP}(D_1,D_2)} &= \mathcal{S}_0k.\,(\lambda f.\,(\lambda x.\,k\,\$\,f\,x)\,\$\,\mathcal{G}[\![e_2]\!]_{D_2})\,\$\,\mathcal{G}[\![e_1]\!]_{D_1}\\
\mathcal{G}[\![\mathcal{S}_0x.\,e]\!]_{\mathrm{SFT}(D)} &= \mathcal{S}_0x.\,\mathcal{G}[\![e]\!]_D\\
\mathcal{G}[\![e_1\,\$\,e_2]\!]_{\mathrm{PDOL}(D_1,D_2)} &= \mathcal{G}[\![e_1]\!]_{D_1}\,\$\,\mathcal{G}[\![e_2]\!]_{D_2}\\
\mathcal{G}[\![e_1\,\$\,e_2]\!]_{\mathrm{DOL}(D_1,D_2)} &= \mathcal{S}_0k.\,(\lambda f.\,k\,\$\,f\,\$\,\mathcal{G}[\![e_2]\!]_{D_2})\,\$\,\mathcal{G}[\![e_1]\!]_{D_1}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{G}[\![\alpha\leq\alpha]\!][e] &= e\\
\mathcal{G}[\![\tau_1'\xrightarrow{\sigma_1}\tau_1\leq\tau_2'\xrightarrow{\sigma_2}\tau_2]\!][e] &= \lambda x.\,\mathcal{G}[\![\tau_1\ \sigma_1\leq\tau_2\ \sigma_2]\!][e\,\mathcal{G}[\![\tau_2'\leq\tau_1']\!][x]]\\
\mathcal{G}[\![\tau_1\ \epsilon\leq\tau_2\ \epsilon]\!][e] &= \mathcal{G}[\![\tau_1\leq\tau_2]\!][e]\\
\mathcal{G}[\![\tau\ \epsilon\leq\tau'\ [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2]\!][e] &= \mathcal{S}_0k.\,\mathcal{G}[\![\tau_1\ \sigma_1\leq\tau_2\ \sigma_2]\!][k\,\mathcal{G}[\![\tau\leq\tau']\!][e]]\\
\mathcal{G}[\![\tau\ [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2\leq\tau'\ [\tau_1'\ \sigma_1']\ \tau_2'\ \sigma_2']\!][e] &= \mathcal{S}_0k.\,\mathcal{G}[\![\tau_1'\ \sigma_1'\leq\tau_2'\ \sigma_2']\!][\\
&\qquad(\lambda x.\,\mathcal{G}[\![\tau_1'\ \sigma_1'\leq\tau_1\ \sigma_1]\!][k\,\mathcal{G}[\![\tau\leq\tau']\!][x]])\,\$\,e]
\end{aligned}
$$

Figure 9 Type-directed selective CGS translation of $\lambda^{\leq}_{\$}$ to $\lambda^{\rightarrow}_{\$}$.

▶ Property 20 (CPS translation). $\mathcal{C}[\![e]\!]_D = \mathcal{P}[\![\mathcal{G}[\![e]\!]_D]\!]$

In contrast to the untyped case, the soundness of $\lambda^{\leq}_{\$}$ axioms is not trivial. The reason is that the typed CGS (and CPS) translation depends on the typing derivation. It is easy to show that every axiom is sound in some particular derivation, but we need to have them sound in any derivation to have soundness. Therefore, we need coherence – the property that, no matter the derivation, the terms resulting from the translation are equal.

▶ **Theorem 21** (Coherence). *For every two derivations D_1, D_2 of the same typing judgment $\Gamma\vdash e:\tau\ \sigma$ $\lambda^{\leq}_{\$}$ we have $\lambda^{\rightarrow}_{\$};\Gamma\vdash\mathcal{G}[\![e]\!]_{D_1}=\mathcal{G}[\![e]\!]_{D_2}$.*

Proof. In the appendix. ◀

We can now prove soundness for $\lambda^{\leq}_{\$}$ with respect to the typed CGS translation:

▶ **Lemma 22.** *Suppose that for some two $\lambda^{\leq}_{\$}$ terms e_1 and e_2 we have $\lambda^{\leq}_{\$};\Gamma\vdash e_1=e_2:\tau\ \sigma$. Then for every two derivations D_1 and D_2 for $\Gamma\vdash e_1:\tau\ \sigma$ and $\Gamma\vdash e_2:\tau\ \sigma$ we have $\lambda^{\rightarrow}_{\$};\Gamma\vdash\mathcal{G}[\![e_1]\!]_{D_1}=\mathcal{G}[\![e_2]\!]_{D_2}$.*

As in the untyped case, we can prove that the target terms of the typed CGS translation are equal in $\lambda^{\leq}_{\$}$ to the source terms:

▶ **Lemma 23.** *For every derivation D of $\Gamma\vdash e:\tau\ \sigma$ we have $\lambda^{\rightarrow}_{\$};\Gamma\vdash e=\mathcal{G}[\![e]\!]_D$.*

Every equality in $\lambda^{\rightarrow}_{\$}$ is also valid in $\lambda^{\leq}_{\$}$:

▶ **Lemma 24.** *If $\lambda^{\rightarrow}_{\$};\Gamma\vdash e_1=e_2$ and $\Gamma\vdash e_1:\tau\ \sigma$, then $\lambda^{\leq}_{\$};\Gamma\vdash e_1=e_2:\tau\ \sigma$.*

We can now prove completeness of $\lambda^{\leq}_{\$}$ axioms with respect to $\lambda^{\rightarrow}_{\$}$:

▶ **Lemma 25.** *If D_1 and D_2 are derivations of $\Gamma\vdash e_1:\tau\ \sigma$ and $\Gamma\vdash e_2:\tau\ \sigma$, and $\lambda^{\rightarrow}_{\$};\Gamma\vdash\mathcal{G}[\![e_1]\!]_{D_1}=\mathcal{G}[\![e_2]\!]_{D_2}$, then $\lambda^{\leq}_{\$};\Gamma\vdash e_1=e_2:\tau\ \sigma$.*

$$\frac{\mathrm{lnf}\$(\Gamma, x : \tau_1)\ e : \tau_2\ \sigma}{\mathrm{lnf}\$(\Gamma)\ \lambda x : \tau_1.\, e : \tau_1 \xrightarrow{\sigma} \tau_2} \qquad \frac{\mathrm{lnf}\$(\Gamma, x : \tau_1 \xrightarrow{\sigma} \tau_2)\ e : \tau\ \sigma'}{\mathrm{lnf}\$(\Gamma)\ \mathcal{S}_0 x : \tau_1 \xrightarrow{\sigma} \tau_2.\, e : \tau_1\ [\tau_2\ \sigma]\ \tau\ \sigma'} \qquad \frac{\mathrm{lnf}\$^{\downarrow}(\Gamma)\ e : \alpha}{\mathrm{lnf}\$(\Gamma)\ e : \alpha}$$

$$\frac{}{\mathrm{lnf}\$^{\downarrow}(\Gamma, x : \tau)\ x : \tau} \qquad \frac{\mathrm{lnf}\$^{\downarrow}(\Gamma)\ e_1 : \tau_1 \xrightarrow{\sigma} \tau_2 \qquad \mathrm{lnf}\$(\Gamma)\ e_2 : \tau_1}{\mathrm{lnf}\$^{\downarrow}(\Gamma)\ e_1\ e_2 : \tau_2\ \sigma}$$

$$\frac{\mathrm{lnf}\$(\Gamma)\ e_1 : \tau_1 \xrightarrow{\sigma} \tau_2 \qquad \mathrm{lnf}\$^{\downarrow}(\Gamma)\ e_2 : \tau_1\ [\tau_2\ \sigma]\ \tau\ \sigma'}{\mathrm{lnf}\$^{\downarrow}(\Gamma)\ e_1\ \$\ e_2 : \tau\ \sigma'}$$

Figure 10 $-beta eta long form.

5.4 From $\lambda^{\to}_{\$}$ to $\lambda^{\to}$

We can easily prove soundness of the typed CGS axioms with respect to $\lambda^{\to}$:

▶ **Lemma 26.** *For any two $\lambda^{\to}_{\$}$ terms e_1, e_2 and any typing environment Γ such that $\lambda^{\to}_{\$}; \Gamma \vdash e_1 = e_2$ we have $\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$.*

Proving completeness of $\lambda^{\to}_{\$}$ is done differently than in the untyped case. We still get unwanted redexes when translating with $\mathcal{P}[\![\cdot]\!]$. For example, if $\Gamma \vdash e : \tau_1\ [\tau_2\ \sigma]\ \tau'\ \sigma'$, then $\Gamma \vdash \lambda x.\, x\ \$\ e : (\tau_1 \xrightarrow{\sigma} \tau_2) \xrightarrow{\sigma'} \tau'$ – but their translations are equal in $\lambda^{\to}$:

$$\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{P}[\![\lambda x.\, x\ \$\ e]\!] = \lambda x.\, \mathcal{P}[\![e]\!]\ x = \mathcal{P}[\![e]\!]$$

The problem exists only for eta reductions and beta expansions: every beta reduction in the translated term corresponds to a β_p or $\beta^p_{\$}$ reduction in the source term, and similarly, every eta expansion in the translated term corresponds to a η_p or $\eta_{\$}$ expansion in the source term.[1] We can use beta eta long forms [1] to solve this issue. Let us define $\mathrm{lnf}(\Gamma)\ e : \tau$ to mean "in the type environment Γ the expression e has type τ and is in beta eta long form". We have the following:

▶ Property 27. If $\lambda^{\to}; \Gamma \vdash e_1 = e_2$, then there exists a $\lambda^{\to}$ term e, in beta eta long form, such that e_1 and e_2 both reduce to e using only beta reductions and eta expansions.

We can easily prove the following:

▶ **Lemma 28.** *If $\lambda^{\to}_{\$}$ term e_1 is typable in Γ and $\mathcal{P}[\![e_1]\!]$ reduces to e in $\mathcal{C}[\![\Gamma]\!]$ using only beta reductions and eta expansions, then there exists a $\lambda^{\to}_{\$}$ term e_2 such that $e = \mathcal{P}[\![e_2]\!]$ and $\lambda^{\to}_{\$}; \Gamma \vdash e_1 = e_2$.*

But this lemma alone cannot be used to prove completeness. Applying the lemma to the two reduction sequences from Property 27 give us two $\lambda^{\to}_{\$}$ terms which translate to the same $\lambda^{\to}$ term, but we do not know yet if they are equal. Fortunately, we can use the fact that their translation is in beta eta long form to give a positive answer to this question.

Let us begin with presenting the syntax of the $\lambda^{\to}_{\$}$ analogue of the beta eta long forms, we call them $-beta eta long forms (Figure 10). We prove that if the translated $\lambda^{\to}_{\$}$ term is in the beta eta long form, then the original term is in the $-beta eta long form:

▶ **Lemma 29.** *If $\mathit{lnf}(\mathcal{C}[\![\Gamma]\!])\ \mathcal{P}[\![e]\!] : \mathcal{C}[\![\tau\ \sigma]\!]$, then $\mathit{lnf}\$(\Gamma)\ e : \tau\ \sigma$.*

[1] Conventionally, we define β_p, $\beta^p_{\$}$, η_p and $\eta_{\$}$ reductions as left-to-right directed versions of the corresponding equations.

Then we prove that if we have two $\lambda^{\to}_{\$}$ terms in \$-beta eta long forms which translate to the same $\lambda^{\to}$ term, then they are (syntactically) equal:

▶ **Lemma 30.** *If* $\mathit{lnf}\$(\Gamma)\ e_1 : \tau\ \sigma$, $\mathit{lnf}\$(\Gamma)\ e_2 : \tau\ \sigma$ *and* $\mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$, *then* $e_1 = e_2$.

Now we can prove completeness of $\lambda^{\to}_{\$}$:

▶ **Lemma 31.** *If we have* $\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$, *then* $\lambda^{\to}_{\$}; \Gamma \vdash e_1 = e_2$.

Proof. Suppose that $\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!] : \tau\ \sigma$. Using Property 27 we get a $\lambda^{\to}$ term e in eta long form such that both $\mathcal{P}[\![e_1]\!]$ and $\mathcal{P}[\![e_2]\!]$ reduce to e using only beta reductions and eta expansions. Applying Lemma 28 we get $\lambda^{\to}_{\$}; \Gamma \vdash e_1 = e'_1$, $\lambda^{\to}_{\$}; \Gamma \vdash e_2 = e'_2$ and $\mathcal{P}[\![e'_1]\!] = \mathcal{P}[\![e'_2]\!] = e$. By Lemma 29 we get that e'_1 and e'_2 are in \$-beta eta long form. Lemma 30 gives us $e'_1 = e'_2$, which finishes the proof. ◀

We can now prove soundness and completeness for $\lambda^{\leq}_{\$}$:

▶ **Theorem 32** (Soundness). *Suppose that for some two* $\lambda^{\leq}_{\$}$ *terms* e_1 *and* e_2 *we have* $\lambda^{\leq}_{\$}; \Gamma \vdash e_1 = e_2 : \tau\ \sigma$. *Then for every two derivations* D_1 *and* D_2 *for* $\Gamma \vdash e_1 : \tau\ \sigma$ *and* $\Gamma \vdash e_2 : \tau\ \sigma$ *we have* $\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{C}[\![e_1]\!]_{D_1} = \mathcal{C}[\![e_2]\!]_{D_2}$.

▶ **Theorem 33** (Completeness). *For every two derivations* D_1, D_2 *of* $\Gamma \vdash e_1 \tau\ \sigma$ *and* $\Gamma \vdash e_2 \tau\ \sigma$, *if* $\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{C}[\![e_1]\!]_{D_1} = \mathcal{C}[\![e_2]\!]_{D_2}$, *then* $\lambda^{\leq}_{\$}; \Gamma \vdash e_1 = e_2 : \tau\ \sigma$.

6 Related work

6.1 Kameyama and Hasegawa's axioms for *shift*/*reset*

We can express the *shift*/*reset* control operators in $\lambda_{\mathcal{S}_0}$ by leaving the occurrences of *reset* without changes and replacing the occurrences of the *shift* operator $\mathcal{S}k.\, e$ with $\mathcal{S}_0 k.\, \langle e \rangle$. It is an interesting question if the axioms of Kameyama and Hasegawa [6] can be validated in this embedding using the axioms for $\lambda_{\mathcal{S}_0}$.

The answer is negative. The axioms *reset-lift*, $\mathcal{S}$*-elim* and $\mathcal{S}$*-reset* cannot be validated. The reason is that the *shift*$_0$ operator, in contrast to *shift*, can reach beyond the nearest delimiter; the three equations above significantly change the structure of delimiters, which can be distinguished by repeated uses of *shift*$_0$.

Our axioms β_v, η_v, β_Ω and $\langle v \rangle$ are identical to corresponding Kameyama and Hasegawa's axioms. The axiom $\langle \mathcal{S}_0 \rangle$ is taken from the reduction semantics for *shift*$_0$. The remaining two axioms $\eta_{\langle\cdot\rangle}$ and $\langle \lambda \rangle$ are different, but are related to $\mathcal{S}$*-elim* and *reset-lift*.

The Kameyama and Hasegawa's $\mathcal{S}$*-elim* axiom $\mathcal{S}k.\, k\, e = e$ is unsound in $\lambda_{\mathcal{S}_0}$. To see why, take a look at the $\lambda_{\mathcal{S}_0}$ term $\langle f\, \langle g\, e \rangle \rangle$. If we apply the $\mathcal{S}$-elim-derived equality $\mathcal{S}_0 k.\, \langle k\, e \rangle = e$ right-to-left on g, we get:

$$\langle f\, \langle (\mathcal{S}_0 k.\, \langle k\, g \rangle)\, e \rangle \rangle \to \langle f\, \langle (\lambda x.\, \langle x\, e \rangle)\, g \rangle \rangle \to \langle f\, \langle \langle g\, e \rangle \rangle \rangle$$

We see that one of the *reset*$_0$'s got duplicated. The *reset*$_0$ operator is not idempotent, so the equation must be unsound. To fix the equation, we need to ensure the superfluous *reset*$_0$ gets removed in the course of evaluation. This way we obtain the axiom $\eta_{\langle\cdot\rangle}$. Let us check this using our previous example:

$$\begin{aligned}&\langle f\, \langle (\mathcal{S}_0 k.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, k\, x)\, g \rangle)\, e \rangle \rangle \to \langle f\, \langle (\lambda x.\, \mathcal{S}_0 z.\, (\lambda y.\, \langle y\, e \rangle)\, x)\, g \rangle \rangle \\ &\to \langle f\, \langle \mathcal{S}_0 z.\, (\lambda y.\, \langle y\, e \rangle)\, g \rangle \rangle \to \langle f\, ((\lambda y.\, \langle y\, e \rangle)\, g) \rangle \to \langle f\, \langle g\, e \rangle \rangle\end{aligned}$$

The Kameyama and Hasegawa's *reset-lift* axiom $\langle(\lambda x.\, e_1)\,\langle e_2\rangle\rangle = (\lambda x.\,\langle e_1\rangle)\,\langle e_2\rangle$ is invalid in $\lambda_{\mathcal{S}_0}$. Notice that the main fact stated by the axiom is that the subexpression e_1 is always evaluated in an empty context. The same fact is the basis for the $\langle\lambda\rangle$ axiom.

6.2 Kameyama and Hasegawa's axioms in the typed setting

It is shown in [8] that the typed *shift*/*reset* [2] can be embedded in $\lambda^{\leq}_{\mathcal{S}_0}$ so that the type-directed CPS translation for this embedding gives terms which are beta eta equal to the standard CPS translation for *shift*/*reset*. This means that the axioms of Kameyama and Hasegawa are validated for this embedding by the axioms for $\lambda^{\leq}_{\mathcal{S}_0}$.

How is this possible, even though the embedding is the same on the term level as the untyped embedding? The answer, of course, lies in the types. Consider, for example, the Kameyama and Hasegawa's *reset-lift* axiom $(\langle(\lambda x.\, e_1)\,\langle e_2\rangle\rangle = (\lambda x.\,\langle e_1\rangle)\,\langle e_2\rangle)$. In the untyped setting, this is obviously invalid – the left hand side and the right hand side have obviously different structure of delimiters, which can be distinguished by two shift_0's. But in the typed setting, we know that the types in the derivations generated by the embedding are shallow: the typing annotations are only of the form ϵ or $[\tau_1]\,\tau_2$. This means that any term in the target of the embedding which has the form $\langle e\rangle$ has a pure typing. So the following is derivable (we use the β_p axiom):

$$\lambda^{\leq}_{\mathcal{S}_0};\Gamma \vdash \langle(\lambda x.\, e_1)\,\langle e_2\rangle\rangle = \langle e_1[\langle e_2\rangle/x]\rangle = (\lambda x.\,\langle e_1\rangle)\,\langle e_2\rangle$$

Let us see another example – the $\mathcal{S}$-elim axiom $(\mathcal{S}k.\, k\, e = e)$. It is embedded into $\lambda^{\leq}_{\mathcal{S}_0}$ in the form $\mathcal{S}_0 k.\,\langle k\, e\rangle = e$. We have the following:

$$\begin{aligned}\lambda^{\leq}_{\mathcal{S}_0};\Gamma \vdash \mathcal{S}_0 k.\,\langle k\, e\rangle &= \mathcal{S}_0 k.\,\langle(\lambda x.\, k\, x)\, e\rangle = \mathcal{S}_0 k.\,\langle(\lambda x.\, \mathcal{S}_0 z.\,\langle k\, x\rangle)\, e\rangle \\ &= \mathcal{S}_0 k.\,\langle(\lambda x.\, \mathcal{S}_0 z.\, k\, x)\, e\rangle = e\end{aligned}$$

We used, in sequence, the axioms η_p, $\langle\lambda\rangle$, $\langle p\rangle$ and $\eta_{\langle\cdot\rangle}$. Notice that the use of the axiom $\langle p\rangle$ was correct only because the return type of k was pure. So the equality is not valid in general, but it is valid in the image of the embedding of typed *shift*/*reset*.

The conclusion is as follows: the type system of $\lambda^{\leq}_{\mathcal{S}_0}$ tracks how the program accesses the context stack, and the typed axioms can use the type information to make some reasoning valid which is not valid in general. The typed axioms of Kameyama and Hasegawa are valid in $\lambda^{\leq}_{\mathcal{S}_0}$ when the type annotations are shallow.

6.3 Connection with the axioms for the CPS Hierarchy

We can embed the CPS Hierarchy λ_H [3] in the calculus $\lambda_\$$, as shown in [9]. A natural question is whether Kameyama's axioms for the CPS Hierarchy [5] are validated by the axioms for $\lambda_\$$. The answer is yes. Let $\mathcal{C}_H[\![\cdot]\!]$ be the CPS translation for the CPS Hierarchy and $\mathcal{H}[\![\cdot]\!]$ be the embedding of λ_H inside $\lambda_\$$. In [9] it is proven that $\lambda \vdash \mathcal{C}_H[\![e]\!] = \mathcal{C}[\![\mathcal{H}[\![e]\!]]\!]$. So the following sequence of equivalences is true:

$$\lambda_H \vdash e_1 = e_2 \Leftrightarrow \lambda \vdash \mathcal{C}_H[\![e_1]\!] = \mathcal{C}_H[\![e_2]\!] \Leftrightarrow \lambda \vdash \mathcal{C}[\![\mathcal{H}[\![e_1]\!]]\!] = \mathcal{C}[\![\mathcal{H}[\![e_2]\!]]\!] \Leftrightarrow \lambda_\$ \vdash \mathcal{H}[\![e_1]\!] = \mathcal{H}[\![e_2]\!]$$

Because Kameyama's axioms specialized for the first level coincide with the axioms of Kameyama and Hasegawa for *shift*/*reset*, the result may seem paradoxical: we said in Section 6.1 that these axioms are not valid in $\mathit{shift}_0/\mathit{reset}_0$! There is no paradox because $\mathcal{H}[\![\cdot]\!]$ gives a different embedding of *shift*/*reset* than the one used in Section 6.1:

$$\begin{aligned}\mathcal{H}[\![\mathcal{S}_1 x.\, e]\!] &= \mathcal{S}_0 k.\,\langle e[\lambda x.\, \mathcal{S}_0 f.\, \mathcal{S}_0 g.\,(\lambda y.\, g\,\$\, f\, y)\,\$\, k\, x/x]\rangle \\ \mathcal{H}[\![\langle e\rangle_1]\!] &= \mathcal{S}_0 f.\, \mathcal{S}_0 g.\,(\lambda x.\, g\,\$\, f\, x)\,\$\,\langle\mathcal{H}[\![e]\!]\rangle\end{aligned}$$

7 Conclusion

We have presented sound and complete axioms for untyped languages $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$ and typed languages with subtyping $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$. In future work we will explore polymorphic and call-by-name variants of the languages considered.

Acknowledgments. Many thanks to Dariusz Biernacki, Maciej Piróg and the anonymous referees for their valuable comments. This work was funded by Polish NCN grant DEC-2011/03/B/ST6/00348, and co-funded by the European Social Fund.

References

1 Henk Barendregt. Lambda calculi with types. In Samson Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Vol. 2*, chapter 2, pages 118–309. Oxford University Press, Oxford, 1992.

2 Olivier Danvy and Andrzej Filinski. A functional abstraction of typed contexts. DIKU Rapport 89/12, DIKU, Computer Science Department, University of Copenhagen, Copenhagen, Denmark, July 1989.

3 Olivier Danvy and Andrzej Filinski. Abstracting control. In Mitchell Wand, editor, *Proceedings of the 1990 ACM Conference on Lisp and Functional Programming*, pages 151–160, Nice, France, June 1990. ACM Press.

4 Paul Downen and Zena M. Ariola. A systematic approach to delimited control with multiple prompts. In Helmut Seidl, editor, *ESOP'12*, Lecture Notes in Computer Science, pages 234–253, Tallinn, Estonia, April 2012. Springer-Verlag.

5 Yukiyoshi Kameyama. Axioms for control operators in the CPS hierarchy. *Higher-Order and Symbolic Computation*, 20(4):339–369, 2007. A preliminary version was presented at the Fourth ACM SIGPLAN Workshop on Continuations (CW'04).

6 Yukiyoshi Kameyama and Masahito Hasegawa. A sound and complete axiomatization of delimited continuations. In Olin Shivers, editor, *ICFP'03*, SIGPLAN Notices, Vol. 38, No. 9, pages 177–188, Uppsala, Sweden, August 2003. ACM Press.

7 Oleg Kiselyov and Chung-chieh Shan. A substructural type system for delimited continuations. In Simona Ronchi Della Rocca, editor, *TLCA'07*, number 4583 in Lecture Notes in Computer Science, pages 223–239, Paris, France, June 2007. Springer-Verlag.

8 Marek Materzok and Dariusz Biernacki. Subtyping delimited continuations. In Oliver Danvy, editor, *ICFP'11*, pages 81–93, Tokyo, Japan, September 2011. ACM Press.

9 Marek Materzok and Dariusz Biernacki. A dynamic interpretation of the CPS hierarchy. In Ranjit Jhala and Atsushi Igarashi, editors, *APLAS'12*, number 7705 in Lecture Notes in Computer Science, pages 296–311, Kyoto, Japan, December 2012.

10 Amr Sabry. *The Formal Relationship between Direct and Continuation-Passing Style Optimizing Compilers: A Synthesis of Two Paradigms*. PhD thesis, Computer Science Department, Rice University, Houston, Texas, August 1994. Technical report 94-242.

11 Amr Sabry. Note on axiomatizing the semantics of control operators. Technical Report CIS-TR-96-03, Department of Computer and Information Science, University of Oregon, 1996.

12 Amr Sabry and Matthias Felleisen. Reasoning about programs in continuation-passing style. *Lisp and Symbolic Computation*, 6(3/4):289–360, 1993. A preliminary version was presented at the 1992 ACM Conference on Lisp and Functional Programming (LFP 1992).

A Useful lemmas

▶ **Lemma 34** ($\$_{R\beta}$). $\lambda_{\$} \vdash k \,\$\, (\lambda x.\, e_1) \, e_2 = (\lambda x.\, k \,\$\, e_1) \,\$\, e_2$

Proof. $k \,\$\, (\lambda x.\, e_1) \, e_2 \overset{\$_E}{=} (\lambda x.\, k \,\$\, (\lambda x.\, e_1) \, x) \,\$\, e_2 \overset{\beta_v}{=} (\lambda x.\, k \,\$\, e_1) \,\$\, e_2$ ◀

▶ **Definition 35** (CGS translation of values).

$$\mathcal{G}_v[\![x]\!] \;=\; x \qquad \mathcal{G}_v[\![\lambda x.\, e]\!] \;=\; \lambda x.\, \mathcal{G}[\![e]\!]$$

▶ **Lemma 36.** $\mathcal{G}[\![v]\!] = \mathcal{S}_0 k.\, k \, \mathcal{G}_v[\![v]\!]$

▶ **Lemma 37.** *$\lambda_{\$}$ axiom $\$_E$ is equivalent to the following three equations:*

$$\begin{aligned} v \,\$\, e_1 \, e_2 &= (\lambda x.\, v \,\$\, x \, e_2) \,\$\, e_1 && (\$_L) \\ v \,\$\, v' \, e &= (\lambda x.\, v \,\$\, v' \, x) \,\$\, e && (\$_R) \\ v \,\$\, e_1 \,\$\, e_2 &= (\lambda x.\, v \,\$\, x \,\$\, e_2) \,\$\, e_1 && (\$_{\$}) \end{aligned}$$

Proof. Equations $\$_L$, $\$_R$ and $\$_{\$}$ are obviously instances of $\$_E$. In the other direction the proof is by induction on the context E.

- $E = \bullet$

$$v \,\$\, e \overset{\eta_v}{=} (\lambda x.\, v \, x) \,\$\, e \overset{\$_v}{=} (\lambda x.\, v \,\$\, x) \,\$\, e$$

- $E = E' \, e$

$$\begin{aligned} & v \,\$\, (E' \, e)[e'] \overset{\text{def}}{=} v \,\$\, E'[e' \, e] \overset{\text{ind}}{=} (\lambda x.\, v \,\$\, E'[x]) \,\$\, e' \, e \overset{\$_L}{=} (\lambda x.\, (\lambda x.\, x \,\$\, E'[x]) \,\$\, x \, e) \,\$\, e' \\ \overset{\text{ind}}{=} \;& (\lambda x.\, v \,\$\, E'[x \, e]) \,\$\, e' \overset{\text{def}}{=} (\lambda x.\, v \,\$\, (E' \, e)[x]) \,\$\, e' \end{aligned}$$

The other two cases are similar. ◀

B Proof of Property 1

1-4 proven by induction on the expression e. Only the nontrivial cases are shown.

1. For every $\lambda_{\mathcal{S}_0}$ term e we have $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![\mathcal{D}[\![e]\!]]\!] = e$.

$$\begin{aligned} & \mathcal{D}^{-1}[\![\mathcal{D}[\![\langle e \rangle]\!]]\!] \overset{\text{def}}{=} \mathcal{D}^{-1}[\![(\lambda x.\, x) \,\$\, \mathcal{D}[\![e]\!]]\!] \overset{\text{def}}{=} (\lambda f.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, \mathcal{D}^{-1}[\![\mathcal{D}[\![e]\!]]\!] \rangle) \, (\lambda x.\, x) \\ \overset{\text{ind}}{=} \;& (\lambda f.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, e \rangle) \, (\lambda x.\, x) \overset{\beta_v}{=} \langle (\lambda x.\, \mathcal{S}_0 z.\, x) \, e \rangle \overset{\langle v \rangle}{=} \langle (\lambda x.\, \mathcal{S}_0 z.\, \langle x \rangle) \, e \rangle \\ \overset{\langle \lambda \rangle}{=} \;& \langle (\lambda x.\, x) \, e \rangle \overset{\beta_\Omega}{=} \langle e \rangle \end{aligned}$$

2. For every $\lambda_{\$}$ term e we have $\lambda_{\$} \vdash \mathcal{D}[\![\mathcal{D}^{-1}[\![e]\!]]\!] = e$.

$$\begin{aligned} & \mathcal{D}[\![\mathcal{D}^{-1}[\![e_1 \,\$\, e_2]\!]]\!] \overset{\text{def}}{=} \mathcal{D}[\![(\lambda f.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, \mathcal{D}^{-1}[\![e_2]\!] \rangle) \, \mathcal{D}^{-1}[\![e_1]\!]]\!] \\ \overset{\text{def}}{=} \;& (\lambda f.\, (\lambda x.\, x) \,\$\, (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, \mathcal{D}[\![\mathcal{D}^{-1}[\![e_2]\!]]\!]) \, \mathcal{D}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!] \\ \overset{\text{ind}}{=} \;& (\lambda f.\, (\lambda x.\, x) \,\$\, (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, e_2) \, e_1 \\ \overset{\eta_{\$}}{=} \;& \mathcal{S}_0 k.\, k \,\$\, (\lambda f.\, (\lambda x.\, x) \,\$\, (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, e_2) \, e_1 \\ \overset{\$_{R\beta}}{=} \;& \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, (\lambda x.\, x) \,\$\, (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, e_2) \,\$\, e_1 \\ \overset{\$_{R\beta}}{=} \;& \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, (\lambda x.\, (\lambda x.\, x) \,\$\, \mathcal{S}_0 z.\, f \, x) \,\$\, e_2) \,\$\, e_1 \\ \overset{\beta_{\$}}{=} \;& \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, (\lambda x.\, f \, x) \,\$\, e_2) \,\$\, e_1 \\ \overset{\eta_v}{=} \;& \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, f \,\$\, e_2) \,\$\, e_1 \overset{\$_E}{=} \mathcal{S}_0 k.\, k \,\$\, e_1 \,\$\, e_2 \overset{\eta_{\$}}{=} e_1 \,\$\, e_2 \end{aligned}$$

3. For every $\lambda_{\mathcal{S}_0}$ term e we have $\lambda \vdash \mathcal{C}[\![e]\!] = \mathcal{C}[\![\mathcal{D}[\![e]\!]]\!]$.

$$\begin{aligned}
& \mathcal{C}[\![\langle e\rangle]\!] \overset{\text{def}}{=} \mathcal{C}[\![e]\!]\,(\lambda x.\,\lambda k.\,k\,x) \overset{\eta}{=} \lambda k.\,\mathcal{C}[\![e]\!]\,(\lambda x.\,\lambda k.\,k\,x)\,k \\
\overset{\beta}{=}\ & \lambda k.\,(\lambda f.\,\mathcal{C}[\![e]\!]\,f\,k)\,(\lambda x.\,\lambda k.\,k\,x) \overset{\beta}{=} \lambda k.\,(\lambda k'.\,k'\,(\lambda x.\,\lambda k.\,k\,x))\,(\lambda f.\,\mathcal{C}[\![e]\!]\,f\,k) \\
\overset{\text{ind}}{=}\ & \lambda k.\,(\lambda k'.\,k'\,(\lambda x.\,\lambda k.\,k\,x))\,(\lambda f.\,\mathcal{C}[\![\mathcal{D}[\![e]\!]]\!]\,f\,k) \\
\overset{\text{def}}{=}\ & \mathcal{C}[\![(\lambda x.\,x)\,\$\,\mathcal{D}[\![e]\!]]\!] \overset{\text{def}}{=} \mathcal{C}[\![\mathcal{D}[\![\langle e\rangle]\!]]\!]
\end{aligned}$$

4. For every $\lambda_{\$}$ term e we have $\lambda \vdash \mathcal{C}[\![e]\!] = \mathcal{C}[\![\mathcal{D}^{-1}[\![e]\!]]\!]$.

$$\begin{aligned}
& \mathcal{C}[\![e_1\,\$\,e_2]\!] \overset{\text{def}}{=} \lambda k.\,\mathcal{C}[\![e_1]\!]\,(\lambda f.\,\mathcal{C}[\![e_2]\!]\,f\,k) \\
\overset{\eta}{=}\ & \lambda k.\,\mathcal{C}[\![e_1]\!]\,(\lambda f.\,\mathcal{C}[\![e_2]\!]\,(\lambda x.\,\lambda k.\,f\,x\,k)\,k) \\
\overset{\beta}{=}\ & \lambda k.\,\mathcal{C}[\![e_1]\!]\,(\lambda f.\,\mathcal{C}[\![e_2]\!]\,(\lambda x.\,(\lambda z.\,\lambda k.\,f\,x\,k)\,(\lambda x.\,\lambda k.\,k\,x))\,k) \\
\overset{\text{def}}{=}\ & \lambda k.\,\mathcal{C}[\![e_1]\!]\,(\lambda f.\,\mathcal{C}[\![e_2]\!]\,(\lambda x.\,\mathcal{C}[\![\mathcal{S}_0 z.\,f\,x]\!]\,(\lambda x.\,\lambda k.\,k\,x))\,k) \\
\overset{\beta}{=}\ & \lambda k.\,\mathcal{C}[\![e_1]\!]\,(\lambda f.\,(\lambda k'.\,\mathcal{C}[\![e_2]\!]\,(\lambda x.\,(\lambda x.\,\mathcal{C}[\![\mathcal{S}_0 z.\,f\,x]\!])\,x\,k'))(\lambda x.\,\lambda k.\,k\,x)\,k) \\
\overset{\text{ind}}{=}\ & \lambda k.\,\mathcal{C}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!]\,(\lambda f.\,(\lambda k'.\,\mathcal{C}[\![\mathcal{D}^{-1}[\![e_2]\!]]\!]\,(\lambda x.\,(\lambda x.\,\mathcal{C}[\![\mathcal{S}_0 z.\,f\,x]\!])\,x\,k'))(\lambda x.\,\lambda k.\,k\,x)\,k) \\
\overset{\text{def}}{=}\ & \lambda k.\,\mathcal{C}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!]\,(\lambda f.\,\mathcal{C}[\![\langle(\lambda x.\,\mathcal{S}_0 z.\,f\,x)\,\mathcal{D}^{-1}[\![e_2]\!]\rangle]\!]\,k) \\
\overset{\beta}{=}\ & \lambda k.\,\mathcal{C}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!]\,(\lambda x.\,(\lambda f.\,\mathcal{C}[\![\langle(\lambda x.\,\mathcal{S}_0 z.\,f\,x)\,\mathcal{D}^{-1}[\![e_2]\!]\rangle]\!])\,x\,k) \\
\overset{\text{def}}{=}\ & \mathcal{C}[\![(\lambda f.\,\langle(\lambda x.\,\mathcal{S}_0 z.\,f\,x)\,\mathcal{D}^{-1}[\![e_2]\!]\rangle)\,\mathcal{D}^{-1}[\![e_1]\!]]\!] \overset{\text{def}}{=} \mathcal{C}[\![\mathcal{D}^{-1}[\![e_1\,\$\,e_2]\!]]\!]
\end{aligned}$$

5. $\lambda_{\mathcal{S}_0} \vdash e_1 = e_2$ implies $\lambda_{\$} \vdash \mathcal{D}[\![e_1]\!] = \mathcal{D}[\![e_2]\!]$.

- (β_Ω) $(\lambda x.\,E[x])\,e = E[e]$

$$\begin{aligned}
& \mathcal{D}[\![(\lambda x.\,E[x])\,e]\!] \overset{\text{def}}{=} (\lambda x.\,(\mathcal{D}[\![E]\!])[x])\,\mathcal{D}[\![e]\!] \\
\overset{\eta_\$}{=}\ & \mathcal{S}_0 k.\,k\,\$\,(\lambda x.\,(\mathcal{D}[\![E]\!])[x])\,\mathcal{D}[\![e]\!] \\
\overset{\$_{R\beta}}{=}\ & \mathcal{S}_0 k.\,(\lambda x.\,k\,\$\,(\mathcal{D}[\![E]\!])[x])\,\$\,\mathcal{D}[\![e]\!] \overset{\$_E}{=} \mathcal{S}_0 k.\,k\,\$\,(\mathcal{D}[\![E]\!])[\mathcal{D}[\![x]\!]] \\
\overset{\text{def}}{=}\ & \mathcal{S}_0 k.\,k\,\$\,\mathcal{D}[\![E[x]]\!] \overset{\eta_\$}{=} \mathcal{D}[\![E[x]]\!]
\end{aligned}$$

- $(\langle\mathcal{S}_0\rangle)$ $\langle E[\mathcal{S}_0 x.\,e]\rangle = e[\lambda x.\,\langle E[x]\rangle/x]$

$$\begin{aligned}
& \mathcal{D}[\![\langle E[\mathcal{S}_0 x.\,e]\rangle]\!] \overset{\text{def}}{=} (\lambda x.\,x)\,\$\,(\mathcal{D}[\![E]\!])[\mathcal{S}_0 x.\,\mathcal{D}[\![e]\!]] \\
\overset{\$_E}{=}\ & (\lambda x.\,(\lambda x.\,x)\,\$\,(\mathcal{D}[\![E]\!])[x])\,\$\,\mathcal{S}_0 x.\,\mathcal{D}[\![e]\!] \\
\overset{\beta_\$}{=}\ & \mathcal{D}[\![e]\!][\lambda x.\,(\lambda x.\,x)\,\$\,(\mathcal{D}[\![E]\!])[x]/x] \overset{\text{def}}{=} \mathcal{D}[\![e[\lambda x.\,\langle E[x]\rangle/x]]\!]
\end{aligned}$$

- $(\langle v\rangle)$ $\langle v\rangle = v$

$$\mathcal{D}[\![\langle v\rangle]\!] \overset{\text{def}}{=} (\lambda x.\,x)\,\$\,\mathcal{D}[\![v]\!] \overset{\$_v}{=} (\lambda x.\,x)\,\mathcal{D}[\![v]\!] \overset{\beta_v}{=} \mathcal{D}[\![v]\!]$$

- $(\eta_{\langle\cdot\rangle})$ $\mathcal{S}_0 k.\,\langle(\lambda x.\,\mathcal{S}_0 z.\,k\,x)\,e\rangle = e$

$$\begin{aligned}
& \mathcal{D}[\![\mathcal{S}_0 k.\,\langle(\lambda x.\,\mathcal{S}_0 z.\,k\,x)\,e\rangle]\!] \overset{\text{def}}{=} \mathcal{S}_0 k.\,(\lambda x.\,x)\,\$\,(\lambda x.\,\mathcal{S}_0 z.\,k\,x)\,\mathcal{D}[\![e]\!] \\
\overset{\$_{R\beta}}{=}\ & \mathcal{S}_0 k.\,(\lambda x.\,(\lambda x.\,x)\,\$\,\mathcal{S}_0 z.\,k\,x)\,\$\,\mathcal{D}[\![e]\!] \\
\overset{\beta_\$}{=}\ & \mathcal{S}_0 k.\,(\lambda x.\,k\,x)\,\$\,\mathcal{D}[\![e]\!] \overset{\eta_v}{=} \mathcal{S}_0 k.\,k\,\$\,\mathcal{D}[\![e]\!] \overset{\eta_\$}{=} \mathcal{D}[\![e]\!]
\end{aligned}$$

- $(\langle\lambda\rangle)$ $\langle(\lambda x.\,\mathcal{S}_0 k.\,\langle e_1\rangle)\,e_2\rangle = \langle(\lambda x.\,e_1)\,e_2\rangle$

$$\begin{aligned}
& \mathcal{D}[\![\langle(\lambda x.\,\mathcal{S}_0 k.\,\langle e_1\rangle)\,e_2\rangle]\!] \overset{\text{def}}{=} (\lambda x.\,x)\,\$\,(\lambda x.\,\mathcal{S}_0 k.\,(\lambda x.\,x)\,\$\,\mathcal{D}[\![e_1]\!])\,\mathcal{D}[\![e_2]\!] \\
\overset{\$_{R\beta}}{=}\ & (\lambda x.\,(\lambda x.\,x)\,\$\,\mathcal{S}_0 k.\,(\lambda x.\,x)\,\$\,\mathcal{D}[\![e_1]\!])\,\$\,\mathcal{D}[\![e_2]\!] \overset{\beta_\$}{=} (\lambda x.\,(\lambda x.\,x)\,\$\,\mathcal{D}[\![e_1]\!])\,\$\,\mathcal{D}[\![e_2]\!] \\
\overset{\$_{R\beta}}{=}\ & (\lambda x.\,x)\,\$\,(\lambda x.\,\mathcal{D}[\![e_1]\!])\,\mathcal{D}[\![e_2]\!] \overset{\text{def}}{=} \mathcal{D}[\![\langle(\lambda x.\,e_1)\,e_2\rangle]\!]
\end{aligned}$$

6. $\lambda_{\$} \vdash e_1 = e_2$ implies $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![e_1]\!] = \mathcal{D}^{-1}[\![e_2]\!]$.

- $(\beta_{\$})$ $v \,\$\, \mathcal{S}_0 x.\, e = e[v/x]$

$$\begin{array}{rl} & \mathcal{D}^{-1}[\![v \,\$\, \mathcal{S}_0 x.\, e]\!] \overset{\text{def}}{=} (\lambda f.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, f\, x)\, (\mathcal{S}_0 x.\, \mathcal{D}^{-1}[\![e]\!]) \rangle)\, \mathcal{D}^{-1}[\![v]\!] \\ \overset{\beta_v}{=} & \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, (\mathcal{S}_0 x.\, \mathcal{D}^{-1}[\![e]\!]) \rangle \\ \overset{\langle \mathcal{S}_0 \rangle}{=} & \mathcal{D}^{-1}[\![e]\!][\lambda y.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, y \rangle / x] \\ \overset{\beta_v}{=} & \mathcal{D}^{-1}[\![e]\!][\lambda y.\, \langle \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, y \rangle / x] \overset{\langle \mathcal{S}_0 \rangle}{=} \mathcal{D}^{-1}[\![e]\!][\lambda y.\, \mathcal{D}^{-1}[\![v]\!]\, y / x] \\ \overset{\eta_v}{=} & \mathcal{D}^{-1}[\![e]\!][\mathcal{D}^{-1}[\![v]\!]/x] \overset{\text{def}}{=} \mathcal{D}^{-1}[\![e[v/x]]\!] \end{array}$$

- $(\eta_{\$})$ $\mathcal{S}_0 x.\, x \,\$\, e = e$

$$\mathcal{D}^{-1}[\![\mathcal{S}_0 x.\, x \,\$\, e]\!] \overset{\text{def}}{=} \mathcal{S}_0 x.\, \langle (\lambda y.\, \mathcal{S}_0 z.\, x\, y)\, \mathcal{D}^{-1}[\![e]\!] \rangle \overset{\eta_{\langle\cdot\rangle}}{=} \mathcal{D}^{-1}[\![e]\!]$$

- $(\$_v)$ $v_1 \,\$\, v_2 = v_1\, v_2$

$$\begin{array}{rl} & \mathcal{D}^{-1}[\![v_1 \,\$\, v_2]\!] \overset{\text{def}}{=} \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v_1]\!]\, x)\, \mathcal{D}^{-1}[\![v_2]\!] \rangle \overset{\beta_v}{=} \langle \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v_1]\!]\, \mathcal{D}^{-1}[\![v_2]\!] \rangle \\ \overset{\langle \mathcal{S}_0 \rangle}{=} & \mathcal{D}^{-1}[\![v_1]\!]\, \mathcal{D}^{-1}[\![v_2]\!] \overset{\text{def}}{=} \mathcal{D}^{-1}[\![v_1\, v_2]\!] \end{array}$$

- $(\$_E)$ $v \,\$\, E[e] = (\lambda x.\, v \,\$\, E[x]) \,\$\, e$

$$\begin{array}{rl} & \mathcal{D}^{-1}[\![v \,\$\, E[e]]\!] \overset{\text{def}}{=} \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, ((\mathcal{D}^{-1}[\![E]\!])[\mathcal{D}^{-1}[\![e]\!]]) \rangle \\ \overset{\beta_\Omega}{=} & \langle (\lambda x.\, (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, ((\mathcal{D}^{-1}[\![E]\!])[x]))\, \mathcal{D}^{-1}[\![e]\!] \rangle \\ \overset{\text{def}}{=} & \langle (\lambda x.\, (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, \mathcal{D}^{-1}[\![E[x]]\!])\, \mathcal{D}^{-1}[\![e]\!] \rangle \\ \overset{\langle \lambda \rangle}{=} & \langle (\lambda x.\, \mathcal{S}_0 z.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, \mathcal{D}^{-1}[\![E[x]]\!] \rangle)\, \mathcal{D}^{-1}[\![e]\!] \rangle \\ \overset{\beta_v}{=} & \langle (\lambda x.\, \mathcal{S}_0 z.\, (\lambda x.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, \mathcal{D}^{-1}[\![E[x]]\!] \rangle)\, x)\, \mathcal{D}^{-1}[\![e]\!] \rangle \\ \overset{\text{def}}{=} & \mathcal{D}^{-1}[\![(\lambda x.\, v \,\$\, E[x]) \,\$\, e]\!] \end{array}$$

C Proof of Lemma 6

For every $\lambda_{\$}$ term e we have $\lambda_{\$} \vdash e = \mathcal{G}[\![e]\!]$.

Proof is by induction on the expression e. Only the nontrivial cases are presented.

- $e = \lambda x.\, e'$

$$\lambda x.\, e' \overset{\eta_{\$}}{=} \mathcal{S}_0 k.\, k \,\$\, \lambda x.\, e' \overset{\$_v}{=} \mathcal{S}_0 k.\, k\, (\lambda x.\, e') \overset{\text{ind}}{=} \mathcal{S}_0 k.\, k\, (\lambda x.\, \mathcal{G}[\![e']\!]) \overset{\text{def}}{=} \mathcal{G}[\![\lambda x.\, e']\!]$$

- $e = e_1\, e_2$

$$\begin{array}{rl} & e_1\, e_2 \overset{\eta_{\$}}{=} \mathcal{S}_0 k.\, k \,\$\, e_1\, e_2 \overset{\$_E}{=} \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, f\, e_2) \,\$\, e_1 \overset{\$_E}{=} \mathcal{S}_0 k.\, (\lambda f.\, (\lambda x.\, k \,\$\, f\, x) \,\$\, e_2) \,\$\, e_1 \\ \overset{\text{ind}}{=} & \mathcal{S}_0 k.\, (\lambda f.\, (\lambda x.\, k \,\$\, f\, x) \,\$\, \mathcal{G}[\![e_2]\!]) \,\$\, \mathcal{G}[\![e_1]\!] \overset{\text{def}}{=} \mathcal{G}[\![e_1\, e_2]\!] \end{array}$$

- $e = e_1 \,\$\, e_2$

$$\begin{array}{rl} & e_1 \,\$\, e_2 \overset{\eta_{\$}}{=} \mathcal{S}_0 k.\, k \,\$\, e_1 \,\$\, e_2 \overset{\$_E}{=} \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, f \,\$\, e_2) \,\$\, e_1 \\ \overset{\text{ind}}{=} & \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, f \,\$\, \mathcal{G}[\![e_2]\!]) \,\$\, \mathcal{G}[\![e_1]\!] \overset{\text{def}}{=} \mathcal{G}[\![e_1 \,\$\, e_2]\!] \end{array}$$

$$\frac{}{\epsilon \ll^{\leq}} \qquad \frac{}{[\tau\ \sigma]\ \tau\ \sigma \ll^{\leq}} \qquad \frac{\sigma \ll^{\leq} \vec{\sigma}}{\sigma \ll^{\leq} \epsilon\ \vec{\sigma}} \qquad \frac{[\tau_3\ \sigma_3]\ \tau_2\ \sigma_2 \ll^{\leq} \vec{\sigma} \qquad \tau_2'\ \sigma_2' \leq \tau_2'\ \sigma_2}{[\tau_3\ \sigma_3]\ \tau_1\ \sigma_1 \ll^{\leq} [\tau_2'\ \sigma_2']\ \tau_1\ \sigma_1\ \vec{\sigma}}$$

$$\frac{}{\Gamma, x : \tau \rhd x : \tau}\ \text{VAR} \qquad \frac{\Gamma, x : \tau_1 \rhd e : \tau_2\ \sigma}{\Gamma \rhd \lambda x : \tau_1.\, e : \tau_1 \xrightarrow{\sigma} \tau_2}\ \text{ABS} \qquad \frac{\Gamma, x : \tau_1 \xrightarrow{\sigma} \tau_2 \rhd e : \tau_3\ \sigma'}{\Gamma \rhd \mathcal{S}_0 x : \tau_1 \xrightarrow{\sigma} \tau_2.\, e : \tau_1\ [\tau_2\ \sigma]\ \tau_3\ \sigma'}\ \text{SFT}$$

$$\frac{\Gamma \rhd e_1 : \tau_1 \xrightarrow{\sigma_3} \tau_2\ \sigma_1 \qquad \Gamma \rhd e_2 : \tau_1\ \sigma_2 \qquad \sigma \ll^{\leq} \sigma_1\ \sigma_2\ \sigma_3}{\Gamma \rhd e_1\ e_2 : \tau_2\ \sigma}\ \text{APP}$$

Rule for $\lambda^{\leq}_{\mathcal{S}_0}$:

$$\frac{\Gamma \rhd e : \tau''\ [\tau'\ \sigma']\ \tau\ \sigma \qquad \tau'' \leq \tau'\ \sigma'}{\Gamma \rhd \langle e \rangle : \tau\ \sigma}\ \text{RST}$$

Rule for $\lambda^{\leq}_{\$}$:

$$\frac{\Gamma \rhd e_1 : \tau_1 \xrightarrow{\sigma'} \tau_2\ \sigma_1 \qquad \Gamma \rhd e_2 : \tau_1\ [\tau_2\ \sigma']\ \tau_3\ \sigma_2 \qquad \sigma \ll^{\leq} \sigma_1\ \sigma_2}{\Gamma \rhd e_1\ \$\ e_2 : \tau_3\ \sigma}\ \text{DOL}$$

Figure 11 The type system giving minimal types for $\lambda_{\$}$ and $\lambda^{\leq}_{\$}$.

D Proof of Theorem 21 (coherence)

The language $\lambda^{\leq}_{\$}$ has minimal types in the following sense:

▶ **Lemma 38** (Minimal types). *For every e, Γ, τ, σ such that $\Gamma \vdash e : \tau\ \sigma$ there exist τ', σ' such that $\Gamma \vdash e : \tau'\ \sigma'$ and for every τ'', σ'' such that $\Gamma \vdash e : \tau''\ \sigma''$ we have $\tau'\ \sigma' \leq \tau''\ \sigma''$.*

Proof. The type system in Figure 11 gives the minimal type. This can be proven by induction on the derivation of $\Gamma \vdash e : \tau\ \sigma$. ◀

For every derivation D_M of the minimal type judgement of Figure 11 we can find a type derivation for the same type which corresponds to the derivation D_M; let us call it $\mathcal{D}(D_M)$. We can prove the following lemma:

▶ **Lemma 39.** *For every derivation D_M of $\Gamma \rhd e : \tau\ \sigma$ and every derivation D of $\Gamma \vdash e : \tau'\ \sigma'$ we have $\lambda^{\rightarrow}_{\$}; \Gamma \vdash \mathcal{G}[\![e]\!]_D = \mathcal{G}[\![\tau\ \sigma \leq \tau'\ \sigma']\!][\mathcal{G}[\![e]\!]_{\mathcal{D}(D_M)}]$.*

Proof. Induction on the derivation D. ◀

Coherence follows immediately.

E Kameyama and Hasegawa's axiomatization of shift/reset

The axioms were presented in [6]. Syntax was adapted to the one used in this paper.

$$\begin{array}{rcll}
(\lambda x.\, e)\, v & = & e[v/x] & & (\beta_v) \\
\lambda x.\, v\, x & = & v & x \notin \mathrm{V}(v) & (\eta_v) \\
(\lambda x.\, E[x])\, e & = & E[e] & x \notin \mathrm{V}(E) & (\beta_\Omega) \\
\langle v \rangle & = & v & & (\textit{reset-value}) \\
\langle (\lambda x.\, e_1)\, \langle e_2 \rangle \rangle & = & (\lambda x.\, \langle e_1 \rangle)\, \langle e_2 \rangle & & (\textit{reset-lift}) \\
\mathcal{S}k.\, k\, e & = & e & k \notin \mathrm{V}(e) & (\mathcal{S}\textit{-elim}) \\
\langle E[\mathcal{S}k.\, e] \rangle & = & e[\lambda x.\, \langle E[x] \rangle / k] & x \notin \mathrm{V}(E) & (\textit{reset-}\mathcal{S}) \\
\mathcal{S}k.\, \langle e \rangle & = & \mathcal{S}k.\, e & & (\mathcal{S}\textit{-reset})
\end{array}$$

On dialogue games and coherent strategies*

Paul-André Melliès

CNRS, Laboratoire PPS (Preuves, Programmes, Systèmes), UMR 7126
Université Paris Diderot, Sorbonne Paris Cité, F–75205 Paris, France.
mellies@pps.univ-paris-diderot.fr

Abstract

We explain how to see the set of positions of a dialogue game as a coherence space in the sense of Girard or as a bistructure in the sense of Curien, Plotkin and Winskel. The coherence structure on the set of positions results from a Kripke translation of tensorial logic into linear logic extended with a necessity modality. The translation is done in such a way that every innocent strategy defines a clique or a configuration in the resulting space of positions. This leads us to study the notion of configuration designed by Curien, Plotkin and Winskel for general bistructures in the particular case of a bistructure associated to a dialogue game. We show that every such configuration may be seen as an interactive strategy equipped with a backward as well as a forward dynamics based on the interplay between the stable order and the extensional order. In that way, the category of bistructures is shown to include a full subcategory of games and coherent strategies of an interesting nature.

1998 ACM Subject Classification D.3.1 Formal Definitions and Theory, F.3.2 Semantics of Programming Languages, F.4.1 Mathematical Logic

Keywords and phrases Game semantics, Stable order, Extensional order, Bistructures, Tensorial logic, Innocent strategies

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.540

1 Introduction

An important dichotomy in the denotational semantics of a programming language like PCF is provided by the distinction between the *qualitative* and the *quantitative* interpretations of the language. The distinction is important but recent since the first quantitative model emerged in the work by Girard on quantitative domains [9] only a few months before the discovery of linear logic. All the denotational models of PCF were qualitative before that. This includes the domain-theoretic models either based on continuous functions between Scott domains [24] or on stable functions between dI-domains [2] as well as the precursor of game semantics based on sequential algorithms between concrete data structures [3]. The difference between qualitative and quantitative models is best understood today by translating the intuitionistic types of PCF into formulas of linear logic. There, the distinction between the two classes of models boils down to the way the exponential modality ! of linear logic is interpreted. As shown by Ehrhard in his work on differential linear logic, the quantitative models of linear logic are usually better behaved and closer to a mathematical understanding of resource (formal series, differential calculus) because they incorporate the number of times a procedure is called by its environment. On the other hand, the qualitative models are precious tools for automatic verification of software because they interpret finite types (typically limited to booleans or to finite approximations of the natural numbers) as

* This work has been partly supported by the ANR Project RECRE.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 540–562

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

finite mathematical structures, and thus provide mechanical procedures to *decide* specific properties of programs.

Quite interestingly, most interactive models based on game semantics are quantitative, rather than qualitative. There is a good reason for that: once understood how to track the several copies of a game A in the game $!A$ by using indices or pointers, it is generally easier to describe the behaviour of a PCF program in exactly the same way as it proceeds in time, typically in a Krivine machine. As a consequence, the number of times a program of type $(!A) \multimap B$ calls its procedure of type A is generally reflected in the game model. As we already mentioned, a remarkable counter-example to this general principle is provided by the sequential algorithm model of PCF [3] which is indeed operational and qualitative at the same time. Lamarche and Curien [15, 5] have shown very early in the history of game semantics how to reformulate this model of PCF as a model of intuitionistic linear logic based on *sequential data structures* — which we prefer to call here *simple games* in order to distinguish them among the more general *dialogue games*. Because of the qualitative nature of the model, simple games are defined as alternating decision trees, without the need for extra indexing or pointer structure. The key idea of the model is to define the simple game $!A$ as the tree of partial explorations of a given strategy σ of the simple game A. The contraction $!A \to !A \circledast !A$ of linear logic is then interpreted by a clever *memoisation* procedure which keeps track of the portion of the strategy σ of the game A on the left explored by the two environments playing independently on the copies of $!A$ on the right. In this way, one obtains a qualitative model of intuitionistic linear logic whose co-Kleisli category embeds in the category of sequential algorithms originally introduced by Berry and Curien. Note that we write $A \circledast B$ for the tensor product of simple games defined by interleaving, in order to distinguish it from the tensor product of dialogue games $A \otimes B$.

The interest in this specific Curien-Lamarche modality ! has been recently revived by the observation that the category of dialogue games and innocent strategies defined by Hyland and Ong [12] may be reconstructed as a bi-Kleisli category from the category of simple games, using a quantitative (or repetitive) version of the modality [11]. For the sake of completeness, we find instructive to take the reverse point of view here, and to see the category **Simple** of simple games as a specific full subcategory of a category **Dialogue** of dialogue games and innocent strategies. This category **Dialogue** should be understood as a resource-aware and linear variant of the original category in [12]. At this point, it is worth recalling the definition of a dialogue category, see [23] for instance:

▶ **Definition 1** (Dialogue category)**.** A dialogue category $\mathcal{C}$ is a symmetric monoidal category equipped with an object $\bot$ together with a functor

$$\neg \quad : \quad \mathcal{C}^{\,op} \quad \longrightarrow \quad \mathcal{C} \tag{1}$$

and a family of bijections

$$\varphi_{A,B} \quad : \quad \mathcal{C}(A \otimes B, \bot) \quad \cong \quad \mathcal{C}(A, \neg B)$$

natural in A and B. A dialogue category is called affine when it is equipped with a natural family of morphisms (called weakening)

$$e_A \quad : \quad \neg A \quad \longrightarrow \quad 1.$$

The category **Dialogue** of dialogue games and total innocent strategies may be concisely defined as the free affine dialogue category with finite sums (and tensors distributing over these finite sums). A more concrete definition will appear in §3 but the conceptual definition is convenient at this introductory stage. Similarly,

▶ **Definition 2** (Negation category)**.** A negation category $\mathcal{C}$ is a category equipped with a functor (1) and with a family of bijections

$$\nu_{A,B} \quad : \quad \mathcal{C}(A, \neg B) \quad \cong \quad \mathcal{C}(B, \neg A)$$

natural in A and B.

The category **Simple** of simple games and total sequential strategies may be concisely defined as the free negation category with finite sums. Note that **Simple** coincides with the free category $Fam(\mathcal{G})$ with finite sums (or finite family construction) generated by the category $\mathcal{G}$ of finite Opponent starting games and total strategies considered in [5]. As a dialogue category, the category **Dialogue** is also a negation category. This implies the existence of a negation and finite sum preserving functor

$$\textit{embedding} \quad : \quad \textbf{Simple} \quad \longrightarrow \quad \textbf{Dialogue}$$

The functor is full and faithful, and injective on objects. As such, it identifies the category **Simple** of simple games to a full subcategory of the category **Dialogue** of dialogue games. The category **Simple** is symmetric monoidal closed with tensor product and linear implication noted $\circledast$ and $-\!\!*$ respectively. As such, it defines a dialogue category with negation defined as $\neg A = A -\!\!* \bot$ where $\bot$ is the simple game with one initial Player move $*$ (which may be also seen as a unique initial position $*$ of the game) followed by a single Opponent move q. Again, by the universal characterization of the category **Dialogue**, this induces a finite sum, tensor and negation preserving functor

$$\textit{pathification} \quad : \quad \textbf{Dialogue} \quad \longrightarrow \quad \textbf{Simple}$$

which we call *pathification* because it transports every dialogue game A to a simple game entirely defined by its alternating paths. Despite its name, the *pathification* functor is a brutal transformation on the original dialogue game, since it destroys the asynchronous structure of the asynchronous game A and only retains its alternating paths. On the other hand, every simple game is already a tree, and thus the composite functor

$$\textbf{Simple} \quad \xrightarrow{\textit{embedding}} \quad \textbf{Dialogue} \quad \xrightarrow{\textit{pathification}} \quad \textbf{Simple}$$

is equal to the identity. One preliminary observation of the paper is that the tensor product $A \circledast B$ between simple games factors as

$$A \circledast B \quad = \quad \textit{pathification}(\textit{embedding}(A) \otimes \textit{embedding}(B))$$

and similarly, that the Curien-Lamarche exponential modality ! factors as

$$\textbf{Simple} \quad \xrightarrow{\textit{shriek}} \quad \textbf{Dialogue} \quad \xrightarrow{\textit{pathification}} \quad \textbf{Simple} \tag{2}$$

Note that the transformation *shriek* is entirely described by the recursive equation

$$\textit{shriek}\left(\bigoplus_{i\in I} \neg \bigoplus_{j\in J_i} \neg A_{ij}\right) \quad = \quad \bigoplus_{i\in I} \bigotimes_{j\in J_i} \neg\neg\, \textit{shriek}\,(A_{ij})$$

whose purpose is to replace every cartesian product (or negated sum indexed by $j \in J_i$) by the corresponding tensor product. We will illustrate the construction in (3) and (5). In particular, for every pair of simple games A, B, there exists a bijection

$$\textbf{Simple}(!A, B) \quad \cong \quad \textbf{Dialogue}(\textit{shriek}(A), \textit{embedding}(B)).$$

This basic observation seems to underlie a lot of work in the field of game semantics, in particular the graph-theoretic formulation of the sequential algorithm model by Hyland and Schalk [13]. A fundamental difficulty (or phenomenon) arises at this point of our analysis: the transformation *shriek* is *not* functorial — and this is precisely the reason why we prefered to indicate it with a dotted line in (2). In order to understand what is going wrong, let us define 1 as the simple game with a unique Player move $*$ (or initial position) and the Sierpinski game $\Sigma = \neg\neg 1$ as the simple game with a unique initial Player move $*$ (or initial position) followed by a unique Opponent move *done*, itself followed by a unique Player move *done*. The cartesian product $\Sigma \,\&\, \Sigma$ in the category **Simple** is equal to the simple game $\neg\big((\neg 1) \oplus (\neg 1)\big)$. Now, consider the morphism

$$\boldsymbol{\Sigma} \;=\; \begin{array}{c} \mathit{done} \\ {\scriptstyle P}\uparrow \\ q \\ {\scriptstyle O}\uparrow \\ \bot \end{array} \quad \xrightarrow{\;\sigma\;} \quad \begin{array}{ccc} \mathit{done}_L & & \mathit{done}_R \\ {\scriptstyle P}\uparrow & & {\scriptstyle P}\uparrow \\ q_L & & q_R \\ {\scriptstyle O}\uparrow & \bot & {\scriptstyle O}\uparrow \end{array} \quad = \boldsymbol{\Sigma} \,\&\, \boldsymbol{\Sigma} \tag{3}$$

in the category **Simple**, where σ denotes the strategy which starts at the initial position $(\bot, \bot)$ and consists of the two sequences of moves below:

$$\begin{array}{rcccccccccc} s_L & : & (\bot,\bot) & \xrightarrow{O} & (\bot, q_L) & \xrightarrow{P} & (q, q_L) & \xrightarrow{O} & (\mathit{done}, q_L) & \xrightarrow{P} & (\mathit{done}, \mathit{done}_L) \\ s_R & : & (\bot,\bot) & \xrightarrow{O} & (\bot, q_R) & \xrightarrow{P} & (q, q_R) & \xrightarrow{O} & (\mathit{done}, q_R) & \xrightarrow{P} & (\mathit{done}, \mathit{done}_R) \end{array} \tag{4}$$

together with their even-length prefixes. We indicate with a grey orb in (3) the fact that the moves q_L and q_R are incompatible and thus cannot appear in the same play of the game $\Sigma \,\&\, \Sigma$. By definition, *shriek* transports the simple game Σ into itself, and the simple game $\Sigma \,\&\, \Sigma = \neg\big((\neg 1) \oplus (\neg 1)\big)$ into the dialogue game $\Sigma \otimes \Sigma = (\neg\neg 1) \otimes (\neg\neg 1)$. We claim that *shriek* is not functorial because it cannot transport the strategy σ to any innocent strategy

$$\boldsymbol{\Sigma} \;=\; \begin{array}{c} \mathit{done} \\ {\scriptstyle P}\uparrow \\ q \\ {\scriptstyle O}\uparrow \\ \bot \end{array} \quad \xrightarrow{\;\tau\;} \quad \begin{array}{ccc} \mathit{done}_L & & \mathit{done}_R \\ {\scriptstyle P}\uparrow & & {\scriptstyle P}\uparrow \\ q_L & & q_R \\ {\scriptstyle O}\uparrow & \bot & {\scriptstyle O}\uparrow \end{array} \quad = \boldsymbol{\Sigma} \otimes \boldsymbol{\Sigma} \tag{5}$$

in the category **Dialogue**. Note that we remove the grey orb between the moves q_L and q_R in the case of the dialogue game $\Sigma \otimes \Sigma$ to indicate that the two moves are compatible in the game. Imagine that there exists such an innocent strategy $\tau = \mathit{shriek}(\sigma)$. In order to make our argument work, we will make the mild hypothesis that any reasonable functorial definition of *shriek* should transport the projection $\pi_i : \Sigma \,\&\, \Sigma \to \Sigma$ to the expected strategy $\mathit{shriek}(\pi_i) : \Sigma \otimes \Sigma \to \Sigma$ which plays a copycat strategy between Σ and the first or second component of $\Sigma \otimes \Sigma$ depending on the value of $i = 1, 2$. With this additional hypothesis, it is easy to deduce from the equality $\pi_i \circ \sigma = id_\Sigma$ (for $i = 1, 2$) and from the totality of τ that the strategy $\tau = \mathit{shriek}(\sigma)$ coincides with the strategy consisting of the two plays

$$\begin{array}{rcccccc} s_{LR} & : & (\bot,\bot,\bot) & \xrightarrow{O^{(*)}} & (\bot, q_L, \bot) & \xrightarrow{P^{(*)}} & (q, q_L, \bot) \\ & & & \xrightarrow{O} & (\mathit{done}, q_L, \bot) & \xrightarrow{P} & (\mathit{done}, \mathit{done}_L, \bot) \\ & & & \xrightarrow{O} & (\mathit{done}, \mathit{done}_L, q_R) & \xrightarrow{P} & (\mathit{done}, \mathit{done}_L, \mathit{done}_R) \\ s_{RL} & : & (\bot,\bot,\bot) & \xrightarrow{O} & (\bot, \bot, q_R) & \xrightarrow{P} & (q, \bot, q_R) \\ & & & \xrightarrow{O} & (\mathit{done}, \bot, q_R) & \xrightarrow{P} & (\mathit{done}, \bot, \mathit{done}_R) \\ & & & \xrightarrow{O^{(**)}} & (\mathit{done}, q_L, \mathit{done}_R) & \xrightarrow{P^{(**)}} & (\mathit{done}, \mathit{done}_L, \mathit{done}_R) \end{array}$$

together with their even-length prefixes. One recognizes here the contraction strategy $\Sigma \to \Sigma \circledast \Sigma$ of the Curien-Lamarche model for the simple game $\Sigma = !\Sigma$. Our whole point is that the strategy τ is *not* innocent as a strategy $\Sigma \to \Sigma \otimes \Sigma$ because it does not play in the same way in the move $P^{(*)}$ of the play s_{LR} and in the move $P^{(**)}$ of the play s_{RL} although the Player views are the same seen from the move $O^{(*)}$ and from the move $O^{(**)}$.

In order to repair the situation, we introduce the notion of *coherent strategy* which relaxes the familiar notion of innocent strategy between dialogue games in such a way that (1) there exists a functor

$$\textbf{Dialogue} \xrightarrow{\ embedding\ } \textbf{Coherent}$$

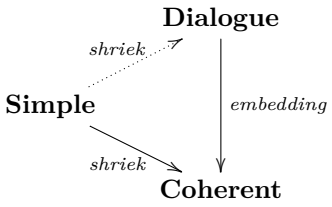
which enables one to transport every strategy $\sigma : A \to B$ between simple games into a coherent strategy using the composite functor

$$\textbf{Simple} \xrightarrow{\ embedding\ } \textbf{Dialogue} \xrightarrow{\ embedding\ } \textbf{Coherent}$$

and moreover (2) a functor

$$\textbf{Simple} \xrightarrow{\ shriek\ } \textbf{Coherent} \tag{6}$$

making the diagram below commute



(7)

One main purpose of this paper is thus to introduce the notion of *coherent strategy* on a dialogue game. We proceed in the same (slightly unconventional) way as the notion emerged in our work. First, we recall in §2 the relationship between dialogue games and tensorial logic, and then define in §3 the notion of innocent strategy we have in mind. Then, we introduce in §4 a Kripke translation of tensorial logic into linear logic extended with a necessity modality (noted $\Box$) which enables us to interpret the set of positions of a dialogue game as a coherence space in the sense of Girard or as a bistructure in the sense of Curien, Plotkin and Winskel [6]. After briefly recalling in the Appendix this model of bistructures, we show in §5 that the configurations σ of the bistructure $[A]$ of positions of a dialogue game A are positional strategies extending the familiar notion of innocent strategies. These strategies are precisely what we call the *coherent strategies* of a dialogue game. Accordingly, the category **Coherent** is defined as the category of coherent strategies between dialogue games. We thus obtain a series of functorial translations:

$$\textbf{Simple} \xrightarrow{\ embedding\ } \textbf{Dialogue} \longrightarrow \textbf{Coherent} \xrightarrow{\ forgetful\ } \textbf{Bistr}$$

where **Bistr** denotes the category of bistructures and configurations introduced by Curien, Plotkin and Winskel [6] and where *forgetful* adapts to **Coherent** the forgetful functor U from the category $\mathcal{M}$ of coalgebras of the comonad $\Box$ to the category **Bistr**. One interesting observation is that the functor

$$\textbf{Simple} \xrightarrow{\ shriek\ } \textbf{Coherent} \xrightarrow{\ forgetful\ } \textbf{Bistr} \xrightarrow{\ \Box\ } \textbf{Bistr} \tag{8}$$

transports every simple game A to the bistructure $![A]$ where $!$ denotes the qualitative exponential modality of **Bistr**. From this follows that the functor lifts to a functor between the Kleisli categories associated to **Simple** and to **Bistr**. We deduce that every sequential algorithm $\sigma : !A \to B$ defines a stable and extensional function $\Gamma(A) \to \Gamma(B)$ between the associated bidomains of configurations.

2 Dialogue games and tensorial logic

Tensorial logic is a primitive logic of tensor and negation which refines linear logic by relaxing the hypothesis that negation is involutive. At the same time, tensorial logic may be seen as a resource-aware version of polarized linear logic developed by Laurent [17] which itself was based on the ideas by Girard on polarities in classical logic [10]. In particular, it extends the connection between polarized linear logic and dialogue games formulated in [16] to the positional and resource-aware notion of dialogue game defined below.

▶ **Definition 3.** A dialogue game is defined as a family of rooted trees (or forest) where every node m is equipped with an equivalence relation **conflict**$[m]$ on its set of children. A node of the forest A is called a *move* of the dialogue game. One writes $m \vdash_A n$ when the node n is a child of the node m in the forest A, and one declares in that case that the move m justifies the move n. Accordingly, a root of the forest A is called an *initial move* of the dialogue game because it is not justified by any other move. A position of the dialogue game A is then defined as a (non-empty) subtree x of the forest A containing only pairwise non-conflicting moves. The set of positions of a dialogue game is denoted **Pos**(A). By convention, we declare that every move of odd depth is Player, and every move of even depth is Opponent. In other words, every initial move is Player, and every branch of the forest is then alternating between Opponent and Player moves.

We will make use of the fact that every finite dialogue game A may be alternatively seen as a formula of tensorial logic:

$$A, B \quad ::= \quad 0 \mid 1 \mid A \oplus B \mid A \otimes B \mid \neg A$$

modulo the equations:

$$A \otimes (B \oplus C) \cong (A \otimes B) \oplus (A \otimes C) \qquad\qquad 0 \otimes A \cong 0$$

together with the associativity and commutativity of $\oplus$ and $\otimes$ and the fact that the formulas 0 and 1 are their respective units. Let us briefly explain how the correspondence works. The dialogue game 0 is the empty forest and the sum $A \oplus B$ of two dialogue games is obtained by putting the two forests A and B side by side. As already mentioned, the game 1 is the tree with a unique Player move $*$. The negation $\neg A$ of a dialogue game A is defined by "lifting" the game A with a move $*$ which justifies the initial moves of the game A. The equivalence relation **conflict**$[*]$ is defined as the total relation, hence every two moves justified by the unique initial move $*$ are conflicting in the game $\neg A$. The tensor product of two dialogue games is required to satisfy the distributivity law

$$\bigoplus_{i \in I} A_i \quad \otimes \quad \bigoplus_{j \in J} B_j \quad = \quad \bigoplus_{(i,j) \in I \times J} A_i \otimes B_j$$

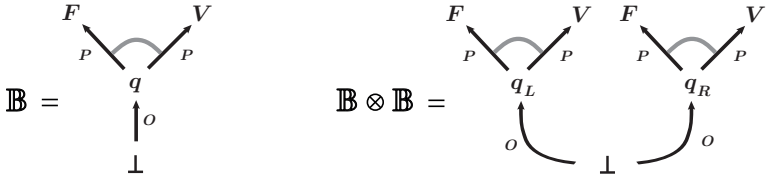
For that reason, the tensor product of two finite dialogue games is entirely described by the equality

$$A \quad = \quad \bigotimes_{i \in I} \neg A_i$$

where the dialogue game A is defined as the *coalesced sum* of the trees $\neg A_i$. This coalesced sum A is the dialogue game with unique initial move $*$ obtained (1) by taking the disjoint sum of the trees $\neg A_i$ and then (2) by identifying the unique initial move $*_i$ of each tree $\neg A_i$ to the unique initial move $*$ of the game A. By definition of a coalesced sum, this dialogue game A is a tree whose unique initial move justifies the moves justified by the root $*_i$ in the game A_i. Its compatibility relation is defined as follows:

$$\mathbf{conflict}[*_A] \quad = \quad \biguplus_{i\in I} \mathbf{conflict}[*_i]$$

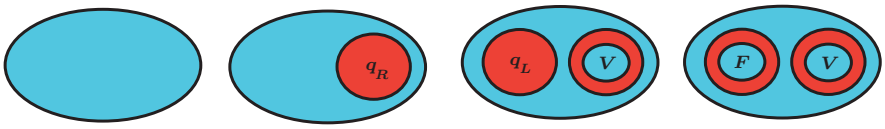
Typically, the boolean formula $1 \oplus 1$ is interpreted as the forest with only two nodes V and F (for *Vrai* and *Faux*, true and false in French) whereas its double negation $\mathbb{B} = \neg\neg(1 \oplus 1)$ and the tensor product $\mathbb{B} \otimes \mathbb{B}$ define the following dialogue games:



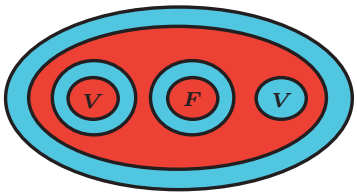
A dialogue game is called *simple* when the conflict relation is full over every move of the game. For instance, the dialogue game $\mathbb{B}$ is simple whereas the dialogue game $\mathbb{B} \otimes \mathbb{B}$ is not simple because the two moves q_L and q_R are not in the same equivalence class of $\mathbf{conflict}[*]$. Note that the set of positions of a dialogue game may be defined inductively as follows:

$$\begin{array}{rcl}
\mathbf{Pos}\,(0) & = & \varnothing \\
\mathbf{Pos}\,(1) & = & \{*\} \\
\mathbf{Pos}\,(A \oplus B) & = & \mathbf{Pos}\,(A) \;+\; \mathbf{Pos}\,(B) \\
\mathbf{Pos}\,(A \otimes B) & = & \mathbf{Pos}\,(A) \;\times\; \mathbf{Pos}\,(B) \\
\mathbf{Pos}\,(\neg A) & = & \mathbf{Pos}\,(A) \;+\; \{*\}
\end{array}$$

Every such position x may be nicely depicted by drawing every move m in it as a circle (or as an ellipse) containing the circles corresponding to the moves n justified by m. The colour convention is to depict the Player moves as blue circles, and the Opponent moves as red circles. Typically, the four positions $\{\bot\}$, $\{\bot, q_R\}$, $\{\bot, q_R, V_R, q_L\}$ and $\{\bot, q_L, F_L, q_R, V_R\}$ of the game $\mathbb{B} \otimes \mathbb{B}$ are respectively depicted as



Similarly, the maximal position of the dialogue game $(\mathbb{B} \otimes \mathbb{B}) \multimap \mathbb{B} = \neg(\mathbb{B} \otimes \mathbb{B} \otimes \neg(1 \oplus 1))$ is depicted as



(9)

where, by convention, we write $A \multimap B$ for the dialogue game $\neg(A \otimes B')$ when $B = \neg B'$. The intuition behind these pictures is that every move m of a dialogue game is a *memory*

cell of a more advanced technology than in the case of concrete data structures, since it may contain several *independent* cells, each of them filled by a value. Quite obviously, each of these cells corresponds to a specific equivalence class of **conflict**$[m]$. This is typically the case of the Opponent move q in the position (9) which is filled by the three independent "values" q_L, q_R and *done*. Note that one recovers the traditional notion of memory cell when the dialogue game is simple, since in that case every memory cell is filled by at most one value.

3 Innocent strategies

In order to define the notion of innocent strategy on dialogue games, we find convenient to recall the asynchronous formulation of innocence formulated in [20]. The starting point of the approach is the idea that every dialogue game A defines an asynchronous transition system

- whose nodes are the positions of the game,
- with a transition $m : x \to y$ between two positions whenever $y = x \uplus \{m\}$ where m is a move of the game and $\uplus$ means disjoint sum,
- with a permutation $(x \to y_1 \to z) \sim (x \to y_2 \to z)$ whenever $y_1 = x \uplus \{m\}$, $y_2 = x \uplus \{n\}$ and $z = x \uplus \{m, n\}$ for two different moves m and n of the game.

Every transition $m : x \to y$ is polarized either as Player or Opponent depending on the polarity of the move m added to the position x in order to obtain the position y. Every initial Player move $*$ of the dialogue game defines an initial position $\{*\}$ of the associated asynchronous transition system. By convention, we generally identify the initial position $\{*\}$ with the initial Player move $*$.

▶ **Definition 4.** A sequential play of a dialogue game A is defined as a path

$$* \xrightarrow{m_1} x_1 \xrightarrow{m_2} \cdots \xrightarrow{m_k} x_k$$

starting from an initial position $*$ of the asynchronous transition system and then alternating between Opponent and Player moves. In particular, every move m_k is Opponent when k is odd and Player when k is even. The position x is called the *target position* of the play s. A play is called *empty* when $k = 0$. There is a one-to-one correspondence between the initial positions of a dialogue game and its empty plays.

▶ **Definition 5.** A sequential strategy σ of a dialogue game A is defined as set of even-length sequential plays which

- has a starting point: σ contains the empty play $*$ for exactly one initial position $*$,
- is closed under even-length prefix: $s \cdot m \cdot n \in \sigma$ implies that $s \in \sigma$,
- is deterministic: $s \cdot m \cdot n_1 \in \sigma$ and $s \cdot m \cdot n_2 \in \sigma$ implies that $n_1 = n_2$

for all plays s and all moves m, n, n_1, n_2 of the dialogue game.

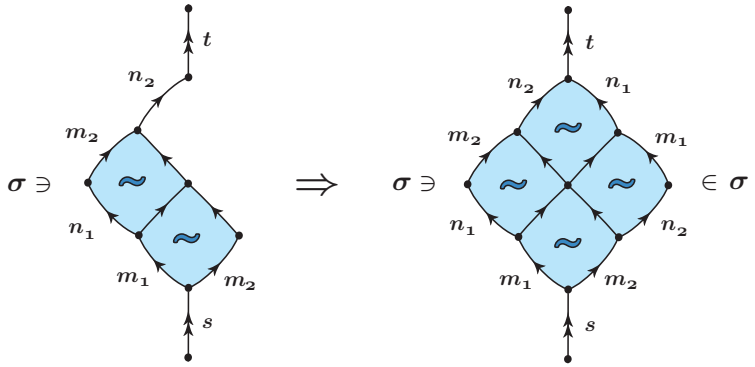
▶ **Definition 6.** A sequential strategy is called backward innocent when every play $s \in \sigma$, every path t, every pair of Opponent moves m_1, m_2, and every pair of Player moves n_1, n_2 which satisfy the properties:

$$s \cdot m_1 \cdot n_1 \cdot m_2 \cdot n_2 \cdot t \in \sigma \text{ and } \neg(n_1 \vdash m_2) \text{ and } \neg(m_1 \vdash m_2)$$

satisfy also the properties:

$$\neg(n_1 \vdash n_2) \text{ and } \neg(m_1 \vdash n_2) \text{ and } s \cdot m_2 \cdot n_2 \cdot m_1 \cdot n_1 \cdot t \in \sigma.$$

Backward innocence may be depicted as the following diagrammatic property:



(10)

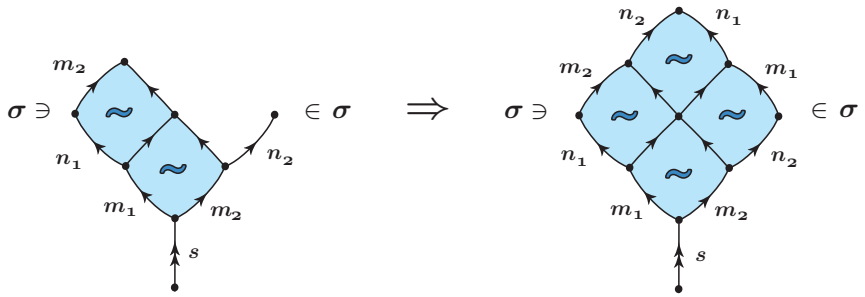
▶ **Definition 7.** A strategy σ is forward innocent when every play $s \in \sigma$, every pair of Opponent moves m_1, m_2, and every pair of Player moves n_1, n_2 satisfying the properties:

$$s \cdot m_1 \cdot n_1 \in \sigma \text{ and } s_1 \cdot m_2 \cdot n_2 \in \sigma \text{ and } m_1 \neq m_2$$

satisfy also the properties:

$$n_1 \neq n_2 \text{ and } s \cdot m_1 \cdot n_1 \cdot m_2 \cdot n_2 \in \sigma.$$

Forward innocence may be depicted as the following diagrammatic property:



(11)

▶ **Definition 8.** A strategy is called innocent when it is backward and forward innocent.

One important property of innocence established in [20] is that every innocent strategy is positional, in the sense that it is entirely described by its set of halting positions. By halting position of the innocent strategy σ, we mean a position x of the dialogue game such that there exists a play $s \in \sigma$ with target position x.

▶ **Definition 9.** A sequential strategy σ of a dialogue game A is called total when for every play $s \in \sigma$, and for every Opponent move m such that $s \cdot m$ is a play of the dialogue game A, there exists a Player move n such that $s \cdot m \cdot n \in \sigma$.

Note that the notion of total strategy considered here is weaker than in [23] since we do not require that every maximal position (x, y) of the strategy in $\mathbf{Dialogue}(A, B)$ reaches two maximal positions x and y of the dialogue games A and B. A typical illustration is provided by the strategy $e_A : \neg A \to 1$ which contains exactly the empty play on the unique initial position $*$. Its unique maximal position is the pair $(*, *)$ of initial positions in $\neg A$ and 1 although the position $*$ is not maximal in the dialogue game $\neg A$.

One main application of tensorial logic is the following characterization of the category **Dialogue** of dialogue games and total innocent strategies. The proof of the proposition may be done directly in a proof-theoretic style or by extending to finite sums the combinatorial presentation of innocence in [23].

▶ **Proposition 10.** *The category* **Dialogue** *is the free affine dialogue category with finite sums (and tensor product distributing over these finite sums) generated by the empty category.*

Although Proposition 10 looks like a purely conceptual statement, it provides a very useful tool in order to relate game semantics to various models of tensorial or linear logic. In particular, it states that there exists a canonical (and functorial) interpretation of dialogue games and total innocent strategies

$$[-] \quad : \quad \mathbf{Dialogue} \longrightarrow \mathcal{D} \tag{12}$$

in any affine dialogue category $\mathcal{D}$ with finite sums, where the tensor distributes over finite sums. Moreover, by its mere construction, the functor $A \mapsto [A]$ preserves the monoidal structure, the finite sums, the negation and the weakening map $e_A : \neg A \to 1$ up to coherent isomorphism.

4 A Kripke translation of tensorial logic into linear logic + necessity

One preliminary insight of the paper is that the construction $A \mapsto \mathbf{Pos}(A)$ which transports a dialogue game to its set of positions may be understood as an instance of the semantic functor (12). After all, a simple example of such an affine dialogue category $\mathcal{D}$ is provided by the category **Rel** of sets and relations with weakening $e_A : A^\perp \to 1$ defined as the empty relation. As in the case of any such affine $*$-autonomous category, the tensorial negation $\neg A$ is interpreted as the involutive negation:

$$[\neg A] \quad = \quad [A]^\perp. \tag{13}$$

In the specific case of **Rel**, this implies that $[A]$ coincides with the set of *maximal positions* of the dialogue game A. This preliminary observation leads to the idea of replacing the inappropriate interpretation (13) of tensorial negation by the following one

$$[\neg A] \quad = \quad \Box\, [A]^\perp \tag{14}$$

where the modality $\Box$ would be typically defined as

$$\Box A \quad = \quad A \,\&\, \perp \tag{15}$$

in order to add a point to the relational interpretation of A. The idea is tempting, but there remains to justify it from a logical and algebraic point of view. In order to understand where we stand, it is worth recalling that tensorial logic enjoys the same position with respect to linear logic as intuitionistic logic does with respect to classical logic. From that point of view, it makes sense to translate tensorial logic into linear logic in just the same way as one translates intuitionistic logic into classical logic. A typical solution is to adapt the well-known Kripke translation of intuitionistic logic in the modal logic S4 consisting of classical logic extended with a necessity modality $\Box$. Recall that the Kripke translation is based on the following interpretation of the intuitionistic implication:

$$[\, A \Rightarrow_{int} B \,] \quad = \quad \Box\, (\, [A]^\perp \,\vee\, [B] \,) \tag{16}$$

Note that one recovers an intuitionistic variant of (14) by taking the formula B equal to false in (16). Consequently, our next purpose will be to design a linear logic extended with a necessity modality $\Box$ in such a way as to make our tensorial version of the Kripke translation (14) work. We could proceed syntactically and define a sequent calculus for the logic, which we will call linear S4 for simplicity. Since this is essentially equivalent, we prefer to remain at an algebraic level, and to define a categorical semantics of linear S4. To that purpose, we introduce the following notion:

▶ **Definition 11.** A necessity modality on a symmetric monoidal category $\mathcal{L}$ is defined as a symmetric monoidal comonad $\Box$. By this, one means a comonad $\Box$ thus equipped with two natural families of morphisms

$$\varepsilon_A : A \longrightarrow \Box A \qquad\qquad \delta_A : \Box A \longrightarrow \Box\Box A$$

making the expected associativity and unit diagrams commute, together with a natural family of coercions

$$m_{A,B} : \Box A \otimes \Box B \to \Box(A \otimes B) \qquad\qquad m_1 : 1 \longrightarrow \Box 1$$

making $\Box$ a lax symmetric monoidal functor, and compatible with the structure of the comonad.

It is well-known and not difficult to check that in that case, the comonad factors as

$$\Box \quad = \quad \mathit{Forget} \circ \mathit{Necessary}$$

where *Forget* and *Necessary* define a symmetric monoidal adjunction

$$(\mathcal{M}, \otimes, 1) \underset{\mathit{Necessary}}{\overset{\mathit{Forget}}{\rightleftarrows}} (\mathcal{L}, \otimes, 1) \qquad \bot \tag{17}$$

and the category $\mathcal{M}$ is typically defined as the category of Eilenberg-Moore coalgebras of the comonad. The adjunction is called a *symmetric monoidal adjunction* because it is the same thing as a formal adjunction in the 2-category of symmetric monoidal categories and symmetric monoidal functors in the lax sense, see [21] for details. The notion of symmetric monoidal adjunction is important in tensorial logic because it enables one to transport the tensorial negations of the category $\mathcal{L}$ into the category $\mathcal{M}$. Suppose for instance that the category $\mathcal{L}$ is $*$-autonomous. In this case, the category $\mathcal{M}$ inherits a tensorial negation

$$\neg A \quad = \quad \mathit{Necessary}\,(\,(\mathit{Forget}\, A)^{\,\bot}) \tag{18}$$

from the linear negation in the category $\mathcal{L}$. Hence, $\mathcal{M}$ defines a dialogue category. This establishes that every $*$-autonomous category $\mathcal{L}$ equipped with a necessity modality $\Box$ induces a model of tensorial logic, simply defined as its dialogue category $\mathcal{M}$ of Eilenberg-Moore coalgebras. Note that the category $\mathcal{M}$ has finite sums as soon as the underlying category $\mathcal{L}$ has finite sums. One shows moreover that the dialogue category $\mathcal{M}$ is affine when the necessity modality $\Box$ is affine in the following sense.

▶ **Definition 12.** An affine necessity modality $\Box$ on a symmetric monoidal category $\mathcal{L}$ is a necessity modality equipped with a family of coalgebra maps $e_A : \Box A \longrightarrow 1$ natural in A.

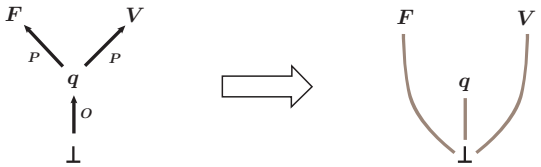
The notion of affine necessity modality is quite familiar in models of linear logic. In particular, the exponential modality ! of a linear category $\mathcal{L}$ defines an affine necessity modality, see [21] for details. The ongoing discussion establishes that

▶ **Proposition 13.** *Every $*$-autonomous category with finite sums equipped with an affine necessity modality $\Box$ induces a functor*

$$(\mathbf{Dialogue}, \otimes, 1) \quad \longrightarrow \quad (\mathcal{M}, \otimes, 1)$$

where $\mathcal{M}$ denotes the category of Eilenberg-Moore coalgebras of the comonad $\Box$.

It is not very difficult to check that equation (15) defines an affine necessity modality $\Box$ in the category **Rel**, with weakening $e_A : A \,\&\, 1 \to 1$ defined as the projection on the second component. Much more interesting is the fact that the same equation (15) defines an affine necessity modality in the category **Coh** of coherence spaces. The resulting semantic functor $A \mapsto [A]$ enables us to identity the set of positions $\mathbf{Pos}\,(A)$ as the web of the coherence space $[A]$. By way of illustration, the dialogue game $\mathbb{B} = \neg\neg(1 \oplus 1)$ is transported to the following coherence space:



(19)

where the initial position $\bot$ is coherent with the three other positions $q = \{\bot, q\}$, $F = \{\bot, q, F\}$ and $V = \{\bot, q, V\}$ which are themselves pairwise incoherent. One main benefit of our logical approach to game semantics is that every innocent strategy σ playing on the dialogue game A is shown to be interpreted as a *clique* of halting positions $[\sigma]$ in the coherence space of positions $[A]$. This fact that the set of halting positions of an innocent strategy σ defines a clique in $[A]$ is reasonable, but it does not seem so easy to establish by a direct and purely combinatorial proof.

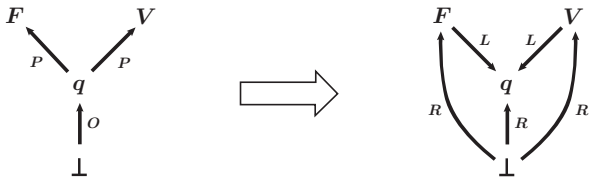
5 Dialogue categories and coherent strategies

Our next task is to apply our general method in order to interpret the positions of a dialogue game A as the web of a bistructure. The bistructure model of linear logic was introduced by Curien, Plotkin and Winskel about ten years ago [6] and it remains today one of the most clever and enigmatic models ever designed for linear logic. Its main achievement is to integrate the causality principles underlying Berry's notion of stable function — later revisited by Girard in his coherence space model of linear logic — to the information structure underlying the notion of continuous function between Scott domains [24]. The definition of bistructure is recalled in the appendix. In order to achieve our task on dialogue games, we introduce an affine necessity modality on bistructures:

$$\Box \quad : \quad \mathbf{Bistr} \quad \longrightarrow \quad \mathbf{Bistr}$$

simply defined by extending a given bistructure E with one element $*$ in such a way that $* \leqslant^R e$ for all $e \in E$. Note that by definition of a bistructure, this implies that $* \subset e$ for all $e \in E$. We then apply Proposition 13 in order to interpret the set of positions of a dialogue game A as the web of a bistructure $[A]$, and an innocent strategy $\sigma : A \to B$ as a configuration $[\sigma]$ defining a morphism $[A] \to [B]$ in the category $\mathcal{M}$ of coalgebras of the

comonad $\square$. Typically, the bistructure associated to the dialogue game $\mathbb{B} = \neg\neg(1 \oplus 1)$ refines the coherence space (19) with the extra $\leqslant^L$ and $\leqslant^R$ ordering information:

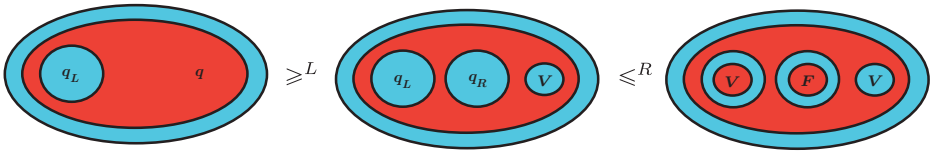


The diagram should be read as follows: it states that $F, V \leqslant^L q$ and that $\bot \leqslant^R q, F, V$. An easy induction on the formulas of tensorial logic enables one to characterize the two orders $\leqslant^L$ and $\leqslant^R$ on the set of positions of a dialogue game A.

▶ **Proposition 14.** *For every dialogue game A, two positions $x, y \in \mathbf{Pos}(A)$ satisfy*

- *$x \leqslant^L y$ precisely when $y \subseteq x$ and the position y may be obtained from x by removing subtrees with Player moves as roots,*
- *$x \leqslant^R y$ precisely when $x \subseteq y$ and the position x may be obtained from y by removing subtrees with Opponent moves as roots.*

Proposition 14 is important because it provides an elementary and purely combinatorial account of the two orders $\leqslant^L$ and $\leqslant^R$. A typical illustration of these orderings is provided by the three positions of the dialogue game $\mathbb{B} \otimes \mathbb{B} \multimap \mathbb{B}$ considered earlier:



Unfortunately, the coherence relation $\frown_{[A]}$ between positions of a dialogue game A appears more difficult to formulate in a similarly simple combinatorial way. We will not try to do that here. Rather, we establish the following useful property.

▶ **Proposition 15.** *The set-theoretic intersection $x \cap y$ of two positions $x, y \in \mathbf{Pos}(A)$ included in a position $z \in \mathbf{Pos}(A)$ is itself a position of the dialogue game A. Moreover, the two positions x and y are coherent in the bistructure $[A]$ of positions in the sense that $x \frown_{[A]} y$ whenever they satisfy the inequalities:*

$$x \cap y \leqslant^R x \qquad\qquad x \cap y \leqslant^R y.$$

Dually, the two positions x and y are incoherent in the bistructure of positions in the sense that $x \asymp_{[A]} y$ whenever they satisfy the inequalities:

$$x \leqslant^L x \cap y \qquad\qquad y \leqslant^L x \cap y.$$

Proof. See the appendix. ◀

An interesting and non trivial consequence of Proposition 13 is the following statement:

▶ **Proposition 16.** *The set of halting positions $[\sigma]$ of a total innocent strategy σ playing on a dialogue game A defines a configuration of the bistructure of positions $[A]$.*

Once this result established, a natural question is to understand more generally the behaviour of *any* configuration σ of the bistructure $[A]$ of positions associated to a dialogue game A. We know already that every such configuration σ is secured, and thus has a *backward*

dynamics which recovers from every position $x \in \sigma$ the causal cascade which produced it from the initial position of the dialogue game $*$. Indeed, in the case of a bistructure of positions $[A]$, securedness means that for every position x in the configuration σ and for every position y obtained by removing some Opponent information from x, there exists a position $z \in \sigma$ obtained by removing some Player information from y. This interpretation of securedness follows from Proposition 14. The somewhat surprising observation is that *every* configuration σ of a bistructure of positions $[A]$ is also equipped with a *forward dynamics* and thus behaves like a (usually not sequential) strategy. This last claim is formulated as the following result:

▶ **Proposition 17.** *For every configuration σ of the bistructure $[A]$ of positions of a dialogue game A, and for every pair of positions $x \in \sigma$ and $z \in \sigma$, such that $x \subseteq z$, and for every Opponent transition $m : x \to y$ to a position $y \in \mathbf{Pos}(A)$ such that $y \subseteq z$, there exists a (possibly empty) path of Player transitions $t : y \to y_1 \to \cdots \to y_n \to y'$ such that $y' \in \sigma$ and $y' \subseteq z$. The position y' is moreover unique.*

Proof. See the appendix. ◀

The result of Proposition 17 justifies to introduce the following definition.

▶ **Definition 18.** A coherent strategy on a dialogue game A is defined as a configuration on the bistructure of positions $[A]$. Accordingly, the category **Coherent** is defined as the category with dialogue games as objects and with configurations σ of the bistructure $[A] \multimap [B]$ making the diagram below commute

$$\begin{array}{ccc} [A] & \xrightarrow{\ \sigma\ } & [B] \\ {\scriptstyle d_A}\downarrow & & \downarrow{\scriptstyle d_B} \\ \square[A] & \xrightarrow{\ \square\sigma\ } & \square[B] \end{array}$$

where d_A and d_B are the coalgebra structures wrt. the comonad $\square$ of the bistructures of positions $[A]$ and $[B]$.

as morphisms. By construction, the category **Coherent** is an affine dialogue category with finite sums, and its tensor product distributes over these finite sums. Moreover, there is a functor of dialogue category

$$\mathbf{Dialogue} \xrightarrow{\ \textit{embedding}\ } \mathbf{Coherent}$$

and the category **Coherent** embeds fully and faithfully as a dialogue category in the dialogue category $\mathcal{M}$ of coalgebras of the comonad $\square$.

6 Sequential algorithms as stable extensional functions

The connection between dialogue games and bistructures provided by the functor $[-]$ only works at this stage for the linear fragment of tensorial logic. In particular, it does not include the quantitative exponential modality of dialogue games and innocent strategies. However, we explain that this connection is sufficient in order to interpret the qualitative exponential modality ! of simple games. The connection is provided by the following observation:

▶ **Proposition 19.** *For every simple game A, there exists an isomorphism*

$$\lambda_A \quad : \quad !\,[A] \quad \to \quad \square\,[\mathit{shriek}(A)] \tag{20}$$

in the category of bistructures, where ! denotes the qualitative exponential modality on bistructures introduced by Curien, Plotkin and Winskel.

Proof. See the appendix. ◀

Just as announced in the introduction, using this result, one constructs a functor

$$\mathit{shriek} \quad : \quad \mathbf{Simple} \quad \longrightarrow \quad \mathbf{Coherent}$$

making the diagram (7) commute. The functor *shriek* is constructed in such a way that the composite functor (8) coincides with the functor

$$\mathbf{Simple} \xrightarrow{[-]} \mathbf{Coherent} \xrightarrow{\mathit{forgetful}} \mathbf{Bistr} \xrightarrow{!} \mathbf{Bistr}$$

One observes moreover that the bistructure $[A]$ of positions of a simple game A is a B-bistructure in the sense of Curien, Plotkin and Winskel, see [6]. From this follows that its set of configurations $\Gamma(A)$ equipped with the stable order $\sqsubseteq^R$ and the extensional order $\sqsubseteq$ defines a bidomain in the sense of Berry [2]. From all this, one deduces that

▶ **Proposition 20.** *There exists a functor*

$$\Gamma \quad : \quad \mathit{Kleisli}(\mathbf{Simple}, !) \quad \longrightarrow \quad \mathit{Kleisli}(\mathbf{Bistr}, !)$$

between the co-Kleisli categories induced by the Curien-Lamarche modality ! *on simple games and the Curien-Plotkin-Winskel modality* ! *on bistructures. The definition of the functor* Γ *is based on the fact that every sequential algorithm*

$$\sigma \quad : \quad A \quad \Rightarrow \quad B \tag{21}$$

may be alternatively seen as a sequential strategy

$$\sigma \quad : \quad !\,A \quad \longrightarrow \quad B$$

in the category **Simple** *of simple games, which may be itself seen as an innocent strategy*

$$\varphi(\sigma) \quad : \quad \mathit{shriek}(A) \quad \longrightarrow \quad B$$

in the category **Dialogue** *of dialogue games. By definition, the functor* Γ *transports the sequential strategy (21) to the composite morphism*

$$!\,[A] \xrightarrow{\lambda_A} \square\,[\,\mathit{shriek}(A)\,] \xrightarrow{\mathit{counit}} [\,\mathit{shriek}(A)\,] \xrightarrow{[\varphi(\sigma)]} [B]$$

in the category of bistructures, which itself corresponds to the stable and extensional function

$$\Gamma(\sigma) \quad : \quad \Gamma(A) \quad \Rightarrow \quad \Gamma(B)$$

between the bidomains of configurations $\Gamma(A)$ *and* $\Gamma(B)$ *induced by the bistructures of positions* $[A]$ *and* $[B]$ *of the simple games* A *and* B*.*

7 Conclusion

This work on coherent strategies between dialogue games is still at a pretty preliminary stage but we find useful to share the general methodology of our approach based on tensorial logic as well as the somewhat unexpected discovery that the category of bistructures contains a subcategory of dialogue games and coherent strategies. Our final result that every sequential algorithm between two simple games A and B induces a stable and extensional function $\Gamma(A) \to \Gamma(B)$ between the associated bidomains of configurations is related to the extensional description of sequential algorithms investigated by Curien, Laird and Streicher [14, 7, 18]. In

particular, Streicher made the important observation that the set of *sequential strategies with errors* on a simple game defines a bidomain in the sense of Berry. In that line of research, it should be possible to refine our Proposition 20 in order to characterize the sequential algorithms between A and B as a specific class of stable and extensional functions, but we prefer to leave that aspect for future work. Note that such a characterization has already been given by Calderon and McCusker [4] for sequential strategies between simple games. Another question of interest would be to understand the relationship between the present work on dialogue games and bistructures with the tight connection between sequential games and Ehrhard's hypercoherence spaces [8, 22].

References

1. S. Abramsky, P.-A. Melliès. Concurrent games and full completeness. In *Proceedings of the Fourteenth Annual IEEE Symposium on Logic in Computer Science* (LICS '99), IEEE Computer Society Press, 1999.
2. G. Berry. Modèles Complètement Adéquats et Stables des Lambda-calculs Typés. Thèse de Doctorat d'Etat, Université Paris VII (1979).
3. G. Berry and P.-L. Curien. Sequential algorithms on concrete data structures. In *Theoretical Computer Science*, 20:265-321, 1982.
4. A. Calderon and G. McCusker. Understanding Game Semantics Through Coherence Spaces. Electr. Notes Theor. Comput. Sci. 265: 231-244 (2010)
5. P.-L. Curien. On the symmetry of sequentiality. In *Proceedings of Mathematical Foundations of Programming Semantics*, MFPS'93, LNCS 802, Springer Verlag, 1993.
6. P.-L. Curien, G. Plotkin and G. Winskel. Bistructures, Bidomains and Linear Logic. In Milner Festschrift, MIT Press (2000).
7. P.-L. Curien. Sequential algorithms as bistable maps. From Semantics to Computer Science. Essays in Honour of Gilles Kahn. Cambridge University Press, 2009.
8. T. Ehrhard. Parallel and serial hypercoherences. Theoretical Computer Science, 2000.
9. J.-Y. Girard. *Normal functors, power series and lambda-calculus.* Journal of Pure and Applied Logic, 1986.
10. J.-Y. Girard. A new constructive logic: Classical logic, Mathematical Structures in Computer Science 1 (3) (1991) 255–296.
11. R. Harmer, M. Hyland and P-A. Melliès. *Categorical combinatorics for innocent strategies.* Proceedings of **LiCS'07**, the 22nd Annual IEEE Symposium on Logic in Computer Science. Wroclaw, 2007.
12. M. Hyland and L. Ong, On full abstraction for PCF: I, II and III, Information and Computation 163 (2) (2000) 285–408.
13. M. Hyland and A. Schalk. Games on Graphs and Sequentially Realizable Functionals. Proceedings of LICS 2002: 257-264.
14. J. Laird. Bistability: A sequential domain theory. Logical Methods in Computer Science, volume 3, issue 2, 2007.
15. F. Lamarche. Sequentiality, games and linear logic. *Manuscript*, 1992.
16. O. Laurent. Polarized Games. Annals of Pure and Applied Logic, number 1–3, volume 130, 79–123, 2004.
17. O. Laurent. Syntax vs. Semantics: a polarized approach. Theoretical Computer Science, volume 343, number 1–2, 177–206, 2005.
18. T. Loew and T. Streicher. Universality results for models in locally boolean domains. Proceedings of CSL 2006.
19. P.-A. Melliès. Comparing hierarchies of types in models of linear logic Information and Computation, Volume 189, Issue 2, Pages 202-234, March 2004.

20 P.-A. Melliès. *The true concurrency of innocence.* Special Issue Selected papers of CONCUR 2004 of Theoretical Computer Science Volume 358, Issues 2-3, pages 200-228, 2006.

21 P.-A. Melliès. *Categorical semantics of linear logic.* Survey in Interactive models of computation and program behaviour. P.-L. Curien, H. Herbelin, J.-L. Krivine, P.-A. Melliès Panoramas et synthèses 27 (2009).

22 P.-A. Melliès. Sequential algorithms and strongly stable functions. Special Issue "Game Theory Meets Theoretical Computer Science" of Theoretical Computer Science, Volume 343, Issue 1, Pages 237-281, 2005.

23 P.-A. Melliès. *Game semantics in string diagrams.* Proceedings of the Annual ACM/IEEE Symposium on Logic in Computer Science 2012.

24 D. S. Scott. *Data types as lattices.* Proceedings of the International Summer Institute and Logic Colloquium, Kiel, in Lecture Notes in Mathematics (Springer-Verlag) 499: 579?651. 1975.

A Appendix: a short account of bistructures

We recall below the notion of bistructure as well as the main definitions of the theory.

▶ **Definition 21.** A (countable) bistructure is a quadruple $(E, \leqslant^L, \leqslant^R, \frown)$ where E is a countable set called the *web* of the bistructure, $\leqslant^L$, $\leqslant^R$ are partial orders on E and $\frown$ is a binary reflexive, symmetric relation on E such that:

1. defining $\leqslant$ as the transitive closure of $(\leqslant^L \cup \leqslant^R)$, we have the following factorisation property:
$$e \leqslant e' \quad \Rightarrow \quad \exists e'' \in E, \quad e \leqslant^L e'' \leqslant^R e'$$
2. defining $\preceq$ as the transitive closure of $(\geqslant^L \cup \leqslant^R)$, we have the following properties:
   a. $\preceq$ is finitary, i.e., $\{e' | e' \preceq e\}$ is finite, for all $e \in E$,
   b. $\preceq$ is a partial order,
   3. (a) $\downarrow^L \subseteq \asymp$ and (b) $\uparrow^R \subseteq \frown$.

Here, the two compatibility relations are defined by:

$$\begin{array}{rcl} e \downarrow^L e' & \iff & \exists\, e'' \in E, \quad e'' \leqslant^L e \text{ and } e'' \leqslant^L e' \\ e \uparrow^R e' & \iff & \exists\, e'' \in E, \quad e \leqslant^R e'' \text{ and } e' \leqslant^R e''. \end{array}$$

and we write $\asymp$ for the reflexive closure of the complementary of $\frown$. We then recall below the definition of configuration.

▶ **Definition 22.** A configuration of a bistructure $(E, \leqslant^L, \leqslant^R, \frown)$ is a subset $\sigma \subseteq E$ which is:

- consistent: $\forall e, e' \in \sigma$, $e \frown e'$, and
- secured: $\forall e \in \sigma$, $\forall e' \leqslant^R e$, $\exists e'' \in \sigma$, $e' \leqslant^L e''$.

We write $\Gamma(E)$ for the set of configurations of a bistructure E, and $\Gamma_{fin}(E)$ for the subset of *finite* configurations. At this point, we recall how Curien, Plokin and Winskel [6] define a stable order $\sqsubseteq^R$ and an extensional order $\sqsubseteq$ on the configurations $\sigma, \tau \in \Gamma(E)$ of a given bistructure E.

▶ **Definition 23.** Let E be a bistructure. The stable order $\sqsubseteq^R$ and the extensional order $\sqsubseteq$ on configurations are defined as:

- $\sqsubseteq^R$ is set-theoretic inclusion,
- $\sigma \sqsubseteq \tau \iff \forall e \in \sigma, \exists e' \in \tau, e \leqslant^L e'$.

Note that it follows from the reflexivity of $\leqslant^L$ that $\sqsubseteq^R$ is included in $\sqsubseteq$. A third relation $\sqsubseteq^L$ is then defined as follows:

$$\sigma \sqsubseteq^L \tau \iff \sigma \sqsubseteq \tau \text{ and } (\forall \upsilon \in \Gamma(E), (\sigma \sqsubseteq \upsilon \text{ and } \upsilon \sqsubseteq^R \tau) \Rightarrow \tau = \upsilon)$$

Thus, $\sigma \sqsubseteq^L \tau$ means that τ is a $\sqsubseteq^R$-minimal configuration such that $\sigma \sqsubseteq \tau$. We also write $\sigma \uparrow^R \tau$ when there exists a configuration $\upsilon \in \Gamma(E)$ such that $\sigma \sqsubseteq^R \upsilon$ and $\tau \sqsubseteq^R \upsilon$.

We briefly recall from [6] that the category **Bistr** has bistructures as objects and configurations of $A \multimap B$ as morphisms $\sigma : A \to B$. The category is $*$-autonomous and has finite sums provided by the following definitions.

- the negation $E^\perp$ of a bistructure $(E, \leqslant^L, \leqslant^R, \frown)$ is defined as $(E, \geqslant^R, \geqslant^L, \asymp)$,
- the sum $E_1 \oplus E_2$ of two bistructures $(E_1, \leqslant^L_1, \leqslant^R_1, \frown_1)$ and $(E_2, \leqslant^L_2, \leqslant^R_2, \frown_2)$ is defined as $(E_1 + E_2, \leqslant^L_1 + \leqslant^L_2, \leqslant^R_1 + \leqslant^R_2, \frown_1 + \frown_2)$,
- the tensor product $E_1 \otimes E_2$ of two bistructures $(E_1, \leqslant^L_1, \leqslant^R_1, \frown_1)$ and $(E, \leqslant^L_2, \leqslant^R_2, \frown_2)$ is defined as $(E_1 \times E_2, \leqslant^L_1 \times \leqslant^L_2, \leqslant^R_1 \times \leqslant^R_2, \frown_1 \times \frown_2)$,
- the bistructure 0 has an empty web, and the bistructure 1 has a singleton web,
- the exponential $!E$ of a bistructure $(E, \leqslant^L, \leqslant^R, \frown)$ is defined as $(\Gamma_{fin}(E), \sqsubseteq^L, \sqsubseteq^R, \uparrow^R)$ where these structures are introduced in Definition 23.

B Appendix: Proof of Proposition 15

Proof. The proof is established by an easy induction on the formula defining the dialogue game A. The property is obvious in the case of the two unit games 0 and 1. We treat in turn the inductive case of the game $A \otimes B$, of the game $A \oplus B$ and of the game $\neg A$.

First inductive case: the dialogue game $A \otimes B$

By definition of the dialogue game $A \otimes B$, the two positions x and y are of the form $x = x_A \otimes x_B$ and $y = y_A \otimes y_B$. Suppose that the two positions x and y are included in a position $z = z_A \otimes z_B$. In that case, the positions x_A and y_A obtained by projecting x and y on the component A are included in the position z_A. By induction hypothesis, it follows that $x_A \cap y_A$ is a position of the dialogue game A. One establishes symmetrically that the intersection $x_B \cap y_B$ is a position of the dialogue game B. The set-theoretic intersection $x \cap y$ is equal to $(x_A \cap y_A) \otimes (x_B \cap y_B)$ which is a position of the dialogue game $A \otimes B$. We conclude that $x \cap y$ is a position of the game $A \otimes B$.

Now, suppose that two positions $x = x_A \otimes x_B$ and $y = y_A \otimes y_B$ are included in a position $z = z_A \otimes z_B$ and moreover that $x \cap y \leqslant^R x$ and $x \cap y \leqslant^R y$. In that case, the two positions x_A and y_A are included in the position z_A. Moreover, $x_A \cap y_A \leqslant^R x_A$ and $x_A \cap y_A \leqslant^R y_A$ since $x \cap y = (x_A \cap y_A) \otimes (x_B \cap y_B)$ and the order $\leqslant^R$ is defined in the bistructure $[A \otimes B] = [A] \otimes [B]$ as the componentwise product of $\leqslant^R$ in the bistructures $[A]$ and $[B]$. By induction hypothesis applied to the game A, it follows that $x_A \frown_{[A]} y_A$. One establishes symmetrically that $x_B \frown_{[B]} y_B$. From this, we conclude by definition of coherence in the bistructure $[A \otimes B] = [A] \otimes [B]$ that $x_A \otimes x_B \frown_{[A \otimes B]} y_A \otimes y_B$ and thus, that $x \frown_{[A \otimes B]} y$.

There remains to establish the last statement of the proposition. Suppose that two positions $x = x_A \otimes x_B$ and $y = y_A \otimes y_B$ are included in a position $z = z_A \otimes z_B$ and moreover that $x \leqslant^L x \cap y$ and $y \leqslant^L x \cap y$. The proof that $x \asymp_{[A \otimes B]} y$ is done in the same way as in the previous paragraph. In that case, the two positions x_A and y_A are included in the position z_A.

Moreover $x_A \leqslant^L x_A \cap y_A$ and $y_A \leqslant^L x_A \cap y_A$ because $x \cap y = (x_A \cap y_A) \otimes (x_B \cap y_B)$ and the order $\leqslant^L$ is defined in the bistructure $[A \otimes B] = [A] \otimes [B]$ as a componentwise product of $\leqslant^L$ in the bistructures $[A]$ and $[B]$. By induction hypothesis applied to the game A, it follows that $x_A \asymp_{[A]} y_A$. One establishes symmetrically that $x_B \asymp_{[B]} y_B$. From this, we conclude by definition of coherence in the bistructure $[A \otimes B] = [A] \otimes [B]$ that $x_A \otimes x_B \asymp_{[A \otimes B]} y_A \otimes y_B$ and thus, that $x \asymp_{[A \otimes B]} y$.

Second inductive case: the dialogue game $A \oplus B$

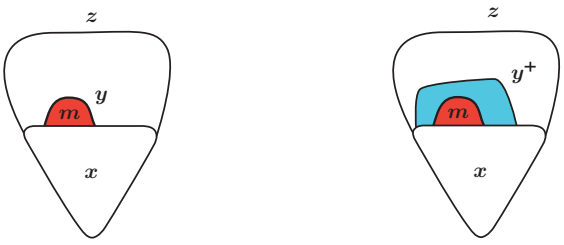
By definition of the dialogue game $A \oplus B$, the fact that the two positions x and y are included in a position z implies that the three positions x, y, z lie in the same component A or B of the game $A \oplus B$. We may suppose without loss of generality that the three positions x, y, z are positions of the component A. From this, it follows easily by induction hypothesis applied to the dialogue game A that the intersection $x \cap y$ is a position in the game A and thus in the game $A \oplus B$. The two remaining statements of the proposition are just as easy to establish by induction.

Third inductive case: the dialogue game $\neg A$

By definition of the dialogue game $\neg A$, we are in one of the two possible situations: either the three positions x, y, z are in the component A or one of the two positions x, y is the initial position $*$ itself. The first case is easily treated by induction hypothesis on A. In the second case, one of the two positions x and y is equal to the initial position $*$. For the sake of discussion, we may suppose without loss of generality that the position x is equal to the initial position $*$. The intersection $x \cap y = *$ is a position of the dialogue game $\neg A$. Moreover, it follows from the definition of the bistructure $[\neg A]$ of positions of the game $\neg A$ as the bistructure $\square([A]^\perp)$ that the two positions x and y are coherent in the bistructure $[\neg A]$ since $x = *$ is the position added to the bistructure $[A]^\perp$ by the necessity modality. Note that, by definition of $\square([A]^\perp)$, the position $x = *$ also satisfies $* \leqslant^R y$. Moreover, if $y \leqslant^L *$ then $y = *$, and thus $x \asymp_{[\neg A]} y$. This concludes the proof by induction of Proposition 15. ◀

C Appendix: Proof of Proposition 17

Proof. The proof is based on the very specific properties of the bistructure $[A]$ associated to a dialogue game A, and in particular on the two Propositions 14 and 15. Given the position $x \in \sigma$ and the Opponent transition $m : x \to y$ such that $y \subseteq z$ for $z \in \sigma$, let y^+ denote the smallest position (with respect to inclusion) containing the position y as a subset: $y \subseteq y^+$, and satisfying $y^+ \leqslant^R z$. This position y^+ exists and is defined according to Proposition 14 by removing from the position z all the subtrees with an Opponent root n not element of the position y. It is important to observe that the position y^+ only contains Player moves besides the moves already in the position y. The situation may be depicted as follows. Note that the Opponent move m in the position y is depicted in red, and the layer of Player moves between y and y^+ is depicted in blue.



By the securedness property of the configuration σ, there exists a position $y' \in \sigma$ such that $y^+ \leqslant^L y'$. By Proposition 14, the position y' is obtained from the position y^+ by removing a series of subtrees with Player roots. From this follows in particular that the position $x \cap y'$ is obtained from the position $x \cap y^+ = x$ by removing a series of subtrees with Player roots. Hence, $x \leqslant^L x \cap y'$.

First claim: the move m appears in the position y'

We claim that the move m appears in the position y'. Suppose that this is no the case, and that the move m does not appear in the position y'. In that case, a simple argument shows that the position $x \cap y'$ is obtained from the position y' by removing only subtrees with roots in the position y^+ but not in the position y. An important point is that the roots of these subtrees removed from y' in order to obtain $x \cap y'$ are all Player moves. By Proposition 14, it thus follows that $y' \leqslant^L x \cap y'$. Recall moreover that the two positions x and y' are included in the position z and that $x \leqslant^L x \cap y'$. All this put together establishes thanks to Proposition 15 that the positions x and y' are incoherent in the bistructure $[A]$. Since the two positions x and y' are also coherent as elements of the clique σ, they are necessarily equal. This contradicts the definition of y and of y^+ and more specifically the fact that $y^+ \leqslant^L y'$. The point is that the position $x = y'$ can be obtained from the position y (and thus from the position y^+) only at the condition of removing the subtree with Opponent root m. From this, we conclude that the move m necessarily appears in the position y'.

Second claim: the position x is a subset of the position y'

Now, we want to prove that $x \subseteq y'$. Suppose that this is not the case, and let the position x^+ be obtained by removing the subtree with Opponent root m from the position $y' \in \sigma$. By construction, one has $x^+ \leqslant^R y'$. One also has $x \cap x^+ = x \cap y'$ since m is not an element of x. From this follows that $x \leqslant^L x \cap x^+$ since we already know that $x \leqslant^L x \cap y'$. By securedness of σ, there exists a position $x' \in \sigma$ such that $x^+ \leqslant^L x'$. By Proposition 14, the position x' is obtained from the position x^+ by removing subtrees with Player roots. From this follows that the position $x \cap x'$ is obtained from the position $x \cap x^+$ by removing subtrees with Player roots. Hence, $x \cap x^+ \leqslant^L x \cap x'$ by Proposition 14 again. From this and $x \leqslant^L x \cap x^+$, we conclude by transitivity that $x \leqslant^L x \cap x'$. At this point, a simple argument shows that the position $x \cap x'$ is obtained from the position x' by removing subtrees whose roots stand among the Player moves in y^+ but not in y. The fact that these moves are all Player moves implies that $x' \leqslant^L x \cap x'$. The two inequalities $x \leqslant^L x \cap x'$ and $x' \leqslant^L x \cap x'$ together with the fact that the positions x and x' are included in the position z implies by Proposition 15 that x and x' are incoherent in the bistructure $[A]$. Since the two positions x and x' are also coherent as elements of the clique σ, they are equal. The equality $x = x'$ establishes that $x \subseteq y'$ since $x' \subseteq x^+ \subseteq y'$ by definition of the position $x' \in \sigma$.

From the two claims just established, we conclude that $y = x \uplus \{m\}$ is a subset of the position y'. By construction, the position y' is at the same time a subset of y^+ which only contains Player moves besides the moves already in the position y. From this, we deduce that the position y' only contains Player moves besides the moves already in position y, and thus that there exists a path of Player transitions from the position y to the position $y' \in \sigma$. ◀

D Appendix: Proof of Proposition 19

Proof. We construct the isomorphism

$$\lambda_A \quad : \quad !\,[A] \quad \to \quad \square\,[\mathit{shriek}(A)]$$

in the category of bistructures, for every simple game A. The first step of the construction is to characterize the configurations σ of the associated bistructure $[A]$ of positions. A preliminary observation is that there is a one-to-one relationship between (1) the positions of the simple game A seen as a dialogue game (2) the sequential plays

$$* \xrightarrow{m_1} x_1 \xrightarrow{m_2} \cdots \xrightarrow{m_k} x_k$$

of the simple game A and (3) the elements of the web of the bistructure $[A]$. Since every position x of the bistructure $[A]$ corresponds to a specific sequential play of the simple game A, every configuration σ is alternatively described by a set of sequential plays (or positions) $x \in \sigma$ of the simple game A. We claim that every configuration σ of the bistructure $[A]$ is closed under even-length prefix in the sense that every sequential play (or position) y which is even-length prefix of a sequential play (or position) $x \in \sigma$ is also an element of the configuration σ. In order to establish our claim, we first observe that by Proposition 14, a position y is an even-length prefix of the position x precisely when $y \leqslant^R x$. Suppose that we are in that case, and that $y \leqslant^R x$. By securedness of σ, we know that there exists a position $z \in \sigma$ such that $y \leqslant^L z$. We would like to prove that $z = y$. Suppose that it is not the case and that z is a strict prefix of y. In that case, $z \in \sigma$ is also a strict prefix of $x \in \sigma$. By Proposition 14, the position z is obtained from the position y by removing a subtree (in that case, a branch) with a Player root. Hence, the position z is also obtained from the position x by removing a subtree with a Player root since y is a prefix of x. From this, we conclude by Proposition 14 that $x \leqslant^L z$. By definition of a bistructure, the two positions x and z are thus incoherent in the bistructure $[A]$. The two positions x and z are also coherent as elements of the configuration σ. From this, we conclude that $x = z$. This contradicts the fact that the position z is a strict prefix of the position y and thus of the position x. From this, we conclude that $z = y$, and thus, that the configuration σ is closed under even-length prefix.

Similarly, we may establish a complementary property of the configuration σ, which states that every strict prefix $y \in \sigma$ of a position $x \in \sigma$ is of even length. The reason is that the two positions x and y of the configuration σ are coherent, whereas the relation $x \leqslant^L y$ (and thus $x \asymp_{[A]} y$) would hold if the position y was of odd length. This second observation leads us to introduce a useful variant of our Definition 5 of sequential strategy on a dialogue game A, see for instance [19].

▶ **Definition 24** (sequential strategy with errors)**.** A sequential strategy σ with errors on a dialogue game A is defined as set of sequential plays which

- has a starting point: σ contains the empty play $*$ for exactly one initial position $*$,
- is closed under even-length prefix, in the sense that for every even-length prefix s of a sequential play t, one has $t \in \sigma \Rightarrow s \in \sigma$,
- has no intermediate errors, in the sense that for every odd-length prefix s of a sequential play t, one has ($s \in \sigma$ and $t \in \sigma$) $\Rightarrow s = t$,
- is deterministic, in the sense that for every even-length sequential play s, $s \cdot m \cdot n_1 \in \sigma$ and $s \cdot m \cdot n_2 \in \sigma$ implies that $n_1 = n_2$,

for all plays s and all moves m, n, n_1, n_2 of the dialogue game.

Note that by definition of a strategy σ with errors, every odd-length position s of the strategy σ is maximal among the positions in σ. Such an odd-length position of the strategy σ is called an error of the strategy. We have just established that every non-empty configuration σ of the bistructure $[A]$ of positions of a simple game A satisfies the three first properties of Definition 5. We prove the fourth property (determinism) below. Suppose given an even-length position x of the configuration σ, alternatively seen as a sequential play:

$$s \quad = \quad * \xrightarrow{m_1} x_1 \xrightarrow{m_2} \cdots \xrightarrow{m_{2k}} x_{2k} = x$$

and suppose that the two sequential plays $y_1 = s \cdot m \cdot n_1$ and $y_2 = s \cdot m \cdot n_2$ are positions in the configuration σ. We claim that $n_1 = n_2$. The proof is very easy, since it simply consists in observing that the two positions y_1 and y_2 are strictly incoherent in the bistructure $[A]$ when the moves n_1 and n_2 are different. Since the positions y_1 and y_2 are elements of the configuration σ, and thus coherent, we conclude that $n_1 = n_2$. This establishes that every non-empty configuration σ of the bistructure $[A]$ defines a sequential strategy with errors of the underlying simple game A. Conversely, it is easy to check that every sequential strategy σ with errors of the simple game A defines a non-empty configuration of the bistructure $[A]$ of configurations. From this we conclude that

Fact: there is a one-to-one relationship between the non-empty configurations of the bistructure $[A]$ and the sequential strategies with errors of the simple game A

At this point, an obvious but important observation is that every sequential strategy σ with errors of the simple game A may be alternatively seen as a non-empty subtree of A which only branches on Opponent moves. This subtree is entirely described by its set $\mathbf{maxpos}(\sigma)$ of maximal positions. Note that the positions in $\mathbf{maxpos}(\sigma)$ may be either of even-length or of odd-length. The sequential strategy σ with errors is then recovered from $\mathbf{maxpos}(\sigma)$ as

$$\sigma \quad = \quad \mathbf{maxpos}(\sigma) \quad \cup \quad \mathbf{even\text{-}length\text{-}prefix}(\mathbf{maxpos}(\sigma))$$

where $\mathbf{even\text{-}length\text{-}prefix}(X)$ denotes the set of even-length prefixes of a position in X. This establishes that there is a one-to-one relationship between the non-empty configurations of $[A]$ and the non-empty subtrees of the simple game A which only branch on Opponent moves. Now, such a non-empty subtree which only branches on Opponent moves in the simple game A is the same thing as a position in the dialogue game $\mathit{shriek}(A)$. From this, we conclude that:

Fact: there is a one-to-one relationship between the non-empty configurations of the bistructure $[A]$ and the positions of the dialogue game $\mathit{shriek}(A)$

We use the notation $\mathbf{config}(x)$ for the non-empty configuration σ of the bistructure $[A]$ associated to the position x in the dialogue game $\mathit{shriek}(A)$. At this point, starting from Proposition 14, it is not difficult to establish that

$$\mathbf{config}(x) \sqsubseteq^R \mathbf{config}(y) \iff x \leqslant^R y$$

because $\mathbf{config}(x) \subseteq \mathbf{config}(y)$ precisely when $x \subseteq y$ and the position x may be obtained from the position y by removing subtrees with Opponent moves as roots ; that

$$\mathbf{config}(x) \sqsubseteq^L \mathbf{config}(y) \iff x \leqslant^L y$$

precisely when $y \subseteq x$ and the position y may be obtained from the position x by removing subtrees with Player moves as roots ; and finally that

$$\mathbf{config}(x) \uparrow^R \mathbf{config}(y) \iff x \frown_{[\mathit{shriek}(A)]} y$$

for every two positions x, y of the dialogue game $\mathit{shriek}(A)$. This establishes that the bistructure $[\mathit{shriek}(A)]$ of positions of the dialogue game $\mathit{shriek}(A)$ is *isomorphic* to the bistructure $![A]$ restricted to its non-empty configurations. As for the empty configuration, one has that

$$\varnothing \sqsubseteq^R \sigma \qquad\qquad \varnothing \frown_{!A} \sigma$$

for every configuration σ of the bistructure $![A]$. This concludes our proof that the bistructure $![A]$ is isomorphic to the bistructure $\square\,[\mathit{shriek}(A)]$ for every simple game A. ◀

Elementary Modal Logics over Transitive Structures*

Jakub Michaliszyn[1,2] and Jan Otop[1,3]

1 **University Of Wrocław**
2 **Imperial College London**
3 **IST Austria**

Abstract

We show that modal logic over universally first-order definable classes of transitive frames is decidable. More precisely, let $\mathcal{K}$ be an arbitrary class of transitive Kripke frames definable by a universal first-order sentence. We show that the global and finite global satisfiability problems of modal logic over $\mathcal{K}$ are decidable in NP, regardless of choice of $\mathcal{K}$. We also show that the local satisfiability and the finite local satisfiability problems of modal logic over $\mathcal{K}$ are decidable in NExpTime.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Modal logic, Transitive frames, Elementary modal logics, Decidability

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.563

1 Introduction

Modal logic was first introduced by philosophers as the study of the deductive behaviour of the expressions 'it is necessary that' and 'it is possible that'. Nowadays, it is widely used in several areas of computer science, including formal verification and artificial intelligence.

Syntactically, modal logic extends propositional logic by two unary operators: $\Diamond$ and $\Box$. The formal semantics is usually given in terms of Kripke structures. Basically, a Kripke structure is a directed graph, called a *frame*, together with a valuation of propositional variables. Vertices of this graph are called *worlds*. For each world truth values of all propositional variables can be defined independently. In this semantics, $\Diamond\varphi$ means *the current world is connected to some world in which* φ *is true*; and $\Box\varphi$, equivalent to $\neg\Diamond\neg\varphi$, means φ *is true in all worlds to which the current world is connected.*

"Classical" modal logic, defined as above, is very simple, and therefore it has limited applications. For that reason, many modifications of modal logic are studied. One way to enrich modal logic is to add more modalities and obtain so called *multimodal logic*. Another popular modification is to add some constraints on the interpretation of operators, e.g. by requiring that the modal operator represents a relation that is reflexive and transitive (S4). Finally, by combining these two techniques, we may obtain multimodal logics with nonuniform modal operators, like Linear Temporal Logic (LTL), Computation Tree Logic (CTL) or Halpern–Shoham logic (HS).

* The first author was generously supported by Polish National Science Center based on the decision number DEC-2011/03/N/ST6/00415. The second author was supported in part by the Austrian Science Fund NFN RiSE (Rigorous Systems Engineering) and by the ERC Advanced Grant QUAREM (Quantitative Reactive Modeling).

Variants of modal logic vary in the complexity and the decidability of the satisfiability problem. While there are some logics, like S5, that are NP-complete, most of them are PSpace-hard. Modal logic itself is PSpace-complete, and so is the temporal logic LTL. Logics CTL and CTL* are even harder, ExpTime-complete and 2-ExpTime-complete, resp. Finally, the Halpern–Shoham logic is a simple example of a temporal logic that is undecidable, even if we consider some unimodal fragments [3, 16].

There are several ways of adding constraints on the interpretation of operators. The syntactic way goes by adding some additional axioms and considering only modal logics that satisfy these axioms. Another way is by restricting the class of the admissible frames. It also can be done in many ways, but one of the simplest is to define the class of frames by a first order logic sentence that uses a single binary relation R, which is interpreted as the *transition relation*. For example, the sentence $\forall xyz.xRy \wedge yRz \Rightarrow xRz$ defines the class of all transitive frames. Modal logic over a class of frames definable by a first order logic sentence is called *an elementary modal logic*.

The main goal of our work is to classify all elementary modal logics with respect to decidability of their satisfiability problems. In [9], it was shown that there is an universal first-order formula such that the global satisfiability problem over the class of frames that satisfy this formula is undecidable. A slight modification of that formula yields an analogous result for the local satisfiability problem. In [12] it was shown that even a very simple formula with three variables without equality leads to undecidability.

Many modal logics used in automatic verification contain operators that are interpreted as transitive relations. For example, Linear Temporal Logic (LTL) contains transitive operators F and G [2]. In epistemic modal logics, the knowledge operators K_i are interpreted as relations that are not only transitive, but also reflexive and symmetric [4]. Another example is the logic of subintervals [16], which is a fragment of Halpern–Shoham logic with a single modality that can be read as "in some subinterval".

Main results. In this paper, an logic is called a *subframe logic* if it is an unimodal logic defined by restricting the class of the admissible frames to a class that is closed under subframes. Fine[5] showed that there are undecidable transitive subframe logics. An open question suggested in [9] is whether there is an undecidable transitive subframe logic that is an elementary modal logic. Due to [5], such a logic would not be finitely axiomatizable. We show that such a logic does not exist.

The Łoś–Tarski preservation theorem (see, e.g., [10]) states that a first order definable class of frames is closed under subframes if and only if it is universally first order definable. Therefore, to answer the question discussed above, we may restrict our attention to universal formulae. We prove the following theorem.

▶ **Theorem 1.** *Let $\mathcal{K}$ be a class of frames defined by a universal first order formula that implies transitivity. The local and the global satisfiability problems for unimodal logic over $\mathcal{K}$ are decidable.*

The finiteness constraint can make the satisfiability problem easier or harder. There are decidable modal logics that are finitely undecidable, and there are undecidable modal logics that are finitely decidable [6, 19]. However, this is not the case here — the finite satisfiability problems are decidable as well.

▶ **Theorem 2.** *Let $\mathcal{K}$ be a class of frames defined by a universal first order formula that implies transitivity. The local and the global finite satisfiability problems for unimodal logic over $\mathcal{K}$ are decidable.*

We focus on the case where a first-order formula, that defines the class of frames, is a parameter of the problem, and the input consists of a modal formula only. However, our results hold even if we allow a first–order formula to be a part of the input.

Related work. Decidability of modal logic over various classes of frames can be shown by employing the so-called standard translation of modal logic to first-order logic [2]. Indeed, the satisfiability of a modal formula φ in $\mathcal{K}_\Phi$ is equivalent to satisfiability of $ST(\varphi)\wedge\Phi$, where $ST(\varphi)$ is the standard translation of φ. In this way, we can show that (multi)modal logic is decidable over any class defined by two-variable logic [20], even extended with a linear order [21], counting quantifiers [22], one transitive relation [24], or equivalence closures of two distinguished binary relations [13]. The same holds for formulae of the guarded fragment [7], even if we allow for some restricted application of transitive relations [14, 23], fixed-points [1, 8] and transitive closures [17]. In many cases, the decidability results hold also when only finite frames are considered. However, the complexity bounds obtained this way are high — usually between ExpTime and 2NExpTime.

Many natural classes of frames, including transitive, reflexive, symmetric and Euclidean, can be defined by first-order sentences even if we further restrict the language to universal Horn formulae. In [18], it was shown that the problems defined by universal Horn formulae are always decidable, and the precise complexity (depending on properties of first–order formulae) was established. In [11], the decidability of elementary modal logics defined by universal Horn formulae was extended to the finite satisfiability case.

Observe however, that the above results do not solve our problem – for example, the class of transitive frames that do not contain cliqus of size 3 cannot be defined in any of the languages mentioned above.

Overview of the paper. The proof of Theorem 2 is much easier than the proof of Theorem 1. For a given modal formula φ and its transitive model, we show that we can remove a world from any path containing at least $|\varphi|+1$ worlds such that the resulting structure is a model. By iterating this procedure, we obtain a model in which a there are no paths longer than $|\varphi|$, and the existence of such a model can be verified by an NExpTime algorithm. The whole proof of Theorem 2 is in Section 3.

The general satisfiability case is discussed in Section 4. We start with the global satisfiability problem in Section 4.1. We show that each satisfiable formula has a model which is a clique or an infinite path.

In Section 4.2 we present the proof for the local satisfiability problem. This proof is much more complicated than the other proofs. The general shape of the proof is similar to the previous proofs: we show that each satisfiable formula has a "nice" model, i.e., a model whose description is exponential w.r.t. the size of the modal formula, and that description admits efficient algorithm checking whether described model satisfies a given modal (or first order) formula.

The goal of this paper is to prove the decidability. However, we also discuss the complexity that arises from our algorithms. In the case of the global (finite and infinite) satisfiability problems, we provide the optimal complexity proving that these problems is always in NP (see Proposition 10). The corresponding lower bound comes from the trivial reduction of SAT problem. For the local satisfiability problem we show only that the membership is in NExpTime (see Propositions 5 and 15), which is not optimal – for example, the satisfiability problem of modal logic over the class of all transitive frames is PSpace-complete.

Finally, the possible future work is discussed in Section 5.

2 Preliminaries

As we work with both first-order logic and modal logic we help the reader to distinguish them in our notation; we denote first-order formulae with Greek capital letters, and modal formulae with Greek small letters. We assume that the reader is familiar with first-order logic and propositional logic.

Modal logic. Formulae of modal logic are interpreted in Kripke structures, which are triples of the form $\langle M, R, \pi\rangle$, where M is a set of worlds, $\langle M, R\rangle$ is a directed graph called a *frame*, and π is *a labelling*, a function that assigns to each world a set of propositional variables which are true at this world. We say that a structure $\langle M, R, \pi\rangle$ is *based* on the frame $\langle M, R\rangle$. For a given class of frames $\mathcal{K}$, we say that a structure is $\mathcal{K}$-based if it is based on some frame from $\mathcal{K}$. We will use calligraphic letters $\mathcal{M}, \mathcal{N}$ to denote frames and Fraktur letters $\mathfrak{M}, \mathfrak{N}$ to denote structures. Whenever we consider a structure $\mathfrak{M}$, we assume that its frame is $\mathcal{M}$ and its universe is M (and the same holds for other letters).

The semantics of modal logic is defined recursively. A modal formula φ is (locally) *satisfied* in a world w of a model $\mathfrak{M} = \langle M, R, \pi\rangle$, denoted as $\mathfrak{M}, w \models \varphi$ if

- $\varphi = p$, where p is a variable, and $\varphi \in \pi(w)$,
- $\varphi = \neg\varphi'$ and $\mathfrak{M}, w \not\models \varphi'$,
- $\varphi = \varphi_1 \wedge \varphi_2$ and $\mathfrak{M}, w \models \varphi_1$ and $\mathfrak{M}, w \models \varphi_2$,
- $\varphi = \Diamond\varphi'$ and there is a world $v \in M$ such that $(w, v) \in R$ and $\mathfrak{M}, v \models \varphi'$,

Boolean connectives $\vee, \Rightarrow, \Leftrightarrow$ and constants $\top, \bot$ are introduced in the standard way. We abbreviate $\neg\Diamond\neg\varphi$ by $\Box\varphi$.

We say that a formula φ is *globally* satisfied in $\mathfrak{M}$, denoted as $\mathfrak{M} \models \varphi$, if for all worlds w of $\mathfrak{M}$, we have $\mathfrak{M}, w \models \varphi$. By $|\varphi|$ we denote the length of φ.

For a given class of frames $\mathcal{K}$, we say that a formula φ is *locally* (resp. *globally*) $\mathcal{K}$-*satisfiable* if there exists a $\mathcal{K}$-based structure $\mathfrak{M}$, and a world $w \in W$ such that $\mathfrak{M}, w \models \varphi$ (resp. $\mathfrak{M} \models \varphi$). We study four versions of the satisfiability problem.

For a given formula φ, a Kripke structure $\mathfrak{M}$, and a world $w \in M$ we define the *type* of w (with respect to φ) in $\mathfrak{M}$ as $tp^{\varphi}_{\mathfrak{M}}(w) = \{\psi : \mathfrak{M}, w \models \psi \text{ and } \psi \text{ is a subformula of } \varphi\}$. We write $tp_{\mathfrak{M}}(w)$ if the formula is clear from the context. Note that $|tp^{\varphi}_{\mathfrak{M}}(w)| \leq |\varphi|$.

First-order logic. The class of (equality-free) universal first order sentences is defined as a subclass of first–order sentences such that each sentence is of the form $\forall\vec{x}\,\Psi(\vec{x})$, where $\Psi(\vec{x})$ is quantifier–free formula over the language $\{R\}$, where R is a binary relation symbol interpreted as the successor relation in modal logic. As we work only with universally quantified formulae, we often skip the quantifier prefix, e.g., write xRx instead of $\forall x.xRx$.

The set of *transitive formulae*, $\forall_T$, is defined as the set of those Φ over the language $\{R\}$ which are of the form $(\forall xyz.xRy \wedge yRz \Rightarrow xRz) \wedge \Psi$, where Ψ is an arbitrary universal first-order formula.

Decision problems. The *local* (resp. *global*) *satisfiability problem* $\mathcal{K}$-SAT (resp. $\mathcal{K}$-GSAT) as follows. Is a given modal formula locally (resp. globally) $\mathcal{K}$-satisfiable? The *finite local* (global) satisfiability problem, $\mathcal{K}$-FINSAT ($\mathcal{K}$-GFINSAT), is defined in the same way, but we are only interested in finite models (the class $\mathcal{K}$ may still contain infinite structures).

In context of decision problems, we use the word "general" as an antonym of "finite". Furthermore, for a given universal first-order formula Φ, we define $\mathcal{K}_\Phi$ as the class of frames satisfying Φ.

3 Finite satisfiability

In this section, we show that if a modal formula has a finite model in which the transition relation is transitive, then it has one with the size bounded exponentially in the size of modal formula. This clearly leads to an NExpTime algorithm that simply guesses an exponential model and verifies it.

For a given modal formula φ, we say that a world w in a model $\mathfrak{M}$ is *special* w.r.t. φ, if there is a subformula ψ of φ such that w satisfies ψ, no successor of w different than w satisfies ψ. Worlds that are not special w.r.t. φ are *regular* (w.r.t. φ). We omit explicit reference to φ when it is clear from the context.

The decidability of FINSAT and GFINSAT comes from the following lemma.

▶ **Lemma 3.** *Let Φ be a $\forall_T$ formula, φ be a modal formula, $\mathfrak{M}$ be a model satisfying Φ, w be a regular (w.r.t. φ) world in $\mathfrak{M}$, and $\mathfrak{N}$ be the result of removing w (and all connected edges) from $\mathfrak{M}$. Then, for each $v \in N$, $tp^{\varphi}_{\mathfrak{M}}(v) = tp^{\varphi}_{\mathfrak{N}}(v)$.*

Proof. The proof goes by induction on φ. Consider a subformula ψ of φ. The only non-trivial case is when $\psi = \Diamond\psi'$, w satisfies ψ' and a world w' is a predecessor of w. Since w is regular, there is a world $v \neq w$ reachable from w that satisfies ψ' in $\mathfrak{M}$ and, by inductive assumptions, in $\mathfrak{N}$. Since R is transitive, v is a successor of w', so w' satisfies ψ in $\mathfrak{N}$. ◀

▶ **Example 4.** Consider a formula Φ that states that R is transitive, $\varphi = \Box\Box\bot$ and the structure $\mathfrak{M} = \langle\{a, b, c\}, \{(a, b), (b, c), (a, c)\}, \pi\}$ with $\mathfrak{M}, a \not\models \varphi$. It may seem that the worlds b and c are regular, but if we removed any of them, the type of a would not contain φ.

However, $\Box$ is an abbreviation, thus an unrolled form of φ is $\neg\Diamond\neg\neg\Diamond\neg\bot$. Then, only a satisfies $\Diamond\neg\neg\Diamond\neg\bot$, b (but not c) satisfies $\Diamond\neg\bot$, and only c satisfies $\neg\Diamond\neg\bot$. Thus, all the worlds in $\mathfrak{M}$ are special w.r.t. φ.

We would like to thank an anonymous reviewer for suggesting the above example. Theorem 2 follows from the following proposition.

▶ **Proposition 5.** *Let Φ be a $\forall_T$ formula. Then $\mathcal{K}_\Phi$-FINSAT and $\mathcal{K}_\Phi$-GFINSAT are decidable in* NExpTime.

Proof. Let φ be a modal formula with a finite model $\mathfrak{M}$ based on a frame from $\mathcal{K}_\Phi$. Since the relation of $\mathfrak{M}$ is transitive, by iterating Lemma 3, we can obtain a model $\mathfrak{N}$ without regular worlds. Since all worlds in $\mathfrak{N}$ are special, the maximal length of a path of different worlds in $\mathfrak{N}$ is bounded by $|\varphi|$.

We obtain a small model by applying standard selection technique [2, 15]. In first stage, we mark an arbitrary world satisfying φ. Then, in consecutive stages, for each world w marked in the previous stage and each subformula ψ of φ, if w is connected to a world satisfying ψ, but not to a marked world satisfying ψ, then we mark one successor of w satisfying ψ. Note that this procedure ends after $|\varphi|$ stages — otherwise there would be a path containing more than $|\varphi|$ different worlds. Finally, we remove all worlds but marked. It is not hard to see that the types of remaining worlds do not change and that the size of obtained model is bounded by $|\varphi|^{|\varphi|}$. We have shown that the considered logic has the exponential model property. The NExpTime algorithm simply guesses an exponential model and verifies it. ◀

4 General satisfiability

Consider the formula that defines the class of transitive and irreflexive frames:

$$\Gamma = (\forall xyz.xRy \wedge yRz \Rightarrow xRz) \wedge \forall x.\neg xRx$$

A quick check shows that a modal formula $\Diamond\top \wedge \Box\Diamond\top$ has only infinite models satisfying Γ. In order to illustrate some later concepts, we consider a more complex formula ν which is a conjunction of the following properties.

1. $\Box\neg((p \wedge q) \vee (p \wedge r) \vee (q \wedge r)) \wedge \Box(p \vee q \vee r)$ stating that each reachable world satisfies exactly one of variables p, q, r (recall that we consider transitive structures).
2. $\Diamond p \wedge \Box(p \Rightarrow \Diamond p)$ stating that there is a reachable world satisfying p, and each reachable world satisfying p has a successor satisfying p.
3. $\Box(p \Rightarrow \Diamond r) \wedge \Box(r \vee q \Rightarrow \Diamond q \wedge \Box q)$ stating that each reachable world satisfying p has a successor satisfying r, and each reachable world satisfying r or q has a successor satisfying q and all its successors satisfy q.

Fig. 1 contains two models of ν satisfying Γ (the formula is satisfied at the root). It is not hard to see that each such model has to contain two infinite chains: one labelled by p and one labelled by q.

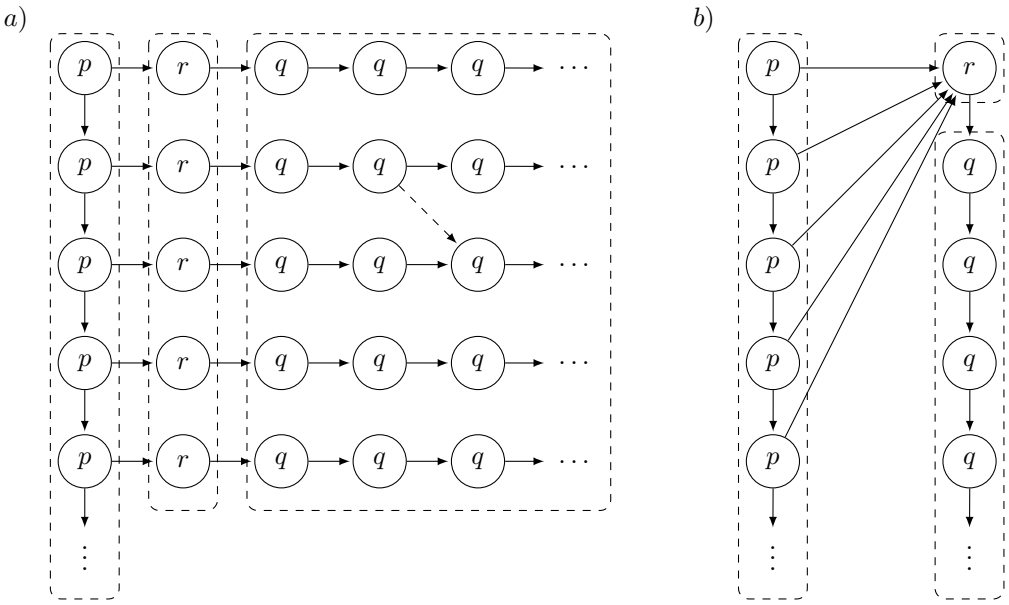


Figure 1 The running example — two models of ν over Γ. The edges that follows from the transitivity are omitted for better readability. Note that the structure a) is a model of ν even if we remove the dashed edge.

4.1 Disentangled structures and global satisfiability

By employing the standard translation [2] and the Löwenheim–Skolem theorem we know that each satisfiable modal formula has a countable model over any class of first-order definable frames. In this section we consider only countable structures.

In order to simplify the exposition in this section, we assume that a modal formula φ is fixed and all notions defined below (a tight structure, a special clique, ...) depend on φ and are defined with respect to φ.

We say that a (sub)structure is a *clique* if it is a single strongly-connected component, i.e., for all two different worlds w, v we have $wRv \wedge vRw$. Note that a single world is a clique, regardless of whether it is reflexive; we will call such a world a *trivial clique*.

Let φ be a modal formula. We say that a structure is *tight* if the size of its cliques is bounded by $|\varphi|$. We omit explicit reference to φ when it is clear from the context.

▶ **Lemma 6.** *Let Φ be a $\forall_T$ formula, $\mathfrak{M}$ be a structure satisfying Φ and φ be a modal formula. Then there is a tight (w.r.t. φ) substructure $\mathfrak{N}$ of $\mathfrak{M}$ such that for each $v \in N$, $tp^{\varphi}_{\mathfrak{M}}(v) = tp^{\varphi}_{\mathfrak{N}}(v)$.*

Proof. In each clique C in $\mathfrak{M}$ with size greater than $|\varphi|$, we do the following. For each subformula ψ of φ satisfied in some world of C, we mark one world satisfying ψ in C. We remove all worlds from C that are not marked. A straightforward induction w.r.t. the size of subformulae shows that the types (w.r.t. φ) of remaining worlds do not change. We do the operation described above independently for all cliques in the model. ◀

Let M be a model containing a world w. A formula ψ is a *reachable formula* of w if it is a subformula of φ and there is a world v satisfying ψ such that $wRv \wedge \neg vRw$. Let B_w denote the set of all reachable formulae of w. We extend this notation to cliques — for a clique C that contains a world w, we put $B_C = B_w$ (notice that B_C does not depend on the choice of w from C).

We say that a clique is *special* if some world w of this clique satisfies some subformula ψ of φ with $\psi \notin B_w$, and regular otherwise.

For a set B of subformulae of φ, we define (B, s)-*zones* in a structure $\mathfrak{M}$ as follows. The $(B, 0)$-zone is the substructure of $\mathfrak{M}$ that consists of worlds from all regular cliques C such that $B_C = B$. The $(B, 1)$-zone is the substructure of $\mathfrak{M}$ that consists of worlds from all special cliques C such that $B_C = B$. We say that a (B, s)-zone is *special* if $s = 1$ and *regular* otherwise. Clearly, each world is in exactly one zone and the number of (B, s)-zones is bounded by $O(2^{|\varphi|})$.

We define a partial order $\preccurlyeq$ on zones as follows. We say that (B, s)-zone $\preccurlyeq$ (B', s')-zone if $B \subset B'$ or $B = B'$ and $s \leq s'$.

▶ **Lemma 7.** *If $\mathfrak{M}$ is transitive, then for any B, B', s, s', world w from a (B, s)-zone and w' from a (B', s')-zone, if wRw', then $(B', s') \preccurlyeq (B, s)$.*

Proof. Suppose that wRw'. Of course, $B_{w'} \subseteq B_w$. Assume that $(B', s') \not\preccurlyeq (B, s)$, i.e., $B_{w'} = B_w$, $s = 0$ and $s' = 1$. Then, w' is a special (trivial) clique and w is not, so there is no edge from w' to w. As w' is special, it satisfies some $\psi \notin B$. But, since wRw', $\psi \in B_w = B$ — a contradiction. ◀

We say that a frame is a *line* if its reflexive closure is isomorphic with $\langle \mathbb{N}, \leq \rangle$. We extend this notation to structures, saying that a (sub)structure is a *line* if its frame is a line. We jointly call cliques and lines *units*. For a line l, we denote by l^i the *ith world* of l, i.e., the world that have i proper predecessors in l.

Our aim is to show that all modal formulae satisfiable over some class definable by a $\forall_T$ formula Φ have models with a bounded number of units.

We say that a structure $\mathfrak{M}$ is *disentangled* if each special zone of $\mathfrak{M}$ consist of not connected cliques and each regular zone of $\mathfrak{M}$ consists of lines $l_1, l_2, \ldots$ such that if there is an edge from a world in l_i to a world in l_j, then $i \leq j$.

▶ **Example 8.** Consider structures presented in Fig. 1 and a modal formula ν as a reference formula. All worlds satisfying r are special (w.r.t. ν) — none of their successors satisfies r. All other worlds are regular (w.r.t. ν).

Consider any of the structures presented in Fig. 1. Let $\mathfrak{p}, \mathfrak{q}$ and $\mathfrak{r}$ be arbitrary worlds of that structure labelled by p, q and r, respectively. Observe that the structure consists of three zones: the $(B_{\mathfrak{p}}, 0)$-zone that contains all elements satisfying p, the $(B_{\mathfrak{r}}, 1)$-zone that contains elements satisfying r and the $B_{\mathfrak{q}}$-zone that contains all elements satisfying q. Note that $B_{\mathfrak{q}} \subseteq B_{\mathfrak{r}} \subseteq B_{\mathfrak{p}}$ — if there is an edge from one zone to another, then by transitivity the first one contains all the successors of some world in the second one. The second inclusion is strict — case in point, $r \in B_{\mathfrak{p}} \setminus B_{\mathfrak{r}}$. But the first one is not — it is not hard to see that $B_{\mathfrak{r}} = B_{\mathfrak{q}}$.

The model b) is disentangled, and each of its zones contains precisely a single unit — a line or a trivial clique (consisting of a single world). The model a) is not disentangled, but it contains a disentangled submodel — it is enough to remove the dashed edge.

▶ **Lemma 9.** *Let Φ be a $\forall_T$ formula, φ be a modal formula and $\mathfrak{M}$ be a tight (w.r.t. φ) structure satisfying Φ and φ. Then there is a tight (w.r.t. φ) disentangled substructure $\mathfrak{N}$ of $\mathfrak{M}$ such that for each $v \in N$, $tp^{\varphi}_{\mathfrak{M}}(v) = tp^{\varphi}_{\mathfrak{N}}(v)$.*

Proof. We show that units in a $(B, 1)$-zone in $\mathfrak{M}$ are not connected. Assume that for some worlds w, v from two different units of this zone we have wRv. World v is in a special clique, therefore there is a world v' in this clique and a subformula $\psi \notin B$ of φ satisfied by v'. Since $\neg v'Rw$ (otherwise v', w would be in the same clique) and wRv' it follows that $\psi \in B_w = B$. This is a contradiction. The structure $\mathfrak{N}$ therefore contains all the $(B, 1)$-zones of $\mathfrak{M}$.

Now we consider regular zones; we transform a $(B, 0)$-zone to a substructure that consists of ordered lines. First, we remove all points with only finite numbers of successors. Such worlds are regular, as they are in a regular zone, therefore Lemma 3 guarantees that the types of the remaining worlds are not changed.

Let $w_1, w_2, \ldots$ be an enumeration of those worlds in the $(B, 0)$-zone such that for all worlds v, if v has a successor in the $(B, 0)$-zone then it has a successor among $w_1, w_2, \ldots$, and no two worlds among $w_1, w_2, \ldots$ are in the same clique. Below we inductively define a sequence $l_1, l_2, \ldots$ such that each l_i is a line or it is empty.

For each i, if w_i has a successor in some l_j with $j < i$, then we leave l_i empty. Otherwise, let l_i be an arbitrary path starting in w_i that does not contain non-trivial cliques, i.e., cliques of consisting of more than one world.

We remove all the worlds that are not in any l_j. Clearly, there are no edges from l_i to l_j for any $i > j$. We add the remaining worlds to $\mathfrak{N}$ – these worlds form the $(B, 0)$-zone of $\mathfrak{N}$. To obtain a sequence matching the requirements of the definition of disentangled structure, we simply remove empty elements from $l_1, l_2, \ldots$. ◀

▶ **Proposition 10.** *For any $\forall_T$ formula Φ, $\mathcal{K}_\Phi$-GSAT and $\mathcal{K}_\Phi$-GFINSAT are in* NP.

Proof. We prove that each satisfiable modal formula has a model which is a single unit. Clearly, the existence of such a model can be verified in NP.

Assume that φ has a model satisfying Φ. By Lemma 9 there is a disentangled model $\mathfrak{M}$ of φ and Φ. Let a (B, s)-zone be a maximal nonempty zone w.r.t. $\preccurlyeq$. It is not hard to see (Lemma 7) that there are no edges from the (B, s)-zone to another zone. Let $\mathfrak{N}$ be any unit of this zone. Of course, $\mathfrak{N}$ satisfies Φ as it is a submodel of $\mathfrak{M}$. Now we define a single-unit structure $\mathfrak{N}'$ that satisfies φ.

If $s = 1$, then we simply put $\mathfrak{N}' = \mathfrak{N}$. By the definition of disentangled models, there are no edges from $\mathfrak{N}$ to different units in the same zone. Moreover, by choice of the (B, s)-zone

there are no edges in $\mathfrak{M}$ from any world from $\mathfrak{N}$ to any different zone. Therefore, each world in $\mathfrak{N}$ has the same successors in $\mathfrak{N}$ as in $\mathfrak{M}$, and therefore its type is the same in both models.

If $s = 0$, then we define a line $\mathfrak{N}'$ over a universe $\underline{0}, \underline{1}, \dots$ containing fresh worlds as follows. Let the frame $\mathcal{N}'$ be isomorphic with $\mathcal{N}$ and let $\{t_0, t_1, \dots, t_{k-1}\}$ be a set of labels of worlds of the (B, s)-zone (restricted to the propositional variables of ϕ). For every $i \geq 0$, we label a world $\underline{i}$ as $t_{i \mod k}$. A quick check shows that the types of worlds in $\mathfrak{N}'$ are the same as the types of worlds in the (B, s)-zone.

This proof can be simply adjusted to the finite global satisfiability — in any finite model, we can find a clique without successors. ◀

4.2 Local satisfiability

First, we define a property of sets of worlds called *homogeneity*. Roughly speaking, all the worlds from a homogeneous set have the same set of predecessors and successors outside of this set.

A world v is called *a proper predecessor (successor)* of w w.r.t. W if v is a predecessor (successor, resp.) of w and $v \notin W$.

A set of worlds W is (P, S)-*homogeneous* (in $\mathfrak{M}$) if for every world $w \in W$ the set of proper successors of w w.r.t. W is equal to S and the set of proper predecessors of w w.r.t. W is equal to P.

A structure $\mathfrak{M}'$ is the (P, S)–*homogenization* of W in a structure $\mathfrak{M}$ if $\mathfrak{M}'$ is a structure with the same universe as $\mathfrak{M}$ such that W is (P, S)–homogeneous in $\mathfrak{M}'$ and $\mathfrak{M}'$ is obtained by changing the proper predecessors and the proper successors (w.r.t W) of worlds from W to P and S respectively. More precisely, there is an edge from w to v in $\mathfrak{M}'$ if and only if one of the following holds.

- w and v are both in W or both outside W and there is an edge from w to v in $\mathfrak{M}$.
- w is in P and v is in W or it is a successor of some world of W.
- v is in Q and w is in W or it is a predecessor of some world of W.

A set W is *homogeneous* (in $\mathfrak{M}$) if there are P, S such that W is (P, S)–homogeneous (in $\mathfrak{M}$). Figure 2 contains an example of a homogenization. Observe that for all P, S, W and $\mathfrak{M}$ there is a unique a (P, S)–homogenization of W in $\mathfrak{M}$.

A set of worlds W is a *witness for a set of worlds* V if for each world $v \in V$ and each subformula $\Diamond\psi$ of φ, if v has a proper successor w.r.t. V satisfying ψ, then it has one in W. Note that if V is a finite set of worlds within one unit, there is a witness for V of size bounded by $|\varphi|$.

Our aim is to prove the following lemma.

▶ **Lemma 11.** *Let Φ be a $\forall_T$ formula and φ be a modal formula satisfiable over $\mathcal{K}_\Phi$. Then there is a tight (w.r.t. φ) disentangled model of φ satisfying Φ s.t. the number of units is bounded exponentially in the size of φ and all its units are homogeneous.*

The first step towards the proof of Lemma 11 is enforcing an additional restriction on lines. Namely, we can assume, without loss of generality, that every line consists of either only reflexive worlds or only irreflexive worlds.

▶ **Lemma 12.** *Let Φ be a $\forall_T$ formula and φ be a modal formula satisfiable over $\mathcal{K}_\Phi$. Then there exists a tight (w.r.t. φ) disentangled model of φ satisfying Φ such that every line is isomorphic with $\langle \mathbb{N}, \leq \rangle$ or $\langle \mathbb{N}, < \rangle$.*

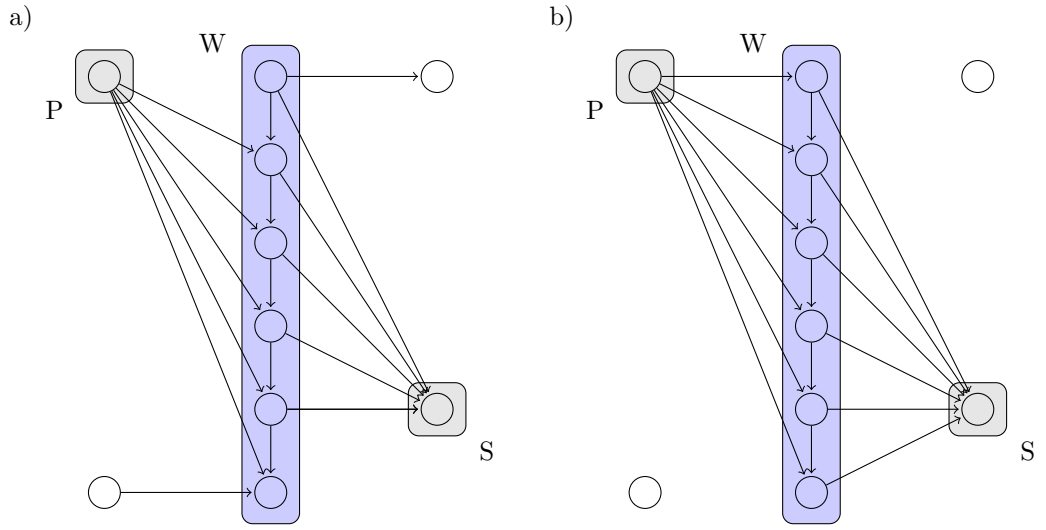


Figure 2 The structure on the picture b) is the result of the (P, S)–homogenization of W in the structure on the picture a). Some edges that follow from the transitivity are omitted for better readability.

Proof. If φ is a modal formula satisfiable over $\mathcal{K}_\Phi$, there exists a tight disentangled model $\mathfrak{M}$ of φ satisfying Φ.

Let $l_1, l_2, \ldots$ be an enumeration of lines in $\mathfrak{M}$. We show that for every $i > 0$, there is a model $\mathfrak{M}_i$ of φ satisfying Φ such that for every $j \in \{1, \ldots, i-1\}$, $\mathcal{M}_i \cap l_j$ is isomorphic with $\langle \mathbb{N}, \leq \rangle$ or $\langle \mathbb{N}, < \rangle$.

Let l_i be a line in $\mathfrak{M}_i$. If l_i contains finitely many reflexive worlds, they can be removed. Let $\mathfrak{M}_{i+1}$ be the resulting structure. Due to Lemma 3, $\mathfrak{M}_{i+1}$ satisfies φ and, as a substructure of $\mathfrak{M}_i$, it satisfies Φ. Clearly, the line $\mathcal{M}_{i+1} \cap l_i$ is isomorphic with $\langle \mathbb{N}, < \rangle$.

Otherwise, l_i contains infinitely many reflexive worlds. Let $\mathfrak{N}$ be a substructure of $\mathfrak{M}_i$ resulting from removing all irreflexive worlds from l_i. Clearly, $\mathcal{N} \cap l_i$ is isomorphic with $\langle \mathbb{N}, \leq \rangle$. The structure $\mathfrak{N}$ satisfies Φ, but it may violate φ.

Therefore, we define $\mathfrak{M}_{i+1}$ as a structure defined on the frame $\mathcal{N}$ such that the labelling of $\mathfrak{M}_{i+1}$ coincides with the labelling of $\mathfrak{M}_i$ on worlds the do not belong to l_i, i.e., $w \in M_{i+1} \setminus l_i$. Now, on worlds $w \in M_{i+1} \cap l_i$ the labelling is defined in such a way that every labelling that occurs on l_i in $\mathfrak{M}_{i+1}$, occurs infinitely often in $M_{i+1} \cap l_i$ in $\mathfrak{M}_{i+1}$. It is easy to verify that the types for all proper predecessors of $M_{i+1} \cap l_i$ in $\mathfrak{M}_{i+1}$ remain the same as in $\mathfrak{M}_i$. ◀

Before we show Lemma 11, we provide an outline. Define $n = |\varphi|$ and $N = |\Phi|$. We start from an arbitrary tight disentangled model $\mathfrak{M}$ such that every line is isomorphic with $\langle \mathbb{N}, \leq \rangle$ or $\langle \mathbb{N}, < \rangle$ and modify it in three steps.

Step 1. Select recursively worlds from $\mathfrak{M}$. Initially, select an arbitrary element a of $\mathfrak{M}$ such that $\mathfrak{M}, a \models \varphi$. Then, recursively, for every world w selected in the previous stage do the following. If w is in a non-trivial clique, select a witness for this clique of size bounded by $|\varphi|$.

If w is in a line l, select from l the world l^k with the following property: for every selected world v preceding l, if v has a successor in l, then l^k is a successor of v as well. (There are finitely many selected v preceding l, thus such l^k exists.) Then, remove all worlds from l preceding w', and select worlds l^k, l^{k+1}, …, $l^{k+N \cdot n^{2 \cdot n}}$. Finally, choose for them a witness

W and select worlds from $W \setminus l$.

Let $\mathfrak{M}_1$ be the structure obtained in this step. We show that $\mathfrak{M}_1$ satisfies Φ and φ and at most $n^{2\cdot n}$ units contain selected worlds.

Step 2. Remove from $\mathfrak{M}_1$ those units that do not contain selected worlds, obtaining the structure $\mathfrak{M}_2$. Clearly, $\mathfrak{M}_2$ satisfies Φ as it is a substructure of $\mathfrak{M}$. However, it may happen that the modal formula φ is not satisfied in $\mathfrak{M}_2$.

Consider a model presented at Fig. 1 a) and the unit that consists of all worlds satisfying p. If we select only finitely many units satisfying r and remove all other such units, then the obtained structure would not satisfy the ν. We find a witness for the set of first $N \cdot n^{2\cdot n}$ worlds from the line and select all its worlds. Later we will show how we can use them as witnesses for all other worlds from this line.

Step 3. In each line l in $\mathfrak{M}_2$, starting from the greatest zone w.r.t. $\preccurlyeq$, we find a homogeneous set of N worlds that is before the world $l^{N\cdot n^{2\cdot n}}$. Since the number of all units in $\mathfrak{M}_2$ is bounded by $n^{2\cdot n}$, such a sequence can be easily found. Then, we use this sequence to obtain a line such that all worlds in this line have the same set of successors as the worlds in the sequence and that transformation does not violate Φ (this trick will be presented as Lemma 13).

Let $\mathfrak{M}_3$ be the structure obtained by applying this procedure to all lines. We conclude by an inductive proof that shows that $\mathfrak{M}_3$ satisfies φ.

Let us discuss the properties of $\mathfrak{M}_1$ defined in Step 1. At every stage of the construction, either the selected worlds are from the same unit or they belong to strictly greater zones w.r.t. $\preccurlyeq$. Since chains w.r.t. $\preccurlyeq$ are bounded by $2 \cdot n$, there is at most $2 \cdot n$ stages of the construction. Every unit that contains a selected world has at most n selected witnesses. Thus, there are at most $n^{2\cdot n}$ units with selected worlds. The structure $\mathfrak{M}_1$ results from $\mathfrak{M}$ by removing finitely many regular worlds, therefore by Lemma 3, for every $v \in \mathfrak{M}_1$, $tp_{\mathfrak{M}_1}(v) = tp_{\mathfrak{M}}(v)$. In particular, $\mathfrak{M}_1$ satisfies φ. As a substructure of $\mathfrak{M}$, $\mathfrak{M}_1$ satisfies Φ.

Observe that $\mathfrak{M}_1$ satisfies the following property: if a selected world v has a successor in a unit U with a selected world, all worlds of U are successors of v. This is clear if U is a clique, and we remove from lines worlds that violate this. Every unit in $\mathfrak{M}_2$ contains a selected world, therefore $\mathfrak{M}_2$ satisfies the following property:

(1) if a selected world v has a successor in a unit U, all worlds from U are successors of v.

Now, we elaborate on Step 3, i.e., we show a construction of $\mathfrak{M}_3$ from $\mathfrak{M}_2$ and we show that it satisfies Φ. We start with the following lemma stating that careful homogenization of a structure satisfying Φ satisfies Φ as well.

▶ **Lemma 13.** *Let Φ be a $\forall_T$ formula, $N = |\Phi|$, $\mathfrak{M}$ be a structure over $\mathcal{K}_\Phi$, l be a line in $\mathfrak{M}$ isomorphic with $\langle \mathbb{N}, \leq \rangle$ or $\langle \mathbb{N}, < \rangle$. If there is k such that $\{l^k, l^{k+1}, \ldots, l^{k+N-1}\}$ is (P, S)–homogeneous, then the $(P \setminus l, S \setminus l)$–homogenization of l in $\mathfrak{M}$ satisfies Φ.*

Proof. Let $\mathfrak{M}'$ be the $(P \setminus l, S \setminus l)$–homogenization of l in $\mathfrak{M}$. Let us assume that l is isomorphic with $\langle \mathbb{N}, < \rangle$. The case where l is isomorphic with $\langle \mathbb{N}, \leq \rangle$ is similar. We show that $\mathfrak{M}'$ satisfies Φ. To avoid confusion with modal logic, we show that $\mathcal{M}'$, which is a relational structure, satisfies the first order formula Φ.

The formula Φ is a conjunction of the fromula $(\forall xyz.xRy \wedge yRz \Rightarrow xRz)$ and a universal first-order formula. Thus, it can be transformed to an equivalent formula of the form $\forall \vec{x}.\Psi(\vec{x})$, where $\Psi(\vec{x})$ is quantifier-free and $|\vec{x}| \leq N$. Suppose that $\mathfrak{M}'$ does not satisfy Φ and let $\vec{u} = \langle u_0, \ldots, u_{|\vec{x}|-1} \rangle$ be such that $\mathcal{M}'$ violate $\Psi(\vec{u})$. We can rearrange $\vec{u}$ such that

$\vec{u} = \vec{u_1} \cup \vec{u_2}$, $\vec{u_1}$ are from l, $\vec{u_2}$ are disjoint with l, and $\vec{u_1}$ are ordered w.r.t. the successors relation of $\mathfrak{M}'$. Let $f : \vec{u} \mapsto \mathcal{M}'$ be defined as

$$f(u_i) = \begin{cases} l^{k+i} & \text{if } u_i \in \vec{u}_1 \\ u_i & \text{otherwise} \end{cases}$$

Let $f(\vec{u})$ denote image of $\vec{u}$ under f, i.e., $f(\vec{u}) = \langle f(u_0), \ldots, f(u_{|\vec{x}|-1})\rangle$. We show that

(i) for all $v, v' \in \vec{u}$, $\mathcal{M}'$ satisfies vRv' iff $\mathcal{M}'$ satisfies $f(v)Rf(v')$,
(ii) for all $v, v' \in \vec{u}$, $\mathcal{M}'$ satisfies $f(v)Rf(v')$ iff $\mathcal{M}$ satisfies $f(v)Rf(v')$.

It follows that if $\mathcal{M}'$ violates $\Psi(\vec{u})$, then $\mathcal{M}$ violates $\Psi(f(\vec{u}))$. Thus, if $\mathcal{M}$ satisfies Φ, then $\mathcal{M}'$ satisfies Φ.

(i) : Clearly, (i) holds when v, v' are both from $\vec{u_1}$ or both from $\vec{u_2}$. If $v \in \vec{u_1}$ and $v' \in \vec{u_2}$, then $v \in l$ and $v' \notin l$. $(P \setminus l, S \setminus l)$-homogeneity implies that $\mathcal{M}'$ satisfies vRv' iff $\mathcal{M}'$ satisfies $f(v)Rv'$. Since $v' \notin l$, $v' = f(v')$. Therefore, $\mathcal{M}'$ satisfies vRv' iff $\mathcal{M}'$ satisfies $f(v)Rf(v')$. The case $v \in \vec{u_2}$ and $v' \in \vec{u_1}$ is similar.

(ii) : $(P \setminus l, S \setminus l)$-homogenization only changes successors and predecessors of worlds from l. This implies that (ii) holds if v, v' are both form $\vec{u_1}$ or both from $\vec{u_2}$.

If $v \in \vec{u_1}$ and $v' \in \vec{u_2}$, then $f(v) \in \{l^k, \ldots, l^{k+N-1}\}$ and $v' \notin l$. Thus, $\mathcal{M}'$ satisfies $f(v)Rf(v')$ iff $v' \in S \setminus l$ iff $\mathcal{M}$ satisfies $f(v)Rf(v')$. Similarly, if $v \in \vec{u_2}$ and $v' \in \vec{u_1}$, then $f(v') \in \{l^k, \ldots, l^{k+N-1}\}$ and $\mathcal{M}'$ satisfies $f(v)Rf(v')$ iff $v \in P \setminus l$ iff $\mathcal{M}$ satisfies $f(v)Rf(v')$. ◀

Let $\sqsubset$ be a linear order relation on units of $\mathfrak{M}_2$ compatible with the reversed $\preccurlyeq$, i.e., if U_1 is from a greater zone than U_2, then $U_1 \sqsubset U_2$. Since $\mathfrak{M}_2$ has finitely many units, $\sqsubset$ is well-founded. We define structures $\mathfrak{N}_U$, where U ranges over units of $\mathfrak{M}_2$. The structures $\mathfrak{N}_U$ will be defined inductively w.r.t. the order $\sqsubset$. They will satisfy the following induction assumption: $\mathfrak{N}_U$ satisfies Φ and for every unit U' such that $U' \sqsubseteq U$, U' is homogeneous.

For the induction base we introduce an empty unit $\emptyset$ and we state that it is the least unit w.r.t. $\sqsubset$. Clearly, $\mathfrak{N}_\emptyset = \mathfrak{M}_2$ satisfies all the assumptions.

For the induction step assume that $\mathfrak{N}_V$ has been defined, it satisfies Φ and every unit $U' \sqsubseteq V$ is homogenous in $\mathfrak{N}_V$. Let U be the least unit greatest than V (w.r.t. $\sqsubset$). Since cliques are homogeneous, $\mathfrak{N}_U = \mathfrak{N}_V$ satisfies all the assumptions.

Assume that U is a line l. (1) implies the following dichotomy for selected worlds: for every selected world w and every unit U' either all worlds from U' are successors of w or no world from U' is a successor of w. Therefore, a subset of units determines the set of successors of a given selected world. Due to transitivity, for all w, w'; if w' is a successor of w, the set of successors of w' is a subset of the set of successors of w. Since there are $n^{2 \cdot n}$ different units, among $N \cdot n^{2 \cdot n}$ consecutive selected worlds there are N consecutive worlds $\vec{u}$ that have the same set of successors outside the line l. Observe (1) implies that all selected worlds in l have the same predecessors. Therefore, $\vec{u}$ have the same sets of predecessors and successors outside l. As they are consecutive on l, predecessors of u_0 are predecessors of $\vec{u}$ and successors of u_{N-1} are successors of $\vec{u}$. Hence, there are P, S such that $\vec{u}$ is a (P, S)-homogeneous set. Let $\mathfrak{N}_U$ be $(P \setminus l, S \setminus l)$-homogenization of l in $\mathfrak{N}_V$. By Lemma 13, $\mathfrak{N}_U$ satisfies Φ. All units homogeneous in $\mathfrak{N}_V$ remain homogeneous in $\mathfrak{N}_U$ and $U = l$ is homogeneous.

We have shown that the construction given in Steps 1, 2 and 3 can be executed and that $\mathfrak{M}_3$ satisfies all the postulated conditions from Lemma 11 but one. It remains to be verified that $\mathfrak{M}_3$ satisfies φ.

▶ **Lemma 14.** *The structure $\mathfrak{M}_3$ satisfies φ.*

Proof. We show that worlds from $\mathfrak{M}_3$ have the same types in $\mathfrak{M}_3$ as in $\mathfrak{M}_1$, i.e., for each $v \in M_3$, $tp_{\mathfrak{M}_1}(v) = tp_{\mathfrak{M}_3}(v)$. Observe that $\mathfrak{M}_3$ contains the root of $\mathfrak{M}_1$, thus it satisfies φ. To show that the types do not change, we show the following property:
(2) for every $v \in \mathfrak{M}_3$, $\Diamond\psi \in tp_{\mathfrak{M}_1}(v)$ iff v has a successor u in $\mathfrak{M}_3$ such that $\psi \in tp_{\mathfrak{M}_1}(u)$.
We show (2) in two steps. First, observe that
(3) for every selected $v \in \mathfrak{M}_2$, $\Diamond\psi \in tp_{\mathfrak{M}_1}(v)$ iff v has a successor u in $\mathfrak{M}_2$ s.t. $\psi \in tp_{\mathfrak{M}_1}(u)$.
Clearly, $\mathfrak{M}_2$ is a substructure of $\mathfrak{M}_1$, therefore if v has a successor u in $\mathfrak{M}_2$ such that $\psi \in tp_{\mathfrak{M}_1}(u)$, then $\Diamond\psi \in tp_{\mathfrak{M}_1}(v)$.

Conversely, assume that $\Diamond\psi \in tp_{\mathfrak{M}_1}(v)$. We need to show that v has a successor $u \in \mathfrak{M}_1$ such that $\psi \in tp_{\mathfrak{M}_1}(u)$ and u is in a unit that contains a selected world from $\mathfrak{M}_1$. Clearly, such u belongs to $\mathfrak{M}_2$, thus (3) holds.

Recall that, for every $w \in \mathfrak{M}_1$, $tp_{\mathfrak{M}}(w) = tp_{\mathfrak{M}_1}(w)$. Let v' be the last selected successor (in $\mathfrak{M}$) of v that satisfies $\Diamond\psi$ (it can be v itself). The world v' has a selected successor u satisfying ψ. If u belongs to $\mathfrak{M}_1$, we are done. We show that $u \notin M_1$ is impossible. Suppose that u belongs to a line l in $\mathfrak{M}$, but it has been removed at Step 1. Then, due to regularity of l, there is a successor u' of u in l, which is selected and belongs to $\mathfrak{M}_1$. Since every world on l is regular, u' has a successor in l satisfying ψ. Thus, u' is a selected strict successor of v' and it satisfies $\Diamond\psi$, which contradicts the assumption that v' is the last selected successor of v that satisfies $\Diamond\psi$.

Observe that (3) implies (2): If v belongs to a clique, then it has exactly the same successors in $\mathfrak{M}_3$ as in $\mathfrak{M}_2$, thus (3) implies (2).

Assume that v belongs to a line l. Let w be a selected world from the same line l which belongs to a chosen homogeneous set. Since w has the same successors in $\mathfrak{M}_2$ and $\mathfrak{M}_3$, (3) implies (2) for w, i.e., $\Diamond\psi \in tp_{\mathfrak{M}_1}(w)$ iff w has a successor u' in $\mathfrak{M}_3$ such that $\psi \in tp_{\mathfrak{M}_1}(u')$.

All worlds in l have the same set of reachable formulae, therefore $\Diamond\psi \in tp_{\mathfrak{M}_1}(v)$ iff $\Diamond\psi \in tp_{\mathfrak{M}_1}(w)$. Since all worlds in l share the set B, the set of reachable formulae, alike in $\mathfrak{M}_1$ as in $\mathfrak{M}_3$, v has a successor u in l satisfying $\psi \in tp_{\mathfrak{M}_1}(u)$ iff w has a a successor u' in l satisfying $\psi \in tp_{\mathfrak{M}_1}(u')$. Also, l is homogeneous in $\mathfrak{M}_3$, therefore v and w have the same successors outside l. Thus, w has a successor u' in $\mathfrak{M}_3$ such that $\psi \in tp_{\mathfrak{M}_1}(u')$ iff u has a successor u'' in $\mathfrak{M}_3$ such that $\psi \in tp_{\mathfrak{M}_1}(u'')$. This implies (2).

Finally, we show that the property (2) implies that for each $v \in M_3$, $tp_{\mathfrak{M}_1}(v) = tp_{\mathfrak{M}_3}(v)$.
The *modal depth*, denoted by $MD(\varphi)$, of a modal formula φ is defined recursively:

- $MD(p) = 0$ where p is a propositional variable or a logical constant ($\top$, $\bot$).
- $MD(\neg\varphi) = MD(\varphi)$.
- $MD(\varphi_1 \wedge \varphi_2) = max(MD(\varphi_1), MD(\varphi_2))$.
- $MD(\Diamond\varphi) = MD(\varphi + 1)$.

We show by induction on n that for every $v \in M_3$, for every subformula ψ of φ with $MD(\psi) \leq k$, $\psi \in tp_{\mathfrak{M}_1}(v)$ iff $\psi \in tp_{\mathfrak{M}_3}(v)$.

Induction base. The labelling π is not changed at any stage of construction, therefore the types $tp_{\mathfrak{M}_1}(v)$ and $tp_{\mathfrak{M}_2}(v)$ agree on formulae of modal depth zero.

Induction step. Assume that for every $v \in \mathfrak{M}_3$, $tp_{\mathfrak{M}_1}(v)$ and $tp_{\mathfrak{M}_3}(v)$ agree on formulae of modal depth n. Let $w \in \mathfrak{M}_3$. Consider a subformula $\Diamond\psi$ of φ, where modal depth of ψ is n. By (2), $\Diamond\psi \in tp_{\mathfrak{M}_1}(w)$ iff w has a successor u in $\mathfrak{M}_3$, such that $\psi \in tp_{\mathfrak{M}_1}(u)$. Since ψ has modal depth n, the induction assumption implies that $\psi \in tp_{\mathfrak{M}_1}(u)$ iff $\psi \in tp_{\mathfrak{M}_3}(u)$. Therefore, $\Diamond\psi \in tp_{\mathfrak{M}_1}(w)$ iff $\Diamond\psi \in tp_{\mathfrak{M}_3}(w)$.

Observe that if $tp_{\mathfrak{M}_1}(w)$ and $tp_{\mathfrak{M}_3}(w)$ agree on formulae of modal depth $n+1$ of the form $\Diamond\psi$, then they agree on all formulae of modal depth $n+1$. ◀

▶ **Proposition 15.** *Let Φ be a $\forall_T$ formula. Then $\mathcal{K}_\Phi$-SAT is decidable in* NExpTime.

Proof. For a given $\forall_T$ formula Φ, we describe an algorithm that checks whether a given modal formula φ has a model that satisfies Φ, φ and consists of exponentially many homogeneous units. Lemma 11 implies that such an algorithm is complete. The algorithm works as follows:

Guess a finite description of a structure $\mathfrak{N}$. Observe that if every unit of $\mathfrak{N}$ is homogeneous, then for all units U_1, U_2, either all worlds from U_1 are successors (predecessors) of all worlds from U_2 or no world from U_1 is a successor (predecessor) of any world from U_2. Therefore, if $\mathfrak{N}$ has at most M units, it has a description of size $M \cdot (M \cdot 2^{|\varphi|})$. For every unit U it needs to be stated whether U is a line or a clique, which units are successors of U and which types occur in $\{tp_{\mathfrak{N}}(u) : u \in U\}$. The order of types in a line is irrelevant and we can assume that each type occurs infinitely often.

Verify the modal formula. It suffices to check whether every type in every unit has appropriate witnesses and if a formula $\neg\Diamond\psi$ belongs to this type, the formula ψ does not belong to any type in the current unit or successor units.

Verify the first-order formula. The first order formula is universal, i.e., $\Phi = \forall\vec{x}.\Psi(\vec{x})$, where Ψ is quantifier-free. Therefore, it has to be checked whether every instantiation of $\vec{x}$ in $\mathfrak{N}$ satisfies Ψ. However, observe that for all $\vec{a}, \vec{b}$ satisfying (a) for every $i \in [1, N]$, a_i is in the same unit as b_i, and (b) for all a_i, a_j from the same unit, a_i is a successor of a_j iff b_i is a successor of b_j, we have $\mathcal{M}_3$ (the frame of $\mathfrak{M}_3$) satisfies $\Psi(\vec{a})$ iff $\mathcal{M}_3$ satisfies $\Psi(\vec{b})$.

It follows that we have to check at most finitely many different types of tuples $\vec{a}$. Each type of a tuple is determined by the assignment of its elements to the units of $\mathfrak{N}$ (cond. (a)) and relative order of elements in the same unit (cond. (b)). Notice that these two characteristics determine relations among worlds. Hence, it can be easily verified whether tuples of a given type satisfy Ψ.

The number of types of tuples $\vec{a}$ that have to be checked is bounded by $c^N(2N)!$, where c is the number of units. Since N is a parameter of the problem and c is of exponential order, they can be checked in exponential time in $|\varphi|$. Even if Φ was a part of an instance, N is linear in the size of the input and $c^N(2N)!$ is still exponential in the input. ◀

5 Conclusion and future work

We proved that all elementary modal logics over universally defined classes of transitive structures are decidable. In case of the global satisfiability problem, we proved that all satisfiability problems and finite satisfiability problems are in NP and, by a straightforward reduction from SAT, we can conclude that these problems are NP-complete. The case of the local satisfiability problem is more complicated. We proved that all the satisfiability problems and all the finite satisfiability problems are in NExpTime, but the precise complexity may vary. For example, the logic K4 is PSpace-complete, while S5 is NP-complete. Providing the full characterisation with respect to complexity is left as an open question.

Modal logic over a class of transitive frames can be seen as a temporal logic. Indeed, the logic K4 may be defined as a syntactic variant of a fragment of CTL that allows only two modalities — AG and EF. Therefore, Theorem 1 may be treated as a first step in study of *elementary temporal logics*. A natural question that arises is whether the decidability results can be extended for different CTL operators, such as EG and AF.

References

1 Vince Bárány and Mikolaj Bojanczyk. Finite satisfiability for guarded fixpoint logic. *Inf. Process. Lett.*, 112(10):371–375, 2012.

2 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Comp. Sc.* Cambridge University Press, Cambridge, 2001.

3 Davide Bresolin, Dario Della Monica, Valentin Goranko, Angelo Montanari, and Guido Sciavicco. The dark side of interval temporal logic: Sharpening the undecidability border. In *TIME*, pages 131–138. IEEE, 2011.

4 R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

5 Kit Fine. Logics containing K4. Part II. *J. of Symbolic Logic 50*, pages 619–651, 1985.

6 Igor Gorbunov. A decidable modal logic that is finitely undecidable. In *Advances in Modal Logic*, pages 247–258. College Publications, 2006.

7 Erich Grädel. On the restraining power of guards. *J. Symbolic Logic*, 64:1719–1742, 1999.

8 Erich Grädel and Igor Walukiewicz. Guarded fixed point logic. In *Fourteenth Annual IEEE Symposium on Logic in Computer Science*, pages 45–54, 1999.

9 Edith Hemaspaandra and Henning Schnoor. A universally defined undecidable unimodal logic. In *MFCS*, volume 6907 of *LNCS*, pages 364–375. Springer, 2011.

10 Wilfrid Hodges. *Model Theory*. Cambridge University Press, 1993.

11 Emanuel Kieroński and Jakub Michaliszyn. Two-variable universal logic with transitive closure. In *Proceedings of Computer Science Logic 2012*, 2012.

12 Emanuel Kieroński, Jakub Michaliszyn, and Jan Otop. Modal logics definable by universal three-variable formulas. In *FSTTCS*, volume 13 of *LIPIcs*, pages 264–275, 2011.

13 Emanuel Kieroński, Jakub Michaliszyn, Ian Pratt-Hartmann, and Lidia Tendera. Extending two-variable first-order logic with equivalence closure. In *LICS '12: Proceedings of the 29th IEEE symposium on Logic in Computer Science*, 2012.

14 Emanuel Kieroński and Lidia Tendera. On finite satisfiability of the guarded fragment with equivalence or transitive guards. In *LPAR*, volume 4790 of *LNCS*, pages 318–332. Springer, 2007.

15 Marcus Kracht. *Tools and Techniques in Modal Logic*. Studies in Logic and the Foundations of Mathematics. Elsevier, Amsterdam, 1999.

16 Jerzy Marcinkowski and Jakub Michaliszyn. The ultimate undecidability result for the Halpern-Shoham logic. In *LICS*, pages 377–386. IEEE Computer Society, 2011.

17 Jakub Michaliszyn. Decidability of the guarded fragment with the transitive closure. In *ICALP (2)*, volume 5556 of *LNCS*, pages 261–272. Springer, 2009.

18 Jakub Michaliszyn and Jan Otop. Decidable elementary modal logics. In *LICS '12: Proceedings of the 29th IEEE symposium on Logic in Computer Science*, 2012.

19 Jakub Michaliszyn, Jan Otop, and Piotr Witkowski. Satisfiability vs. finite satisfiability in elementary modal logics. In *Proceedings of GandALF 2012*, pages 141–154, 2012.

20 Michael Mortimer. On languages with two variables. *Mathematical Logic Quarterly*, 21(1):135–140, 1975.

21 Martin Otto. Two variable first-order logic over ordered domains. *Journal of Symbolic Logic*, 66:685–702, 1998.

22 Ian Pratt-Hartmann. Complexity of the two-variable fragment with (binary-coded) counting quantifiers. *Journal of Logic, Language and Information*, 2005.

23 Wiesław Szwast and Lidia Tendera. The guarded fragment with transitive guards. *Annals of Pure and Applied Logic*, 128:227–276, 2004.

24 Lidia Tendera and Wiesław Szwast. FO^2 with one transitive relation is decidable. In *Proc. of STACS 2013*, 2013.

A Fully Abstract Game Semantics for Parallelism with Non-Blocking Synchronization on Shared Variables

Susumu Nishimura

Dept. of Mathematics, Graduate School of Science, Kyoto University
Sakyo-ku, Kyoto 606-8502, JAPAN
susumu@math.kyoto-u.ac.jp

Abstract

We present a fully abstract game semantics for an Algol-like parallel language with non-blocking synchronization primitive. Elaborating on Harmer's game model for nondeterminism, we develop a game framework appropriate for modeling parallelism. The game is a sophistication of the wait-notify game proposed in a previous work, which makes the signals for thread scheduling explicit with a certain set of extra moves. The extra moves induce a Kleisli category of games, on which we develop a game semantics of the Algol-like parallel language and establish the full abstraction result with a significant use of the non-blocking synchronization operation.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages, F.1.1 Models of Computation, F.1.2 Modes of Computation

Keywords and phrases shared variable parallelism, non-blocking synchronization, full abstraction, game semantics

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.578

1 Introduction

In shared memory parallel programming, parallel threads competing for shared memory cells (or shared variables) must be appropriately synchronized to avoid race conditions. A synchronization method is called *non-blocking*, if each individual thread spins over a shared resource until it acquires an exclusive access to it. In contemporary architectures including multicores, non-blocking synchronization is supported via the read-modify-write operation, most notably known as *compare-and-set (CAS)* operation. [12]

This paper concerns with game theoretical analysis of an Algol-like parallel language that supports non-blocking synchronization on shared variables. Game semantics for PCF and Idealized Algol have been well investigated and shown fully abstract [13, 3]. However, the standard methods used in the game modeling do not directly apply to the parallel extension considered in this paper:

- The models for the above deterministic languages solely concern *may-convergence*, i.e., they just observe if a program has the possibility of termination. The parallel programs, on the other hand, are inherently nondeterministic and thus may-convergence is too imprecise to give a pleasant discrimination of parallel programs: Even if two programs are judged equivalent, they can nondeterministically exhibit different convergences.
- One might expect that the parallel execution would be modeled by interleaved game plays of simultaneously running threads, but this fails to properly shuffle variable accesses, due to the parity restriction originating from the Hyland-Ong game [13], in which the opponent moves and the player moves must strictly alternate.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 578–596

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In a preliminary work [20], Watanabe and the present author proposed *wait-notify games* as a means to remedy the above issues. They developed the wait-notify games based on Harmer's game semantics [9, 10], in order to capture the nondeterministic nature of parallel computation more precisely. Harmer's games substantially extend Hyland-Ong's with the notion of divergence, giving the full abstraction result for a nondeterministic variant of Algol-like sequential language. It concerns both may-convergence and *must-convergence*, i.e., it also discriminates those programs which are obliged to terminate from those which are not.

The fundamental idea in wait-notify games is to have each game play interspersed by a suitable number of pairs of extra wait and notify moves, written `W` and `N`, respectively. A wait move `W` represents a delay imposed by the scheduler of the operating system, each time a single execution thread attempts to access a shared variable; A subsequent notify move `N` represents the resumption of the delayed variable access by the scheduler. However, wait-notify games are defined only for the type of parallel computation and are not well integrated with the computational structure of other types, including higher-order ones. The resulting parallel language thereby supports parallelism only under a fairly limited context: Within parallel contexts, nothing but shared variables can be parametrized.

The present paper sophisticates the idea in the wait-notify game to give a fully abstract game semantics for an Algol-like parallel language with non-blocking synchronization, in which parallelism is allowed under much wider contexts of arbitrary types, though subject to a few modest syntactic restrictions. We will develop the game model in a Kleisli category of games, induced from the monadic structure introduced by the wait and notify moves. This not only enables us to reach to the full abstraction result in a standard way but also reveals the computational structure hindered behind the parallel computation.

Here we emphasize that we do *not* intend to model parallel computation as a game between individual parallel threads. Rather, we model parallel computation as a game between the collection of simultaneously running threads and the scheduler, which is the entity invisible in the program text. The extra wait and notify moves enable the game semantical construction with the scheduler's interference explicit. The extra moves, on the other hand, should not be counted when we discuss observational behavior of programs. Thus we need to introduce a scheduler strategy that ignores these extra moves all together, later in Section 5.

The development in this paper also gives some indications on the nature of parallel computing:

- As we will discuss later, the game model in this paper has no ability to observe the termination of the entire collection of threads running in parallel. This implies that no language whose parallel running threads can join to a single sequential thread would be fully abstract with respect to the present game model or its modest extension. Due to this fact, we are driven to design our parallel language so that no parallel threads join: There is no means to merge the set of parallel threads into a single sequential thread, even after all the parallel threads have terminated.
- In the course of establishing full abstraction, we need to separate out a history-insensitive part from a given game strategy. This is usually done by the so-called *innocent factorization* [3], which does not directly apply to our parallel setting, though. The parallel threads would compete for a shared variable in which the factorized strategy keeps the history. There we make an indispensable use of the non-blocking synchronization operation CAS, as a means for mutual exclusion on the shared variable. This indicates that a language without CAS or its equivalent would be strictly less able to define strategies than the language with CAS would be.

Related work. The game model in this paper can be seen as a resumption model, which has been used for denotational modeling of concurrency [17, 19] and later examined for giving a game semantics for parallelism [1]. The present paper investigates the computational structure in the resumption-style (wait-notify) game and develops the full abstraction result for a suitable Algol-like shared variable parallel language.

There have been several approaches to full abstraction for shared variable parallel languages, e.g., [5] by Brookes and [7, 8] by Ghica, Murawski, et al. These studies concern blocking synchronization primitives, i.e., some locking mechanisms, while the present paper concerns non-blocking ones. More significantly, whereas they solely discuss may-convergence, we discuss may&must-convergence. This enables a more precise analysis of parallel computation that is inherently nondeterministic, though we need certain quotienting on the game model.

Unlike in the parallel languages they consider, parallel threads in our language never join to continue with a sequential computation, as mentioned earlier. This rigid distinction of sequential and parallel computation in our language might be more suitable for distributed computing, where a collection of parallel threads execute in concert but their computation results not necessarily need to coalesce. In a distributed environment, the indefinability without CAS might seem a reminiscent of the impossibility result for the wait-free distributed consensus [11], but one should mind their difference. Wait-freeness assumes a fairness in scheduling, while the present game model does not: A parallel computation in the latter is judged to diverge as soon as at least one out of the parallel threads does, while the former is judged irrespective of divergence of a subset of threads.

Outline. The rest of the paper is organized as follows. Section 2 introduces an Algol-like parallel language and gives its operational semantics. Reviewing Harmer's games for nondeterminism in Section 3, we develop in Section 4 a game framework, in which parallel programs are interpreted in a Kleisli category of wait-notify games. Section 5 defines the game interpretation for terms and its soundness is shown. In Section 6, we show the full abstraction, which is derived by combining two factorizations followed by a definability result. Finally, Section 7 concludes the paper with some topics for future investigations.

2 The IA_{par} Language

Let us define a programming language for shared variable parallelism, called IA_{par}, which is yet another variant of Idealized Algol [18]. IA_{par} is a strongly typed language, whose types and syntax are defined as below.

$$\begin{aligned}
\mathtt{T} &::= \mathtt{nat} \mid \mathtt{com} \mid \mathtt{var} \mid \mathtt{par} \mid \mathtt{T} \to \mathtt{T} \\
M &::= x \mid n \mid M \ \star \ M \mid \lambda x^{\mathtt{T}}.M \mid MM \mid \mathsf{fix}_{\mathtt{T}}\, M \mid \mathsf{skip} \mid \mathsf{seq}\ M\ M \mid \mathsf{if0}\ M\ \mathsf{then}\ M\ \mathsf{else}\ M \\
&\quad \mid \mathsf{assign}\ M\ M \mid \mathsf{deref}\ M \mid \mathsf{cas}\ M\ M\ M \mid \mathsf{mkvar}\ M\ M\ M \mid \mathsf{newvar}\ v = n\ \mathsf{in}\ M \\
&\quad \mid M\ \mathsf{or}\ M \mid M_1 \| \cdots \| M_\zeta \mid M \gg_j M
\end{aligned}$$

The types consist of base types and arrow types built from them. Base types are either `nat` for natural numbers, `com` for sequential commands, `var` for mutable variables, or `par` for parallel commands. A *type judgment* of the form $\Gamma \vdash M : \mathtt{T}$ assigns the type τ for the term M, where Γ is a *typing context*, a finite mapping from identifiers to types. The typing rules are given in Figure 1 of Appendix A.

The terms consist of PCF terms (natural numbers, λ-terms, and general recursion, where the binary operator $\star$ on natural numbers at least includes the addition $+$ and the cut-off

subtraction $-$, Algol terms (commands for mutable variables and the bad variable constructor mkvar), the erratic nondeterminism or [9, 10], and parallel constructs. The *parallel command* $M_1 \| \cdots \| M_\zeta$ executes the sequential commands M_1, ..., M_ζ simultaneously, where ζ is the fixed degree of parallelism. The *thread extension* $P \gg_j M$ $(1 \leq j \leq \zeta)$ extends the command execution of the j-th thread in the parallel command P by a sequential command M. That is, as soon as the execution in the j-th thread as specified in P has terminated, the same thread continues to execute the sequential command M. The *preamble sequencing* $\mathsf{seq}\ M\ P$, where P is of type $\mathtt{par}$, executes the sequential command M in advance of the parallel command P. Within a parallel command, every simultaneously running thread has parallel access to shared variables, which are ranged over by the identifiers v, v', ... of type $\mathtt{var}$. In addition to the primitives assign and deref for atomic read and write on mutable variables, respectively, it also provides compare-and-set (CAS) operation as a means for non-blocking synchronization on shared variables. A CAS operation $\mathsf{cas}\ v\ m\ n$ is an atomic uninterruptible operation that conditionally updates the value stored in v: If the present value stored in v is equal to m then it updates the value to n and returns n; otherwise, it leaves v as it is and returns the stored value.

The formal operational semantics for IA_{par} is given in the style of small-step operational semantics (Figure 2 of Appendix A). Each *1-step reduction* $\langle M, s\rangle \longrightarrow \langle M', s'\rangle$ corresponds to a single atomic sequential execution that may update the store s to s' as the side effect, where a *store* is a finite mapping from mutable variables to natural numbers.

We say the evaluation of a term M *may-converges* at initial state s, if $\langle M, s\rangle \longrightarrow^* \langle M', s'\rangle$ for some state s' and no reduction rules apply to $\langle M', s'\rangle$ further, where $\longrightarrow^*$ is the reflexive transitive closure of $\longrightarrow$. Also, we say the evaluation of a term M *must-converges* at initial state s, if there is no infinite reduction sequence $\langle M, s\rangle \longrightarrow \langle M', s'\rangle \longrightarrow \cdots$. In particular when M is a closed term, we write $M{\Downarrow}^{\mathrm{may}}$ for may-convergence and also write $M{\Downarrow}^{\mathrm{must}}$ for must-convergence.

A *contextual preorder* on terms is defined by means of both may- and must-convergences. We define $M \lesssim_{\mathrm{may}} N$ iff, for any context $C[-]$ of type $\mathtt{par}$, $C[M]{\Downarrow}^{\mathrm{may}}$ implies $C[N]{\Downarrow}^{\mathrm{may}}$. Also, $M \lesssim_{\mathrm{must}} N$ iff, for any context $C[-]$ of type $\mathtt{par}$, $C[M]{\Downarrow}^{\mathrm{must}}$ implies $C[N]{\Downarrow}^{\mathrm{must}}$. Overall, we define the approximation of convergence by: $M \lesssim_{\mathrm{m\&m}} N$ iff $M \lesssim_{\mathrm{may}} N$ and $M \lesssim_{\mathrm{must}} N$.

3 Harmer's game for nondeterminism

This section reviews Harmer's game model together with a little sophistication specifically needed for the development in the present paper. It is intended to make the present paper self-contained as much as possible, but some details are omitted due to page limitation. For the full details, see Harmer's thesis [10]. More general aspects on game semantics for Algol-like languages can be found in [4].

Arenas. The definition of arenas is standard, except that the arena moves are ordered. An *arena* A is a triple $((M_A, <_A), \lambda_A, \vdash_A)$, where M_A is a countable set of *moves*, associated with a nonreflexive total order $<_A$ on them; $\lambda_A : M_A \to \{\mathsf{O}, \mathsf{P}\} \times \{\mathsf{Q}, \mathsf{A}\}$ is a *labeling* function that assigns each move $m \in M_A$ its attributes, either O (opponent) or P (player) and either Q (question) or A (answer); the *enabling relation* $\vdash_A$ is a binary relation over the moves satisfying: (e1) $(a \vdash_A b \wedge a \neq b) \implies \lambda_A^{\mathsf{OP}}(a) \neq \lambda_A^{\mathsf{OP}}(b)$; (e2) $b \vdash_A b \implies (\lambda_A(b) = (\mathsf{O}, \mathsf{Q}) \wedge (a \neq b \implies a \not\vdash_A b))$; (e3) $(a \vdash_A b \wedge \lambda_A^{\mathsf{QA}}(b) = \mathsf{A}) \implies \lambda_A^{\mathsf{QA}}(a) = \mathsf{Q}$, where we write $\lambda_A^{\mathsf{OP}}(b)$ (resp., $\lambda_A^{\mathsf{QA}}(b)$) for the opponent/player (resp., question/answer) attribution of

the move b. When $a \vdash_A b$ ($a \neq b$), we say a *justifies* b. A move a is called an *initial move* if $a \vdash_A a$.

The most trivial arena is $\mathbf{1} = (\emptyset, \emptyset, \emptyset)$. The arena $\mathbf{C} = ((M_{\mathbf{C}}, <_{\mathbf{C}}), \lambda_{\mathbf{C}}, \vdash_{\mathbf{C}})$ corresponding to type com of commands is specified, as usual, by the data $M_{\mathbf{C}} = \{\mathtt{run}, \mathtt{done}\}$, $\lambda_{\mathbf{C}}(\mathtt{run})=(\mathsf{O}, \mathsf{Q})$, $\lambda_{\mathbf{C}}(\mathtt{done}) = (\mathsf{P}, \mathsf{A})$, $\vdash_{\mathbf{C}}= \{(\mathtt{run}, \mathtt{run}), (\mathtt{run}, \mathtt{done})\}$, together with the ordering $\mathtt{run} <_{\mathbf{C}} \mathtt{done}$. For the arenas of other base types and type constructors, see Appendix B.

Here we notice that the arena definition does *not* preclude the possibility that question moves are justified by answer moves. We say such a question move justified by an answer move a a *subquestion* move (inferior to a). Let us write $Subq_A(a) = \{q \mid a \vdash_A q\}$ for the set of all subquestions inferior to the answer move a. We call an answer move a a *subquestioning answer* iff $Subq_A(a) \neq \emptyset$. Throughout the paper, we assume that $Subq_A(a)$ is a finite set for each answer move a. In this paper, subquestion moves in lifted arenas will play a significant role in modeling parallelism (Section 4.1).

Justified strings, legal plays, and strategies. Let us write ε for an empty string of arena moves and st (occasionally written $s \cdot t$ for clarity) for concatenation of strings of moves s and t. We write $s \sqsubseteq t$ to mean that s is a prefix of t and also write $s \sqsubseteq^{\mathrm{even}} t$ in particular when s is an even-length string. Typically, strings of arena moves are ranged over by s, t, u, etc., while arena moves are ranged over by a, b, p, q, etc.

A *justified string* in an arena A is a sequence of moves in which every non-initial move p has a pointer to an earlier occurrence of a justifying move q (i.e., $q \vdash p$), written like $\cdots\ q \overset{\frown}{\cdots}\ p\ \cdots$. In particular when p is an answer move (and hence q is a question move), we say "p answers q."

We say a justified string s is *well-opened*, if s has at most a single occurrence of initial move. We also define the *player view* of a justified string s, denoted by $\mathsf{V}(s)$, by induction on the length of s as follows: (i) $\mathsf{V}(sq) = q$ if q is initial; (ii) $\mathsf{V}(sqtp) = \mathsf{V}(s)qp$ if p is an opponent move and q justifies p; (iii) $\mathsf{V}(sq) = \mathsf{V}(s)q$ if q is a player move.

A justified string s in arena A is called a *legal play*, if s strictly alternates opponent/player moves, that is, $s = o_1p_1o_2p_2\cdots$ where o_i's are opponents and p_i's are players. We write L_A to denote the set of legal plays in the arena A and also L_A^{even} (resp., L_A^{odd}) to denote the set of even (resp., odd) length legal plays.

In order to appropriately model may&must-convergence in nondeterministic programs, Harmer defined each game strategy by a pair of execution traces and witnesses of divergence.

A *strategy* σ is a pair (T_σ, D_σ), where the trace set T_σ is an even-length prefix closed subset of L_A^{even} and the divergence set D_σ is a subset of L_A^{odd} satisfying: (d1) if $s \in T_\sigma$, $sa \in L_A$, and $sa \notin dom(\sigma)$, then there exists $d \in D_\sigma$ such that $d \sqsubseteq sa$; (d2) if $sa \in D_\sigma$, then $s \in T_\sigma$; If $rng_\sigma(sa)$ is an infinite set, then there exists $d \in D_\sigma$ such that $d \sqsubseteq sa$, where $rng_\sigma(sa) = \{sab \mid sab \in T_\sigma\}$ is the *range* of moves that follows sa and $dom(\sigma) = \{sa \mid rng_\sigma(sa) \neq \emptyset\}$ is the *domain* of σ.

We say a divergence $d \in D_\sigma$ *interesting*, if $d \in dom(\sigma)$; otherwise, it is called *uninteresting*. A strategy σ is called *deterministic* if $rng_\sigma(sa)$ is a singleton set for every $sa \in dom(\sigma)$; A strategy σ is called *reliable* if it is deterministic and further every $sa \in D_\sigma$ is uninteresting.

The composition of two strategies $\sigma : A \to B$ and $\tau : B \to C$, written $\sigma;\tau : A \to C$, is obtained by parallel composition and hiding on the pair of plays taken from each of the strategies. Given $s \in L_{A\to B}$ and $s' \in L_{B\to C}$, a parallel composition of s and s' is a play t of moves in $M_A \cup M_B \cup M_C$ such that s (resp., s') is a restriction of t to the moves $M_A \cup M_B$ (resp., $M_B \cup M_C$). Hiding the moves M_B occurring in t, we obtain a composite play.

The trace part $T_{\sigma;\tau}$ is the set of composite plays of any $s \in T_\sigma$ and $s' \in T_\tau$. The divergence $D_{\sigma;\tau}$ is the set of divergences that is a union of subsets generated by the following two ways. One subset is obtained by parallel composition and hiding on the pair of a trace from one strategy and a divergence from the other strategy. The other subset is obtained from *infinite* traces generated by a pair of traces taken from both: Let u^∞ be an infinite play of moves in $M_A \cup M_B \cup M_C$ such that only a finite number of moves from M_A or M_B are witnessed in u^∞. Then the restriction of u^∞ to $M_A \cup M_B$ moves is a divergent play, as u^∞ can be understood as exhibiting an *infinite chatter*, where two strategies make infinite (thus diverging) interaction with each other in the arena B.

We say strategy τ is more likely to converge than σ, denoted by $\sigma \leq^\natural \tau$, iff $T_\sigma \subseteq T_\tau \wedge \forall d' \in D_\tau.\exists d \in D_\sigma.d \sqsubseteq d' \wedge \forall sab.(sab \in T_\tau \wedge sab \notin T_\sigma \Longrightarrow \exists d \in D_\sigma.d \sqsubseteq sab)$.

For every arena A, there is the least element $\bot_A$ subject to $\leq^\natural$, specified by $(T_{\bot_A}, D_{\bot_A}) = (\{\varepsilon\}, \{q \mid q\text{: initial}\})$. We will define $\sigma =^\natural \tau$ iff $\sigma \leq^\natural \tau$ and $\tau \leq^\natural \sigma$.

Strategy subclasses. Throughout the paper, we will only concern the class of *single-threaded* strategies [9, 10]. Intuitively, a single-threaded strategy consists of plays that are closed under arbitrary interleaving of several copies of plays. This intuition is supported by the fact that single-threaded strategies have a bijective correspondence with the so-called well-opened ones, representing a single execution of a sequential program.

We say a strategy σ is *well-opened* iff every play $s \in T_\sigma$ is well-opened and so is every interesting divergence $d \in D_\sigma$. The bijective correspondence between single-threaded strategies and well-opened ones, up to $=^\natural$, is established by a pair of mappings $WO\,(-)$ and $ST\,(-)$: for every single-threaded strategy σ and well-opened strategy υ, we have $\sigma =^\natural ST\,(WO\,(\sigma))$ and $WO\,(ST\,(\upsilon)) =^\natural \upsilon$, where $WO\,(\sigma)$ restricts the plays in σ to those well-opened ones and $ST\,(\upsilon)$ interleaves several copies of the well-opened plays in υ.

Due to this bijective correspondence, we may specify a single-threaded strategy σ by just giving the well-opened plays contained in the trace T_σ and the interesting divergences in D_σ. Every uninteresting divergence is implicitly identified by (d1), i.e., $sa \in L_A^{\mathrm{odd}}$ is identified as an uninteresting divergence whenever $s \in T_\sigma$ but $sa \notin dom(\sigma)$. Furthermore, the trace set can be identified by (the prefix closure of) the longest well-opened plays. Also, we may not even mention divergences, when the strategy has no interesting divergences. For example, when we say a single-threaded strategy $\sigma : \mathbf{C}$ is specified by the trace set $\{\mathtt{run} \cdot \mathtt{done}\}$ (of the longest well-opened plays), σ is formally a strategy defined by the pair $(T_\sigma, D_\sigma) = (\{(\mathtt{run}\cdot\mathtt{done})^k \mid k \geq 0\}, \{(\mathtt{run}\cdot\mathtt{done})^k \cdot \mathtt{run} \mid k \geq 1\})$.

In this paper, we will mostly concern a further limited class of strategies satisfying the following closure properties, except for the scheduler strategy to be presented in Section 5.

- A strategy σ is called *player visible*, if for every $sa \in T_\sigma$, the player move a is justified by a move in $\mathsf{V}(s)$; A strategy σ is called *player bracketing*, if for every $sa \in T_\sigma$, a is the answer to the *pending question*, i.e., the last occurrence of unanswered question in $\mathsf{V}(s)$.

Harmer gave a fully abstract semantics for Idealized Algol with erratic nondeterminism on a cartesian closed category $\mathcal{C}$ of games, whose objects are the arenas and morphisms are the ($=^\natural$-equivalence classes of) single-threaded strategies. The identity arrow assigned to an object A in $\mathcal{C}$ is the *copycat* strategy $id_A : A \to A$, which is specified by the trace set $T_{id_A} = \{s \in L^{\mathrm{even}}_{A_1 \to A_0} \mid \forall t \sqsubseteq^{\mathrm{even}} s.(t \upharpoonright A_1 = t \upharpoonright A_0)\}$.[1] Restricting morphisms in $\mathcal{C}$ to those player visible and player bracketing ones, we have a lluf subcategory, denoted by $\mathcal{C}_{vb}$.

[1] We may occasionally put subscripts or primes in order to distinguish different copies of the same arena.

In what follows, by abuse of notation, we denote each morphism in $\mathcal{C}$ (and their subcategories as well) by σ, a representative of the equivalence class containing it.

4 The Game for Parallelism

As usual, each IA_{par} term is assigned a game strategy whose arena is determined by the type of the term. The base types (except for $\mathtt{par}$) and function types are each interpreted as $[\![\mathtt{nat}]\!] = \mathbf{N}$, $[\![\mathtt{com}]\!] = \mathbf{C}$, $[\![\mathtt{var}]\!] = \mathbf{Var}$, and $[\![\mathtt{T} \to \mathtt{T}]\!] = [\![\mathtt{T}]\!] \Rightarrow [\![\mathtt{T}]\!]$. Each typing context $\Gamma = x_1 : \mathtt{T}_1, \cdots x_k : \mathtt{T}_k$ is interpreted by a product arena, namely, $[\![\Gamma]\!] = [\![\mathtt{T}_1]\!] \times \cdots \times [\![\mathtt{T}_k]\!]$.

4.1 Interleaving game plays

While a single-threaded strategy just interleaves *independent* execution of threads, the execution of threads in a shared variable parallel program can be affected by the order of interleaved variable access operations. This possible dependency between threads can be properly modeled in lifted arenas [16, 10] within a single-threaded strategy.

- A *lifted arena* $A_\perp$ is a triple $((M_{A_\perp}, <_{A_\perp}), \lambda_{A_\perp}, \vdash_{A_\perp})$, where $M_{A_\perp} = \{?, \surd\} + M_A$, $\lambda_{A_\perp} = [\lambda', \lambda_A]$ with $\lambda'(?) = (\mathsf{O}, \mathsf{Q})$ and $\lambda'(\surd) = (\mathsf{P}, \mathsf{A})$, and $p \vdash_{A_\perp} q$ iff $p = q =?\ \vee\ (p = ? \wedge q = \surd) \vee (p = \surd \wedge q \vdash_A q) \vee (p \vdash_A q \wedge p \neq q)$. The associated ordering extends $<_A$ with $? <_{A_\perp} \surd$ and also $p <_{A_\perp} q$ for every $p \in \{?, \surd\}$ and $q \in M_A$.

We interpret $\mathtt{par}$ type by $[\![\mathtt{par}]\!] = (\mathbf{C}_1 \times \cdots \times \mathbf{C}_\zeta)_\perp$. The individual threads in this arena is ordered by: $\mathtt{run}_1 <_{[\![\mathtt{par}]\!]} \mathtt{run}_2 <_{[\![\mathtt{par}]\!]} \cdots <_{[\![\mathtt{par}]\!]} \mathtt{run}_\zeta$. An IA_{par} term $\Gamma \vdash \mathsf{seq}\ M\ (M_1 \| \cdots \| M_\zeta) : \mathtt{par}$ is interpreted by a strategy whose play has the form $?\, s\, \surd\, t$, where s models the interaction of the command M with Γ and t models the execution of the subsequent parallel command $M_1 \| \cdots \| M_\zeta$. whose interleaved parallel execution is modeled by the subsequent play t. As mentioned in [10], the lifting construction gathers parallel threads of computation into a single sequence of play, allowing single-threaded strategies to express history-sensitive execution.

The arenas that interpret $\mathtt{par}$ and higher-types involving it, however, still contain some strategies that are not definable by the terms of IA_{par}: The language IA_{par} is carefully designed to force affine uses of parallel computational contents. Further, a parallel computational content cannot be converted to a sequential computational content either; it can only be either discarded or modified by means of a few parallel constructs.

A suitable (fully abstract) game model for IA_{par} is obtained by further restricting the class of strategies to *subquestion-affine* ones, which satisfy the following properties.

(sq1) For every $s{\cdot}?' \in T_\sigma$, if $?' \vdash_A \surd'$ for some subquestioning answer $\surd'$, then the occurrence of $?'$ is justified by an occurrence of $?$ in s, where $?$ is a question move satisfying $? \vdash_A \surd$ for some subquestioning answer $\surd$.

(sq2) For every $s{\cdot}\surd' \in \mathit{dom}(\sigma)$ where $\surd'$ is a subquestioning answer, $s{\cdot}\surd' \notin D_\sigma$ and $\mathit{rng}_\sigma(s{\cdot}\surd') = \{s{\cdot}\surd'{\cdot}\surd\}$ for some subquestioning answer $\surd$.

(sq3) For every $s{\cdot}\surd'{\cdot}\surd{\cdot}t{\cdot}q \in \mathit{dom}(\sigma)$ such that $\surd'$ and $\surd$ are subquestioning answers and $q \in \mathit{Subq}_A(\surd)$, it holds that $s{\cdot}\surd'{\cdot}\surd{\cdot}t{\cdot}q \notin D_\sigma$ and $\mathit{rng}_\sigma(s{\cdot}\surd'{\cdot}\surd{\cdot}t{\cdot}q) = \{s{\cdot}\surd'{\cdot}\surd{\cdot}t{\cdot}q{\cdot}\rho(q)\}$, where $\rho(q)$ is justified by $\surd'$ and $\rho : \mathit{Subq}_A(\surd) \mapsto \mathit{Subq}_A(\surd')$ is the bijection that preserves the order $<_A$.

(sq4) For every $u{\cdot}b{\cdot}\surd{\cdot}s{\cdot}q{\cdot}t \in T_\sigma$ where q is a subquestion move justified by $\surd$ which is further justified by an initial move, if $u{\cdot}b{\cdot}\surd{\cdot}s{\cdot}q{\cdot}t{\cdot}q \in L_A^{\mathrm{odd}}$ in which the both occurrences of q are justified by $\surd$ and also b is *not* a subquestioning move, then $u{\cdot}b{\cdot}\surd{\cdot}s{\cdot}q{\cdot}t{\cdot}q \notin \mathit{dom}(\sigma)$.

To see how these conditions compel the affine use of parallel computation, let $\sigma \in [\![\mathtt{T}]\!]$ be a subquestion-affine strategy for some type $\mathtt{T}$. The condition (sq4) applies to the case where $\mathtt{T}$ has the form $\cdots \to \mathtt{par}$, with q being any $\mathtt{run}_j \in [\![\mathtt{par}]\!]$ $(1 \leq j \leq \zeta)$, prohibiting any duplicated occurrences of the same move $\mathtt{run}_j$. This compels each thread of a parallel command to execute once and only once.

The conditions (sq1)–(sq3) apply to the case where $\mathtt{T}$ contains a positive occurrence of $\mathtt{par}$ and a negative occurrence of $\mathtt{par}'$ in the form $(\mathtt{T}' \to \mathtt{par}') \to \cdots \to \mathtt{par}$. Any occurrence of $?' \in M_{[\![\mathtt{par}']\!]}$ in a trace of σ must be justified $? \in M_{[\![\mathtt{par}]\!]}$ [(sq1)]; Any occurrence of $\surd' \in M_{[\![\mathtt{par}']\!]}$ in σ must be immediately followed by $\surd \in M_{[\![\mathtt{par}']\!]}$ without diverging [(sq2)]; Any occurrence of $\mathtt{run}_j \in M_{[\![\mathtt{par}]\!]}$ in σ must be immediately followed by $\mathtt{run}'_j \in M_{[\![\mathtt{par}']\!]}$ without diverging, unless the subquestioning move $\surd$ that justifies $\mathtt{run}_j$ is ever preceded by $\surd'$ [(sq3)]. Thus a typical trace of σ has the form like: $?\cdot s\cdot ?'\cdots\surd'\cdot\surd\cdot\mathtt{run}_1\cdot\mathtt{run}'_1\cdots\mathtt{run}_2\cdot\mathtt{run}'_2\cdots\mathtt{done}'_1\cdot t_1\cdot\mathtt{done}_1\cdots\mathtt{done}'_2\cdot t_2\cdot\mathtt{done}_2\cdots\mathtt{run}_1\cdot\mathtt{run}'_1\cdots$. It is intended that the corresponding function makes just a single copy of computation of the argument type $\mathtt{par}'$, possibly augmenting it by preamble sequencing and thread extension (as denoted by subsequences s and t_i's in the trace above, respectively). Notice that, as opposed to the case of (sq4) that does not copy parallel computation, the trace can contain duplicated occurrences of the same move $\mathtt{run}_j$, each immediately followed by $\mathtt{run}'_j$. These duplicates are not harmful, since they are superficial in a sense that solely the earliest one of the duplicates can come into play, eventually when the strategy is combined with some strategy in $[\![\mathtt{T}' \to \mathtt{par}']\!]$.

In what follows, we will develop a game semantical framework on a category $\mathcal{G}$ of games, a lluf subcategory of $\mathcal{C}_{vb}$, whose morphisms of player visible and player bracketing strategies are further restricted to subquestion-affine ones. The category $\mathcal{C}$ and their subcategories $\mathcal{C}_{vb}$ and $\mathcal{G}$ are cartesian closed. For any objects A and B, $A \Rightarrow B$ is the exponential object with its associated evaluation map $ev_{A,B} : (A \Rightarrow B) \times A \to B$ being a copycat strategy between the copies of arenas A and B, respectively. The currying isomorphism is written $\Lambda_{A,B}(f) : C \to (A \Rightarrow B)$ for every $f : C \times A \to B$.

4.2 Wait-notify games for shared variable access

We model interleaved access to shared variables in wait-notify games [20], as we discussed earlier. The arena $\mathbf{WN}$ of wait and notify moves is defined as below.

- $\mathbf{WN} = ((M_{\mathbf{WN}}, <_{\mathbf{WN}}), \lambda_{\mathbf{WN}}, \vdash_{\mathbf{WN}})$ is the arena of wait-notify signals, where $M_{\mathbf{WN}} = \{\mathtt{W}, \mathtt{N}\}$, $\lambda_{\mathbf{WN}}(\mathtt{W}) = (\mathsf{O}, \mathsf{Q})$, $\lambda_{\mathbf{WN}}(\mathtt{N}) = (\mathsf{P}, \mathsf{A})$, $\vdash_{\mathbf{WN}} = \{(\mathtt{W}, \mathtt{W}), (\mathtt{W}, \mathtt{N})\}$, and and $\mathtt{W} <_{\mathbf{WN}} \mathtt{N}$.

Let $F : \mathcal{G} \to \mathcal{G}$ be the functor $\mathbf{WN} \Rightarrow (-)$ and (F, η, μ, t) be the canonical commutative strong monad, where each natural transformation is given the following game interpretation:

- The unit $\eta_A : A \to \mathbf{WN} \Rightarrow A$ is a trivial copycat between the two copies of arena A, with no witnesses of $\mathbf{WN}$ moves.
- The multiplication $\mu_A : \mathbf{WN}_1 \Rightarrow (\mathbf{WN}_2 \Rightarrow A) \to \mathbf{WN}_0 \Rightarrow A$ is a copycat in which (i) each opponent move of an arena A is immediately copied by the same move of the other A arena; (ii) each opponent $\mathtt{W}$ move of $\mathbf{WN}_1$ or $\mathbf{WN}_2$ is copied by a $\mathtt{W}$ move of $\mathbf{WN}_0$; (iii) each opponent $\mathtt{N}$ move of $\mathbf{WN}_0$ is copied by a $\mathtt{N}$ move of $\mathbf{WN}_1$ (resp., $\mathbf{WN}_2$) when the opponent $\mathtt{N}$ move is justified by a $\mathtt{W}$ move that copies a move of $\mathbf{WN}_1$ (resp., $\mathbf{WN}_2$).
- The tensorial strength $t_{A,B} : A \times (\mathbf{WN} \Rightarrow B) \to \mathbf{WN} \Rightarrow (A \times B)$ is the trivial copycat between the arenas $A \times (\mathbf{WN} \Rightarrow B)$ and $\mathbf{WN} \Rightarrow (A \times B)$.

The monadic structure in $\mathcal{G}$, in the usual way, gives rise to the Kleisli category $\mathcal{G}_F$, whose objects are the same as $\mathcal{G}$ and the homset $\mathcal{G}_F(A,B)$ is $\mathcal{G}(A,FB)$. We write $f : A \to B$ to mean f is in $\mathcal{G}_F(A,B)$.[2] The composition of two Kleisli arrows $f : A \to B$, $g : B \to C$, written $f \fatsemi g$, is the morphism $f; g^* : A \to FC$ (in $\mathcal{G}$), where $g^* : FB \to FC$ is the Kleisli extension of g. For each object A, $\mathcal{G}_F$ has the identity arrow $\mathbf{id}_A : A \to A = \eta_A$ and the terminal arrow $!_A : A \to \mathbf{1} = \Lambda_{\mathbf{WN}}(!_{A \times \mathbf{WN}})$.

The category $\mathcal{G}_F$ is also cartesian closed. For objects A and B, the exponential object is $A \Rightarrow B$ (the same as that of $\mathcal{G}$) and the evaluation map is given by $\mathbf{ev}_{A,B} : (A \Rightarrow B) \times A \to B = ev_{A,B}; \eta_B$. For each $f : C \times A \to B$, the currying isomorphism is given by $\boldsymbol{\Lambda}_A(f) : C \to (A \Rightarrow B) = \Lambda_A(f); \delta^{-1}_{A,B}$, where $\delta_{A,B} : F(A \Rightarrow B) \to A \Rightarrow FB$ is the trivial copycat strategy between the arenas $\mathbf{WN} \Rightarrow (A \Rightarrow B)$ and $A \Rightarrow (\mathbf{WN} \Rightarrow B)$. Given Kleisli arrows $f_1 : A \to B_1$, ..., $f_k : A \to B_k$, we write, as usual, $\langle f_1, \ldots, f_k \rangle : A \to B_1 \times \cdots \times B_k$ for pairing operations and $\boldsymbol{\pi}_i : B_1 \times \cdots \times B_k \to B_i$ $(1 \leq i \leq k)$ for projections. $\boldsymbol{\pi}_i$ may be instead written as $\boldsymbol{\pi}_{B_i}$ when there arises no confusion.

The categories $\mathcal{G}$ and $\mathcal{G}_F$ inherit some significant properties [10] from $\mathcal{C}$: They are CPO-enriched, where the underlying CPO is an algebraic CPO with the least element $\bot_A$ for every arena A, w.r.t. the ordering $\leq^\natural$. Furthermore, a strategy σ is a *compact* element in the CPO iff $T_{WO(\sigma)}$ is a finite set.

5 The Game Model and the Soundness

The game model is given in a quite standard way except that each IA_{par} term $\Gamma \vdash M : \mathtt{T}$ is interpreted by a Kleisli arrow in $\mathcal{G}_F$, denoted by $[\![\Gamma \vdash M : \mathtt{T}]\!] : [\![\Gamma]\!] \to [\![\mathtt{T}]\!]$, as given in Fig. 3 of Appendix B. (We may instead write $[\![\Gamma \vdash M]\!]$ or $[\![M]\!]$, unless ambiguity arises.) We notice that the strategy given to every term is player visible, player bracketing, and subquestion-affine.

The $\mathtt{W}$ and $\mathtt{N}$ moves in the Kleisli arrows model the wait and notify events that come into play when accessing shared variables: Each time a program tries to access a shared variable, it is forced to yield its execution to another running thread and *wait* until it is *notified* to resume execution after an arbitrary amount of delays. Thus the terms assign, deref, and cas are modeled by strategies whose every interaction with the **Var** arena is preceded by a sequence of moves $\mathtt{W}\cdot\mathtt{N}$, which are kept throughout a series of Kleisli compositions. Let us consider, for instance, a term $v : \mathtt{var} \vdash \mathsf{seq}\ M_1\ M_2$ where $M_1 = \mathsf{assign}\ v\ 1$ and $M_2 = \mathsf{assign}\ v\ 2$. The term $\mathsf{seq}\ M_1\ M_2$ is interpreted by the strategy $\langle [\![M_1]\!], [\![M_2]\!] \rangle \fatsemi seq_{\mathtt{com}} = \langle [\![M_1]\!], [\![M_2]\!] \rangle; seq^*_{\mathtt{com}}$, where the Kleisli extension $seq^*_{\mathtt{com}} : F'(\mathtt{com} \times \mathtt{com}) \to F\,\mathtt{com}$ has a trace $\mathtt{run}\cdot\mathtt{run}_1\cdot\mathtt{W}'\cdot\mathtt{W}\cdot\mathtt{N}\cdot\mathtt{N}'\cdot\mathtt{done}_1\cdot\mathtt{run}_2\cdot\mathtt{W}'\cdot\mathtt{W}\cdot\mathtt{N}\cdot\mathtt{N}'\cdot\mathtt{done}_2\cdot\mathtt{done}$ that augments the trace in $seq_{\mathtt{com}} : \mathtt{com}_1 \times \mathtt{com}_2 \to F\,\mathtt{com}$ with extra **WN** moves that copies each **WN** move associated to $\mathtt{com}_1$ or $\mathtt{com}_2$ to a **WN** move associated to $\mathtt{com}$. When this is composed with the strategies $[\![M_i]\!] : \mathtt{var} \to F'\mathtt{com}_i$ $(i = 1, 2)$ of subterms, each of which is specified by a trace $\mathtt{run}\cdot\mathtt{W}'\cdot\mathtt{N}'\cdot\mathtt{wr}_i\cdot\mathtt{ok}\cdot\mathtt{done}$, the extra moves $\mathtt{W}'$ and $\mathtt{N}'$ in both strategies are synchronized and hidden but the copies of them are kept intact in the composed trace as: $\mathtt{run}\cdot\mathtt{W}\cdot\mathtt{N}\cdot\mathtt{wr}_1\cdot\mathtt{ok}\cdot\mathtt{W}\cdot\mathtt{N}\cdot\mathtt{wr}_2\cdot\mathtt{ok}\cdot\mathtt{done}$.

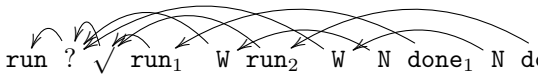
Several term constructors would worth further explanation. Thread extension $P \gg_j M$ makes use of the strategy $cont_j : (\mathbf{C}'_1 \times \cdots \times \mathbf{C}'_\zeta)_\bot \times \mathbf{C} \to (\mathbf{C}_1 \times \cdots \times \mathbf{C}_\zeta)_\bot$, which combines the strategy of the parallel command P with that of the sequential command M at the end of the j-th thread's execution. The interpretation of the parallel command $M_1 \| \cdots \| M_\zeta$ is given for its equivalent: $(\mathsf{skip} \parallel \cdots \parallel \mathsf{skip}) \gg_1 M_1 \gg_2 M_2 \cdots \gg_\zeta M_\zeta$. The

[2] Notice the uses of heavier symbols in the Kleisli category.

trivial parallel command $\mathsf{skip} \parallel \cdots \parallel \mathsf{skip}$ is interpreted by a strategy (the strategy *pskip* in Appendix B) whose well-opened trace set consist of the ζ copies of the neutral command skip in the arena $[\![\mathtt{par}]\!]$.

Since every IA_{par} term is interpreted by a Kleisli arrow in our game model, two terms that are comparable in an operational sense are not necessarily so by their corresponding strategies, due to the excess $\mathtt{W}$ and $\mathtt{N}$ moves. These excessive moves are necessary for modeling interleaved parallel execution but should be ignored when we discuss the observable behavior of programs.

Specifically for this purpose, we introduce a *scheduler* strategy $sched : F(\mathbf{C}_1 \times \cdots \times \mathbf{C}_\zeta)_\perp \to \mathbf{C}$ in $\mathcal{C}$, which is identified by the set of (well-opened longest) traces $\{\mathtt{run}\cdot?\cdot(\mathtt{W}\cdot\mathtt{N})^k\cdot\surd\cdot s\cdot \mathtt{done} \mid k \geq 0, s \in CPP\}$, where CPP is the set of *complete parallel plays*: we say a legal play $s \in F(\mathbf{C}_1 \times \cdots \times \mathbf{C}_\zeta)$ is a complete parallel play if s arbitrarily interleaves the plays $\mathtt{run}_j\cdot(\mathtt{W}\cdot\mathtt{N})^{k_j}\cdot\mathtt{done}_j$ $(1 \leq j \leq \zeta,\ k_j \geq 0)$ and both $\mathtt{run}_j$ and $\mathtt{done}_j$ have exactly a single occurrence in s for each j. We remark that the strategy *sched* in $\mathcal{C}$ but not in $\mathcal{G}$ or $\mathcal{C}_{vb}$, because it is neither player visible nor player bracketing. For example, *sched* has a trace



, where the second $\mathtt{N}$ is not justified by a move in its player view $\mathtt{run}\cdot?\cdot\surd\cdot\mathtt{run}_1\cdot\mathtt{done}_1\cdot\mathtt{N}$ and also the first $\mathtt{N}$ does not answer to the last unanswered question in its player view, i.e., the second $\mathtt{W}$. This indicates that the scheduler strategy is not definable in IA_{par} without using higher-order references and control primitives [2, 14, 15].

In what follows, the soundness property is discussed up to the so-called *intrinsic quotient*, a game model quotiented by contexts [3, 16, 10]. In most game models, the intrinsic quotient is only needed for establishing the full abstraction result, but the present paper needs it for establishing the soundness result as well, because of the above mentioned issue.

Using the scheduler strategy, we define the intrinsic quotient as follows: Given morphisms $f, g : A \to B$ in $\mathcal{G}_F$, we define $f \preccurlyeq g$ iff $(`f' \fatsemi h); sched \leq^\natural (`g' \fatsemi h); sched$ for every $h : (A \Rightarrow B) \to \mathbf{C}^\zeta{}_\perp$, where $`f' : \mathbf{1} \to A \Rightarrow B$ is the name of $f$, namely, $`f' = \Lambda_A(\pi_A \fatsemi f)$. We write $f \simeq g$ to mean $f \preccurlyeq g$ and $g \preccurlyeq f$. Let us write $\mathcal{G}_F/\simeq$ for the quotient category. $\mathcal{G}_F/\simeq$ is cartesian closed and is also rational, which is a sufficient condition for modeling recursions in Algol-like languages [16, 10]. We write $\mathcal{E}[\![M]\!] = [[\![M]\!]]_\simeq$, the *extensional* interpretation of the term M in $\mathcal{G}_F/\simeq$.

The soundness property follows from the consistency and adequacy.

▶ Proposition 1 (consistency). Suppose M is a closed term of type $\mathtt{par}$. If $M\Downarrow^{\mathrm{may}}$, then $\mathtt{run}\cdot\mathtt{done} \in T_{[\![M]\!];sched}$; If $M\Downarrow^{\mathrm{must}}$, then $\mathtt{run} \notin D_{[\![M]\!];sched}$.

▶ Proposition 2 (adequacy). Suppose M is a closed term of type $\mathtt{par}$. If $\mathtt{run}\cdot\mathtt{done} \in T_{[\![M]\!];sched}$, then $M\Downarrow^{\mathrm{may}}$; If $\mathtt{run} \notin D_{[\![M]\!];sched}$, then $M\Downarrow^{\mathrm{must}}$.

▶ **Theorem 3** (soundness). *If M and N are closed terms of type $\mathtt{T}$ and $\mathcal{E}[\![M]\!] \lesssim \mathcal{E}[\![N]\!]$, then $M \lesssim_{\mathrm{m\&m}} N$.*

6 Definability and Full Abstraction

As Harmer did for his nondeterministic Algol-like language [10], we can also show the full abstraction result for our parallel language IA_{par} by combining two factorizations and a definability result. However, we need to make his techniques more precise, due to extra intricacies involved in parallelism, most notably race conditions on shared variables.

6.1 Reliable factorization

We first factor out possible nondeterminism as the oracle strategy $\mathit{oracle} : 1 \to \mathbf{N}'\Rightarrow\mathbf{N}$, whose (longest well-opened) traces are $\{\mathsf{qq}'0n \mid n \geq 0\} \cup \{\mathsf{q}(\mathsf{q}'m)^{n+1}n \mid m > 0 \wedge 0 \leq n \leq m\}$ and the (sole) well-opened interesting divergence is $\mathsf{qq}'0$. Given a constant 0, the *oracle* strategy nondeterministically diverges or converges to produce an arbitrary number; Given a positive constant m, it never diverges but reliably returns a number not greater than m. The strategy *oracle* is definable by the term $\lambda x^{\mathtt{nat}}.\mathsf{if0}\ x\ \mathsf{then}\ \mho\ \mathsf{else}\ \mho' x$, where $\mho = \mathsf{fix}^{\mathtt{nat}}(\lambda x.0\ \mathsf{or}\ (x+1))$ and $\mho' = \mathsf{fix}^{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathsf{if0}\ x\ \mathsf{then}\ 0\ \mathsf{else}\ (0\ \mathsf{or}\ f(x-1)+1))$.

Assuming a fixed coding function code_A that assigns a unique natural number to each distinct legal play in arena A, we separate out the reliable part of a strategy σ as follows.

▶ Proposition 4. If $\sigma : 1 \to A$ is a compact strategy in $\mathcal{G}_F$, then there exists a compact, reliable strategy $\mathit{det}(\sigma) : (\mathbf{N}' \Rightarrow \mathbf{N}) \to A$ such that $\sigma =^{\natural} \mathit{oracle}; \mathrm{det}(\sigma)$.

Proof. The proof basically follows that of Proposition 4.6.2 in [10], but we must be careful that our oracle strategy is different from the one employed in Harmer's original proof, in that Harmer's oracle strategy makes use of local variables whereas ours doesn't. The effect is that ours makes duplicated copies of the argument, witnessed as the subsequence $(\mathsf{q}'m)^{n+1}$ in the trace. This does not matter for factorization, though. Wherever Harmer factors out a finitely branching nondeterminism at $sa \in \mathit{dom}(\sigma)$ by a trace $\cdots a{\cdot}\mathsf{q}{\cdot}\mathsf{q}'m{\cdot}j{\cdot}b$ in $\mathit{det}(\sigma)$, we do it by a bit longer trace $\cdots a{\cdot}\mathsf{q}{\cdot}(\mathsf{q}'{\cdot}m)^{j+1}{\cdot}j{\cdot}b$. Factorization at diverging points is similarly done.

Further, in order to have the factorization process closed under the subquestion-affine property, we have to make Harmer's factorization more precise so that any oracle moves are not inserted where the subsequent player move is uniquely determined. ◀

Our preference to the oracle that is definable without local variables is due to the extra **WN** moves to be introduced otherwise. Harmer's oracle strategy would also work in our game model modulo $\simeq$. We will deal with $\simeq$-quotients in the next step, where we can work with reliable strategies, without being bothered with divergence or nondeterminism.

6.2 Innocent factorization

The second step toward full abstraction is *innocent factorization* [3], which separates an innocent (history-free, in other words) strategy from history-sensitive one. A strategy σ is called *innocent* if σ is reliable and player visible and for every $sab \in T_\sigma$ and $t \in L_\sigma^{\mathrm{odd}}$ such that $\mathsf{V}(sa) = \mathsf{V}(t)$, $tb \in T_\sigma$ and $\mathsf{V}(sa)b = \mathsf{V}(t)b$, where b is justified by the matching move in the player view. An innocent strategy σ is uniquely identified by its *view function*, $\mathsf{fun}(\sigma) = \{\mathsf{V}(s) \mid s \in T_\sigma\}$.

The idea in innocent factorization of a strategy is to determine its behavior up to, instead of the trace that it has played so far, a record of execution history kept in a variable. The standard innocent factorization procedure builds an innocent strategy, whose view function contains a player view of the form $s'{\cdot}a{\cdot}\mathtt{rd}{\cdot}\mathsf{code}_A(s){\cdot}\mathtt{wr}_{\mathsf{code}_A(s\cdot a\cdot b)}{\cdot}\mathtt{ok}{\cdot}b$, for every $sab \in T_\sigma$ and the player view s' of the factorization of s. This construction violates, however, the subquestion-affine conditions (sq2) and (sq3). Thus we need a factorization procedure with improved precision.

Let σ be a reliable and player visible strategy. A player view $\mathsf{V}(sab)$ at an opponent move a of σ is called *locally innocent*, if it holds that $\mathsf{V}(sab) = \mathsf{V}(s'ab')$ for every $s'ab' \in T_\sigma$ satisfying $\mathsf{V}(sa) = \mathsf{V}(s'a)$. Wherever a player view at an opponent move is locally innocent,

as the next player move that follows is uniquely determined by the player view, we can skip and postpone the history update until we reach a point that is not locally innocent.

More fundamentally, we need another change in the factorization procedure, in order to avoid possible race conditions caused by parallel accesses to the shared variable. Suppose we have a strategy in arena $F[\![\mathtt{par}]\!]$ that has a factorized view function containing plays like $?\cdot\surd\cdot\mathtt{run}_j\cdot s_j\cdot\mathtt{done}_j$ $(1 \leq j \leq \zeta)$, where each s_j contains successive moves $a_j\cdot\mathtt{W}\cdot\mathtt{N}\cdot\mathtt{rd}\cdot m_j\cdot\mathtt{W}\cdot\mathtt{N}\cdot\mathtt{wr}_{n_j}\cdot\mathtt{ok}\cdot p_j$. (Remember that every move representing a variable access operation is preceded by $\mathtt{W}\cdot\mathtt{N}$ in our game modeling.) Then the factorized strategy, which arbitrarily interleaves the plays in the view function, contains a play that competes for the shared variable, e.g., $?\cdot\surd\cdots a_1\cdot\mathtt{W}\cdot\mathtt{N}\cdot\mathtt{rd}\cdot m_1\cdot\mathtt{W}\cdot a_2\cdot\mathtt{W}\cdot\mathtt{N}\cdot\mathtt{rd}\cdot m_2\cdot\mathtt{W}\cdot\mathtt{N}\cdot\mathtt{wr}_{n_1}\cdot\mathtt{ok}\cdot p_1\cdot\mathtt{N}\cdot\mathtt{wr}_{n_2}\cdot\mathtt{ok}\cdot p_2$. At the end of the play, thread 1 has already reached its player move p_1 but it is not recorded in the storage, overwritten by the subsequent moves of thread 2.

Here we achieve innocent factorization by a *thread-safe programming* on game plays. We make use of the atomic read-modify-write ability provided by the CAS operation in order to avoid race conditions, making each history update inseparable from the completion of a single computation step. To do this, we may just spin over the variable storing the history, repeatedly trying to atomically update the variable by CAS until successful. However, the spin lock mechanism is too naïve as a means for factorization, as it may introduce an undesired divergence when the update fails forever. Instead, we repeatedly try to update, but bounded by a sufficiently large number of times. Such a bound exists, if we assume a compact strategy, i.e., a strategy whose well-opened traces are finite, because there can be at most a bounded number of successful writes throughout the entire traces, meaning that each repeated execution of update by CAS is guaranteed to succeed within the bound.

To sum up, given a compact, player visible, and reliable strategy σ on an arena $A \to FB$ and also a positive number d, we construct an innocent strategy $\mathsf{inn}_d(\sigma)$, identified by the view function $\mathsf{fun}(\mathsf{inn}_d(\sigma))$ that contains the following player views for every $s\cdot a\cdot b \in T_\sigma$:

- When the player view $\mathsf{V}(sa)$ is locally innocent, we just add player views of the form $s'\cdot a\cdot b$ to the view function, where s' is a factorization obtained from s;
- Otherwise, for any partial function ν on natural numbers such that $\nu(n) = m$ iff $n = \mathsf{code}(s')$, $m = \mathsf{code}(s'ab)$ for some $s'ab \in T_\sigma$ satisfying $\mathsf{V}(s'a) = \mathsf{V}(sa)$, where s' is a factorization obtained from s. Then, we add (every even-length prefixes of) player views of the following form $s'\cdot a\cdot(\mathtt{W}\cdot\mathtt{N})^d\cdot\mathtt{rd}\cdot\mathsf{code}(s)\cdot u_{c_0,c_1}\cdot u_{c_1,c_2}\cdots u_{c_i,c_{i+1}}\cdots u_{c_k,\nu(c_k)}\cdot(\mathtt{W}\cdot\mathtt{N})^d\cdot b$, where $u_{m,m'}$ is a sequence of moves $(\mathtt{W}\cdot\mathtt{N})^d\cdot\mathtt{cas}_{m,\nu(m)}\cdot m'$, $c_0 = \mathsf{code}(s)$, and for every i $(1 \leq i \leq k)$, $c_i \neq \nu(c_{i-1})$ and there exists $ta'b' \in T_{WO(\sigma)}$ satisfying $\mathsf{code}(t) = c_{i-1}$ and $\mathsf{code}(ta'b') = c_i$. k is bounded by a number determined by each given strategy, i.e., the length of the longest well-opened trace.

We call the strategy $\mathsf{inn}_d(\sigma)$ a *d-delayed* innocent strategy, as every $sa \in dom(\sigma)$ must be followed by at least d successive sequences of $\mathtt{W}\cdot\mathtt{N}$ moves, unless the player view $\mathsf{V}(sa)$ uniquely determines the next player move. Formally, an innocent strategy σ is called *d-delayed* iff for every trace $sab \in T_\sigma$, either b is $\mathtt{W}$, the player view $\mathsf{V}(sa)$ is locally innocent, or $\mathsf{V}(sab) = t\cdot(\mathtt{W}\cdot\mathtt{N})^d\cdot b$ for some t. The extra d-delays, not just a single delay, are needed for obtaining the definability result (Section 6.3).

▶ Proposition 5. Let $f : 1 \to A$ be a compact reliable strategy in $\mathcal{G}_F$. Then, for every $d \geq 1$, there is a compact d-delayed innocent strategy $\mathsf{inn}_d(f) : \mathbf{Var} \to A$ in $\mathcal{G}_F$ such that $(cell_{\mathsf{code}(\varepsilon)}; \eta_{\mathbf{Var}}) \S \mathsf{inn}_d(f) \simeq f$.

We notice that factorization is modulo $\simeq$, which ignores the extra **WN** moves introduced during the factorization procedure.

6.3 Definability

The last step toward the full abstraction is the definability: We are obliged to show that every strategy in a suitable strategy subclass is definable by an IA_{par} term.

In the construction of IA_{par} terms below, we need a general conditional expression $\mathsf{case}_{\ell,\mathtt{T}} : \mathtt{nat} \times \mathtt{T}^{\ell} \to \mathtt{T}$, where $\ell \geq 0$ and $\mathtt{T}$ is either $\mathtt{nat}$, $\mathtt{com}$, or $\mathtt{par}$. This conditional expression is an operationally conservative extension to the language, as it is defined by the following IA_{par} term:

$$\begin{aligned}&\lambda x x_1 \cdots x_\ell.\big(\mathsf{newvar}\ v = 0\ \mathsf{in}\\ &\quad \mathsf{seq}\ (\mathsf{assign}\ v\ x)\ \ (\mathsf{if0}\ (\mathsf{deref}\ v)\ \mathsf{then}\ x_1\ \mathsf{else}\\ &\qquad\qquad (\mathsf{if0}\ (\mathsf{deref}\ v) - 1\ \mathsf{then}\ x_2\ \mathsf{else}\ \cdots\ (\mathsf{if0}\ (\mathsf{deref}\ v) - \ell\ \mathsf{then}\ x_\ell\ \mathsf{else}\ \Omega)\cdots)))\big)\end{aligned}$$

where Ω is the divergence at type $\mathtt{T}$.

Due to the variable access operations being involved, however, the game interpretation of $\mathsf{case}_{\ell,\mathtt{T}}$ contains extra **WN** moves in its traces: Given a natural number m less than ℓ as its first argument, $\mathsf{case}_{k,\mathtt{T}}$ has $m+1$ accesses to the local variable v, leaving $(\mathtt{W}{\cdot}\mathtt{N})^{m+1}$ in its trace as the footprint. The extra **WN** moves are canceled by the aforementioned d-delayed innocent strategy $\mathsf{inn}_d(\sigma)$, where d is a number larger than the maximum number of possible opponent moves that immediately follow the same player view, i.e., $\max\{\#\{ta'b' \mid ta'b' \in \mathsf{fun}(\sigma)\} \mid tab \in \mathsf{fun}(\sigma)\}$.

In case some trace in the innocent strategy contains excess $\mathtt{W}{\cdot}\mathtt{N}$ sequences than those to be canceled out, we let the term $\mathsf{newvar}\ v = 0\ \mathsf{in}\ \mathsf{assign}\ v\ 0$, denoted by touch hereafter, cancel out the remaining ones. The term touch has the trace $\mathtt{run}{\cdot}\mathtt{W}{\cdot}\mathtt{N}{\cdot}\mathtt{done}$, which witnesses a single $\mathtt{W}{\cdot}\mathtt{N}$ sequence as the footprint of a single access to the local variable v. Further, for brevity, we will write touch^k for the command $\overbrace{\mathsf{seq}\ \mathsf{touch}\ (\mathsf{seq}\ \mathsf{touch}\ \cdots (\mathsf{seq}\ \mathsf{touch}}^{k\ \text{times}}\ \mathsf{skip})\cdots)$.

Let us show that any compact, d-delayed innocent, player bracketing strategy $\sigma : [\![\mathtt{T}_1]\!] \times \cdots \times [\![\mathtt{T}_n]\!] \to [\![\mathtt{T}]\!]$, where d is a sufficiently large number as analyzed above, is definable by an IA_{par} term of the form $x_1 : \mathtt{T}_1, \cdots, x_n : \mathtt{T}_n \vdash M : \mathtt{T}$. We construct such a term by induction on the size of view function $\mathsf{fun}(\sigma)$, where the base case is $\mathsf{fun}(\sigma) = \{\varepsilon\}$ that is trivially definable by Ω. Below we will mostly concern $\mathtt{par}$ types and the extra **WN** moves. (We will omit some details not concerning these extra complications. See [10] for the missing details.)

We may assume that $\mathtt{T}$ is a base type. If otherwise, say, $\mathtt{T} = \mathtt{T}'_1 \to \cdots \to \mathtt{T}'_m \to \mathtt{T}'$ with $\mathtt{T}'$ being a base type, we instead work on the isomorphic strategy on $[\![\mathtt{T}_1]\!] \times \cdots \times [\![\mathtt{T}_n]\!] \times [\![\mathtt{T}'_1]\!] \times \cdots \times [\![\mathtt{T}'_m]\!] \to [\![\mathtt{T}']\!]$.

Here we consider solely the case $\mathtt{T} = \mathtt{par}$. The remaining cases $\mathtt{T} = \mathtt{nat}$, $\mathtt{T} = \mathtt{com}$, and $\mathtt{T} = \mathtt{var}$ are shown in almost the same way, except that, when $\mathtt{T} = \mathtt{var}$, we need to construct a *bad variable* $\mathsf{mkvar}\ M\ N\ L$ with L being the term that simulates a CAS-like operation.

Suppose that $?{\cdot}(\mathtt{W}{\cdot}\mathtt{N})^d{\cdot}\mathtt{q}_j \in \mathsf{fun}(\sigma)$, where $\mathtt{q}_j$ is a move from the arena $\mathbf{N}$ in particular when $\mathtt{T}_j$ has the form $\mathtt{T}'_1 \to \cdots \to \mathtt{T}'_m \to \mathtt{nat}$, with d being the sufficiently large number. We derive a class of substrategies from σ that separate out the threads of play in $[\![\mathtt{T}_j]\!]$ induced by the move $\mathtt{q}_j$, as follows. Let $\sigma_i : [\![\mathtt{T}_1]\!] \times \cdots \times [\![\mathtt{T}_n]\!] \to [\![\mathtt{par}]\!]$ $(i \geq 0)$ and $\sigma'_h : [\![\mathtt{T}_1]\!] \times \cdots \times [\![\mathtt{T}_n]\!] \to [\![\mathtt{T}'_h]\!]$ $(1 \leq h \leq m)$ be substrategies identified by view functions $\mathsf{fun}(\sigma_i) = \{?{\cdot}s \mid ?{\cdot}(\mathtt{W}{\cdot}\mathtt{N})^d{\cdot}\mathtt{q}_j{\cdot}i{\cdot}s \in \mathsf{fun}(\sigma)\}$ and $\mathsf{fun}(\langle\!\langle \sigma'_1, \cdots, \sigma'_m \rangle\!\rangle) = \{s \mid ?{\cdot}(\mathtt{W}{\cdot}\mathtt{N})^d{\cdot}\mathtt{q}_j{\cdot}s \in \mathsf{fun}(\sigma),\ s \text{ contains no answer to } \mathtt{q}_j\}$, respectively. By the compactness, there exists a natural number ℓ such that $\mathsf{fun}(\sigma_i) = \{\varepsilon\}$ for every $i \geq \ell$. Since each substrategy is again a compact, d-delayed innocent, player bracketing strategy that is strictly smaller than $\mathsf{fun}(\sigma)$,

by induction hypothesis, we have terms M_i and M'_h defining σ_i and σ'_h, respectively, for each i and h. Then the strategy σ is definable by the term:

$$\mathsf{case}_{\ell,\mathtt{par}}\ (x_j M'_1 \cdots M'_m)\ \ (\mathsf{seq\ touch}^{d-1}\ M_0)\ (\mathsf{seq\ touch}^{d-2}\ M_1)\ \cdots\ (\mathsf{seq\ touch}^{d-\ell}\ M_{\ell-1}).$$

The case the right-most base type in $\mathtt{T}_j$ being `com` or `par` is similarly defined by using seq in place of case; The case for `var` is also similar, except that we need to additionally deal with CAS operations.

When $?{\cdot}\mathtt{W}{\cdot}\mathtt{N}{\cdot}p \in \mathsf{fun}(\sigma)$ with p not being a move from $[\![T_j]\!]$'s, the strategy σ is simply definable by the term seq touch M, where M is the defining term of the substrategy σ' specified by $\mathsf{fun}(\sigma_i) = \{?{\cdot}p{\cdot}s \mid ?{\cdot}\mathtt{W}{\cdot}\mathtt{N}{\cdot}p{\cdot}s \in \mathsf{fun}(\sigma)\}$. When $?{\cdot}\surd \in \mathsf{fun}(\sigma)$, we define substrategies $\sigma_j : [\![\mathtt{T}_1]\!] \times \cdots \times [\![\mathtt{T}_n]\!] \to \mathbf{C}_j$ $(1 \le j \le \zeta)$ by view functions $\mathsf{fun}(\sigma_j) = \{\mathtt{run}_j{\cdot}t \mid ?{\cdot}\surd{\cdot}\mathtt{run}_j{\cdot}t \in \mathsf{fun}(\sigma)\}$. By induction hypothesis, we have a term M_i that defines σ_i for each i. Then the strategy σ is definable by the term $M_1 \| \cdots \| M_\zeta$.

The remaining case is that there exists $?{\cdot}?'{\cdot}\surd'{\cdot}\surd \in \mathsf{fun}(\sigma)$ where $\surd'$ is a subquestioning answer, which is derived from the $\mathtt{par}'$ type in $\mathtt{T}_j$ of the form $\mathtt{T}'_1 \to \cdots \to \mathtt{T}'_m \to \mathtt{par}'$. Likewise above, we derive substrategies from σ. Let $\sigma_i : [\![\mathtt{T}_1]\!] \times \cdots \times [\![\mathtt{T}_n]\!] \to \mathbf{C}$ $(1 \le i \le \zeta)$ and $\sigma'_h : [\![\mathtt{T}_1]\!] \times \cdots \times [\![\mathtt{T}_n]\!] \to [\![\mathtt{T}'_h]\!]$ $(1 \le h \le m)$ be substrategies identified by view functions $\mathsf{fun}(\sigma_i) = \{\mathtt{run}_i{\cdot}s' \mid ?{\cdot}?'{\cdot}\surd'{\cdot}\surd{\cdot}\mathtt{run}_i{\cdot}\mathtt{run}'_i{\cdot}\mathtt{done}'_i{\cdot}s' \in \mathsf{fun}(\sigma)\}$ and $\mathsf{fun}(\langle\sigma'_1, \cdots, \sigma'_m\rangle) = \{s' \mid ?{\cdot}?'{\cdot}\surd'{\cdot}\surd{\cdot}\mathtt{run}_j{\cdot}\mathtt{run}'_j{\cdot}s' \in \mathsf{fun}(\sigma),\ s' \text{ contains no answer to } \mathtt{run}'_j\}$, respectively. Again, by induction hypothesis, we have terms M_i, and M'_j defining σ_i, and σ'_j, respectively, for each i and j. Then σ is definable by the term $(x_j M'_1 \cdots M'_m) \gg_1 M_1 \gg_2 M_2 \cdots \gg_\zeta M_\zeta$.

▶ **Theorem 6** (definability). *Let $\sigma : [\![\Gamma]\!] \to [\![\mathtt{T}]\!]$ be a compact and player bracketing strategy in $\mathcal{G}_F$ (henceforth, it is also a player visible, subquestion-affine strategy.) Then, σ is definable in IA_{par}.*

▶ **Theorem 7** (Full abstraction). *Suppose M and N are closed terms of type $\mathtt{T}$. Then, $\mathcal{E}[\![M]\!] \lesssim \mathcal{E}[\![N]\!]$ iff $M \lesssim_{\mathrm{m\&m}} N$.*

7 Conclusion and Future Work

We have developed a full abstract game semantics for an Algol-like parallel language with a non-blocking synchronization primitive CAS. Elaborating on the wait-notify game [20] in the framework of Harmer's game model for nondeterminism [10], we exploited the computational structure of the Kleisli category induced by the extra `W` and `N` moves and thereby established the full abstraction, in which we made a significant use of CAS operations.

Based on the full abstraction result, it would be beneficial to find a subset language whose observational equality is decidable, as it would provide a mechanized method for equivalence checking, as done in [6]. Further, though the present paper is limited to an unfair thread scheduling policy specified by the particular strategy *sched*, more flexible variants of scheduling policy (say, the Round-robin scheduling) would worth investigating. This requires a game model for fair nondeterminism, which would provide a deeper insight on parallel computation.

Acknowledgment. I would like to thank Shin-ya Katsumata for his valuable comments and suggestions on an early draft of the paper. I am also grateful for reviewers for their helpful comments. This work was supported by KAKENHI 24500014.

References

1 Samson Abramsky. Game semantics of idealized parallel Algol. Lecture given at the Newton Institute, 1995.

2 Samson Abramsky, Kohei Honda, and Guy McCusker. A fully abstract game semantics for general references. In *13th Annual IEEE Symposium on Logic in Computer Science*, pages 334–344, 1998.

3 Samson Abramsky and Guy McCusker. Linearity, sharing and state: A fully abstract game semantics for idealized Algol with active expressions. In P. W. O'Hearn and R. D. Tennent, editors, *Algol-like Languages*, volume 2 of *Progress in Theoretical Computer Science*, pages 297–329. Birkhäuser, 1997.

4 Samson Abramsky and Guy McCusker. Game semantics. In H. Schwichtenberg and U. Berger, editors, *Computational Logic: Proceedings of the 1997 Marktoberdorf Summer School*, pages 1–56. Springer-Verlag, 1999.

5 Stephen Brookes. Full abstraction for a shared variable parallel language. In *Proceedings of 8th Annual IEEE Symposium on Logic in Computer Science*, pages 98–109, 1993.

6 Dan R. Ghica and Guy McCusker. The regular-language semantics of second-order idealized ALGOL. *Theoretical Computer Science*, 309(1–3):469–502, 2003.

7 Dan R. Ghica and Andrzej S. Murawski. Angelic semantics of fine-grained concurrency. *Annals of Pure and Applied Logic*, 151(2-3):89–114, 2008.

8 Dan R. Ghica, Andrzej S. Murawski, and C.-H. Luke Ong. Syntactic control of concurrency. *Theoretical Computer Science*, 350(2-3):234–251, 2006.

9 Russel Harmer and Guy McCusker. A fully abstract game semantics for finite nondeterminism. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*, pages 422–430, 1999.

10 Russell Harmer. *Games and Full Abstraction for Nondeterministic Languages*. PhD thesis, University of London, 1999.

11 Maurice Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1), 1991.

12 Maurice Herlihy and Nir Shavit. *The Art of Multiprocessor Programming*. Morgan Kaufmann, 2008.

13 J. M. E. Hyland and C.-H. Luke Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000.

14 James Laird. Full abstraction for functional languages with control. In *12th Annual IEEE Symposium on Logic in Computer Science*, pages 58–67, 1997.

15 James Laird. A fully abstract game semantics of local exceptions. In *16th Annual IEEE Symposium on Logic in Computer Science*, pages 105–114, 2001.

16 Guy McCusker. *Games and Full Abstraction for a Functional Metalanguage with Recursive Types*. Distinguished Dissertations. Springer, 1998.

17 Gordon D. Plotkin. A powerdomain construction. In *SIAM J. Comput.*, number 5 in 3, pages 452–487, 1976.

18 John C. Reynolds. The essence of Algol. In *Proceedings of the 1981 International Symposium on Algorithmic Languages*, pages 345–372. North-Holland, 1981.

19 David A. Schmidt. *Denotational Semantics: A Methodology for Language Development*. 1986.

20 Keisuke Watanabe and Susumu Nishimura. May&must-equivalence of shared variable parallel programs in game semantics. *Information Processing Society of Japan Transactions on Programming (PRO)*, 5(4):17–26, 2012. `http://jlc.jst.go.jp/DN/JST.JSTAGE/ipsjtrans/5.167`.

$$\frac{}{\Gamma, x : \mathtt{T} \vdash x : \mathtt{T}} \qquad \frac{}{\Gamma \vdash n : \mathtt{nat}} \qquad \frac{\Gamma \vdash M : \mathtt{nat} \quad \Gamma \vdash N : \mathtt{nat}}{\Gamma \vdash M \ \star \ N : \mathtt{nat}}$$

$$\frac{\Gamma, x : \mathtt{T}' \vdash M : \mathtt{T}}{\Gamma \vdash \lambda x^{\mathtt{T}'}.M : \mathtt{T}' \to \mathtt{T}} \qquad \frac{\Gamma \vdash M_1 : \mathtt{T}' \to \mathtt{T} \quad \Gamma \vdash M_2 : \mathtt{T}'}{\Gamma \vdash M_1 M_2 : \mathtt{T}} \qquad \frac{\Gamma \vdash M : \mathtt{T} \to \mathtt{T}}{\Gamma \vdash \mathsf{fix}_{\mathtt{T}}\ M : \mathtt{T}}$$

$$\frac{}{\Gamma \vdash \mathsf{skip} : \mathtt{com}} \qquad \frac{\Gamma \vdash M_1 : \mathtt{com} \quad \Gamma \vdash M_2 : \mathtt{T} \quad \mathtt{T} \in \{\mathtt{nat}, \mathtt{com}, \mathtt{par}\}}{\Gamma \vdash \mathsf{seq}\ M_1\ M_2 : \mathtt{T}}$$

$$\frac{\Gamma \vdash M : \mathtt{nat} \quad \Gamma \vdash M_1 : \mathtt{T} \quad \Gamma \vdash M_2 : \mathtt{T} \quad \mathtt{T} \in \{\mathtt{nat}, \mathtt{com}, \mathtt{par}\}}{\Gamma \vdash \mathsf{if0}\ M\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \mathtt{T}}$$

$$\frac{\Gamma \vdash M : \mathtt{var} \quad \Gamma \vdash N : \mathtt{nat}}{\Gamma \vdash \mathsf{assign}\ M\ N : \mathtt{com}} \qquad \frac{\Gamma \vdash M : \mathtt{var}}{\Gamma \vdash \mathsf{deref}\ M : \mathtt{nat}}$$

$$\frac{\Gamma \vdash L : \mathtt{var} \quad \Gamma \vdash M : \mathtt{nat} \quad \Gamma \vdash N : \mathtt{nat}}{\Gamma \vdash \mathsf{cas}\ L\ M\ N : \mathtt{nat}}$$

$$\frac{\Gamma \vdash M : \mathtt{nat} \to \mathtt{com} \quad \Gamma \vdash N : \mathtt{nat} \quad \Gamma \vdash L : \mathtt{nat} \to \mathtt{nat} \to \mathtt{nat}}{\Gamma \vdash \mathsf{mkvar}\ M\ N\ L : \mathtt{var}}$$

$$\frac{\Gamma, v : \mathtt{var} \vdash M : \mathtt{T} \quad \mathtt{T} \in \{\mathtt{com}, \mathtt{par}\}}{\Gamma \vdash \mathsf{newvar}\ v = n\ \mathsf{in}\ M : \mathtt{T}} \qquad \frac{\Gamma \vdash M_1 : \mathtt{nat} \quad \Gamma \vdash M_2 : \mathtt{nat}}{\Gamma \vdash M_1\ \mathsf{or}\ M_2 : \mathtt{nat}}$$

$$\frac{\Gamma \vdash M_1 : \mathtt{com} \quad \cdots \quad \Gamma \vdash M_\zeta : \mathtt{com}}{\Gamma \vdash M_1 \| \cdots \| M_\zeta : \mathtt{par}} \qquad \frac{\Gamma \vdash P : \mathtt{par} \quad \Gamma \vdash M : \mathtt{com}}{\Gamma \vdash P \gg_j M : \mathtt{par}}$$

Figure 1 Typing rules.

A Typing Rules and Operational Semantics for IA_{par}

The typing rules of IA_{par} are given in Figure 1.

Let us write $\langle s \mid v \mapsto m \rangle$ for a store that updates s to give the natural number m at v. Let us also write $M[N/x]$ for the term substitution, which replaces every free occurrence of variable x in M with N, assuming the usual variable convention.

The reduction rules are given in Figure 2, relative to *evaluation context*. An evaluation context E is a term with a single hole $[\,]$ defined by the following grammar:

$$E ::= [\,] \mid E \ \star \ M \mid n \ \star \ E \mid EM \mid \mathsf{seq}\ E\ M \mid \mathsf{if0}\ E\ \mathsf{then}\ M\ \mathsf{else}\ N \mid E \gg_j M$$
$$\mid M_1 \| \cdots \| M_{j-1} \| E \| M_{j+1} \| \cdots \| M_\zeta.$$

We write $E[M]$ for the term obtained by filling the hole in E with term M.

Here we notice that there are three term formations whose evaluation can be nondeterministic: M_1 or M_2, the erratic binary choice on natural numbers; $M_1 \parallel \cdots \parallel M_\zeta$, the choice of a single command out of ζ simultaneously running sequential commands; $(M_1 \parallel \cdots \parallel M_\zeta) \gg_j M$, which either extends the j-the thread's execution by the command M or executes just one out of ζ parallel threads a single step forward. The last nondeterminism, nevertheless, is neutral to the observational property, that is, the different reduced terms can conflue even after further reductions. Indeed, they will be given the identical game interpretation.

B Game Interpretation of Terms

The arenas corresponding to common base types found in Algol-like languages are given below. (For notational convenience, let us we write $\overline{\lambda}_A$ for the labeling function whose

$$\langle E[m \star n], s\rangle \longrightarrow \langle E[n'], s\rangle, \text{ where } n' = m \star n \qquad \langle E[\left(\lambda x^{\mathsf{T}}.M\right) N], s\rangle \longrightarrow \langle E[M[N/x]], s\rangle$$

$$\langle E[\mathsf{if0}\ 0\ \mathsf{then}\ M\ \mathsf{else}\ N], s\rangle \longrightarrow \langle E[M], s\rangle \qquad \langle E[\mathsf{if0}\ n+1\ \mathsf{then}\ M\ \mathsf{else}\ N], s\rangle \longrightarrow \langle E[N], s\rangle$$

$$\langle E[\mathsf{fix}\ M], s\rangle \longrightarrow \langle E[M(\mathsf{fix}\ M)], s\rangle \qquad \langle E[\mathsf{seq}\ \mathsf{skip}\ M], s\rangle \longrightarrow \langle E[M], s\rangle$$

$$\langle E[\mathsf{assign}\ v\ n, s)] \longrightarrow \langle E[\mathsf{skip}], \langle s \mid v \mapsto n\rangle\rangle \qquad \langle E[\mathsf{deref}\ v], s\rangle \longrightarrow \langle E[s(v)], s\rangle$$

$$\langle E[\mathsf{cas}\ v\ m\ n], s\rangle \longrightarrow \langle E[n], \langle s \mid v \mapsto n\rangle\rangle, \text{ where } s(v) = m$$

$$\langle E[\mathsf{cas}\ v\ m\ n], s\rangle \longrightarrow \langle E[s(v)], s\rangle, \text{ where } s(v) \neq m$$

$$\langle E[\mathsf{assign}\ (\mathsf{mkvar}\ M\ N\ L)\ n], s\rangle \longrightarrow \langle E[Mn], s\rangle \qquad \langle E[\mathsf{deref}\ (\mathsf{mkvar}\ M\ N\ L)], s\rangle \longrightarrow \langle E[N], s\rangle$$

$$\langle E[\mathsf{cas}\ (\mathsf{mkvar}\ M\ N\ L)\ m\ n], s\rangle \longrightarrow \langle E[Lmn], s\rangle$$

$$\langle E[M\ \mathsf{or}\ N], s\rangle \longrightarrow \langle E[M], s\rangle \qquad \langle E[M\ \mathsf{or}\ N], s\rangle \longrightarrow \langle E[N], s\rangle$$

$$\langle E[\mathsf{newvar}\ v = n\ \mathsf{in}\ \mathsf{skip}], s\rangle \longrightarrow \langle E[\mathsf{skip}], s\rangle$$

$$\langle E[\mathsf{newvar}\ v = n\ \mathsf{in}\ (\mathsf{skip} \| \cdots \| \mathsf{skip})], s\rangle \longrightarrow \langle E[\mathsf{skip} \| \cdots \| \mathsf{skip}], s\rangle$$

$$\frac{\langle M, \langle s \mid v \mapsto n\rangle\rangle \longrightarrow \langle M', s'\rangle}{\langle E[\mathsf{newvar}\ v = n\ \mathsf{in}\ M], s\rangle \longrightarrow \langle E[\mathsf{newvar}\ v = s'(v)\ \mathsf{in}\ M'], \langle s' \mid v \mapsto s(v)\rangle\rangle}$$

$$\langle E[(\mathsf{newvar}\ v = n\ \mathsf{in}\ P) \gg_j M], s\rangle \longrightarrow \langle E[\mathsf{newvar}\ v = n\ \mathsf{in}\ (P \gg_j M)], s\rangle$$

$$\langle E[(M_1 \| \cdots \| M_{j-1} \| M_j \| M_{j+1} \| \cdots \| M_\zeta) \gg_j M], s\rangle$$
$$\longrightarrow \langle E[M_1 \| \cdots \| M_{j-1} \| \mathsf{seq}\ M_j\ M \| M_{j+1} \| \cdots \| M_\zeta], s\rangle$$

Figure 2 Reduction rules.

opponent/player attribution is swapped, i.e., $\overline{\lambda}_A^{\mathsf{OP}}(b) = \mathsf{O}$ iff $\lambda_A^{\mathsf{OP}}(b) = \mathsf{P}$ for every move b.)

- The arena $\mathbf{N} = ((M_{\mathbf{N}}, <_{\mathbf{N}}), \lambda_{\mathbf{N}}, \vdash_{\mathbf{N}})$ of natural numbers, where $M_{\mathbf{N}} = \{\mathtt{q}\} \cup \{n \mid n \geq 0\}$, $\lambda_{\mathbf{N}}(\mathtt{q}) = (\mathsf{O}, \mathsf{Q})$, $\lambda_{\mathbf{N}}(n) = (\mathsf{P}, \mathsf{A})$ for every n, $\vdash_{\mathbf{N}} = \{(\mathtt{q}, \mathtt{q})\} \cup \{(\mathtt{q}, n) \mid n \geq 0\}$.
- The arena $\mathbf{C} = ((M_{\mathbf{C}}, <_{\mathbf{C}}), \lambda_{\mathbf{C}}, \vdash_{\mathbf{C}})$ of commands, where $M_{\mathbf{C}} = \{\mathtt{run}, \mathtt{done}\}$, $\lambda_{\mathbf{C}}(\mathtt{run}) = (\mathsf{O}, \mathsf{Q})$, $\lambda_{\mathbf{C}}(\mathtt{done}) = (\mathsf{P}, \mathsf{A})$, $\vdash_{\mathbf{C}} = \{(\mathtt{run}, \mathtt{run}), (\mathtt{run}, \mathtt{done})\}$.
- The arena $\mathbf{Var} = ((M_{\mathbf{Var}}, <_{\mathbf{Var}}), \lambda_{\mathbf{Var}}, \vdash_{\mathbf{Var}})$ of mutable variables, where $M_{\mathbf{Var}} = \{\mathtt{rd}, \mathtt{ok}\} \cup \{n \mid n \geq 0\} \cup \{\mathtt{wr}_n \mid n \geq 0\} \cup \{\mathtt{cas}_{m,n} \mid m, n \geq 0\}$, $\lambda_{\mathbf{Var}}(\mathtt{rd}) = \lambda_{\mathbf{Var}}(\mathtt{wr}_n) = \lambda_{\mathbf{Var}}(\mathtt{cas}_{m,n}) = (\mathsf{O}, \mathsf{Q})$ and $\lambda_{\mathbf{Var}}(\mathtt{ok}) = \lambda_{\mathbf{Var}}(n) = (\mathsf{P}, \mathsf{A})$ for every $m, n \geq 0$, and $\vdash_{\mathbf{Var}} = \{(\mathtt{rd}, \mathtt{rd})\} \cup \{(\mathtt{rd}, n) \mid n \geq 0\} \cup \{(\mathtt{wr}_n, \mathtt{wr}_n), (\mathtt{wr}_n, \mathtt{ok}) \mid n \geq 0\} \cup \{(\mathtt{cas}_{m,n}, \mathtt{cas}_{m,n}) \mid m, n \geq 0\} \cup \{(\mathtt{cas}_{m,n}, l) \mid l, m, n \geq 0\}$.

We associate an arbitrary (but fixed) ordering to each of the arena moves above, leaving its explicit definition unspecified.

The compound arenas $A \times B$ and $A \Rightarrow B$ are defined as follows. (Below, for any pairs of functions $f : S \to U$ and $g : T \to U$, we write $[f, g] : S + T \to U$ for the coproduct function, where $S + T$ stands for the disjoint sum of S and T.)

- A *product arena* $A \times B$ is a triple $((M_{A\times B}, <_{A\times B}), \lambda_{A\times B}, \vdash_{A\times B})$, where $M_{A\times B} = M_A + M_B$, $\lambda_{A\times B} = [\lambda_A, \lambda_B]$, and $n \vdash_{A\times B} m$ iff $n \vdash_A m \vee n \vdash_B m$. The associated order extends the union of orders $<_A \cup <_B$ with additional orderings $a <_{A\times B} b$ for every $a \in M_A$ and $b \in M_B$.
- An *arrow arena* $A \Rightarrow B$ is a triple $((M_{A\Rightarrow B}, <_{A\Rightarrow B}), \lambda_{A\Rightarrow B}, \vdash_{A\Rightarrow B})$, where $M_{A\Rightarrow B} = M_A + M_B$, $\lambda_{A\Rightarrow B} = [\overline{\lambda}_A, \lambda_B]$, and $n \vdash_{A\Rightarrow B} m$ iff $n \vdash_B m \vee (n \neq m \wedge n \vdash_A m) \vee (n \vdash_B n \wedge m \vdash_A m)$. The associated order $<_{A\Rightarrow B}$ is the same as $<_{A\times B}$.

Fig. 3 gives the game interpretation of each IA_{par} term $\Gamma \vdash M : \mathtt{T}$, specified as a Kleisli arrow $[\![\Gamma \vdash M : \mathtt{T}]\!] : [\![\Gamma]\!] \to [\![\mathtt{T}]\!]$ in $\mathcal{G}_F$.

$$[\![\Gamma, x : \mathtt{T} \vdash x : \mathtt{T}]\!] = \boldsymbol{\pi}_{[\![\mathtt{T}]\!]}$$
$$[\![\Gamma \vdash \lambda x :^{\mathtt{T}} .M : \mathtt{T}']\!] = \boldsymbol{\Lambda}_{[\![\mathtt{T}]\!]}([\![\Gamma, x : \mathtt{T} \vdash M : \mathtt{T}']\!])$$
$$[\![\Gamma \vdash MN : \mathtt{T}]\!] = \langle [\![\Gamma \vdash M : \mathtt{T}' \to \mathtt{T}]\!], [\![\Gamma \vdash N : \mathtt{T}']\!] \rangle_{\S} \mathbf{ev}_{[\![\mathtt{T}']\!],[\![\mathtt{T}]\!]}$$
$$[\![\Gamma \vdash \mathsf{fix}_{\mathtt{T}} M : \mathtt{T}]\!] = \bigsqcup_i \sigma_i, \text{ where } \sigma_0 = \bot \text{ and } \sigma_{i+1} = \langle [\![\Gamma \vdash M : \mathtt{T} \to \mathtt{T}]\!], \sigma_i \rangle_{\S} \mathbf{ev}_{[\![\mathtt{T}]\!],[\![\mathtt{T}]\!]}.$$
$$[\![\Gamma \vdash n : \mathtt{nat}]\!] = !_{[\![\Gamma]\!]}{}_{\S}\, \mathit{cnst}_n$$
$$[\![\Gamma \vdash M \star N : \mathtt{nat}]\!] = \langle [\![\Gamma \vdash M]\!], [\![\Gamma \vdash N]\!] \rangle_{\S}\, \mathit{binop}_\star$$
$$[\![\Gamma \vdash \mathsf{skip} : \mathtt{com}]\!] = !_{[\![\Gamma]\!]}{}_{\S}\, \mathit{skip}$$
$$[\![\Gamma \vdash \mathsf{seq}\ M_1\ M_2 : \mathtt{T}]\!] = \langle [\![\Gamma \vdash M_1 : \mathtt{com}]\!], [\![\Gamma \vdash M_2 : \mathtt{T}]\!] \rangle_{\S}\, \mathit{seq}_{[\![\mathtt{T}]\!]}$$
$$[\![\Gamma \vdash \mathsf{if0}\ M\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \mathtt{T}]\!] = \langle [\![\Gamma \vdash M : \mathtt{nat}]\!], [\![\Gamma \vdash M_1 : \mathtt{T}]\!], [\![\Gamma \vdash M_2 : \mathtt{T}]\!] \rangle_{\S}\, \mathit{cond}_{[\![\mathtt{T}]\!]}$$
$$[\![\Gamma \vdash \mathsf{assign}\ M\ N : \mathtt{com}]\!] = \langle [\![\Gamma \vdash M]\!], [\![\Gamma \vdash N]\!] \rangle_{\S}\, \mathit{asgn}$$
$$[\![\Gamma \vdash \mathsf{deref}\ M : \mathtt{nat}]\!] = [\![\Gamma \vdash M]\!]_{\S}\, \mathit{deref}$$
$$[\![\Gamma \vdash \mathsf{cas}\ L\ M\ N : \mathtt{com}]\!] = \langle [\![\Gamma \vdash L]\!], [\![\Gamma \vdash M]\!], [\![\Gamma \vdash N]\!] \rangle_{\S}\, \mathit{cas}$$
$$[\![\Gamma \vdash \mathsf{mkvar}\ M\ N\ L : \mathtt{var}]\!] = \langle [\![\Gamma \vdash M]\!]_{\S}\, \mathit{acpt}, [\![\Gamma \vdash N]\!], [\![\Gamma \vdash L]\!]_{\S}\, \mathit{cacpt} \rangle$$
$$[\![\Gamma \vdash \mathsf{newvar}\ v = n\ \mathsf{in}\ M : \mathtt{T}]\!] = ST\left(\langle id_{[\![\Gamma]\!]}, !_{[\![\Gamma]\!]}; \mathit{cell}_n \rangle; WO\left([\![\Gamma, v : \mathtt{var} \vdash M : \mathtt{T}]\!]\right)\right)$$
$$[\![\Gamma \vdash M_1\ \mathsf{or}\ M_2 : \mathtt{nat}]\!] = \langle [\![\Gamma \vdash M_1]\!], [\![\Gamma \vdash M_2]\!] \rangle_{\S}\, \mathit{choice}$$
$$[\![\Gamma \vdash M_1 \| \cdots \| M_\zeta : \mathtt{par}]\!] = \sigma_\zeta, \text{ where } \sigma_0 = !_{[\![\Gamma]\!]}{}_{\S}\, \mathit{pskip} \text{ and } \sigma_{i+1} = \langle \sigma_i, [\![\Gamma \vdash M_{i+1} : \mathtt{com}]\!] \rangle_{\S}\, \mathit{cont}_{i+1}.$$
$$[\![\Gamma \vdash P \gg_j M : \mathtt{par}]\!] = \langle [\![\Gamma \vdash P : \mathtt{par}]\!], [\![\Gamma \vdash M : \mathtt{com}]\!] \rangle_{\S}\, \mathit{cont}_j$$

Figure 3 The game interpretation of IA_{par} terms.

The strategies printed in *italic* fonts in the figure are defined (by their longest well-opened traces) as below.

- $\mathit{cnst}_n : \mathbf{1} \to \mathbf{N}$ is defined by the trace set $\{\mathtt{q}n\}$.
- $\mathit{binop}_\star : \mathbf{N}' \times \mathbf{N}'' \to \mathbf{N}$ is defined by the trace set $\{\mathtt{q}\mathtt{q}'m'\mathtt{q}''n''k \mid m \star n = k\}$.
- $\mathit{skip} : \mathbf{1} \to \mathbf{C}$ is defined by the trace set $\{\mathtt{run}\cdot\mathtt{done}\}$.
- $\mathit{seq}_A : \mathbf{C} \times A \to A$ is defined by the trace set $\{q\cdot\mathtt{run}\cdot\mathtt{done}\cdot t \mid q\cdot t \in T_{WO(id_A)}\}$.
- $\mathit{deref} : \mathbf{Var} \to \mathbf{N}$ is defined by the trace set $\{\mathtt{q}\cdot\mathtt{W}\cdot\mathtt{N}\cdot\mathtt{rd}\cdot n\cdot n \mid n \geq 0\}$.
- $\mathit{cas} : \mathbf{Var} \times \mathbf{N}_1 \times \mathbf{N}_2 \to \mathbf{N}$ is defined by the trace set $\{\mathtt{q}\cdot\mathtt{q}_1\cdot m\cdot\mathtt{q}_2\cdot n\cdot\mathtt{W}\cdot\mathtt{N}\cdot\mathtt{cas}_{m,n}\cdot k\cdot k \mid m, n, k \geq 0\}$.
- $\mathit{acpt} : (\mathbf{N} \Rightarrow \mathbf{C}) \to \mathbf{Var}$ is defined by the trace set $\{\mathtt{wr}_n\cdot\mathtt{run}\cdot\mathtt{q}\cdot n\cdot\mathtt{done}\cdot\mathtt{ok} \mid n \geq 0\}$.
- $\mathit{cacpt} : (\mathbf{N}_1 \Rightarrow \mathbf{N}_2 \Rightarrow \mathbf{N}) \to \mathbf{Var}$ is defined by the trace set $\{\mathtt{cas}_{m,n}\cdot\mathtt{q}\cdot\mathtt{q}_1\cdot m\cdot\mathtt{q}_2\cdot n\cdot k\cdot k \mid m, n, k \geq 0\}$.
- $\mathit{choice} : \mathbf{N}_1 \times \mathbf{N}_2 \to \mathbf{N}$ is defined by the trace set $\{\mathtt{q}\cdot\mathtt{q}_i\cdot n\cdot n \mid i \in \{1, 2\}, n \geq 0\}$.
- $\mathit{pskip} : \mathbf{1} \to (\mathbf{C}_1 \times \cdots \times \mathbf{C}_\zeta)_\perp$ is defined by the trace set $\{?\surd\cdot\mathtt{run}_{\rho(1)}\cdot\mathtt{done}_{\rho(1)} \cdots \mathtt{run}_{\rho(\zeta)}\cdot \mathtt{done}_{\rho(\zeta)} \mid \rho : \{1, \ldots, \zeta\} \mapsto \{1, \ldots, \zeta\} \text{ is a bijection }\}$.
- $\mathit{cont}_i : (\mathbf{C}'_1 \times \cdots \times \mathbf{C}'_\zeta)_\perp \times \mathbf{C} \to (\mathbf{C}_1 \times \cdots \times \mathbf{C}_\zeta)_\perp$ is specified by the traces $\{??'\surd'\surd s \mid s \in T_{\langle\sigma_1,\cdots,\sigma_\zeta\rangle}\}$, where the trace of each $\sigma_j : \mathbf{C}'_j \times \mathbf{C} \to \mathbf{C}_j$ is defined by $\{\mathtt{run}_j\cdot\mathtt{run}'_j\cdot\mathtt{done}'_j\cdot \mathtt{done}_j\}$ if $j \neq i$ and σ_i is defined by $\{\mathtt{run}_i\cdot\mathtt{run}'_i\cdot\mathtt{done}'_i\cdot\mathtt{run}\cdot\mathtt{done}\cdot\mathtt{done}_i\}$.
- $\mathit{cell}_n : \mathbf{1} \to \mathbf{Var}$ is the strategy defined by the set of *causal* traces whose initial value is n. A trace in $\mathbf{Var}$ is called a *causal* trace, if every $\mathtt{wr}_n$ move is immediately followed by a move $\mathtt{ok}$, and each $\mathtt{rd}$ or $\mathtt{cas}_{m,n}$ move is followed by a natural number stored in the variable just after the corresponding operation is finished. The value stored in the variable is determined by the last successful write: A *successful write* by n is identified

by a pair of successive moves, either $\mathtt{wr}_n \cdot \mathtt{ok}$ of a write operation or $\mathtt{cas}_{m,n} \cdot n$ of a CAS operation.

The newvar $v = n$ in M construct delimits the scope of variable v local to M and further forces the causality induced by the order of accesses to v. Unless bound in newvar, v behaves like a variable allocated in a volatile memory cell. We notice that, since every term M of type com or par under a scope of a local variable is executed exactly once, it must be interpreted, as done in [10], by a composition of well-opened strategy in $WO(\llbracket M \rrbracket)$ with a cell strategy in $\mathcal{G}$, written $cell_n : \mathbf{1} \to \mathbf{Var}$, which comprises of causal traces whose initial value is n.

Extracting Herbrand trees in classical realizability using forcing*

Lionel Rieg

LIP (UMR 5668 CNRS ENS Lyon UCBL INRIA), ENS de Lyon, Université de Lyon
46 allée d'Italie, 69364 LYON, FRANCE
`lionel.rieg@ens-lyon.fr`

Abstract

Krivine presented in [9] a methodology to combine Cohen's forcing with the theory of classical realizability and showed that the forcing condition can be seen as a reference that is not subject to backtracks. The underlying classical program transformation was then analyzed by Miquel [11] in a fully typed setting in classical higher-order arithmetic (PAω^+).

As a case study of this methodology, we present a method to extract a Herbrand tree from a classical realizer of inconsistency, following the ideas underlying the completeness theorem and the proof of Herbrand's theorem. Unlike the traditional proof based on Kőnig's lemma (using a fixed enumeration of atomic formulas), our method is based on the introduction of a particular Cohen real. It is formalized as a proof in PAω^+, making explicit the construction of generic sets in this framework in the particular case where the set of forcing conditions is arithmetical.
We then analyze the algorithmic content of this proof.

1998 ACM Subject Classification F.4.1 Lambda-calculus and related system

Keywords and phrases classical realizability, forcing, Curry-Howard correspondence, Herbrand trees

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.597

1 Introduction

Forcing is a model transformation initially invented by Cohen [1, 2] to prove the relative consistency of the negation of the continuum hypothesis with respect to the axioms of Zermelo-Fraenkel (ZF) set theory. From a model-theoretic point of view, forcing is a technique to extend a given model of ZF—the *base model*—into a larger model—the *generic extension*—generated around the base model from a new set with good properties: the generic filter G. From a proof-theoretic point of view, forcing can be presented as a logical translation that maps formulas expressing properties of the extended model into formulas expressing (more complex) properties of the base model. Through this translation, the properties of the (fictitious) generic set G (in the extended universe) are reduced to the properties of the forcing poset C (in the base universe) that parametrizes the whole construction.

Recently, Krivine studied [9] Cohen forcing in the framework of the proofs-as-programs correspondence in classical logic [5, 13, 3] and showed how to combine it with the theory of classical realizability [8]. In particular, he discovered a program translation (independent from typing derivations) that captures the computational contents of the logical translation underlying forcing. Surprisingly, this program transformation acts as a *state passing style*

* This work was supported by the ANR project RÉCRÉ.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 597–614

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

translation where the forcing condition is treated as a memory cell that is protected from the backtracks performed by control operators such as callcc [5] —thus opening an intriguing connection between forcing and imperative programming. Reformulating this work in classical higher-order arithmetic (PAω^+) and analyzing the corresponding program transformation, Miquel [11, 12] introduced an extension of the Krivine Abstract Machine (KAM) devoted to execution of proofs by forcing—the KFAM—where the forcing condition is explicitly treated as a memory cell in the context of the execution of a proof by forcing.

These analogies naturally suggest that Cohen forcing can be used not only to prove relative consistency results, but also to write computationally more efficient (classical) proofs by exploiting the imperative flavor of the forcing condition.

In this paper, we propose to instantiate this technique on one example, namely the extraction of a Herbrand tree (see section 2) from a validity proof of an existential formula $\exists\vec{x}.\, F(\vec{x})$ where $F(\vec{x})$ is quantifier-free. Our extraction procedure is based on a proof of a mix between completeness and Herbrand's theorem using the method of forcing. The key ingredient of this proof is the introduction of a Cohen real (using forcing) that represents all valuations at once. From a computational point of view, we will see that the corresponding program uses the forcing condition to store the tree under construction, thus protecting it from the backtracks induced by classical reasoning. The interest of this approach is that since the conclusion of our semantic variant of Herbrand's theorem is Σ^0_1, any proof (program) of the translation of the conclusion (through the forcing translation) can be turned into a proof (program) of the conclusion itself. From this, it is then possible to apply standard witness extraction techniques in classical realizability [10] to extract the desired Herbrand tree.

Contribution of the paper

This work follows on from [9] and [11]. Its contributions are the following:

- The extension of the program transformation underlying forcing to a generic filter G (when the forcing sort and its relativization predicate are invariant under forcing).
- A proof of a semantic variant of Herbrand's theorem (containing completeness) by forcing where a Cohen real represents all valuations at once in the forcing universe.
- A formalization of this proof in the formal system PAω^+ which, through the forcing transformation, gives an extraction process for Herbrand trees.
- An analysis of the computational content of this extraction process in classical realizability.

2 Herbrand trees

2.1 The notion of Herbrand tree

In what follows, we work in a given countable first-order language, and write Term and Atom the countable sets of closed terms and of closed atomic formulas, respectively. Throughout this paper we are interested in the following problem.

Let $\exists\vec{x}.\, F(\vec{x})$ be a purely existential formula, where $F(\vec{x})$ is quantifier-tree. Let us now assume that the formula $\exists\vec{x}.\, F(\vec{x})$ is true in all models, and actually in all *syntactic models*, where variables are interpreted by closed terms $t \in \mathrm{Term}$. From this information, we know that there is a function $H : (\mathrm{Atom} \to \mathrm{Bool}) \to \overrightarrow{\mathrm{Term}}$ that associates to every syntactic valuation $\rho : \mathrm{Atom} \to \mathrm{Bool}$ a tuple of closed terms $H(\rho) = \vec{t} \in \overrightarrow{\mathrm{Term}}$ such that $\rho \models F(\vec{t})$ (i.e. a 'witness' for the formula $\exists\vec{x}.\, F(\vec{x})$ in the valuation ρ).

However, the information provided by the function H is twice infinite: it is infinite in depth since each valuation $\rho : \mathrm{Atom} \to \mathrm{Bool}$ is (a priori) infinite, and it is infinite in

width since the set of all such valuations has the power of continuum. Nevertheless, the completeness theorem combined with Herbrand's theorem says that we can compact the information given by the a priori infinite function H into a finite binary tree, which is called a *Herbrand tree*.

▶ **Definition 2.1** (Herbrand tree for a formula F)**.** A *Herbrand tree* is a finite binary tree H such that:

- The inner nodes of H are labeled with atomic formulas $a \in \mathrm{Atom}$, so that every branch of the tree represents a partial valuation (going left means 'true', going right means 'false').
- Every leaf of H contains a witness for the corresponding branch, that is a tuple $\vec{t} \in \overrightarrow{\mathrm{Term}}$ s.t. $\rho \models F(\vec{t})$ for every (total) valuation ρ extending that partial valuation of the branch.

▶ **Theorem 2.2.** *If the formula $\exists \vec{x}.\, F(\vec{x})$ is true in all syntactic models, then F has a Herbrand tree.*

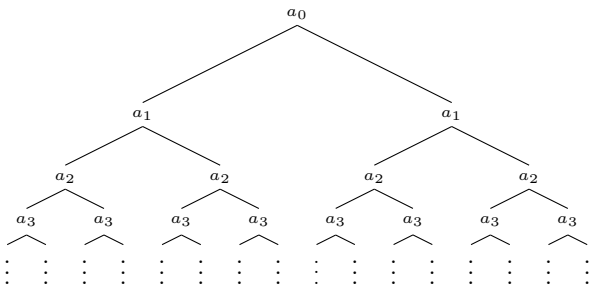
The aim of this paper is to describe a method to effectively extract a Herbrand tree from a proof (actually a classical realizer) of the proposition expressing that 'the formula $\exists \vec{x}.\, F(\vec{x})$ holds in all syntactic models'. Since the latter proposition is directly implied by the formula $\exists \vec{x}.\, F(\vec{x})$ itself (using the trivial implication of the completeness theorem), we will thus get a method to effectively extract a Herbrand tree from a proof/realizer of the formula $\exists \vec{x}.\, F(\vec{x})$.

Note that we will not give a proof of Theorem 2.2 but rather a proof of the validity of the *admissible rule* associated to it, namely: given a proof that 'the formula $\exists \vec{x}.\, F(\vec{x})$ holds in all syntactic models', we can build a proof of existence of a Herbrand tree for F. This statement is enough for extraction.

2.2 Extracting Herbrand trees effectively

In the framework of the Curry-Howard correspondence, the natural method to extract Herbrand trees is to use a classical realizer t_0 obtained from a formal proof of Theorem 2.2. By applying t_0 to a realizer u of the premise of Theorem 2.2, we get a realizer of the Σ^0_1-formula expressing the existence of a Herbrand tree for the formula $\exists x.F(\vec{x})$, from which we can retrieve the desired Herbrand tree using standard classical extraction techniques [10].

However, the efficiency of the extracted code highly depends on the proof of Theorem 2.2. In particular, the simplest proof of this theorem (Fig. 1), which relies on a fixed enumeration of all atoms, is not well suited to this task, since it gives terribly poor performances on



Given an enumeration $(a_i)_{i\in\mathbb{N}}$ of the closed instances of the atomic formulas appearing in $F(\vec{x})$, let us consider the infinite binary tree whose 2^i nodes at depth i are labeled with the atom a_i. Any infinite branch in this infinite tree is an valuation ρ, because all atoms appear along it. From our assumption, we know that there is a tuple $\vec{t} \in \overrightarrow{\mathrm{Term}}$ such that $\rho \models F(\vec{t})$. But since the calculation of the truth value of the closed formula $F(\vec{t})$ only relies on a finite subset of ρ, we can cut the branch along ρ at some depth d, putting a leaf labeled with $\vec{t}$. Doing this in all branches simultaneously, we get a finite tree (by the fan theorem), which is by construction a Herbrand tree.

Figure 1 A proof of Theorem 2.2 by enumerating the atoms.

formulas $F(\vec{x})$ involving atoms that appear late in the chosen enumeration. What we want is a proof/realizer of Theorem 2.2 that chooses the atoms labeling the nodes only in function of the realizer of its premise.

In what follows, we present a novel proof of Theorem 2.2 that is tailored for this purpose, and that relies on the forcing techniques developed in [9, 11, 12]. In this case, the forcing condition is a Cohen real which behaves as a generic valuation, *i.e.* it represents all infinite branches at once. As we will see in section 6, it is computationally a scheduler that will extend the tree under construction on request, depending on which atoms are required by the realizer of the premise. It will scan the whole tree and schedule pending branches until the full Herbrand tree is built.

3 The higher-order arithmetic PAω^+

In this section, we recall PAω^+, the formal proof system in which this work takes place. It is a presentation of classical higher-order arithmetic with explicit (classical) proof terms, inspired by Church's theory of simple types. It features an extra congruence on terms, in the spirit of deduction modulo [4]. This section is a summary of the presentation of PAω^+ in [11], to which we refer the reader for more details and proofs of the results stated here.

3.1 Syntax

System PAω^+ distinguishes three kinds of syntactic entities: *sorts* (or *kinds*), *higher-order terms*, and *proof terms*, whose grammar is recalled in Fig 2.

Sorts	$\tau, \sigma ::= \iota \quad \mid \quad o \quad \mid \quad \tau \to \sigma$
Higher-order terms	$M, N, A, B ::= x^\tau \quad \mid \quad \lambda x^\tau.\, M \quad \mid \quad MN \quad \mid \quad 0 \quad \mid \quad S \quad \mid \quad \mathrm{rec}_\tau$ $\mid A \Rightarrow B \quad \mid \quad \forall x^\tau.\, A \quad \mid \quad M \doteq_\tau N \mapsto A$
Proof-terms	$t, u ::= x \quad \mid \quad \lambda x.\, t \quad \mid \quad tu \quad \mid \quad \mathrm{callcc}$

Figure 2 Syntax of PAω^+.

3.1.1 Sorts and higher-order terms

Sorts are simple types formed from the two basic sorts ι (the sort of *individuals*) and o (the sort of *propositions*). Higher-order terms (also called *terms* for short) are simply-typed λ-terms (à la Church) that are intended to represent *mathematical objects* that inhabit sorts.

Higher-order terms of sort ι, which are called *individuals*, are formed using the two constructors 0 (of sort ι), S (of sort $\iota \to \iota$) and the family of recursors rec_τ (of sort $\tau \to (\iota \to \tau \to \tau) \to \iota \to \tau$).

Higher-order terms of sort o, which are called *propositions* (and written A, B, C, etc. in what follows), are formed using implication $A \Rightarrow B$ (where A and B are propositions), universal quantification $\forall x^\tau.\, A$ (where A is a proposition possibly depending on the variable x^τ) and a new connective $M \doteq_\tau N \mapsto A$ called an *equational implication* (where M and N are of sort τ and where A is a proposition). This new connective must be thought of as a kind of implication, but giving more compact proof terms. It makes the computational contents of the forcing translation more transparent, but it is logically equivalent to the usual implication $M =_\tau N \Rightarrow A$, via the proof terms:

$$\lambda xy.\, y\, x \;:\; (M \doteq N \mapsto A) \Rightarrow (M = N \Rightarrow A), \qquad \lambda x.\, x\, (\lambda y.\, y) \;:\; (M = N \Rightarrow A) \Rightarrow (M \doteq N \mapsto A)$$

(See fFg. 4 for a definition of the proof system.)

As usual, application is left associative whereas implication and equational implication are both right associative and have same precedence: $A \Rightarrow M \doteq N \mapsto B \Rightarrow C \Rightarrow D$ has to be read as $A \Rightarrow (M \doteq N \mapsto (B \Rightarrow (C \Rightarrow D)))$. Logical connectives (absurdity, negation, conjunction, disjunction) are defined using the standard second-order encodings, as well as Leibniz equality, letting: $x =_\tau y := \forall Z^{\tau\to o}.\, Z\, x \Rightarrow Z\, y$. Existential quantification (possibly combined with conjunctions) is encoded classically using De Morgan laws: $\exists x^\tau.\, A_1 \,\&\, \ldots \,\&\, A_k := \neg(\forall x^\tau.\, A_1 \Rightarrow \ldots \Rightarrow A_k \Rightarrow \bot)$. We often omit the sort annotation τ to ease reading when this does not hinder understanding. On the opposite, when we want to give explicitly the sort of a term, we write it in exponent, *e.g.* M^τ, A^o, $\mathrm{rec}_\tau^{\tau\to(\iota\to\tau\to\tau)\to\iota\to\tau}$.

3.1.2 System T is a fragment of PAω^+

Gödel's system T can be recovered from PAω^+ as the subsystem where we restrict sorts to be T-sorts, that is sorts built with ι as the only base sort. This constraint casts out all logical constructions and limits the term construction rules exactly to those of system T. Recall that the expressiveness of system T is exactly the functions which are provably total in first-order arithmetic, which includes (and exceeds) all primitive recursive functions.

3.2 Proof system

3.2.1 Congruence

The proof system PAω^+ differs from higher-order arithmetic by the addition of a congruence $\simeq_\mathcal{E}$ to the proof system. This allows to reason modulo some equivalence on higher-order terms (hence on propositions) without polluting the proof terms with computationally irrelevant parts.

This congruence contains the usual $\beta\eta\iota$-conversion, some semantic equivalences on propositions (mostly commutations) and an equational theory $\mathcal{E}$. This *equational theory* is a finite set of equations $\mathcal{E} = M_1 = N_1, \ldots, M_k = N_k$, where M_i and N_i are higher-order terms of the same sort (that $\simeq_\mathcal{E}$ considers equal). Some rules for the congruence $\simeq_\mathcal{E}$ are given in Fig. 3, the full set is given in annex A.

$$\frac{}{M \simeq_\mathcal{E} N}\ (M = N) \in \mathcal{E} \qquad \frac{M \simeq_\mathcal{E} N \qquad P \simeq_\mathcal{E} Q \qquad A \simeq_{\mathcal{E}, M=P} B}{M \doteq P \mapsto A \simeq_\mathcal{E} N \doteq Q \mapsto B}$$

$$\frac{}{M \doteq M \mapsto A \simeq_\mathcal{E} A} \qquad \frac{}{A \Rightarrow M \doteq N \mapsto B \simeq_\mathcal{E} M \doteq N \mapsto A \Rightarrow B}$$

$$\frac{}{\forall x^\tau.\, M \doteq N \mapsto A \simeq_\mathcal{E} M \doteq N \mapsto \forall x^\tau.\, A}\ x \notin FV(M, N)$$

Figure 3 Some inference rules for the relation $\simeq_\mathcal{E}$.

3.2.2 Proof terms and inference rules

Proof terms (Fig. 2) are pure λ-terms enriched with an extra constant callcc; they are formed from a set of *proof variables* (notation: x, y, z, etc.) distinct from higher-order term variables. The deduction system of PAω^+ is defined around a typing judgment of the form $\mathcal{E}; \Gamma \vdash t : A$, where $\mathcal{E}$ is an equational theory and Γ a *context*, that is: a finite set of bindings of distinct proof variables x_i to propositions A_i. The inference rules, given in Fig 4, are the ones of higher-order arithmetic, with slight modifications to deal with the congruence and equational implication.

$$\frac{}{\mathcal{E};\Gamma,x:A\vdash x:A} \qquad \frac{\mathcal{E};\Gamma\vdash t:A}{\mathcal{E};\Gamma\vdash t:A'}\ A\simeq_{\mathcal{E}} A' \qquad \frac{}{\mathcal{E};\Gamma\vdash \mathrm{callcc}:((A\Rightarrow B)\Rightarrow A)\Rightarrow A}$$

$$\frac{\mathcal{E};\Gamma,x:A\vdash t:B}{\mathcal{E};\Gamma\vdash \lambda x.\,t:A\Rightarrow B} \qquad \frac{\mathcal{E};\Gamma\vdash t:A\Rightarrow B \qquad \mathcal{E};\Gamma\vdash u:A}{\mathcal{E};\Gamma\vdash t\,u:B}$$

$$\frac{\mathcal{E},M^\tau=N^\tau;\Gamma\vdash t:A}{\mathcal{E};\Gamma\vdash t:M\doteq_\tau N\mapsto A} \qquad \frac{\mathcal{E};\Gamma\vdash t:M\doteq_\tau M\mapsto A}{\mathcal{E};\Gamma\vdash t:A}$$

$$\frac{\mathcal{E};\Gamma\vdash t:A}{\mathcal{E};\Gamma\vdash t:\forall x^\tau.\,A}\ x\notin FV(\Gamma) \qquad \frac{\mathcal{E};\Gamma\vdash t:\forall x^\tau.\,A}{\mathcal{E};\Gamma\vdash t:A[N^\tau/x^\tau]}$$

Figure 4 The inference rules of PAω^+.

▶ Remarks.

1. The only inference rules that alter proof terms are the axiom, Peirce's law, and the introduction and elimination rules of implication. The remaining rules do not affect proof terms and are said to be *computationally transparent*.
2. The proof system of PAω^+ enjoys no normalization property since the proposition $\top$ defined by $\top := \lambda xy.\,x\doteq_o \lambda xy.\,y\mapsto\bot$ acts as a type of all (untyped) proof terms [11, section II.E.3]. (Intuitively, $\top$ allows to equate any two propositions so that they are all equivalent to $\bot$.) Nevertheless, the system is sound with respect to the intended classical realizability semantics (see section 3.4).
3. This proof system allows full classical reasoning thanks to Peirce's law. Arithmetical reasoning (including reasoning by induction) can be recovered by relativizing all quantifications over the sort ι using the predicate $x\in\mathbb{N} := \forall Z^o.\,Z\,0\Rightarrow(\forall y^\iota.\,Z\,y\Rightarrow Z\,(S\,y))\Rightarrow Z\,x$ (see below).

3.3 Sets and datatypes

In PAω^+, a set is given by a sort τ together with a relativization predicate P of sort $\tau\to o$ expressing membership in the set. For instance, the set of total relations between individuals is given by the sort $\iota\to\iota\to o$ and the predicate $\mathrm{Tot} := \lambda R.\,\forall x^\iota.\,\exists y^\iota.\,R\,x\,y$.

Because the sort τ can be inferred from the sort of P, we will identify sets with their relativization predicates. For convenience, we use the suggestive notations $x\in P$ (resp. $\forall x\in P.\,A$, $\exists x\in P.\,A$) for $P\,x$ (resp. $\forall x.\,P\,x\Rightarrow A$, $\exists x.\,x\in P\,\&\,A$). In what follows, *datatypes* will be represented as particular sets based on the sort $\tau\equiv\iota$ and whose relativization predicate P is invariant under forcing (see section 4.2). For instance, the datatypes of Booleans and natural numbers are given by

$$\begin{aligned} x\in\mathrm{Bool} &:= \forall Z^{\iota\to o}.\,Z\,0\Rightarrow Z\,1\Rightarrow Z\,x\\ x\in\mathbb{N} &:= \forall Z^{\iota\to o}.\,Z\,0\Rightarrow\big(\forall y^\iota.\,Z\,y\Rightarrow Z\,(S\,y)\big)\Rightarrow Z\,x\end{aligned}$$

(The proof of their invariance under forcing is delayed until section 5.2.) We also consider two abstract datatypes Term and Atom representing closed terms and closed atomic formulas (whose exact implementation is irrelevant). More generally, inductive datatypes are defined by implementing constructors as suitable functions from individuals to individuals and by defining the corresponding predicate by well-known second-order encodings. For instance, the datatype of binary trees

$$t,t' := \mathrm{Leaf}\,\vec{v} \quad\mid\quad \mathrm{Node}\,a\,t\,t' \qquad\qquad \text{where } \vec{v}\in\overrightarrow{\mathrm{Term}},\ a\in\mathrm{Atom}$$

is given by two injective functions $\mathrm{Leaf}^{\iota\to\iota}$ and $\mathrm{Node}^{\iota\to\iota\to\iota\to\iota}$ whose ranges do not overlap (the actual implementation is irrelevant here) and the corresponding relativization predicate $t\in\mathrm{Tree}$ is

$\forall Z^{\iota\to o}.\,(\forall \vec{v} \in \mathrm{Term}.\, Z\,(\mathrm{Leaf}\,\vec{v})) \;\Rightarrow\; (\forall t_1^\iota t_2^\iota\, a \in \mathrm{Atom}.\, Z\,t_1 \Rightarrow Z\,t_2 \Rightarrow Z\,(\mathrm{Node}\,a\,t_1\,t_2)) \Rightarrow Z\,t\,.$

We also introduce the inductive datatype Comp of quantifier-free formulas built above Atom:

$$c, c' := \perp\!\!\!\perp \;\mid\; a \;\mid\; c \Rrightarrow c' \qquad \text{where } a \in \mathrm{Atom}$$

This presentation based on implication is more suited to classical realizability (see below), but Comp is nothing but the free Boolean algebra generated by Atom.

3.4 Realizability semantics

System PAω^+ has a classical realizability semantics in the spirit of Krivine's [8] that is fully described in [11, 12]. This semantics is based on Krivine's λ_c-calculus (which contains all proof terms of PAω^+) and parametrized by a fixed set of processes (the *pole* of the realizability model). According to this semantics, every (closed) proof term t of a (closed) proposition A is a realizer of A (written $t \Vdash A$), and this independently from the choice of the pole. In the particular case where the pole is empty, the realizability model collapses to a Tarski model of PAω^+, from which we deduce the logical consistency of the system. This classical realizability semantics also provides simple methods to extract witnesses from realizers (and thus from proofs) of Σ^0_1-propositions [10].

4 The Forcing Transformation

4.1 Forcing in PAω^+

This section is a reformulation of Cohen's theory of forcing (developed for ZF set theory) in the framework of PAω^+. Here, we see forcing as a translation of facts about objects living in an *extended universe* (where sorts intuitively contain much more inhabitants) to facts about objects living in the *base universe*. Technically, we will first present forcing as a translation from PAω^+ to itself. But in section 4.3, we will see how to add a generic filter G to PAω^+, so that forcing will be actually a translation from PA$\omega^+ + G$ to PAω^+. We follow here the presentation of [11, 12], where the reader may find all missing proofs.

4.1.1 Definition of a forcing structure

As in [9, 11], we introduce the set of conditions as an upward closed subset C of a meet-semilattice $(\kappa, \cdot, 1)$. (Any poset with a greatest element can be presented in this way.)

▶ **Definition 4.1** (Forcing structure). A *forcing structure* is given by:

- a set $C : \kappa \to o$ of *well-formed forcing conditions* ($p \in C$ being usually written $C[p]$),
- an operation $\cdot$ of sort $\kappa \to \kappa \to \kappa$ to form the *meet* of two conditions (denoted by juxtaposition),
- a greatest condition 1,
- nine closed proof terms representing the axioms that must be satisfied by the forcing structure:

$$\begin{array}{lll}
\alpha_0 : C[1] & \alpha_1 : \forall pq.\, C[pq] \Rightarrow C[p] & \alpha_2 : \forall pq.\, C[pq] \Rightarrow C[q] \\
\alpha_3 : \forall pq.\, C[pq] \Rightarrow C[qp] & \alpha_4 : \forall p.\, C[p] \Rightarrow C[pp] & \alpha_5 : \forall pqr.\, C[(pq)r] \Rightarrow C[p(qr)] \\
\alpha_6 : \forall pqr.\, C[p(qr)] \Rightarrow C[(pq)r] & \alpha_7 : \forall p.\, C[p] \Rightarrow C[p1] & \alpha_8 : \forall p.\, C[p] \Rightarrow C[1p]
\end{array}$$

$(\sigma \to \tau)^*$	$:=$	$\sigma^* \to \tau^*$	ι^*	$:=$	ι	o^*	$:=$	$\kappa \to o$
$(x^\tau)^*$	$:=$	x^{τ^*}	0^*	$:=$	0	$(\forall x^\tau.\, A)^*$	$:=$	$\lambda r^\kappa.\, \forall x^{\tau^*}.\, A^*\, r$
$\lambda x^\tau.\, M$	$:=$	$\lambda x^{\tau^*}.\, M^*$	S^*	$:=$	S	$(M \doteq N \mapsto A)^*$	$:=$	$\lambda r^\kappa.\, M^* \doteq N^* \mapsto A^*\, r$
$(M\, N)^*$	$:=$	$M^*\, N^*$	rec_τ^*	$:=$	rec_{τ^*}	$(A \Rightarrow B)^*$	$:=$	$\lambda r^\kappa.\, \forall q^\kappa \forall (r')^\kappa.\, r \doteq qr' \mapsto (\forall s^\kappa.\, C[qs] \Rightarrow A^*\, s) \Rightarrow B^*\, r'$
x^*	$:=$	x	$(\lambda x.\, t)^*$	$:=$	$\gamma_1(\lambda x.\, t^*[(\beta_3 y)/y][(\beta_4 x)/x])$			$y \neq x$
$(t\, u)^*$	$:=$	$\gamma_3\, t^*\, u^*$	callcc^*	$:=$	$\lambda cx.\, \mathrm{callcc}(\lambda k.\, x\, (\alpha_{14}\, c)\, (\lambda cy.\, k\, (y\, \alpha_{15}\, c)))$			

$\beta_3 := \lambda xc.\, x\, (\alpha_9\, c)$ $\quad$ $\beta_4 := \lambda xc.\, x\, (\alpha_{10}\, c)$ $\quad$ $\gamma_1 := \lambda xcy.\, x\, y\, (\alpha_6\, c)$ $\quad$ $\gamma_3 := \lambda xyc.\, x\, (\alpha_{11} c)\, y$

Figure 5 The forcing translations $\tau \mapsto \tau^*$, $M \mapsto M^*$ and $t \mapsto t^*$.

(This set of axioms is not minimal, since α_2, α_6 and α_8 can be defined from the others.)

The above axioms basically express that the set C is upward-closed with respect to the pre-ordering $p \leq q$ ('p is stronger than q') defined by $p \leq q := \forall r^\kappa.\, C[pr] \Rightarrow C[qr]$. From this definition of the preorder $p \leq q$, we easily check that pq is the meet of p and q and that 1 is the greatest element. On the other hand, all the elements of κ outside C are equivalent with respect to the ordering $\leq$; they intuitively represent an 'inconsistent condition' stronger than all well-formed conditions.

In what follows, we will also need the following derived combinators:

$\alpha_9 := \alpha_3 \circ \alpha_1 \circ \alpha_6 \circ \alpha_3 : \forall pqr.\, C[pqr] \Rightarrow C[pr]$ $\quad$ $\alpha_{10} := \alpha_2 \circ \alpha_5 : \forall pqr.\, C[pqr] \Rightarrow C[qr]$
$\alpha_{11} := \alpha_9 \circ \alpha_4 : \forall pq.\, C[pq] \Rightarrow C[p(pq)]$ $\quad$ $\alpha_{12} := \alpha_5 \circ \alpha_3 : \forall pqr.\, C[p(qr)] \Rightarrow C[q(rp)]$
$\alpha_{13} := \alpha_3 \circ \alpha_{12} : \forall pqr.\, C[p(qr)] \Rightarrow C[(rp)q]$
$\alpha_{14} := \alpha_{12} \circ \alpha_{10} \circ \alpha_4 \circ \alpha_2 : \forall pqr.\, C[p(qr)] \Rightarrow C[q(rr)]$ $\quad$ $\alpha_{15} := \alpha_9 \circ \alpha_3 : \forall pqr.\, C[p(qr)] \Rightarrow C[qp]$

where $\alpha_i \circ \alpha_j \circ \cdots \circ \alpha_k$ stands for $\lambda c.\, \alpha_i\, (\alpha_j\, \ldots\, (\alpha_k\, c) \ldots)$ with c a fresh proof variable.

4.1.2 The three forcing translations

Given a forcing structure, the forcing transformation consists of three translations: $\tau \mapsto \tau^*$ on sorts, $M \mapsto M^*$ on higher-order terms (which is extended point-wise to equational theories) and $t \mapsto t^*$ on proof terms. The translations are given figure 5 (see [12] for the definition of all combinators).

▶ Remarks.

1. The translation on sorts simply replaces occurrences of o by $\kappa \to o$. This means that propositions will now depend on an extra parameter which is a forcing condition.
2. The translation on (higher-order) terms changes the sort of the term: N^τ is turned into $(N^*)^{\tau^*}$. The heart of this translation lies in the implication case and it merely propagates through the connectives in all the other cases.
3. The proof term translation instrumentalizes the computational interaction between abstractions and applications in proof terms:
   - it adds the γ_3 combinator in front of applications;
   - it shows the de Bruijn structure of bound variables: if an occurrence of the bound variable x has de Bruijn index n, it will be translated to $\beta_3^n\, (\beta_4\, x)$.

4.1.3 The forcing transformation on propositions

From the translation on terms, we define the usual forcing relation $p \Vdash A$ on propositions, letting:

$$p \Vdash A := \forall r^\kappa.\, C[pr] \Rightarrow A^* r\ .$$

This definition extends point-wise to contexts and we write it $p \Vdash \Gamma$. In addition to the expected properties of substitutivity and compatibility with the congruences $\simeq_{\mathcal{E}}$, this transformation on propositions enjoys the following important properties:

▶ **Proposition 4.2.**
1. *Forcing strongly commutes with universal quantification and equational implication:*
   $p \Vdash \forall x^\tau.\, A \;\simeq\; \forall x^{\tau^*}.\, (p \Vdash A) \qquad p \Vdash (M \doteq_\tau N \mapsto A) \;\simeq\; M^* \doteq_{\tau^*} N^* \mapsto (p \Vdash A)$
2. *Forcing is anti-monotonic:* $\forall pq.\, (p \Vdash A) \Rightarrow (pq \Vdash A)$
3. *Forcing an implication:* $p \Vdash A \Rightarrow B \iff \forall q^\kappa.\, (q \Vdash A) \Rightarrow (pq \Vdash B)$

▶ **Theorem 4.3** (Soundness). *If the judgment $\mathcal{E}; \Gamma \vdash t : A$ is derivable in PAω^+, then the judgment $\mathcal{E}^*; (p \Vdash \Gamma) \vdash t^* : p \Vdash A$ is derivable in PAω^+.*

This theorem is thus an effective way to turn a proof term $t : A$ (expressed in the forcing universe) into a proof term $t^* : p \Vdash A$ (expressed in the base universe).

4.2 Invariance under forcing

Clearly, the sorts that are invariant under the forcing translation are exactly the T-sorts defining Gödel's system T (see section 3.1.2). A proposition A whose free variables live in T-sorts is said to be *invariant under forcing* or *absolute* when there exist two closed proof terms ξ_A and ξ'_A such that

$$\xi_A : \forall p.\, (p \Vdash A) \Rightarrow (C[p] \Rightarrow A) \qquad\qquad \xi'_A : \forall p.\, (C[p] \Rightarrow A) \Rightarrow (p \Vdash A)\ .$$

An important class of absolute propositions is the class of *first-order propositions*, which contains the subclass of *arithmetical propositions* (in which all quantifications are relativized).

▶ **Definition 4.4** (First-order propositions). First-order propositions are defined by

$$A, B \quad := \quad \bot \;\mid\; M^\tau = N^\tau \;\mid\; A \Rightarrow B \;\mid\; \forall x^\sigma.\, A \;\mid\; M^\iota \in \mathbb{N}$$

where σ and τ are T-sorts (see section 3.1.2).

▶ **Theorem 4.5** (Invariance). *All first-order propositions are invariant under forcing.*

▶ **Theorem 4.6** (Elimination of a forced hypothesis). *If the propositions $1 \Vdash A$ and $A \Rightarrow B$ are derivable (in the empty context) and if B is absolute, then B is derivable too (in the empty context).*

Proof. Let u and s be proof terms such that $u : A \Rightarrow B$ and $s : 1 \Vdash A$. Using theorem 4.3, we have $u^* : 1 \Vdash A \Rightarrow B$. Because B is invariant under forcing, the previous theorem gives us $\xi_B : (1 \Vdash B) \Rightarrow C[1] \Rightarrow B$. We finally get $\xi_B\, (\gamma_3\, u^*\, s)\, \alpha_0 : B$. ◀

This theorem will be used to remove forcing in the proof of existence of a Herbrand tree.

4.3 The generic filter G

We now introduce PA$\omega^+ + G$, which extends PAω^+ with a constant G (the generic filter) and its axioms. To do so, we first assume that $\kappa \equiv \kappa^*$ (it is a T-sort) and that the set of well-formed conditions C (of sort $\kappa \to o$) is absolute, so that we have two proof terms ξ_C and ξ'_C such that

$$\xi_C \;:\; \forall pq.\, (p \Vdash C[q]) \Rightarrow (C[p] \Rightarrow C[q]) \qquad\qquad \xi'_C \;:\; \forall pq.\, (C[p] \Rightarrow C[q]) \Rightarrow (p \Vdash C[q])\ .$$

(At this stage, we do not need to know the particular implementation of C.)

The proof system PA$\omega^+ + G$ is defined from PAω^+ by adding a constant G of sort $\kappa \to o$ and five axioms expressing its properties. The first four axioms say that G is a filter in C:

$\boldsymbol{A_1}$: G is a subset of C: $\forall p.\, p \in G \Rightarrow C[p]$,
$\boldsymbol{A_2}$: G is non empty: $1 \in G$,
$\boldsymbol{A_3}$: G is upward closed: $\forall pq.\, pq \in G \Rightarrow p \in G$,
$\boldsymbol{A_4}$: G is closed under product: $\forall pq.\, p \in G \Rightarrow q \in G \Rightarrow pq \in G$,
The last axiom—*genericity*—relies on the following notion:

▶ **Definition 4.7** (Dense subset)**.** A set D of sort $\kappa \to o$ is said *dense* in C if for every element $p \in C$, there is an element $q \in C$ belonging to D and smaller than p. Formally, we let:

$$D \text{ dense } := \forall p^\kappa.\, C[p] \Rightarrow \exists q^\kappa.\, C[pq] \,\&\, pq \in D \quad (\Leftrightarrow \quad \forall p^\kappa.\, C[p] \Rightarrow \exists q^\kappa.\, C[q] \,\&\, q \in D \,\&\, q \leq p)$$

The last axiom on the set G is then:
$\boldsymbol{A_5}$: G intersects every set $D^{\kappa\to o}$ (of the base universe) dense in C:
$(\forall p.\, C[p] \Rightarrow \exists q.\, C[pq] \,\&\, pq \in D) \;\Rightarrow\; \exists p.\, p \in G \,\&\, p \in D$.

Now we need to explain how the forcing translation extends to a translation from $\mathrm{PA}\omega^+ + G$ to $\mathrm{PA}\omega^+$. The term translation on the generic filter G is defined by $G^* \;:=\; \lambda pr.\, C[pr]$. This definition has the advantage of giving a very simple proposition for $p \Vdash q \in G$:

▶ **Fact 4.8.** $p \Vdash q \in G \;:=\; \forall r.\, C[pr] \Rightarrow (q \in G)^* r \;\simeq\; \forall r.\, C[pr] \Rightarrow C[qr] \;\simeq\; p \leq q$

We now need to prove the proposition $\forall p^\kappa.\, p \Vdash A_i$ (in $\mathrm{PA}\omega^+$) for each of the five axioms A_1–A_5 of the generic filter G. Thanks to proposition 4.2 (anti-monotonicity), it is sufficient to prove that $1 \Vdash A_i$. Notice that the proof terms justifying the filter properties of G are small, except the proof term for genericity (the most complex property).

▶ **Proposition 4.9** (Forcing the properties of G)**.**

$$\gamma_1\,(\lambda x.\, \xi'_C\,(\alpha_1 \circ x \circ \alpha_3)) : 1 \Vdash \forall p.\, p \in G \Rightarrow C[p] \tag{4.9.i}$$

$$\lambda x.\, x : 1 \Vdash 1 \in G \tag{4.9.ii}$$

$$\gamma_1\,(\lambda x.\, \alpha_9 \circ x \circ \alpha_{10}) : 1 \Vdash \forall pq.\, pq \in G \Rightarrow p \in G \tag{4.9.iii}$$

$$\gamma_1(\lambda x.\, \gamma_1\,(\lambda y.\, \alpha_{13} \circ y \circ \alpha_{12} \circ x \circ \alpha_2 \circ \alpha_5 \circ \alpha_5)) : 1 \Vdash \forall pq.\, p \in G \Rightarrow q \in G \Rightarrow pq \in G \tag{4.9.iv}$$

$$\begin{aligned}\gamma_1\,(\lambda x.\, \gamma_1\,(\lambda y.\, \xi'_\perp\,(\lambda c.\, \xi_{\exists_2}\, \xi_C\, \xi_D(\gamma_3\, x\,(\xi'_c\,(\lambda\_.\, c)))\,(\alpha_2\,(\alpha_1\, c))\\ (\lambda c' d.\, \xi_\perp\,(\gamma_3\,(\gamma_3\,(\beta_3\,(\beta_4\, y))\;\mathrm{I})\,(\xi'_D\,(\lambda\_.\, d)))\, c'))))\\ :\; 1 \Vdash (\forall p.\, C[p] \Rightarrow \exists q.\, C[pq] \,\&\, pq \in D) \Rightarrow \exists p.\, p \in G \,\&\, p \in D\end{aligned} \tag{4.9.v}$$

where $\xi_{\exists_2}$ is the proof term (built using theorem 4.5) such that

$$\xi_{\exists_2}\, \xi_A\, \xi_B \;:\; (p \Vdash \exists n.\, A \,\&\, B) \Rightarrow (C[p] \Rightarrow \exists n.\, A \,\&\, B)$$

5 A proof of Herbrand's theorem by forcing

In order not to alter the meaning of the forcing poset through the forcing transformation, we choose to let $\kappa \;:=\; \iota$ (the sort of individuals), because $\iota^* \equiv \iota$.

5.1 Interface for finite relations over $\mathrm{Atom} \times \mathrm{Bool}$

We describe here an interface implementing finite relations over pairs of atoms and Booleans together with some operations (union, membership test) and properties. Everything can be implemented for instance by finite ordered lists of pairs (in the sort ι) without repetition. We assume given $\&\&^{\iota\to\iota\to\iota}$ and $||^{\iota\to\iota\to\iota}$, the (infix) Boolean conjunction and disjunction (at

the term level) together with their defining equations (*e.g.* $1 \mathbin{\&\&} b \simeq b$) that must hold at the congruence level (typically by β-reduction for suitable definitions of && and ||). Let us first describe the terms of the interface.

$\emptyset^\iota$	: the empty relation	$\mathrm{sing}^{\iota\to\iota\to\iota}$	: $\mathrm{sing}\, a\, b$ denotes $\{(a,b)\}$ and is written a^b
$\cup^{\iota\to\iota\to\iota}$	: union (infix symbol)	$\mathrm{test}^{\iota\to\iota\to\iota\to\iota}$	: $\mathrm{test}\, p\, a\, b$ tests if the atom a is mapped to b in p

The required properties over this structure are:

- associativity, commutativity and idempotence of $\cup$
- $\emptyset$ is a neutral element for $\cup$
- the specification equations of test: for all a, a', b, b', p, q with $a \neq a'$ or $b \neq b'$,
  $\mathrm{test}\, \emptyset\, a\, b = 0 \qquad \mathrm{test}\, a^b\, a\, b = 1 \qquad \mathrm{test}\, a^b\, a'\, b' = 0 \qquad \mathrm{test}\, (p \cup q)\, a\, b = \mathrm{test}\, p\, a\, b \,||\, \mathrm{test}\, q\, a\, b$

Using these terms and properties, we define two operations:

- testing membership: $\mathrm{mem}\, a\, p := \mathrm{test}\, p\, a\, 1 \,||\, \mathrm{test}\, p\, a\, 0$
- adding the binding (a,b) to p: $p \cup a^b$

Among finite relations, we can distinguish those that are *functional*, *i.e.* those representing finite functions from Atom to Booleans. We call them *finite valuations* and denote their set by FVal. Formally, this set (in the sense of section 3.3) is inductively defined using the following second-order encoding, which encompasses both finiteness and functionality:

$$p \in \mathrm{FVal} := \forall Z^{\iota\to o}.\, Z\, \emptyset \Rightarrow (\forall r^\iota . \forall a \in \mathrm{Atom}.\, \mathrm{mem}\, a\, r \doteq_\iota 0 \mapsto Z\, r \Rightarrow Z\, (r \cup a^1)) \Rightarrow (\forall r^\iota . \forall a \in \mathrm{Atom}.\, \mathrm{mem}\, a\, r \doteq_\iota 0 \mapsto Z\, r \Rightarrow Z\, (r \cup a^0)) \Rightarrow Z\, p$$

This shows the underlying computational structure of finite valuations: they are isomorphic to lists of atoms with two `cons` constructors (one for the atoms mapped to true, one for those mapped to false) without duplicates (thanks to the precondition $\mathrm{mem}\, a\, r \doteq_\iota 0 \mapsto \ldots$ in the `cons` constructors).

Finally, we assume the existence of a function for testing membership, that is a proof term $\mathrm{Tot}_{\mathrm{test}}$ of the totality of test on finite valuations:

$$\mathrm{Tot}_{\mathrm{test}} \quad : \quad \forall p \in \mathrm{FVal}. \forall a \in \mathrm{Atom}. \forall b \in \mathrm{Bool}.\, \mathrm{test}\, p\, a\, b \in \mathrm{Bool}\ .$$

5.2 Programming in PAω^+

In order to ease writing proof terms in PAω^+, we introduce some macros:

$\langle a, b\rangle$	$\lambda f.\, f\, a\, b$	$\mathrm{let}\ (x,y) = c\ \mathrm{in}\ M$	$c\, (\lambda xy.\, M)$
$\mathrm{true}, \mathrm{false}$	$\lambda xy.\, x, \lambda xy.\, y$	$\mathrm{if}\ b\ \mathrm{then}\ f\ \mathrm{else}\ g$	$b\, f\, g$
$\mathrm{consT}\, a\, p$	$\lambda x_1 x_2 x_3.\, x_2\, a\, (p\, x_1\, x_2\, x_3)$	$\mathrm{consF}\, a\, p$	$\lambda x_1 x_2 x_3.\, x_3\, a\, (p\, x_1\, x_2\, x_3)$

They come with the inference rules (admissible in PAω^+) given Fig. 6.

$$\frac{\mathcal{E};\Gamma \vdash M : A \qquad \mathcal{E};\Gamma \vdash N : B}{\mathcal{E};\Gamma \vdash \langle M, N\rangle : A \wedge B} \qquad \frac{\mathcal{E};\Gamma \vdash M : A \wedge B \qquad \mathcal{E};\Gamma, x : A, y : B \vdash N : C}{\mathcal{E};\Gamma \vdash \mathrm{let}\ (x,y) = M\ \mathrm{in}\ N : C}\ {\scriptstyle x,y \notin FV(M)}$$

$$\frac{}{\mathcal{E};\Gamma \vdash \mathrm{true} : 1 \in \mathrm{Bool}} \qquad \frac{}{\mathcal{E};\Gamma \vdash \mathrm{false} : 0 \in \mathrm{Bool}}$$

$$\frac{\mathcal{E};\Gamma \vdash M : b \in \mathrm{Bool} \qquad \mathcal{E};\Gamma \vdash N : b \doteq 1 \mapsto A \qquad \mathcal{E};\Gamma \vdash P : b \doteq 0 \mapsto A}{\mathcal{E};\Gamma \vdash \mathrm{if}\ M\ \mathrm{then}\ N\ \mathrm{else}\ P : A}$$

$$\frac{\mathcal{E};\Gamma \vdash M : a \in \mathrm{Atom} \qquad \mathcal{E};\Gamma \vdash N : p \in \mathrm{FVal}}{\mathcal{E};\Gamma \vdash \mathrm{consT}\, M N : p \cup a^1 \in \mathrm{FVal}}\ {\scriptstyle \mathrm{mem}\, a\, p \simeq_{\mathcal{E}} 0}\ +\ \text{idem for consF with } p \cup a^0 \in \mathrm{FVal}$$

Figure 6 Admissible inference rules in PAω^+.

5.3 Definition of our forcing structure

The interface and functions defined in the previous two sections allow us to build the forcing structure that we will use for Herbrand's theorem. In this setting, finite valuations will represent pieces of information about the current valuation that will be used to decide which closed instance of the proposition $F(\vec{x})$ is false. Note that most combinators are the identity thanks to the properties we imposed on the implementation of finite relations.

▶ **Definition 5.1** (Forcing structure for Herbrand's theorem)**.** Our forcing structure is given by

$$\kappa := \iota \qquad C[p] := p \in \mathrm{FVal} \wedge (\mathrm{subH}\, p \Rightarrow \mathrm{subH}\, \emptyset) \qquad p \cdot q := p \cup q$$
$$1 := \emptyset \qquad \alpha_3 = \alpha_4 = \alpha_5 = \alpha_6 = \alpha_7 = \alpha_8 := \mathrm{I} \qquad \alpha_0 := \langle \lambda xyz.\, z, \mathrm{I} \rangle$$
$$\alpha_1 = \alpha_2 := \lambda c.\, \mathrm{let}\ (p,t) = c\ \mathrm{in}\ \langle \mathrm{Up}_{\mathrm{FVal}}\ p, \lambda x.\, \mathrm{let}\ (x_1,x_2) = x\ \mathrm{in}\ t\ \langle x_1, \mathrm{Mon}_{\mathrm{subHtree}}\ x_2 \rangle \rangle$$

($\mathrm{Up}_{\mathrm{FVal}}$ and $\mathrm{Mon}_{\mathrm{subHtree}}$ will be defined in section 5.4.)

▶ Remarks.

1. We can simplify α_1 further if we replace $\mathrm{Mon}_{\mathrm{subHtree}}$ by I (which is a realizer of the same formula, see the remark after lemma 5.4). Note that in this case, α_1 is no longer a proof term but only a realizer (which is enough for our purpose) and we can write it $\alpha_1 := \lambda c.\, \mathrm{let}\ (c_1,c_2) = c\ \mathrm{in}\ \langle \mathrm{Up}_{\mathrm{FVal}}\ c_1, c_2 \rangle$.
2. Once we have proven the existence of a Herbrand tree (that is $\mathrm{subH}\,\emptyset$), the second part of the definition of the set C ($\mathrm{subH}\, p \Rightarrow \mathrm{subH}\,\emptyset$) is trivial. Therefore, the set C is logically equivalent to its first part FVal, the set of finite functions from Atom to Bool. It is interesting to notice that when Atom $= \mathbb{N}$, this is exactly the forcing conditions used to add a Cohen real [7]. This remark means that our forcing structure actually adds a single Cohen real (in the extended universe) which turns out to be the model we seek. It is a simple exercise of forcing to show that this real number is different from all real numbers of the base universe and that it is non computable.

In order to use all the results of section 4 and to be able to remove forcing using theorem 4.6, we need to prove that both subH and C are absolute.

▶ **Proposition 5.2.** *The sets* Tree, subH, FVal *and* C *are invariant under forcing.*

Proof. There exist proof terms in PAω^+ proving these properties. For instance, we have:

$$\xi_C := \xi_\wedge\, \xi_{\mathrm{FVal}}\, (\xi_\Rightarrow\, \xi'_{\mathrm{subH}}\, \xi_{\mathrm{subH}}) \qquad \xi'_C := \xi'_\wedge\, \xi'_{\mathrm{FVal}}\, (\xi'_\Rightarrow\, \xi_{\mathrm{subH}}\, \xi'_{\mathrm{subH}})$$ ◀

5.4 Formal statement of Herbrand's theorem in PAω^+

We now formalize in PAω^+ the statement of Herbrand's theorem presented in section 2 as:

> If $F(\vec{x})$ is a quantifier-free formula and all syntactic valuations validate $\exists \vec{x}.\, F(\vec{x})$,
> then $\exists \vec{x}.\, F(\vec{x})$ has a Herbrand tree.

Since we consider atomic formulas as elements of an abstract datatype (of sort ι) represented by the set Atom, a valuation is completely determined by its values on atoms and is thus defined as a function from atoms to propositions that we represent by a term of sort $\iota \to o$. We can extend a valuation ρ to quantifier-free formulas by the function $\mathrm{interp}^{(\iota\to o)\to\iota\to o}$ recursively defined by the following equations.

$$\mathrm{interp}\,\rho\,\bot\!\!\!\bot := \bot \qquad \mathrm{interp}\,\rho\,a := \rho\,a \qquad \mathrm{interp}\,\rho\,(c \Rrightarrow c') := (\mathrm{interp}\,\rho\,c) \Rightarrow (\mathrm{interp}\,\rho\,c')$$

The formula $F(\vec{x})$ is represented by a term $V^{\iota\to\iota}$ mapping any $\vec{v}$ to the corresponding quantifier-free formula $F(\vec{v})$ in Comp. The premise of Herbrand's theorem becomes the formula $\forall\rho^{\iota\to o}.\,\exists\vec{v}\in\mathrm{Term}.\,\mathrm{interp}\,\rho\,(V\,\vec{v})$.

We now need to define the proposition expressing that a binary tree is a Herbrand tree. Checking the correctness of a Herbrand tree is completely computational:

1. go down the tree and remember the partial valuation of your current branch,
2. evaluate $V\,\vec{v}$ at the leaves using the partial valuation accumulated so far.

This process is performed by the function subHtree recursively defined by these equations.

$$\begin{aligned}\mathrm{subHtree}\,p\,(\mathrm{Node}\,a\,t_1\,t_2) &:= \mathrm{subHtree}\;pa^1\,t_1\;\&\&\;\mathrm{subHtree}\;pa^0\,t_2\\ \mathrm{subHtree}\,p\,(\mathrm{Leaf}\,\vec{v}) &:= \mathrm{eval}\,p\,(V\,\vec{v})\,1\end{aligned}$$

The case of leaves is treated using a Boolean function $\mathrm{eval}^{\iota\to\iota\to\iota\to\iota}$ checking whether the truth value of $V\,\vec{v}$ (2nd arg.) is equal to b (3rd arg.) in the valuation p (1st arg.). The only non trivial case is the case of an atom where we need to look for the binding (a,b) into p, which can be done by the test function (see section 5.1). Since p is partial, $\mathrm{eval}\,p\,(V\,\vec{v})\,b=0$ can have two causes: either the truth value of $V\,\vec{v}$ in p is $1-b$ or p does not contain enough information to evaluate $V\,\vec{v}$. Conversely, when $\mathrm{eval}\,p\,(V\,\vec{v})\,b=1$, it means both that p contains enough information to evaluate $V\,\vec{v}$ and that the result is b. When $\mathrm{subHtree}\,p\,t=1$, we say that t *is a Herbrand tree below* p. Using subHtree, we finally define the predicate $\mathrm{subH}\,p$ expressing the existence of a Herbrand tree below the finite (and partial) valuation p: $\mathrm{subH}\,p:=\exists t\in\mathrm{Tree}.\,\mathrm{subHtree}\,p\,t=1$.

Summing up, the formal statement of Herbrand's theorem in PAω^+ is

$$(\forall\rho^{\iota\to o}.\,\exists\vec{v}\in\mathrm{Term}.\neg\,\mathrm{interp}\,\rho\,(V\,\vec{v}))\Rightarrow\mathrm{subH}\,\emptyset\;. \tag{H}$$

▶ **Lemma 5.3** (subH-merging). *Let p be a partial valuation and let a be an atom not appearing in p. If we have both* $\mathrm{subH}\;pa^1$ *and* $\mathrm{subH}\;pa^0$*, then we have* $\mathrm{subH}\,p$.

Proof. If t_1 and t_2 are Herbrand trees below pa^1 and pa^0 respectively, then $\mathrm{Node}\,a\,t_1\,t_2$ is a Herbrand tree below p. In PAω^+, this lemma is formally stated and proved as follows.

$$\begin{aligned}\mathrm{merge}\;&:=\;\lambda x_a xy.\,\mathrm{let}\;(x_1,x_2)=x\;\mathrm{in\;let}\;(y_1,y_2)=y\;\mathrm{in}\;\langle\mathrm{Node}\;x_a\,x_1\,y_1,y_2\circ x_2\rangle\\ &:\;\forall p^\iota.\forall a\in\mathrm{Atom}.\,\mathrm{mem}\,a\,p\doteq 0\mapsto\mathrm{subH}\,pa^1\Rightarrow\mathrm{subH}\,pa^0\Rightarrow\mathrm{subH}\,p\end{aligned}$$

◀

▶ **Lemma 5.4** (Monotonicity). *The functions* test, eval *and* subHtree *are monotonic in p.*

Proof. There exists proof terms $\mathrm{Mon}_{\mathrm{test}}$, $\mathrm{Mon}_{\mathrm{eval}}$ and $\mathrm{Mon}_{\mathrm{subHtree}}$ of the propositions

$$\begin{aligned}&\forall pqab.\;\mathrm{test}\,p\,a\,b=1\Rightarrow\mathrm{test}\,(p\cup q)\,a\,b=1\\ &\forall pq.\forall c\in\mathrm{Comp}.\forall b\in\mathrm{Bool}.\,\mathrm{eval}\,p\,c\,b=1\Rightarrow\mathrm{eval}\,(p\cup q)\,c\,b=1\\ &\forall pq.\forall t\in\mathrm{Tree}.\;\mathrm{subHtree}\,p\,t=1\Rightarrow\;\mathrm{subHtree}(p\cup q)\,t=1\;.\end{aligned}$$

For instance, we have $\mathrm{Mon}_{\mathrm{test}}:=\lambda xy.\,x\,y$. ◀

▶ Remark. In practice, there is no need to build formal proofs in PAω^+ of monotonicity since their unrelativized version are realized by the identity (they are Horn formulas, true in the standard model): we can use them in proofs as axioms and later realize them by I.

▶ **Lemma 5.5** (FVal is upward-closed). *For all p and q, if $(p\cup q)\in\mathrm{FVal}$, then $p\in\mathrm{FVal}$.*

Proof. There exists a proof term $\mathrm{Up}_{\mathrm{FVal}}\;:\;\forall pq.\,(p\cup q)\in\mathrm{FVal}\Rightarrow p\in\mathrm{FVal}$. ◀

5.5 The full proof

5.5.1 The big picture

Now that we have our forcing setting, we can turn to the proof itself. It will be split between the base (**B**) and forcing universes (**F**) as shown by the following steps:

1. **B** Assume the premise $\forall \rho^{\iota \to o}. \exists \vec{v} \in \mathrm{Term}. \neg\, \mathrm{interp}\, \rho\, (V\, \vec{v})$.
2. **F** Lift the premise to the forcing universe.
3. **F** Make the proof: $t : \mathrm{subH}\, \emptyset$.
4. **B** Use the forcing translation: $t^* : 1 \Vdash \mathrm{subH}\, \emptyset$.
5. **B** Remove forcing: $\xi_{\mathrm{subH}}\, t^* \alpha_0 : \mathrm{subH}\, \emptyset$.
6. **B** Extract a witness.

▶ Remarks.

1. Steps 1 and 2 are automatic (a proof in the base universe is correct in the forcing one),
2. Step 5 has already been explained in the general case,
3. Step 6 uses standard classical realizability techniques and will not be discussed here.
4. Since the premise is not absolute (because of the quantification over valuations ρ of sort $\iota \to o$), we do not have a proof of Herbrand's theorem (in the base universe) and only get this admissible rule (see section 2.1): $\dfrac{\mathcal{E}; \Gamma \vdash u : \forall \rho^{\iota \to o}. \exists \vec{v} \in \mathrm{Term}.\ \mathrm{interp}\, \rho\, (V\, \vec{v})}{\mathcal{E}; \Gamma \vdash t(u) : \mathrm{subH}\, \emptyset}$.

5.5.2 The proof in the forcing universe (step 3)

Recall the formal statement of Herbrand's theorem (H) given in section 5.4. Since we are now in the forcing universe, we can use the properties of the generic filter G given in section 4.3. As usual with proof in forcing, we start by building the generic valuation $g = \bigcup G$, which is legal because G is a filter. We would like to let $g := \bigcup G$ and prove that it is total. However, its simpler to define $g := \lambda a. \exists p \in G.\ \mathrm{test}\, p\, a\, 1 = 1$ (total by definition) and then prove that it is equal to the union of G. To do so, instead of full genericity, we use a specialized axiom

$$\forall a \in \mathrm{Atom}. \exists p \in G. \exists b \in \mathrm{Bool}.\ \mathrm{test}\, p\, a\, b = 1\ . \tag{A}$$

First of all, we lift this axiom to quantifier-free formulas:

▶ **Lemma 5.6** (Evaluation by G)**.** *There exists a proof term proving the proposition*

$$\forall c \in \mathrm{Comp}. \exists p \in G. \exists b \in \mathrm{Bool}.\ \mathrm{eval}\, p\, c\, b = 1\ \&\ \textit{if}\ b\ \textit{then}\ \mathrm{interp}\, g\, c\ \textit{else}\ \neg(\mathrm{interp}\, g\, c)\ .$$

Proof. The second part of the conjunct simply says that g must interpret a quantifier-free formula c exactly as any p in G would do, which is obvious by definition of g. We can therefore focus our attention on the first part on the conjunct, which is proved by induction on c, using property (4.9.iv) for the case of implication and axiom (A) for the case of atom. ◀

Because g is a valuation, we can feed it to the premise of (H) to get terms $\vec{v}$ such that $\vec{v} \in \mathrm{Term}$ (1) and $\neg\, \mathrm{interp}\, g\, (V\, \vec{v})$ (2). Using lemma 5.6 above with $V\, \vec{v}$, we get $p \in G$ and $b \in \mathrm{Bool}$ such that $\mathrm{eval}\, p\, (V\, \vec{v})\, b = 1$ (3) and if b then $\mathrm{interp}\, g\, (V\, \vec{v})$ else $\neg(\mathrm{interp}\, g\, (V\, \vec{v}))$ (4). Since $b \in \mathrm{Bool}$, we can make a case analysis:

1. $b = 1$: By (4), we have $\mathrm{interp}\, g\, (V\, \vec{v})$ which is in contradiction with (2).
2. $b = 0$: The equation (3) gives us $\mathrm{eval}\, p\, (V\, \vec{v})\, 0 = 1$ which, combined with (1), makes a proof of $\mathrm{subH}\, p$ (take $t := \mathrm{Leaf}\, \vec{v}$). But $p \in G$ and $G \subset C$ so that we have $C[p]$ and thus $\mathrm{subH}\, p \Rightarrow \mathrm{subH}\, \emptyset$ which allows us to conclude.

$$\begin{array}{l}\lambda caf.\ \text{let}\ (p,t) = \alpha_1\, c\ \text{in} \qquad\qquad a' := \xi_{\mathrm{Atom}}\, a\, (\alpha_1\, c)\ :\ a \in \mathrm{Atom}\\ \quad \text{if}\ \ \mathrm{Tot}_{\mathrm{test}}\ p\, a'\ \text{true}\ \ \text{then}\ f\,(\alpha_1\, c)\ \mathrm{I}\ \mathrm{true}^*\ \mathrm{I}^*\ \ \text{else}\\ \quad \text{if}\ \ \mathrm{Tot}_{\mathrm{test}}\ p\, a'\ \text{false}\ \ \text{then}\ f\,(\alpha_1\, c)\ \mathrm{I}\ \mathrm{false}^*\ \mathrm{I}^*\ \ \text{else}\\ \quad f\,\langle \mathrm{Up}_{\mathrm{FVal}}\,(\mathrm{consT}\, a'\, p), \lambda t_1.\, f\,\langle \mathrm{Up}_{\mathrm{FVal}}\,(\mathrm{consF}\, a'\, p), \lambda t_2.\, t\,(\mathrm{merge}\, a'\, t_1\, t_2)\rangle\ \mathrm{I}\ \mathrm{false}^*\ \mathrm{I}^*\rangle\ \mathrm{I}\ \mathrm{true}^*\ \mathrm{I}^*\end{array}$$

Figure 7 The program realizing the axiom (A).

5.5.3 Back to the base universe (step 4)

Converting our proof term $t : \mathrm{subH}\,\emptyset$ in the forcing universe into a proof term $t^* : 1 \Vdash \mathrm{subH}\,\emptyset$ in the base universe follows exactly the methodology of section 4. The only subtlety is that instead of the genericity property of G (property (4.9.v)), we use the axiom (A) and we now need to translate it.

▶ **Proposition 5.7** (Forcing the axiom (A)). *There is a proof term in $PA\omega^+$ proving*

$$1 \Vdash \forall a \in \mathrm{Atom}.\, \neg(\forall pb.\, p \in G \Rightarrow b \in \mathrm{Bool} \Rightarrow \mathrm{test}\, p\, a\, b = 1 \Rightarrow \bot)\ .$$

Proof. The corresponding realizer is given in Fig. 7. Note that we use the simplified version of α_1. The (textual) proof is given in annex B. ◀

6 Computational interpretation

By analyzing the proof from the previous section, we obtain an algorithm for computing Herbrand trees. In order to study this algorithm, we use Krivine's classical realizability (see section 3.4), the setting in which the computational content of forcing has been studied [9, 11].

Overall, the interest of using forcing in this case is twofold. First, it allows to reason (in the forcing universe) on a single valuation, the *generic valuation*, instead of considering all of them. The forcing translation takes care of 'moving' this generic valuation across the tree to make sure we cover every possible branch. In short, forcing transparently manages the tree structure. Second, the forcing condition stores the tree under construction (see below), thus protecting it from any backtrack that might occur in the realizer of the premise of (H).

Computationally, a realizer of $C[p]$ is a dependent type of a zipper [6] at position p. Its first part ($p \in \mathrm{FVal}$) behaves as a finite list of atoms with two *cons* constructors, representing a finite approximation of the generic valuation g. Its second part ($\mathrm{subH}\, p \Rightarrow \mathrm{subH}\,\emptyset$) is the return continuation: provided we can find a Herbrand tree below p, we have a full Herbrand tree; it represents a *tree context* where the hole is at position p.

From this perspective, the key ingredient of the proof is axiom (A), which is responsible for the insertion of new nodes in the Herbrand tree and the scheduling of the computation of the subtrees. Indeed, it is the only place where the second component of the forcing condition (the tree context) is modified. It can be seen as the primitive called by the user program (the premise) to build the tree, like a system call giving access to g: given an atom a, this program (given Fig. 7) computes the truth value b of a in g, together with a witness of its answer: $p \in G$ containing a (remember that $g = \bigcup G$). To do so, it first checks whether a belongs to the current forcing condition q and if so, returns the associated value (lines 2 & 3) by feeding it to its continuation f. When a does not belong to q, we need to extend q. Since a can be mapped to either true or false in g, we consider both cases and hence make two calls to f (last line). These two calls can be understood intuitively as follows: first we lead f to believe we have a tree context for $p := qa^1$ (*i.e.* a fictitious realizer T' of $\mathrm{subH}\, qa^1 \Rightarrow \mathrm{subH}\,\emptyset$) although at the time, we only have one for q. When the computation inside f uses T', it

must provide a Herbrand tree t_1 below qa^1. We then swap branches and call f again with $p := qa^0$ because this time, we do have a tree context for qa^0, namely $\lambda t_2.\, t\,(\mathrm{merge}\, a\, t_1\, t_2)$. Summing up, this last line contains both the extension of the tree (in $\mathrm{merge}\, a\, u\, v$) and the scheduling of the subtree computation (the two calls to f).

Furthermore, our realizer is completely intuitionistic (no callcc), which means that any backtrack during execution originates from the realizer of the premise of (H) and cannot affect the partial tree under construction which is stored in the second part of $C[p]$. Indeed callcc* takes care of saving and restoring the forcing condition. This restricted form of backtrack becomes a real instruction in the KFAM [12] (Krivine's Forcing Abstract Machine) which hard-wires the forcing translation of section 4 and features two *execution modes*:

- a *real mode* where terms have their usual KAM behavior,
- a *forcing mode* (or *protected mode*) where the first slot on the stack is considered as a forcing condition and terms behave as if they were translated through the forcing transformation.

In this machine, the premise of Herbrand's theorem would be executed only in forcing mode and could not affect the forcing condition (stored on the first slot of the stack).

Finally, the proof of section 5.5.2 in the forcing universe $\mathrm{PA}\omega^+ + G$ never uses the upward closure of G (property 4.9.iii). This means that we do not need to erase information from the partial Herbrand tree and suggests that our realizer is efficient.

References

1 Paul J. Cohen. The independence of the continuum hypothesis. *Proceedings of the National Academy of Science of the USA*, 50:1143–1148, 1963.

2 Paul J. Cohen. The independence of the continuum hypothesis II. *Proceedings of the National Academy of Science of the USA*, 51:105–110, 1964.

3 P.-L. Curien and Hugo Herbelin. The duality of computation. In *International Conference on Functional Programming*, pages 233–243, 2000.

4 Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo. *Journal of Automated Reasoning*, 31(1):33–72, 2003.

5 Timothy G. Griffin. A formulae-as-types notion of control. In *Principles of Programming Languages (POPL'90)*, pages 47–58, 1990.

6 Gérard Huet. The zipper. *Journal of Functional Programming*, 7(5):549–554, 1997.

7 Thomas Jech. *Set theory*. Springer Monographs in Mathematics. Springer-Verlag, Berlin, 2003. The third millennium edition, revised and expanded.

8 J.-L. Krivine. Realizability in classical logic. In *Interactive models of computation and program behaviour*, volume 27 of *Panoramas et synthèses*, pages 197–229. SMF, 2009.

9 J.-L. Krivine. Realizability algebras: a program to well order $\mathbb{R}$. *Logical Methods in Computer Science*, 7, 2011.

10 Alexandre Miquel. Existential witness extraction in classical realizability and via a negative translation. In *Logical Methods in Computer Science*, 2010.

11 Alexandre Miquel. Forcing as a program transformation. *Logic in Computer Science*, pages 197–206, 2011.

12 Alexandre Miquel. Forcing as a program transformation. *Mathematical Structure in Computer Science*, 2013. to appear.

13 Michel Parigot. Proofs of strong normalisation for second order classical natural deduction. *Journal of Symbolic Logic*, 62(4):1461–1479, 1997.

A Definition of the congruence relation

Reflexivity, symmetry, transitivity and base case

$$\frac{}{M \simeq_{\mathcal{E}} M} \qquad \frac{M \simeq_{\mathcal{E}} N}{N \simeq_{\mathcal{E}} M} \qquad \frac{M \simeq_{\mathcal{E}} N \qquad N \simeq_{\mathcal{E}} P}{M \simeq_{\mathcal{E}} P}$$

$$\frac{}{M \simeq_{\mathcal{E}} N}\ (M = N) \in \mathcal{E}$$

Context closure

$$\frac{M \simeq_{\mathcal{E}} N}{\lambda x.\, M \simeq_{\mathcal{E}} \lambda x.\, N} \qquad \frac{A \simeq_{\mathcal{E}} B}{\forall x^{\tau}.\, A \simeq_{\mathcal{E}} \forall x^{\tau}.\, B} \qquad \frac{M \simeq_{\mathcal{E}} N \qquad P \simeq_{\mathcal{E}} Q}{M\, P \simeq_{\mathcal{E}} N\, Q}$$

$$\frac{A \simeq_{\mathcal{E}} B \qquad C \simeq_{\mathcal{E}} D}{A \Rightarrow C \simeq_{\mathcal{E}} B \Rightarrow D} \qquad \frac{M \simeq_{\mathcal{E}} N \qquad P \simeq_{\mathcal{E}} Q \qquad A \simeq_{\mathcal{E}, M \doteq P} B}{M \doteq P \mapsto A \simeq_{\mathcal{E}} N \doteq Q \mapsto B}$$

$\beta\,\eta\,\iota$-conversion

$$\frac{}{(\lambda x^{\tau}.\, M)\, N^{\tau} \simeq_{\mathcal{E}} M[N^{\tau}/x^{\tau}]} \qquad \frac{}{\lambda x.\, M\, x \simeq_{\mathcal{E}} M}\ x \notin FV(M)$$

$$\frac{}{\mathrm{rec}_{\tau}\, M\, N\, 0 \simeq_{\mathcal{E}} M} \qquad \frac{}{\mathrm{rec}_{\tau}\, M\, N (S\, P) \simeq_{\mathcal{E}} N\, P\, (\mathrm{rec}_{\tau}\, M\, N\, P)}$$

Semantically equivalent propositions

$$\frac{}{\forall x^{\tau} \forall y^{\sigma}.\, A \simeq_{\mathcal{E}} \forall y^{\sigma} \forall x^{\tau}.\, A} \qquad \frac{}{\forall x^{\tau}.\, A \simeq_{\mathcal{E}} A}\ x \notin FV(A)$$

$$\frac{}{A \Rightarrow \forall x^{\tau}.\, B \simeq_{\mathcal{E}} \forall x^{\tau}.\, A \Rightarrow B}\ x \notin FV(A)$$

$$\frac{}{M \doteq M \mapsto A \simeq_{\mathcal{E}} A} \qquad \frac{}{M \doteq N \mapsto A \simeq_{\mathcal{E}} N \doteq M \mapsto A}$$

$$\frac{}{M \doteq N \mapsto P \doteq Q \mapsto A \simeq_{\mathcal{E}} P \doteq Q \mapsto M \doteq N \mapsto A}$$

$$\frac{}{A \Rightarrow M \doteq N \mapsto B \simeq_{\mathcal{E}} M \doteq N \mapsto A \Rightarrow B}$$

$$\frac{}{\forall x^{\tau}.\, M \doteq N \mapsto A \simeq_{\mathcal{E}} M \doteq N \mapsto \forall x^{\tau}.\, A}\ x \notin FV(M, N)$$

B Proof that the axiom (A) is forced

We want to prove (in PAω^{+}) that $1 \Vdash \forall a \in \mathrm{Atom}.\, \exists p \in G.\, \exists b \in \mathrm{Bool}.\, \mathrm{test}\, p\, a\, b = 1$. Unfolding the existential quantifiers, we need to prove

$$1 \Vdash \forall a \in \mathrm{Atom}.\, \neg(\forall p \in G.\, \forall b \in \mathrm{Bool}.\, \mathrm{test}\, p\, a\, b = 1 \Rightarrow \bot),$$

that is

$$1 \Vdash \forall a.\, a \in \mathrm{Atom} \Rightarrow \neg(\forall p \forall b.\, p \in G \Rightarrow b \in \mathrm{Bool} \Rightarrow \mathrm{test}\, p\, a\, b = 1 \Rightarrow \bot)\ .$$

Using proposition 4.2, it amounts to proving $(1 q_a) q_f \Vdash \bot$ given

$$x_a :\ q_a \Vdash a \in \mathrm{Atom}$$
$$x_f :\ q_f \Vdash \forall p \forall b.\, p \in G \Rightarrow b \in \mathrm{Bool} \Rightarrow \mathrm{test}\, p\, a\, b = 1 \Rightarrow \bot\ .$$

Since 1 is neutral for the product, this is the same as proving that $q_a q_f \Vdash \bot$. With repeated use of γ_3, we can turn x_f into $y := \lambda uv.\, \gamma_3(\gamma_3\, (\gamma_3\, (\beta_4\, y)\, u)\, v)$ which is a proof term for

$$\forall q \forall p \forall b.\, (q q_f \Vdash p \in G) \Rightarrow (q q_f \Vdash b \in \mathrm{Bool}) \Rightarrow (q q_f \Vdash \mathrm{test}\, p\, a\, b = 1) \Rightarrow (q q_f \Vdash \bot)\ .$$

Because test is total and we have $\mathrm{Tot}_{\mathrm{test}} :\ \forall p \in \mathrm{FVal}. \forall a \in \mathrm{Atom}. \forall b \in \mathrm{Bool}.\, \mathrm{test}\, p\, a\, b \in \mathrm{Bool}$ (both assumed in the interface for finite relations), we can proceed by case analysis:

- $\mathrm{test}\,(q_a q_f)\,a\,1 = 1$: We take $p := q_a q_f$ and $b := 1$. We use y with $q := q_a$ to prove $q_a q_f \Vdash \bot$ so that we have to prove its premises:
  - $\mathrm{I} : q_a q_f \leq q_a q_f \equiv q_a q_f \Vdash q_a q_f \in G$,
  - $\gamma_1(\lambda u.\, \gamma_1(\lambda v.\, \beta_4\,(\beta_3\, u))) : q_a q_f \Vdash 1 \in \mathrm{Bool}$,
  - $\mathrm{I}^* : q_a q_f \Vdash \mathrm{test}\,(q_a q_f)\,a\,1 = 1$ because the equality holds in the equational theory (thanks to the case analysis).
- $\mathrm{test}\,(q_a q_f)\,a\,0 = 1$: It is similar to the previous case.
- $\mathrm{test}\,(q_a q_f)\,a\,1 = 0$ and $\mathrm{test}\,(q_a q_f)\,a\,0 = 0$: This case means that a does not appear in $q_a q_f$.

  We use $\xi'_\bot$ and we are left to prove $C[q_a q_f] \Rightarrow \bot$. We first prove $C[(q_a a^1) q_f] \Rightarrow \bot$ by using first $\xi_\bot$ then y with $q \equiv p := q_a a^1$ and $b := 1$.
  - $\alpha_9 : (q_a a^1) q_f \leq q_a a^1 \equiv (q_a a^1) q_f \Vdash q_a a^1 \in G$ because product is the glb for $\leq$,
  - $(q_a a^1) q_f \Vdash 1 \in \mathrm{Bool}$ proved as before,
  - $(q_a a^1) q_f \Vdash \mathrm{test}\,(q_a a^1)\,a\,1 = 1$ proved as before.

  In a similar fashion, we prove $C[(q_a a^0) q_f] \Rightarrow \bot$.

  Let us come back to the proof of $C[q_a q_f] \Rightarrow \bot$. Assume $C[q_a q_f]$. Applying the proof term for $C[(q_a a^1) q_f] \Rightarrow \bot$, we have to prove $C[(q_a a^1) q_f] \equiv (q_a a^1) q_f \in \mathrm{FVal} \wedge (\mathrm{subH}\,(q_a a^1) q_f \Rightarrow \mathrm{subH}\,\emptyset)$. The first part present no difficulty because we have $C[q_a q_f]$ and $\mathrm{mem}\,a\,(q_a q_f) = 0$: we just need to apply the second constructor of FVal. For the second part, we assume $\mathrm{subH}\,(q_a a^1) q_f$ and we want to prove $\mathrm{subH}\,\emptyset$. Instead we choose to prove $\bot \equiv \forall Z.\, Z$. Applying again the same method with $C[(q_a a^0) q] \Rightarrow \bot$, we end up proving $\mathrm{subH}\,\emptyset$ with $\mathrm{subH}\,(q_a a^1) q$ and $\mathrm{subH}\,(q_a a^0) q_f$ as extra hypotheses. For this, we just have to use merge (proposition 5.3) to get $\mathrm{subH}\, q_a q_f$ and apply it to $\mathrm{subH}\, q_a q_f \Rightarrow \mathrm{subH}\,\emptyset$ which we get from $C[q_a q_f]$.

The Complexity of Abduction for Equality Constraint Languages

Johannes Schmidt* **and Michał Wrona†**

Linköping University
IDA
Linköping
{johannes.schmidt, michal.wrona}@liu.se

Abstract

Abduction is a form of nonmonotonic reasoning that looks for an explanation for an observed manifestation according to some knowledge base. One form of the abduction problem studied in the literature is the propositional abduction problem parameterized by a structure Γ over the two-element domain. In that case, the knowledge base is a set of constraints over Γ, the manifestation and explanation are propositional formulas.

In this paper, we follow a similar route. Yet, we consider abduction over infinite domain. We study the equality abduction problem parameterized by a relational first-order structure Γ over the natural numbers such that every relation in Γ is definable by a Boolean combination of equalities, a manifestation is a literal of the form $(x = y)$ or $(x \neq y)$, and an explanation is a set of such literals. Our main contribution is a complete complexity characterization of the equality abduction problem. We prove that depending on Γ, it is Σ_2^P-complete, or NP-complete, or in P.

1998 ACM Subject Classification F.4.1 Mathematical Logic, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Abduction, infinite structures, equality constraint languages, computational complexity, algebraic approach

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.615

1 Introduction

Abduction is a form of logical inference that aims at finding explanations for observed manifestations, starting from some knowledge base. It found many different applications in artificial intelligence [21], in particular to explanation-based diagnosis (e.g. medical diagnosis [10]), text interpretation [18], and planning [17].

In this paper we are interested in the complexity of abduction in a well-defined framework explained below. In a certain sense we follow a series of papers concerning the complexity of propositional abduction [16, 15, 20]. Roughly speaking, an instance of a propositional abduction problem for a relational structure Γ over the two element domain consists of a *knowledge base* KB — a conjunction of constraints over Γ, a set of *hypotheses* H — propositional literals formed upon variables in KB, and a *manifestation* M — a propositional formula. The question is whether there exists an *explanation*, i.e., a set $E \subseteq H$ such that $(\mathrm{KB} \wedge \bigwedge E)$ is satisfiable and $(\mathrm{KB} \wedge \bigwedge E)$ entails M. For every Γ, this propositional abduction problem is in Σ_2^P [16]. Depending on the restrictions, e.g., on Γ, one can obtain variants which are polynomial, NP-complete, coNP-complete or Σ_2^P-complete [20].

* Supported by the National Graduate School in Computer Science (CUGS), Sweden.
† Supported by the *Swedish Research Council* (VR) under grant 621-2012-3239.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca ; pp. 615–633

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Here, we follow a similar scheme. The difference is that Γ is not a structure over the two-element domain but a structure that has a first-order definition in $(\mathbb{N};=)$, that is, the set of natural numbers with equality only. In what follows we call such structures *equality (constraint) languages* and the corresponding abduction problem *the equality abduction problem*. Since all structures first-order definable in $(\mathbb{N};=)$ have also first-order definitions in all other infinite structures, it is natural to start classifying the complexity of abduction for infinite structures considering equality languages first. The motivation for studying abduction for infinite constraint languages is strong and presented below. Equality languages are also of independent interest. They were studied in the context of CSPs [5] and QCSPs [3], see also Section 4.1. In [4], these languages were classified with respect to primitive positive definability.

An instance of the equality abduction problem for Γ consists of a knowledge base ϕ that is a conjunction of constraints over Γ, a subset V of the set of variables occurring in ϕ, and a manifestation which is a literal $L(x, y)$ of the form $(x = y)$, or $(x \neq y)$. The question is whether there exists an explanation, i.e., a conjunction ψ of such literals formed upon variables in V such that $(\phi \wedge \psi)$ is satisfiable and $(\phi \wedge \psi)$ entails $L(x, y)$. For instance, a possible explanation for the knowledge base $((x_1 = y_1 \wedge x_2 = y_2) \rightarrow z = v)$ and the manifestation $(z = v)$ is $(x_1 = y_1 \wedge x_2 = y_2)$. We give a precise definition in Section 4. The notions used in the introduction are pretty standard. Most of them are, nevertheless, defined in Section 2.

The main contribution of this paper is a trichotomous complexity classification of the equality abduction problem. As we show, this problem is always in Σ_2^P. Moreover, depending on Γ, it may be Σ_2^P-hard, or NP-complete, or solvable in polynomial time.

This way of parameterizing computational problems by relational structures is also referred to as *Schaefer's framework* or *Schaefer's approach* and dates back to Schaefer's paper on the complexity of *constraint satisfaction problems (CSPs)* over the two-element domain [22]. Modern proofs of Schaefer's theorem (see, e.g., [12]), as well as many other classifications of the complexity of related problems (for a survey, see [14]) including also propositional abduction take advantage of the so-called *algebraic approach*. In this approach the complexity of the problem, e.g., a constraint satisfaction problem or an abduction problem for a fixed Γ, is related directly to the set of operations preserving Γ. To enjoy the benefits of the algebraic tools, it is not necessary to restrict to the two-element domain, neither to any finite domain. Indeed, algebraic tools are equivalently powerful in classifying the complexity of CSPs for certain infinite structures [2], called ω-categorical structures [19].

Very natural examples of such structures are $(\mathbb{N};=)$ but also $(\mathbb{Q};<)$, that is, the order of rational numbers. Several classifications of constraint satisfaction problems for ω-categorical structures were obtained in the literature [2]. All of them follow the following scheme. One starts from some ω-categorical structure Δ such as $(\mathbb{N};=)$ or $(\mathbb{Q};<)$, then considers the class of all structures Γ with a first-order definition in Δ, which are also ω-categorical. The cases where Δ is $(\mathbb{N};=)$ and $(\mathbb{Q};<)$ were treated in [5] and [6], respectively. In both situations it appeared that the problem CSP(Γ) is either in P, or it is NP-complete. Here, we adopt this framework to study the complexity of the equality abduction problem.

To motivate the study of CSPs for ω-categorical structures, it is worth to mention that many problems studied independently in temporal reasoning, e.g., network satisfaction problems for qualitative calculi such as the Point Algebra [23] or Allen's Interval Algebra [1] can be directly formulated in this framework. In fact the complexity classification of Γ with a first-order definition in $(\mathbb{Q};<)$ substantially generalizes the result on tractability for the network satisfaction problem for the Point Algebra. Furthermore, the network satisfaction

problem for Allen's Interval Algebra may be also modelled as a CSP for an ω-categorical structure, see [2].

Back to the issue of abduction, a kind of this problem handling time dependencies between events is called temporal abduction. Among others it is studied in [8, 13] in the framework [9] based on already mentioned, formalisms: Point Algebra, and Allen's Interval Algebra. It motivates the study of the complexity of abduction for ω-categorical structures, which we initiate in this paper.

1.1 Outline of the paper

We start in Section 2 by providing some preliminaries. In Section 3 we give a general definition of an abduction problem for a relational structure Γ. This definition captures many variants of propositional abduction as well as the equality abduction problem we study in this paper. We believe that the general definition will be employed in our future research on abduction. In that section we also show that two primitive positive interdefinable structures Γ_1 and Γ_2 give rise to abduction problems that are polynomial-time equivalent. Using the Galois connection in [7], it follows that if Γ_1 and Γ_2 are ω-categorical and preserved by the same operations, then their abduction problems are polynomial-time equivalent. We conclude that part of the paper by proving a useful result which links the complexity of the abduction and the constraint satisfaction problem.

A set of operations preserving a given structure Γ forms an algebraic structure called a *clone*. Clones corresponding to equality languages were classified in [4]. To provide our classification, which is presented in detail in Section 4, we express it in terms of clones, see Section 5. Then it remains to prove the complexity results, which are provided in Sections 6, 7, and 8. The paper is concluded in Section 9, where also the issue of future work is adressed.

The appendix contains proofs of: Theorem 5, Proposition 26, Lemma 28, and Proposition 29.

2 Preliminaries

Always, when it is possible, the notation is consistent with [19], [11], and [2], which we recommend as further reading on model theory, ω-categoricity and CSPs over ω-categorical structures, respectively. We write $[n]$ to denote $\{1, \dots, n\}$.

2.1 Structures and Formulas

In this paper, we consider relational structures, which are typically denoted here by capital Greek letters such as Γ, or Δ. A signature is usually denoted by τ. If it is not stated otherwise, then we assume that the signature is finite. For the sake of simplicity, we use the same symbols to denote relations and their corresponding relation symbols. We mainly focus on countably infinite and ω-categorical structures. We say that a countably infinite structure is *ω-categorical* if all countable models of its first-order theory are isomorphic.

Let σ and τ be signatures with $\sigma \subseteq \tau$. When Δ is a σ-structure and Γ is a τ-structure with the same domain such that $R^\Delta = R^\Gamma$ for all $R \in \sigma$, then Γ is called an *expansion* of Δ.

For a τ-structure Γ over the domain D we define $\Delta := \Gamma^k$, where $k > 0$ is a natural number, to be a k-fold direct product of Γ, that is, the τ-structure on the domain D^k such that for every n-ary relation symbol R in τ we have $((d_1^1, \dots, d_k^1), \dots, (d_1^n, \dots, d_k^n)) \in R^\Delta$ iff $(d_i^1, \dots, d_i^n) \in R^\Gamma$ for all $i \in [k]$.

In this paper, we say that a relational structure Γ is first-order definable in Δ if Γ has the same domain as Δ, and for every relation R of Γ there is a first-order formula ϕ in the signature of Δ such that ϕ holds exactly on those tuples that are contained in R. If Γ is first-order definable in Δ, then we say that Γ is a *first-order reduct* of Δ. We say that two formulas are equivalent if they are over the same variables and define the same relation.

We are in particular interested in *equality (constraint) languages*, that is, first-order reducts of $(\mathbb{N}; =)$. All equality languages are ω-categorical structures. Furthermore, since $(\mathbb{N}; =)$ has quantifier elimination, every equality language has a quantifier-free first-order definition in $(\mathbb{N}; =)$ in conjunctive normal form. Such formulas over the signature $\{=, \neq\}$ will be called *equality formulas*, and equality formulas of the form $(x = y)$ and $(x \neq y)$ will be called *(equality) literals*. The set of all literals that can be formed upon a set of variables V will be denoted by $\mathcal{L}(V)$.

A Γ-constraint is an atomic formula over the signature of Γ of the form $R(x_1, \ldots, x_n)$. Of special interest for abduction are Γ-formulas which are conjunctions of Γ-constraints. Furthermore, *primitive positive formulas (pp-formulas)* over the signature of Γ are first-order formulas built exclusively from conjunction, existential quantifiers, Γ-constraints and atomic formulas of the form $(x = y)$. The set of relations with a pp-definition in Γ is denoted by $[\Gamma]$.

For a quantifier-free first-order formula ϕ, we write $\mathrm{Var}(\phi)$ to denote the set of variables occurring in ϕ. Let ϕ_1 and ϕ_2 be two equality formulas over the same set of variables $\{v_1, \ldots, v_n\}$. We say that ϕ_1 entails ϕ_2 if $(\mathbb{N}; =) \models (\forall v_1 \cdots \forall v_n . \phi_1 \rightarrow \phi_2)$.

2.2 Polymorphisms and Clones

Let Γ be a structure. Homomorphisms from Γ^k to Γ are called *polymorphisms* of Γ. When R is a relation over domain D, we say that $f \colon D^k \rightarrow D$ *preserves* R if f is a polymorphism of $(D; R)$, and that f *violates* R otherwise. The set of all polymorphisms of a relational structure Γ, denoted by $\mathrm{Pol}(\Gamma)$, forms an algebraic object called a *clone*. A clone on some fixed domain D is a set of operations on D containing all projections and closed under composition. A clone $\mathcal{C}$ is *locally closed* iff for all natural numbers n, for all n-ary operations g on D, if for all finite $B \subseteq D^n$ there exists an n-ary $f \in \mathcal{C}$ which agrees with g on B, then $g \in \mathcal{C}$. We say that a set of operations F *(locally) generates* an operation f (or that an operation f is (locally) generated by F) if f is in the smallest locally closed clone containing F, denoted by $\langle F \rangle$. If $F = \{g\}$ then we also say that g generates f or that f is generated by g.

▶ Proposition 1 (see e.g. [2]). Let F be a set of operations on some domain D. Then the following are equivalent: (i) F is the polymorphism clone of a relational structure; and (ii) F is a locally closed clone.

For ω-categorical structures we have the following Galois connection.

▶ **Theorem 2** ([7]). *Let Γ_1, Γ_2 be ω-categorical structures. We have that $Pol(\Gamma_1) \subseteq Pol(\Gamma_2)$ if and only if $[\Gamma_2] \subseteq [\Gamma_1]$.*

A special kind of a polymorphism is an automorphism. Observe that the set of automorphisms of $(\mathbb{N}; =)$ is exactly $S_{\mathbb{N}}$, that is, the set of all permutations on $\mathbb{N}$. By the theorem of Engeler, Ryll-Nardzewski and Svenonius (see, e.g., [19]), it follows that a structure is an equality language if and only if it is preserved by $S_{\mathbb{N}}$. Thus, by Theorem 2 and results obtained in Section 3, studying the complexity of the equality abduction problem amounts to studying locally closed clones on $\mathbb{N}$ containing $S_{\mathbb{N}}$. In what follows, such clones will be called *equality clones*. These clones form a complete lattice, where the least element is the clone

generated by $S_{\mathbb{N}}$ and the greatest element is the set of all operations on $\mathbb{N}$, denoted here by $\mathcal{O}$. By $\mathcal{O}^{(k)}$, we denote a subset of $\mathcal{O}$ containing the operations of arity k. For a given family of clones $(C_i)_{i\in I}$, the meet is just an intersection $\bigcap_{i\in I} C_i$, and the join is $\langle \bigcup_{i\in I} C_i\rangle$. The lattice of equality clones was described in [4]. In this paper, we take advantage of this classification. The clones which are important for us are recalled in Section 5.

2.3 Complexity Classes

In this paper we study decision problems. The complexity classes we deal with are P, NP and Σ_2^P. Recall that $\Sigma_2^P = \mathrm{NP}^{\mathrm{NP}}$ is the class of decision problems solvable in nondeterministic polynomial time with access to an NP-oracle. In general, we write $\mathcal{C}_1^{\mathcal{C}_2}$ for the class of languages solvable in $\mathcal{C}_1$ with access to a $\mathcal{C}_2$-oracle.

2.4 Propositional Abduction

Abduction has been intensively studied in the propositional case, see e.g., [15] and [20] for complexity classifications. Similarly as in this paper, these classifications are based on the closure properties of constraint languages. To prove hardness results on the equality abduction problem, we will use the complexity classification for a special kind of propositional abduction problem called PQ-ABDUCTION(Γ), where Γ is a structure over the two-element domain. By $\mathrm{Lit}(V)$ we denote the set of propositional literals that can be formed upon variables in V. An instance of PQ-ABDUCTION(Γ) is a triple (ϕ, V, q), where ϕ is a Γ-formula, $V \subseteq \mathrm{Var}(\phi)$, and $q \in \mathrm{Var}(\phi) \setminus V$. We ask whether there is a set of literals $\mathrm{Lit} \subseteq \mathrm{Lit}(V)$ such that $(\phi \wedge \bigwedge \mathrm{Lit})$ is satisfiable but $(\phi \wedge \bigwedge \mathrm{Lit} \wedge \neg q)$ is not. We will need the following version of Theorem 7.6 in [15]. Beforehand, however, consider the following operations over the two-element domain: i) $\mathrm{majority}(b_1, b_2, b_3) = (b_1 \wedge b_2) \vee (b_1 \wedge b_3) \vee (b_2 \wedge b_3)$, ii) $\mathrm{minority}(b_1, b_2, b_3) = (b_1 \wedge \neg b_2 \wedge \neg b_3) \vee (\neg b_1 \wedge b_2 \wedge \neg b_3) \vee (\neg b_1 \wedge \neg b_2 \wedge b_3) \vee (b_1 \wedge b_2 \wedge b_3)$, iii) $\min(b_1, b_2) = b_1 \wedge b_2$, iv) $\max(b_1, b_2) = b_1 \vee b_2$, v) $c_0(b) = 0$, vi) $c_1(b) = 1$, vii) $\mathrm{opzero}(b_1, b_2, b_3) = b_1 \wedge (b_2 \vee b_3)$, and viii) $\mathrm{opone}(b_1, b_2, b_3) = b_1 \vee (b_2 \wedge b_3)$.

▶ **Theorem 3** ([15]). *Let Γ be a structure over the two-element domain.*

- *If Γ is preserved by i) majority, ii) minority, iii) min and c_1, iv) opzero, or v) opone, then PQ-ABDUCTION(Γ) is in P.*
- *Otherwise, if Γ is preserved by min or max, the problem PQ-ABDUCTION(Γ) is NP-complete.*
- *In all other cases, we have that PQ-ABDUCTION(Γ) is Σ_2^P-hard.*

3 The Abduction Problem and Algebra

Let Δ be a relational structure over some domain D and let $\Gamma, \mathcal{HYP}, \mathcal{M}$ be three first-order reducts of Δ. Let $\Gamma_{\mathcal{HYP}}$ be an expansion of Γ by the relations in $\mathcal{HYP}$, that is, the structure whose relations are either from Γ or $\mathcal{HYP}$; and $\Gamma_{\mathcal{HYP},\mathcal{M}}$ be an expansion of $\Gamma_{\mathcal{HYP}}$ by the relations in $\mathcal{M}$.

▶ **Definition 4.** An instance of the abduction problem $\mathrm{ABD}(\Gamma, \mathcal{HYP}, \mathcal{M})$ is a triple $T = (\phi, V, M)$, where:

- ϕ is a Γ-formula (the knowledge base),
- V is a subset of $\mathrm{Var}(\phi)$,
- M is an $\mathcal{M}$-constraint (the manifestation).

The triple $T = (\phi, V, M)$ is a positive instance of $\mathrm{ABD}(\Gamma, \mathcal{HYP}, \mathcal{M})$ if there exists an *explanation* for T, that is, a $\mathcal{HYP}$-formula ψ built upon variables from V such that both of the following hold:

- $(\phi \wedge \psi)$ is satisfiable in $\Gamma_{\mathcal{HYP}}$,
- $(\phi \wedge \psi)$ entails M (or equivalently, $(\phi \wedge \psi \wedge \neg M)$ is not satisfiable in $\Gamma_{\mathcal{HYP},\mathcal{M}}$).

In this case ψ is called an *explanation* for T.

This definition allows to model many variants of the propositional abduction problem as defined in [20]. For instance, the basic problem PQ-ABDUCTION(Γ) discussed in Section 2 (called V-ABD(Γ, PosLits) in [20]) can be modelled in the following way. We start from $\Delta = (\{0,1\}; T, F)$, where $T = \{(1)\}$ and $F = \{(0)\}$, and consider Γ with a first-order definition in Δ, that is, Γ may be an arbitrary structure over the two-element domain. Then, we set $\mathcal{HYP}$ to $(\{0,1\}; T, F)$, and $\mathcal{M}$ to $(\{0,1\}; T)$.

We observe in the following that the algebraic approach is applicable to the abduction problem under consideration. We first show that when $\mathcal{HYP}$ and $\mathcal{M}$ are fixed, then the complexity of $\mathrm{ABD}(\Gamma, \mathcal{HYP}, \mathcal{M})$ is fully determined by the set $[\Gamma]$, the closure of Γ under primitive positive definitions. For $\Gamma_1 \subseteq [\Gamma_2]$ and an instance $T_1 = (\phi_1, V, M)$ of $\mathrm{ABD}(\Gamma_1, \mathcal{HYP}, \mathcal{M})$, we create an instance $T_2 = (\phi_2, V, M)$ of $\mathrm{ABD}(\Gamma_2, \mathcal{HYP}, \mathcal{M})$ by transforming a Γ_1-formula ϕ_1 into a Γ_2-formula ϕ_2 in the following standard way: (1) replace in ϕ_1 every Γ_1-constraint by its pp-definition in Γ_2, (2) delete all existential quantifiers, (3) delete all equality constraints and identify variables that are linked by a sequence of $=$.

It is easily observed that this transformation preserves satisfiability. Similarly as in [20], one can show that an explanation for T_1 can be easily rewritten into an explanation for T_2, and vice versa.

▶ **Theorem 5.** *Let Δ be a relational structure and $\Gamma_1, \Gamma_2, \mathcal{HYP}, \mathcal{M}$ be first-order reducts of Δ, where Γ_1 and Γ_2 are over finite signatures. If Γ_1 has a pp-definition in Γ_2, then ABD$(\Gamma_1, \mathcal{HYP}, \mathcal{M})$ reduces to ABD$(\Gamma_2, \mathcal{HYP}, \mathcal{M})$ in polynomial time.*

By Theorems 2 and 5, we have that the complexity of $\mathrm{ABD}(\Gamma, \mathcal{HYP}, \mathcal{M})$ for ω-categorical Γ is fully captured by the set of polymorphisms preserving Γ.

▶ **Corollary 6.** *Let Δ be an ω-categorical structure and Γ_1, Γ_2, $\mathcal{HYP}$, and $\mathcal{M}$ first-order reducts of Δ, where Γ_1 and Γ_2 are over finite signatures. If Pol$(\Gamma_2) \subseteq$ Pol(Γ_1), then ABD$(\Gamma_1, \mathcal{HYP}, \mathcal{M})$ reduces to ABD$(\Gamma_2, \mathcal{HYP}, \mathcal{M})$ in polynomial time.*

We will conclude this section by providing a simple but useful link between the complexity of $\mathrm{CSP}(\Gamma_{\mathcal{HYP},\mathcal{M}})$ and $\mathrm{ABD}(\Gamma, \mathcal{HYP}, \mathcal{M})$.

▶ Proposition 7. Let Δ be a relational structure and $\Gamma, \mathcal{HYP}, \mathcal{M}$ be first-order reducts of Δ, where $\mathcal{HYP}$ is over a finite signature. If $\mathrm{CSP}(\Gamma_{\mathcal{HYP},\mathcal{M}})$ is in the complexity class $\mathcal{C}$, then $\mathrm{ABD}(\Gamma, \mathcal{HYP}, \mathcal{M})$ is in $\mathrm{NP}^{\mathcal{C}}$.

Proof. Let $T = (\phi, V, M)$ be an instance of $\mathrm{ABD}(\Gamma, \mathcal{HYP}, \mathcal{M})$. By assumption, the signature of $\mathcal{HYP}$ is finite and therefore we can assume that if an explanation for T exists, then it is of polynomial length with respect to the number of variables in V, and thereby with respect to the length of T. Thus, we can guess a ψ and verify in polynomial time with two calls to the $\mathcal{C}$-oracle whether the instances $(\phi \wedge \psi)$ and $(\phi \wedge \psi \wedge \neg M)$ of $\mathrm{CSP}(\Gamma_{\mathcal{HYP},\mathcal{M}})$ are, respectively, satisfiable and not satisfiable in $\Gamma_{\mathcal{HYP},\mathcal{M}}$. ◀

4 Equality Abduction

In this paper, we treat a special case of the abduction problem $\mathrm{ABD}(\Gamma, \mathcal{HYP}, \mathcal{M})$. In the rest of the paper, Γ is always an equality language over a finite signature, and $\mathcal{HYP}$ and $\mathcal{M}$ are always $(\mathbb{N}; =, \neq)$.

We now formally define the equality abduction problem. Recall from Section 2 that equality literals are equality formulas of the form $(x = y)$ or $(x \neq y)$; and that the set of all equality literals that can be formed upon variables in V is denoted by $\mathcal{L}(V)$.

▶ **Definition 8** (Equality Abduction Problem)**.** The equality abduction problem $\mathrm{ABD}(\Gamma)$ for an equality language Γ (over a finite signature) is the computational problem, whose instance is a triple $T = (\phi, V, L(x, y))$, where:

- ϕ is a Γ-formula,
- V is a subset of $\mathrm{Var}(\phi)$,
- $L(x, y)$ is a literal and $x, y \in \mathrm{Var}(\phi)$.

The triple T is a positive instance of $\mathrm{ABD}(\Gamma)$ if there exists an explanation for T, that is, a set of literals $\mathcal{L} \subseteq \mathcal{L}(V)$ such that both of the following hold:

- $(\phi \wedge \bigwedge \mathcal{L})$ is satisfiable in $(\mathbb{N}; =, \neq)$,
- $(\phi \wedge \bigwedge \mathcal{L})$ entails $L(x, y)$ (or equivalently, $(\phi \wedge \bigwedge \mathcal{L} \wedge \neg L(x, y))$ is not satisfiable in $(\mathbb{N}; =, \neq)$).

We would like to remark that since Γ is always over a finite signature, the complexity of $\mathrm{ABD}(\Gamma)$ does not depend on the representation of relations in Γ.

Consider the following example. Let $\Gamma = (\mathbb{N}; \mathbb{I})$, where $\mathbb{I} = \{(x, y, z) \mid (x = y \to y = z)\}$, be an equality language. Consider the instance $T = (\phi, \{x, y, v\}, (z = w))$ of $\mathrm{ABD}(\Gamma)$ where ϕ is $((x = y \to y = z) \wedge (v = z \to v = w))$. Consider the set of literals $\mathcal{L} = \{x = y, y = v\}$. Observe that $(\phi \wedge \bigwedge \mathcal{L})$ is equivalent to $((x = y \to y = z) \wedge (v = z \to v = w)) \wedge (x = y) \wedge (y = v)$. It is straightforward to verify that this formula is satisfiable and that it entails $(z = w)$. Therefore, $\mathcal{L}$ is an explanation. As we will see, the problem $\mathrm{ABD}(\mathbb{N}; \mathbb{I})$ is NP-complete.

The following classes of equality languages are crucial to understand the complexity of the equality abduction problem.

▶ **Definition 9.** We say that a first-order formula is a *negative (equality) formula* if it is a conjunction of clauses of the form

$$(x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k) \text{ or } (x = y).$$

A relation R is called *negative* if it can be defined by a negative formula. An equality language Γ is *negative* if every its relation is negative.

▶ **Definition 10.** We say that a first-order formula is a *Horn (equality) formula* if it is a conjunction of clauses of the form

$$(x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k \vee x = y),$$

where it is permitted that $k = 0$ and the clause is simply an equality, i.e., of the form $x = y$. It is also permitted that we skip the equality and the clause is simply a disjunction of disequalities. A relation R is called *Horn* if it can be defined by a Horn formula. An equality language Γ is *Horn* if every its relation is Horn.

We will now present the main contribution of this paper. The following theorem completely classifies the complexity of the equality abduction problem.

▶ **Theorem 11** (Complexity Classification of the Equality Abduction Problem)**.** *Let* Γ *be an equality language (over a finite signature). Then exactly one of the following holds.*
1. Γ *is negative and* $ABD(\Gamma)$ *is in P;*
2. Γ *is not negative but Horn and* $ABD(\Gamma)$ *is NP-complete;*
3. Γ *is not Horn and* $ABD(\Gamma)$ *is* Σ_2^P*-complete.*

We will now break the proof of Theorem 11 into smaller steps. Keeping in mind that NP^P is equal to NP, by Proposition 7 and the complexity results in [5], which are also discussed in Section 4.1, we have the following upper bounds.

▶ Proposition 12. Let Γ be an equality language. Then we have both of the following.
1. The problem $\mathrm{ABD}(\Gamma)$ is in Σ_2^P.
2. If Γ is Horn, then $\mathrm{ABD}(\Gamma)$ is in NP.

In the remainder of the paper, we focus on hardness and tractability results. We first characterize those equality languages for which the abduction problem is of the highest complexity.

▶ Proposition 13. Let Γ be an equality language. If Γ is not Horn, then $\mathrm{ABD}(\Gamma)$ is Σ_2^P-hard.

Then, we take care of those that are NP-hard.

▶ Proposition 14. Let Γ be an equality language. If Γ is not negative, then $\mathrm{ABD}(\Gamma)$ is NP-hard.

As we show, there is also a nontrivial class of equality abduction problems that are in P. For those, we will provide an appropriate algorithm.

▶ Proposition 15. Let Γ be an equality language. If Γ is negative, then $\mathrm{ABD}(\Gamma)$ is in P.

Propositions 13, 14, and 15 are proved in Sections 6, 7, and 8, respectively. Now assuming these propositions we will prove Theorem 11.

Proof of Theorem 11. It obviously holds exactly one of the following cases.
1. Γ is negative, or
2. Γ is not negative but Horn, or
3. Γ is not Horn.

We obtain the corresponding memberships by Propositions 15 and 12. The required hardness results follow by Propositions 13 and 14. ◀

4.1 Related Classifications on Equality Languages

As we already mentioned, equality languages were studied in the context of CSPs [5] and QCSPs [3]. Just to recall, an instance of $\mathrm{CSP}(\Gamma)$ may be seen as a Γ-formula where every variable is existentially quantified, and an instance of $\mathrm{QCSP}(\Gamma)$ as Γ-formula where every variable is either existentially or universally quantified. In both cases, the question is whether a given sentence is true in Γ.

The problem $\mathrm{CSP}(\Gamma)$ for an equality language Γ is always in NP, it is in P if Γ is Horn or it is preserved by a constant operation. Preservation under a constant function makes neither $\mathrm{QCSP}(\Gamma)$ nor $\mathrm{ABD}(\Gamma)$ tractable.

The problem $\mathrm{QCSP}(\Gamma)$ is always in PSPACE. Moreover, it is known to be in P if Γ is negative, and to be NP-hard if Γ is *positive*, that is, it may be defined as a conjunction of clauses of the form $(x_1 = y_1 \vee \cdots \vee x_k = y_k)$, but not negative. Otherwise $\mathrm{QCSP}(\Gamma)$ is coNP-hard. In particular $\mathrm{QCSP}(\mathbb{N};\mathbb{I})$ is coNP-hard. We remark that the equality abduction problem for positive equality languages is Σ_2^P-hard unless it may be defined as a conjunction of equalities (i.e., unless it is negative).

5 Equality Clones

To prove Theorem 11 we take advantage of the classification of equality clones classified in [4]. Here, we recall only the definitions of clones that are relevant to our classification.

We say that an operation $f : \mathbb{N}^n \to \mathbb{N}$ is *essentially unary* if there exists $i \in [n]$ and $g : \mathbb{N} \to \mathbb{N}$ such that $f(x_1, \ldots, x_i, \ldots, x_k) = g(x_i)$.

▶ **Definition 16.** For every $i \in \mathbb{N}$, we define $\mathcal{K}_i$ to be the set of operations containing all essentially unary operations as well as all operations whose range has at most i elements.

We say that an operation $f : \mathbb{N}^n \to \mathbb{N}$ is *up to fictitious coordinates, injective* if there exists $\{i_1, \ldots, i_k\} \subseteq [n]$ where $i_1 < \ldots < i_k$ and an injective function $g : \mathbb{N}^k \to \mathbb{N}$ such that for all $(x_1, \ldots, x_n) \in \mathbb{N}^n$ we have that $f(x_1, \ldots, x_n) = g(x_{i_1}, \ldots, x_{i_k})$.

▶ **Definition 17.** (The Horn clone $\mathcal{H}$). We define $\mathcal{H}$ to be the set of operations which are, up to fictitious coordinates, injective.

Let $i \in [n]$. We call an operation $f \in \mathcal{O}^{(n)}$ *injective in the i-th direction* if $f(a) \neq f(b)$ whenever $a, b \in \mathbb{N}^n$ and $a_i \neq b_i$. We say that $f \in \mathcal{O}^{(n)}$ is *injective in one direction* if there is an $i \in [n]$ such that f is injective in the i-th direction.

▶ **Definition 18.** (Richard $\mathcal{R}$). We define $\mathcal{R}$ to be the set of operations injective in one direction.

Let $f_3 \in \mathcal{O}^{(3)}$ be any operation satisfying the following.

- For all $a \in \mathbb{N}$ we have $f_3(a, 1, 1) = 1$, $f_3(2, a, 2) = 2$, and $f_3(3, 3, a) = 3$.
- For all other arguments, the function arbitrarily takes a value that is distinct from all other function values.

▶ **Definition 19.** (The odd clone $\mathcal{S}$). We define $\mathcal{S}$ to be the set of operations generated by f_3, and $\mathcal{S}^+$ to be the superset of $\mathcal{S}$ containing additionally all constant operations.

In [4], one can find the proof that the sets of operations $\mathcal{S}, \mathcal{S}^+, \mathcal{R}, \mathcal{H}$ and $\mathcal{K}_i$ for every $i \in \mathbb{N}$ are locally closed clones. The following theorem is a direct consequence of Theorems 8, 13, and 15 in that paper.

▶ **Theorem 20.** *Let Γ be an equality language. Then either*

1. *Pol(Γ) is contained in $\mathcal{K}_i$ for some $i \in \mathbb{N}$, or*
2. *Pol(Γ) contains $\mathcal{H}$. In this case:*
   1. *either Pol(Γ) is contained in $\mathcal{S}^+$, or*
   2. *Pol(Γ) contains $\mathcal{R}$.*

Further, we get from [4] (Propositions 43 and 68) the algebraic characterizations for negative constraint languages and Horn constraint languages.

▶ Proposition 21. Let Γ be an equality constraint language. Then:

1. Γ is negative if and only if $\mathcal{R} \subseteq \mathrm{Pol}(\Gamma)$;
2. Γ is Horn if and only if $\mathcal{H} \subseteq \mathrm{Pol}(\Gamma)$.

We will use the following version of Corollary 6.

▶ Proposition 22. Let Γ_1 and Γ_2 be equality languages such that $\mathrm{Pol}(\Gamma_2) \subseteq \mathrm{Pol}(\Gamma_1)$, then $\mathrm{ABD}(\Gamma_1)$ has a polynomial time reduction to $\mathrm{ABD}(\Gamma_2)$.

6 Σ_2^P-hard Equality Abduction Problems

We start by presenting an infinite family of relations $\mathbb{H}_2, \mathbb{H}_3, \ldots$ that give rise to the abduction problems whose complexity meets the upper bound from Proposition 12.

Let $i \in \mathbb{N} \setminus \{0,1\}$. We define $\mathbb{H}_i$ to be an equality relation of arity $(i+4)$ which is the union of: (1) $\{(b_0, \ldots, b_i, x, y, z) \in \mathbb{N}^{i+4} \mid (|\{b_0, \ldots, b_i, x, y, z\}| < i+1)\}$ and (2) $\{(b_0, \ldots, b_i, x, y, z) \in \mathbb{N}^{i+4} \mid \bigwedge_{k \neq l; k,l \in \{0,\ldots,i\}} (b_k \neq b_l) \wedge (x = b_0 \vee x = b_1) \wedge (y = b_0 \vee y = b_1) \wedge (z = b_0 \vee z = b_1) \wedge (b_0 = x \vee b_0 = y \vee b_0 = z) \wedge (b_1 = x \vee b_1 = y \vee b_1 = z)\}$.

Observe that item (1) has a first-order definition over $(\mathbb{N}; =)$. Indeed, one can define it by a conjunction of the formulas of the form $\neg(\bigwedge_{v,w \in S} v \neq w)$ where $S \subseteq \{b_0, \ldots, b_i, x, y, z\}$ is of size greater or equal than $(i+1)$.

The real purpose of this chapter is, however, to prove that ABD(Γ) is Σ_2^P-complete whenever Pol(Γ) $\subseteq \mathcal{K}_i$ for some i. The next lemma reduces that problem to showing that every ABD($\mathbb{N}; \mathbb{H}_i$) for every i is Σ_2^P-complete. The lemma also explains why item (1) is included in the definition of $\mathbb{H}_i$: assure that $\mathbb{H}_i$ is preserved by all operations in $\mathcal{K}_i$.

▶ **Lemma 23.** *Let $i \in \mathbb{N} \setminus \{0,1\}$. Then $\mathbb{H}_i$ is preserved by all operations in $\mathcal{K}_i$.*

Proof. Directly from the definition of $\mathbb{H}_i$, it follows that this relation contains all tuples with at most i pairwise different entries. Thus it is preserved by all operations with range of at most i elements. It remains to show that $\mathbb{H}_i$ is preserved also by all essentially unary operations. We therefore consider some $f : \mathbb{N} \to \mathbb{N}$ and $t \in \mathbb{H}_i$. Observe that either $f(t) = \alpha(t)$ for some automorphism α of $(\mathbb{N}; =)$, or the number of pairwise different entries in $f(t)$ is strictly smaller than in t. In the first case $f(t)$ is certainly in $\mathbb{H}_i$. Further, we observe that t has at most $(i+1)$ pairwise different entries. Hence in the second case, the tuple $f(t)$ has at most i pairwise different entries. Thus in this case, we are done by the observation from the first sentence of the proof. ◀

Let NAE $= \{(1,0,0), (0,1,0), (0,0,1), (0,1,1), (1,0,1), (1,1,0)\}$. To prove that ABD($\Gamma$), where $\Gamma = (\mathbb{N}; \mathbb{H}_i)$, is Σ_2^P-hard we reduce from the problem PQ-ABDUCTION(Δ) such that $\Delta = (\{0,1\}; \text{NAE})$. By Theorem 3, this problem is Σ_2^P-hard. Indeed, it is straightforward to verify that Δ is preserved by none of the following operations: majority, minority, min, max, opzero, opone. Let i be a natural number greater than or equal to 2. Observe that a tuple $(b_0, \ldots, b_i, x_p, x_r, x_s)$ of $\mathbb{H}_i$ with $b_0, \ldots, b_i$ pairwise different may be easily translated into a tuple (p, r, s) of NAE such that the value of $t \in \{p, r, s\}$ is $k \in \{0,1\}$ if and only if x_t is assigned to the same value as b_k. Analogously, one can find a tuple in $\mathbb{H}_i$ with $b_0, \ldots, b_i$ pairwise different corresponding to a tuple (p, r, s) of NAE. We produce an instance T_Γ of ABD(Γ) from an instance T_Δ of PQ-ABDUCTION(Δ) by replacing in the knowledge base every constraint NAE(p, r, s) with $\mathbb{H}_i(b_0, \ldots, b_i, x_p, x_r, x_s)$ and setting the manifestation to $(x_q \neq b_0)$, where q is the manifestation in T_Δ. Translating an explanation for T_Δ into an explanation for T_Γ, we ensure that all $b_0, \ldots, b_i$ are forced to be pairwise different. Converting the other way, it turns out that in general every explanation for T_Γ enforces $b_0, \ldots, b_i$ to take pairwise distinct values. We now give more details on that.

▶ Proposition 24. Let $i \in \mathbb{N} \setminus \{0,1\}$. Then the problem ABD($\mathbb{H}_i$) is Σ_2^P-hard.

Proof. Let $T_\Delta = (\phi_\Delta, V_\Delta, q)$ be an instance of PQ-ABDUCTION(Δ). We will now construct an instance $T_\Gamma = (\phi_\Gamma, V_\Gamma, L(x, y))$ of ABD(Γ) from it. First, for every propositional variable p occurring in ϕ_Δ, we introduce a variable x_p ranging over $\mathbb{N}$. Besides, we have $(i+1)$ extra variables $b_0, \ldots, b_i$ in Var(ϕ_Γ). The formula ϕ_Γ is a conjunction of constraints of the form $\mathbb{H}_i(b_0, \ldots, b_i, x_p, x_r, x_s)$ such that NAE(p, r, s) occurs in ϕ_Δ. The set V_Γ we define to be equal

to $\{x_p \mid p \in V_\Delta\} \cup \{b_0, \ldots, b_i\}$, and $L(x, y)$ equal to $(x_q \neq b_0)$. This construction may be certainly performed in polynomial time. We will now prove that $T_\Delta \in$ PQ-ABDUCTION(Δ) if and only if $T_\Gamma \in$ ABD(Γ). We start from the following facts.

▶ Observation 25.
- Let $a_\Delta : \mathrm{Var}(\phi_\Delta) \to \{0,1\}$, and let $F_{\Delta,\Gamma}(a_\Delta) : \mathrm{Var}(\phi_\Gamma) \to \mathbb{N}$ be such that $F_{\Delta,\Gamma}(a_\Delta)(b_k) = k$ for all $k \in \{0, \ldots, i\}$ and $F_{\Delta,\Gamma}(a_\Delta)(x_p) = k$ if and only if $a_\Delta(p) = k$ for all $p \in \mathrm{Var}(\phi_\Delta)$ and $k \in \{0,1\}$. Then, if a_Δ satisfies ϕ_Δ, then $F_{\Delta,\Gamma}(a_\Delta)$ satisfies ϕ_Γ.
- Let $a_\Gamma : \mathrm{Var}(\phi_\Gamma) \to \mathbb{N}$ be such that $a_\Gamma(b_k) = k$ for $k \in \{0, \ldots, i\}$ and $a_\Gamma(x_p) \in \{0,1\}$ for all $x_p \notin \{b_0, \ldots, b_k\}$. Let $F_{\Gamma,\Delta}(a_\Gamma) : \mathrm{Var}(\phi_\Delta) \to \{0,1\}$ such that $F_{\Gamma,\Delta}(a_\Gamma)(p) = k$ if and only if $a_\Gamma(x_p) = k$ for all $p \in \mathrm{Var}(\phi_\Delta)$ and $k \in \{0,1\}$. Then, if a_Γ satisfies ϕ_Γ, then $F_{\Gamma,\Delta}(a_\Gamma)$ satisfies ϕ_Δ. ◀

Suppose that T_Δ has an explanation $\mathrm{Lit}_\Delta \subseteq \mathrm{Lit}(V_\Delta)$ so that $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta)$ is satisfiable by some assignment $a_\Delta : \mathrm{Var}(\phi_\Delta) \to \{0,1\}$ and $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta \wedge \neg q)$ is not satisfiable. We set $\mathcal{L}_\Gamma \subseteq \mathcal{L}(V_\Gamma)$ to be the union of $\{(x_p = b_1) \mid p \in \mathrm{Lit}_\Delta\}$, and $\{(x_p = b_0) \mid (\neg p) \in \mathrm{Lit}_\Delta\}$, and $\bigcup_{k \neq l; k,l \in \{0,\ldots,i\}} \{(b_k \neq b_l)\}$. We will now prove that $\mathcal{L}_\Gamma$ is an explanation for T_Γ. By Observation 25, the assignment $F_{\Delta,\Gamma}(a_\Delta)$ satisfies $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma)$. Assume towards contradiction that $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma \wedge (x_q = b_0))$ is also satisfiable by some a_Γ. By the construction of $\mathcal{L}_\Gamma$, the image of a_Γ has at least $(i+1)$ elements. Indeed, every b_k for $k \in \{0, \ldots, i\}$ has to be set to a different element. We assume without loss of generality that $a_\Gamma(b_k) = k$ for all $k \in \{0, \ldots, i\}$. By the construction of ϕ_Γ, for every $p \in \mathrm{Var}(\phi_\Delta)$ we have that $a_\Gamma(x_p) \in \{0,1\}$. Hence, by Observation 25, the assignment $F_{\Gamma,\Delta}(a_\Gamma)$ satisfies $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta \wedge \neg q)$. It contradicts the assumption and thus we are done with the left-to-right implication.

Suppose now that there is some explanation $\mathcal{L}_\Gamma \subseteq \mathcal{L}(V_\Gamma)$ for T_Γ, that is, (i) the formula $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma)$ is satisfiable, and (ii) $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma \wedge (x_q = b_0))$ is not satisfiable. Observe first that every assignment a satisfying $\mathcal{L}_\Gamma$ has at least $(i+1)$ elements in the image. Indeed, suppose that there is some $a : \mathrm{Var}(\phi_\Gamma) \to \mathbb{N}$ with less than $(i+1)$ elements in the image. Then a can be extended to a' without increasing the size of the range of the assignment so that every variable not occurring in $\mathcal{L}_\Gamma$ is set to the same value as b_0. By the definition of the PQ-ABDUCTION problem and the construction of ϕ_Γ, the variable x_q is not in V_Γ, and hence a' satisfies $(x_q = b_0)$. By the definition of $\mathbb{H}_i$, the assignment a' also satisfies all constraints in ϕ_Γ. But this contradicts (ii). Now, by item (2) of the definition of $\mathbb{H}_i$, every assignment satisfying $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma)$ has exactly $(i+1)$ elements in the image. Indeed, for every $p \in \mathrm{Var}(\phi_\Delta)$, such an assignment assigns to x_p the same value as to b_0 or b_1. We can therefore assume that there is an assignment a_Γ satisfying $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma)$ such that $a_\Gamma(b_k) = k$ for all $k \in \{0, \ldots, i\}$. We now set the explanation Lit_Δ for T_Δ to be the union of $\{(p) \mid p \in V_\Delta \wedge a_\Gamma(x_p) = 1\}$ and $\{(\neg p) \mid p \in V_\Delta \wedge a_\Gamma(x_p) = 0\}$. To complete the proof we have to show that (a) $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta)$ is satisfiable, and (b) $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta \wedge \neg q)$ is not satisfiable. Point (a) follows by (i) and Observation 25. To prove that (b) holds, we suppose that $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta \wedge \neg q)$ is satisfiable by some a_Δ. From Observation 25, it follows that $F_{\Delta,\Gamma}(a_\Delta)$ satisfies $(\phi_\Gamma \wedge (x_q = b_0))$. It is also easy to see that $F_{\Delta,\Gamma}(a_\Delta)(x) = a_\Gamma(x)$ for every x occurring in $\mathcal{L}_\Gamma$, hence $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma \wedge (x_q = b_0))$ is satisfiable by $F_{\Delta,\Gamma}(a_\Delta)$. But it contradicts (ii) and thus completes the proof. ◀

This section will be concluded by proving Proposition 13.

Proof of Proposition 13. Let Γ be not Horn. By Theorem 20 and Proposition 21 we know that there is an $i \in \mathbb{N}$ such that $\mathrm{Pol}(\Gamma) \subseteq \mathcal{K}_i$. From Lemma 23 it follows that $\mathrm{Pol}(\Gamma) \subseteq \mathcal{K}_i \subseteq \mathrm{Pol}(\mathbb{N}; \mathbb{H}_i)$ for some $i \in \mathbb{N}$. By Proposition 22 and Proposition 24 we conclude that ABD(Γ) is Σ_2^P-hard. ◀

7 Equality Horn Languages that are NP-hard

In this section we prove Proposition 14. It turns out that already a very simple Horn relation

$$\mathbb{I} = \{(x, y, z) \in \mathbb{N}^3 \mid (x = y \to y = z)\}$$

gives rise to an abduction problem which is NP-hard. In fact we provide a hardness proof for a structure $\Gamma = (\mathbb{N}; \mathbb{I}_4)$, where $\mathbb{I}_4 = \{(a, b, c, d) \in \mathbb{N}^4 \mid ((a = b \wedge b = c) \to (a = d))\}$. Observe that $\exists z\ (\mathbb{I}(x, y, z) \wedge \mathbb{I}(v, z, w))$ pp-defines $\mathbb{I}_4(x, y, v, w)$.

We reduce from the propositional abduction problem PQ-ABDUCTION(Δ), where $\Delta = (\{0, 1\}; R_{\mathrm{A3}})$ and $R_{\mathrm{A3}} = \{(x, y, z) \mid \neg x \wedge \neg y \to \neg z\}$. By Theorem 3, this problem is NP-hard. Indeed, it is straightforward to verify that Δ is preserved by none of the following operations: majority, minority, min, opzero, opone.

The idea of the proof is similar to what we had in the preceding section. Observe that every tuple (b_0, x_p, x_r, x_s) of $\mathbb{I}_4$ may be translated into a tuple (p, r, s) of R_{A3} such that $t \in \{p, r, s\}$ is 0 if x_t and b_0 are assigned to the same value and t is 1 otherwise. In the analogical way, one can find a tuple (b_0, x_p, x_r, x_s) of $\mathbb{I}_4$ for every (p, r, s) in R_{A3} by setting b_0 to 0, and x_t to the same value which is assigned to $t \in \{p, r, s\}$. We construct an instance of T_Γ from T_Δ by replacing in the knowledge base every constraint of the form $R_{\mathrm{A3}}(p, r, s)$ with $\mathbb{I}_4(b_0, x_p, x_r, x_s)$, and setting the manifestation to $(x_q \neq b_0)$ where q is the manifestation in T_Δ. Now an explanation for T_Γ may be obtained from an explanation for T_Δ when t and $(\neg t)$ are replaced with $(x_t \neq b_0)$ and $(x_t = b_0)$, respectively. Converting the explanation back is analogous.

▶ Proposition 26. The problem ABD($\mathbb{N}; \mathbb{I}$) is NP-hard.

We will conclude this section by proving Proposition 14.

Proof of Proposition 14. Let Γ be not negative. It suffices to concentrate on the case where Γ is Horn (if Γ is not Horn, we conclude with Proposition 13). We obtain then by Theorem 20 and Proposition 21 that $\mathrm{Pol}(\Gamma) \subseteq \mathcal{S}^+$.

By Proposition 62 in [4], we have that if R has a pp-definition by $\mathrm{ODD}_3 = \{(a, b, c) \in \mathbb{N}^3 \mid a = b = c \vee |\{a, b, c\}| = 3\}$, then $\mathcal{S} \subseteq \mathrm{Pol}(\mathbb{N}; R)$. The relation $\mathbb{I}$ has a pp-definition by ODD_3, this follows by Lemma 8.6 in [3]. Further, since $\mathbb{I}$ is preserved by all constant operations, we have that $\mathrm{Pol}(\Gamma) \subseteq \mathcal{S}^+ \subseteq \mathrm{Pol}(\mathbb{N}; \mathbb{I})$. Hence, by Proposition 22, there is a polynomial-time reduction from ABD($\mathbb{N}; \mathbb{I}$) to ABD(Γ). Thus, by Proposition 26, the problem ABD(Γ) is NP-hard. ◀

8 Abduction for Negative Languages is in P

Recall negative equality languages provided in Definition 9. In this section we prove Proposition 15, that is, we show that if Γ is a negative equality language, then ABD(Γ) is in P. The algorithm is presented in Fig. 1. We will first discuss the first line of the procedure. There an instance $T = (\phi, V, L(x, y))$ of ABD(Γ) is transformed into an instance $T_A = (\phi_A, V_A, L_A)$ of the problem $\mathrm{ABD}_{\mathrm{no\_eq}}$ defined below. The instance T_A is equivalent to T w.r.t. existence of explanations but is such that ϕ_A contains no equalities.

▶ **Definition 27.** An instance of the computational problem $\mathrm{ABD}_{\mathrm{no\_eq}}$ consists of:

- a conjunction of disjunctions of disequalities ϕ,
- a subset V of $\mathrm{Var}(\phi)$, and
- an equality literal $L(x, y)$, with $\{x, y\} \subseteq \mathrm{Var}(\phi)$, of the form $(x = y)$, or $(x \neq y)$.

The question is whether there is an explanation $\mathcal{L} \subseteq \mathcal{L}(V)$ such that:

1. $(\phi \wedge \bigwedge \mathcal{L})$ is satisfiable, and
2. $(\phi \wedge \bigwedge \mathcal{L} \wedge \neg L(x,y))$ is not satisfiable.

Let $\sim$ be an equivalence relation on $\mathrm{Var}(\phi)$ such that for all $x_1, x_2 \in \mathrm{Var}(\phi)$ we have $x_1 \sim x_2$ if and only if ϕ entails $(x_1 = x_2)$. We construct $\phi_A, V_A, L_A(x_A, y_A)$ by first replacing in $\phi, V, L(x,y)$, respectively, every variable from $\mathrm{Var}(\phi)$ by its equivalence class in $\mathrm{Var}(\phi)/\sim$. Then, we remove all equalities and disequalities of the form $(v \neq v)$ in ϕ_A.

▶ **Lemma 28.** *Let Γ be a negative language and $T = (\phi, V, L(x,y))$ be an instance of ABD(Γ). Then there exists an instance T_A of ABD_{no_eq} such that $T \in ABD(\Gamma)$ if and only if $T_A \in ABD_{no\_eq}$. Moreover, T_A can be obtained from T in polynomial time.*

Algorithm for ABD(Γ), where Γ is a negative structure.

INPUT: An instance $T = (\phi, V, L(x,y))$ of ABD(Γ), where
- ϕ is a Γ-formula
- $V, \{x,y\} \subseteq \mathrm{Var}(\phi)$, and
- $L(x,y)$ is $(x = y)$, or $(x \neq y)$.

Let $T_A = (\phi_A, V_A, L_A(x_A, y_A))$ be an instance of $\mathrm{ABD}_{\mathrm{no\_eq}}$ from Lemma 28.
if ϕ_A is unsatisfiable **then return** FALSE
if $L_A(x_A, y_A)$ is $(x_A = y_A)$ **then**
  if x_A and y_A is the same variable **then return** TRUE
  else if $x_A, y_A \in V_A$ and $(\phi_A \wedge x_A = y_A)$ is satisfiable **then return** TRUE
  else return FALSE
end if
// from now on we can assume that $L_A(x_A, y_A)$ is $(x_A \neq y_A)$.
if x_A and y_A is the same variable **then return** FALSE
if $x_A, y_A \in V_A$ **then return** TRUE
if $z \in \{x_A, y_A\}$ is in V_A, and
$v \in \{x_A, y_A\} \backslash \{z\}$ is not in V_A, and
ϕ_A contains a clause equivalent to $(x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k \vee w \neq v)$ such that
- $\{x_1, y_1, \ldots, x_k, y_k, w\} \subseteq V_A$, and
- $(\phi \wedge \bigwedge_{i \in [k]} x_i = y_i \wedge z = w)$ is satisfiable

then return TRUE
if $x_A, y_A \notin V_A$ and
ϕ_A contains a clause equivalent to $(x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k \vee x_A \neq y_A)$ such that
- $\{x_1, y_1, \ldots, x_k, y_k\} \subseteq V_A$, and
- $(\phi \wedge \bigwedge_{i \in [k]} x_i = y_i)$ is satisfiable

then return TRUE
return FALSE

Figure 1 Algorithm for Abduction for Negative Languages.

The following proposition states that the algorithm presented in Fig. 1 is correct and complete.

▶ Proposition 29. Let Γ be a negative equality language, and $(\phi, V, L(x,y))$ be an instance of ABD(Γ). Then $(\phi, V, L(x,y)) \in$ ABD(Γ) if and only if the algorithm in Fig. 1 returns TRUE.

By Lemma 28, the first line of the algorithm in Fig. 1 can be performed in polynomial time. Apart from that, the procedure amounts to checking the satisfiability of a number of formulas which are obtained from negative formulas by adding conjuncts, which are equalities. Formulas of this form are always Horn formulas, and hence by the result in [5], each such check may be performed in polynomial time. Since the number of the satisfiability checks is readily polynomial with respect to the size of the input, we have the following.

▶ Proposition 30. Let Γ be a negative equality language. Then, for a given instance $(\phi, V, L(x, y))$ of ABD(Γ), the algorithm in Fig. 1 works in polynomial time in the size of $(\phi, V, L(x, y))$.

We are now ready to conclude this section.

Proof of Proposition 15. The statement follows by Propositions 29 and 30. ◀

9 Conclusion and Future Work

In this paper, we have initiated the study of the abduction problem parameterized by an ω-categorical relational structure Γ. We proved that as in the case of CSPs for these structures, the complexity of the abduction problem is fully captured by the set of operations preserving Γ. We have classified the complexity of the abduction problem parameterized by Γ with a first-order definition in $(\mathbb{N}; =)$ under the assumption that a manifestation is a literal of the form $(x = y)$ or $(x \neq y)$ and an explanation is a set of such literals over a given set of variables.

Our future work will concern similar classifications for first-order reducts of other ω-categorical structures. A natural choice for the next structure to study is $(\mathbb{Q}; <)$. Let Γ be a first-order reduct of $(\mathbb{Q}; <)$. In this case an instance of an abduction problem consists of a *temporal knowledge base* ϕ — a set of Γ-constraints — that describes point-based temporal dependencies between a finite number of events. A manifestation might be, for instance, of the form $(x < y)$, where x, y are events from ϕ. We can ask for an explanation that is a partial order on events in ϕ described by a conjunction of literals ψ of the form $(x \leq y)$ and $(x \neq y)$ such that ψ is consistent with ϕ ($(\phi \wedge \psi)$ is satisfiable) and ordering events from ϕ as described in ψ entails that x has to take place before y ($(\phi \wedge \psi)$ entails $(x < y)$).

Abduction problems for first-order reducts of $(\mathbb{Q}; <)$ do not only form a class of natural computational problems but also are plausible to be classified. The complexity of CSPs for these structures was classified in [6].

References

1 James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

2 Manuel Bodirsky. Complexity classification in infinite-domain constraint satisfaction. Memoire d'habilitation à diriger des recherches, Université Diderot – Paris 7. Available at arXiv:1201.0856, 2012.

3 Manuel Bodirsky and Hubie Chen. Quantified equality constraints. *SIAM Journal on Computing*, 39(8):3682–3699, 2010. A preliminary version of the paper appeared in the proceedings of LICS'07.

4 Manuel Bodirsky, Hubie Chen, and Michael Pinsker. The reducts of equality up to primitive positive interdefinability. *Journal of Symbolic Logic*, 75(4):1249–1292, 2010.

5 Manuel Bodirsky and Jan Kára. The complexity of equality constraint languages. *Theory of Computing Systems*, 3(2):136–158, 2008. A conference version appeared in the proceedings of Computer Science Russia (CSR'06).

6 Manuel Bodirsky and Jan Kára. The complexity of temporal constraint satisfaction problems. *Journal of the ACM*, 57(2):1–41, 2009. An extended abstract appeared in the Proceedings of the Symposium on Theory of Computing (STOC'08).

7 Manuel Bodirsky and Jaroslav Nešetřil. Constraint satisfaction with countable homogeneous templates. *Journal of Logic and Computation*, 16(3):359–373, 2006.

8 Vittorio Brusoni, Luca Console, Paolo Terenziani, and Daniele Theseider Dupré. A spectrum of definitions for temporal model-based diagnosis. *Artif. Intell.*, 102(1):39–79, 1998.

9 Vittorio Brusoni, Luca Console, Paolo Terenziani, and Barbara Pernici. Later: Managing temporal information efficiently. *IEEE Expert*, 12(4):56–64, 1997.

10 Tom Bylander, Dean Allemang, Michael C. Tanner, and John R. Josephson. The computational complexity of abduction. *Artif. Intell.*, 49(1-3):25–60, 1991.

11 Peter J. Cameron. *Oligomorphic Permutation Groups*. Cambridge University Press, Cambridge, 1990.

12 Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Comput. Surv.*, 42(1), 2009.

13 Luca Console, Paolo Terenziani, and Daniele Theseider Dupré. Local reasoning and knowledge compilation for efficient temporal abduction. *IEEE Trans. Knowl. Data Eng.*, 14(6):1230–1248, 2002.

14 Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors. *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, volume 5250 of *Lecture Notes in Computer Science*. Springer, 2008.

15 Nadia Creignou and Bruno Zanuttini. A complete classification of the complexity of propositional abduction. *SIAM J. Comput.*, 36(1):207–229, 2006.

16 Thomas Eiter and Georg Gottlob. The complexity of logic-based abduction. *J. ACM*, 42(1):3–42, 1995.

17 A. Herzig, J. Lang, and Pierre Marquis. Planning as abduction. In *IJCAI-01 Workshop on Planning under Uncertainty and Incomplete Information*, Seattle, Washington, USA, aug 2001.

18 Jerry R. Hobbs, Mark E. Stickel, Douglas E. Appelt, and Paul A. Martin. Interpretation as abduction. *Artif. Intell.*, 63(1-2):69–142, 1993.

19 Wilfrid Hodges. *Model theory*. Cambridge University Press, 1993.

20 Gustav Nordh and Bruno Zanuttini. What makes propositional abduction tractable. *Artif. Intell.*, 172(10):1245–1284, 2008.

21 Harry E. Pople. On the mechanization of abductive logic. In *IJCAI*, pages 147–152, 1973.

22 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.

23 Marc Vilain, Henry Kautz, and Peter van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. *Reading in Qualitative Reasoning about Physical Systems*, pages 373–381, 1989.

A Proof of Theorem 5

Proof of Theorem 5. By assumption, it follows that every n-ary relation R in the structure Γ_1 has a pp-definition $(\exists y_1 \cdots \exists y_m . \phi_R(y_1, \ldots, y_m, x_1, \ldots, x_n))$ in Γ_2.

Let $T_1 = (\phi_1, V_1, M_1)$ be an instance of $\mathrm{ABD}(\Gamma_1, \mathcal{HYP}, \mathcal{M})$. We will now construct an instance $T_2 = (\phi_2, V_2, M_2)$ of $\mathrm{ABD}(\Gamma_2, \mathcal{HYP}, \mathcal{M})$. To obtain ϕ_2 from ϕ_1, we first transform ϕ_1 to a $(\Gamma_2 \cup \{=\})$-formula $\phi_=$. To that end, we replace every Γ_1-constraint in ϕ_1 of the form $R(x_1, \ldots, x_n)$ with $\phi_R(x_1, \ldots, x_n, y_1, \ldots, y_m)$ so that every time $y_1, \ldots, y_m$ are fresh variables. Observe that by the construction of $\phi_=$, we have $\mathrm{Var}(\phi_1) \subseteq \mathrm{Var}(\phi_=)$.

Consider now a partition $\Xi = \{X_1, \ldots, X_n\}$ of $\mathrm{Var}(\phi_=)$ such that $x, y \in \mathrm{Var}(\phi_=)$ are in the same block X_i if and only if $(x = y)$ is entailed by the conjunction of equalities occurring in $\phi_=$. Let $\mathcal{V} = \{v_1, \ldots, v_n\}$ be fresh variables. If $x \in \mathrm{Var}(\phi_=)$ is in X_i, then we say that v_i is the representative of x. We obtain a Γ_2-formula ϕ_2 from $\phi_=$ by first removing all $\{=\}$-constraints and then replacing every occurrence of every variable by its representative in $\mathcal{V}$. Similarly, to obtain V_2 and M_2, we replace all variables in V_1 and M_1, respectively, with their representatives in $\mathcal{V}$. It is easy to see that this procedure can be performed in polynomial time. We will now prove that T_1 is a positive instance of $\mathrm{ABD}(\Gamma_1, \mathcal{HYP}, \mathcal{M})$ if and only if T_2 is a positive instance of $\mathrm{ABD}(\Gamma_2, \mathcal{HYP}, \mathcal{M})$.

As a first step towards this goal, we will show that $T_1 = (\phi_1, V_1, M_1)$ is a positive instance of $\mathrm{ABD}(\Gamma_1, \mathcal{HYP}, \mathcal{M})$ if and only if $T_= = (\phi_=, V_1, M_1)$ is a positive instance of $\mathrm{ABD}(\Gamma_2 \cup \{=\}, \mathcal{HYP}, \mathcal{M})$. Observe that the claim is a consequence of the following two facts. First, every assignment $a_1 : \mathrm{Var}(\phi_1) \to \mathbb{N}$ satisfying ϕ_1 may be extended to $a_= : \mathrm{Var}(\phi_=) \to \mathbb{N}$ satisfying $\phi_=$. Second, every assignment $a_= : \mathrm{Var}(\phi_=) \to \mathbb{N}$ satisfying $a_=$ restricted to variables in $\mathrm{Var}(\phi_1)$ satisfies ϕ_1.

Now, it remains to prove that $T_= = (\phi_=, V_1, M_1)$ is a positive instance of $\mathrm{ABD}(\Gamma_2 \cup \{=\}, \mathcal{HYP}, \mathcal{M})$, if and only if $T_2 = (\phi_2, V_2, M_2)$ is a positive instance of $\mathrm{ABD}(\Gamma_2, \mathcal{HYP}, \mathcal{M})$. We start from an easy observation.

▶ Observation 31. Let $a_= : \mathrm{Var}(\phi_=) \to \mathbb{N}$. If $a_=$ satisfies $\phi_=$, then for all $X_i \in \Xi$ and all $x, y \in X_i$ we have that $a_=(x) = a_=(y)$. ◀

Suppose first that $\psi_=$ is an explanation for $T_=$. Construct ψ_2 from $\psi_=$ by replacing every variable by its representative in $\mathcal{V}$. Since $(\phi_= \wedge \psi_=)$ is satisfiable, by Observation 31, it follows that $(\phi_2 \wedge \psi_2)$ is also satisfiable. Furthermore, if $(\phi_2 \wedge \psi_2 \wedge \neg M_2)$ was satisfiable, we would have that $(\phi_= \wedge \psi_= \wedge \neg M_1)$ is satisfiable. It contradicts the assumption and completes the proof of this implication.

Suppose now that ψ_2 is an explanation for T_2. Construct $\psi_=$ from ψ_2 by replacing each variable v_i by some, always the same, variable $x \in X_i \cap V_1$. Observe that by the construction of V_2, the set $X_i \cap V_1$ is not empty. Since $(\phi_2 \wedge \psi_2)$ is satisfiable, it easily follows that $(\phi_= \wedge \psi_=)$ is also satisfiable. To conclude the proof, observe that $(\phi_= \wedge \psi_= \wedge \neg M_1)$ is not satisfiable. Indeed, if it was satisfiable, then by Observation 31, we would have that $(\phi_2 \wedge \psi_2 \wedge \neg M_2)$ is satisfiable. It contradicts the assumption and completes the proof of the theorem. ◀

B Proof of Proposition 26

Proof of Proposition 26. Consider the relation $\mathbb{I}_4 = \{(a, b, c, d) \in \mathbb{N}^4 \mid ((a = b \wedge b = c) \to (a = d))\}$. Observe that $\exists z\, (\mathbb{I}(x, y, z) \wedge \mathbb{I}(v, z, w))$ pp-defines $\mathbb{I}_4(x, y, v, w)$. By Theorem 5, it is therefore enough to show that $\mathrm{ABD}(\Gamma)$, where $\Gamma = (\mathbb{N}; \mathbb{I}_4)$, is NP-hard.

We reduce from the propositional abduction problem PQ-ABDUCTION(Δ), where $\Delta = (\{0, 1\}; R_{\mathrm{A3}})$ and $R_{\mathrm{A3}} = \{(x, y, z) \mid \neg x \wedge \neg y \to \neg z\}$. By Theorem 3, this problem is NP-hard. Indeed, it is straightforward to verify that Δ is preserved by none of the following operations: majority, minority, min, opzero, opone.

Let $T_\Delta = (\phi_\Delta, V_\Delta, q)$ be an instance of PQ-ABDUCTION(Δ). We will now construct an instance $T_\Gamma = (\phi_\Gamma, V_\Gamma, L(x, y))$ of $\mathrm{ABD}(\Gamma)$. First, for every Boolean variable $p \in \mathrm{Var}(\phi_\Delta)$, we introduce a variable x_p ranging over $\mathbb{N}$. Besides, we have also one extra variable b_0 in $\mathrm{Var}(\phi_\Gamma)$. Then, we set ϕ_Γ to be a conjunction of atomic formulas of the form $\mathbb{I}_4(b_0, x_p, x_r, x_s)$ such that $R_{\mathrm{A3}}(p, r, s)$ occurs in ϕ_Δ; and V_Γ to $\{x_p \mid p \in V_\Delta\} \cup \{b_0\}$. To complete the reduction we set $L(x, y)$ to be equal to $(x_q \neq b_0)$.

The reduction may certainly be performed in polynomial time. To complete the proof, we will now show that $T_\Delta \in$ PQ-ABDUCTION(Δ) if and only if $T_\Gamma \in$ ABD(Γ).

We start from the following facts.

▶ Observation 32. Let $a_\Gamma : \mathrm{Var}(\phi_\Gamma) \to \mathbb{N}$ be any assignment satisfying ϕ_Γ. Then a'_Γ, obtained from a_Γ so that for every $x \in \mathrm{Var}(\phi_\Gamma)$ we have that $a'_\Gamma(x) = 0$ iff $a_\Gamma(x) = a_\Gamma(b_0)$ and $a'_\Gamma(x) = 1$ otherwise, also satisfies ϕ_Γ. ◀

▶ Observation 33.
- Let $a_\Delta : \mathrm{Var}(\phi_\Delta) \to \{0,1\}$, and let $F_{\Delta,\Gamma}(a_\Delta) : \mathrm{Var}(\phi_\Gamma) \to \mathbb{N}$ be such that $F_{\Delta,\Gamma}(a_\Delta)(b_0) = 0$ and $F_{\Delta,\Gamma}(a_\Delta)(x_p) = k$ if and only if $a_\Delta(p) = k$ for $k \in \{0,1\}$. Then, if a_Δ satisfies ϕ_Δ, then $F_{\Delta,\Gamma}(a_\Delta)$ satisfies ϕ_Γ.
- Let $a_\Gamma : \mathrm{Var}(\phi_\Gamma) \to \mathbb{N}$ such that $a_\Gamma(b_0) = 0$ and for every $p \in \mathrm{Var}(\phi_\Delta)$, we have $a_\Gamma(x_p) \in \{0,1\}$. Define $F_{\Gamma,\Delta}(a_\Gamma) : \mathrm{Var}(\phi_\Delta) \to \{0,1\}$ so that for every $p \in \mathrm{Var}(\phi_\Delta)$ and $k \in \{0,1\}$, we have that $F_{\Gamma,\Delta}(a_\Gamma)(x_p) = k$ iff $a_\Gamma(p) = k$. Then, if a_Γ satisfies ϕ_Γ, then $F_{\Gamma,\Delta}(a_\Gamma)$ satisfies ϕ_Δ. ◀

Suppose first that there exists a set of propositional literals $\mathrm{Lit}_\Delta \subseteq \mathrm{Lit}(V_\Delta)$ such that $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta)$ is satisfiable by some assignment $a_\Delta : \mathrm{Var}(\phi_\Delta) \to \{0,1\}$ and $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta \wedge \neg q)$ is not satisfiable. We set the explanation $\mathcal{L}_\Gamma \subseteq \mathcal{L}(V_\Gamma)$ for T_Γ to be the union of $\bigcup_{p \in \mathrm{Lit}_\Delta} \{x_p \neq b_0\}$ and $\bigcup_{(\neg p) \in \mathrm{Lit}_\Delta} \{x_p = b_0\}$. We will now show that $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma)$ is satisfiable and $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma \wedge (b_0 \neq x_q))$ is not satisfiable. The former follows from Observation 33. Indeed, the formula $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma)$ is satisfiable by $F_{\Delta,\Gamma}(a_\Delta)$. To prove the latter, assume on the contrary that $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma \wedge (x_q = b_0))$ is satisfied by some $a_\Gamma : \mathrm{Var}(\phi_\Gamma) \to \mathbb{N}$. By Observation 32, there exists $a'_\Gamma : \mathrm{Var}(\phi_\Gamma) \to \mathbb{N}$ satisfying ϕ_Γ that assigns 0 to b_0 as well as sends every variable to 0, or 1. Thus, by Observation 33, we have that $F_{\Gamma,\Delta}(a'_\Gamma)$ satisfies $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta \wedge \neg q)$. This contradicts the assumption and completes the proof of the left-to-right implication.

Suppose now that there is an explanation $\mathcal{L}_\Gamma \subseteq \mathcal{L}(V_\Gamma)$ of T_Γ, that is, the formula $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma)$ is satisfiable by some $a_\Gamma : \mathrm{Var}(\phi_\Gamma) \to \mathbb{N}$ and $(\phi_\Gamma \wedge \bigwedge \mathcal{L}_\Gamma \wedge (x_q = b_0))$ is not satisfiable. By Observation 32, we can assume that a_Γ sends every variable to 0 or 1 and b_0 to 0. We set the explanation $\mathrm{Lit}_\Delta \subseteq \mathrm{Lit}(V_\Delta)$ for T_Δ to be the union of $\{(p) \mid p \in \mathrm{Lit}(V_\Delta) \wedge a_\Gamma(x_p) = 1\}$ and $\{(\neg p) \mid p \in \mathrm{Lit}(V_\Delta) \wedge a_\Gamma(x_p) = 0\}$. We will now show that $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta)$ is satisfiable and $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta \wedge \neg q)$ is not satisfiable. The former holds by Observation 33. Indeed, we have that $F_{\Gamma,\Delta}(a_\Gamma)$ satisfies $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta)$. Finally, assume on the contrary that $(\phi_\Delta \wedge \bigwedge \mathrm{Lit}_\Delta \wedge \neg q)$ is satisfied by $a_\Delta : \mathrm{Var}(\phi_\Delta) \to \{0,1\}$. By Observation 33, the assignment $F_{\Delta,\Gamma}(a_\Delta)$ satisfies $(\phi_\Gamma \wedge (x_q = b_0))$. It is also easy to see that for every $p \in \mathrm{Var}(\phi_\Delta)$ it holds $F_{\Delta,\Gamma}(a_\Delta)(x_p) = a_\Gamma(x_p)$. Thus $F_{\Delta,\Gamma}(a_\Delta)$ satisfies $\bigwedge \mathcal{L}_\Gamma$ and in consequence, by Observation 33, $(\phi_\Gamma \wedge \mathcal{L}_\Gamma \wedge (x_q = b_0))$. But this contradicts the assumption and hence completes the proof. ◀

C Proof of Lemma 28

Proof of Lemma 28. Let $T_A = (\phi_A, V_A, L_A(x_A, y_A))$ be an instance of $\mathrm{ABD}_{\mathrm{no\_eq}}$ obtained from $T = (\phi, V, L(x,y))$ as it was described before the formulation of the lemma. It is easily observed that T_A can be obtained from T in polynomial time.

We will now show that $T \in \mathrm{ABD}(\Gamma)$ if and only if $T_A \in \mathrm{ABD}_{\mathrm{no\_eq}}$. As we will argue, it is basically a consequence of the following two facts. Let $X_1, \ldots, X_k$ be equivalence classes of $\mathrm{Var}(\phi)/\sim$.

▶ Observation 34. Let $a_\phi : \mathrm{Var}(\phi) \to \mathbb{N}$ be an assignment satisfying ϕ. Then for every $i \in [k]$ and all $v, z \in X_i$ we have that $a_\phi(v) = a_\phi(z)$. ◀

▶ Observation 35. ▬ Let $a_\phi : \mathrm{Var}(\phi) \to \mathbb{N}$ be such that for every $i \in [k]$ and all $v, z \in X_i$ we have that $a_\phi(v) = a_\phi(z)$, and let $F_{\phi,\phi_A}(a_\phi) : \mathrm{Var}(\phi)/\sim\to \mathbb{N}$ be such that for all $i \in [k]$ and all $z \in X_i$ we have that $F_{\phi,\phi_A}(a_\phi)(X_i) = a_\phi(z)$. Then if a_ϕ satisfies ϕ, then $F_{\phi,\phi_A}(a_\phi)$ satisfies ϕ_A.

▬ Let $a_{\phi_A} : \mathrm{Var}(\phi)/\sim\to \mathbb{N}$, and $F_{\phi_A,\phi}(a_{\phi_A}) : \mathrm{Var}(\phi) \to \mathbb{N}$ be such that for all $i \in [k]$ and $z \in X_i$ we have $F_{\phi_A,\phi}(a_{\phi_A})(z) = a_{\phi_A}(X_i)$. Then, if a_{ϕ_A} satisfies ϕ_A, then $F_{\phi_A,\phi}$ satisfies ϕ. ◀

Suppose first that $T \in \mathrm{ABD}(\Gamma)$. Then there exists $\mathcal{L} \subseteq \mathcal{L}(V)$ such that $(\phi \wedge \bigwedge \mathcal{L})$ is satisfiable by some $a_\phi : \mathrm{Var}(\phi) \to \mathbb{N}$ and $(\phi \wedge \bigwedge \mathcal{L} \wedge \neg L(x, y))$ is not satisfiable. We define $\mathcal{L}_A$ to be the set that contains all literals of the form $(X_i \circ X_j)$, where $i, j \in [k]$ and $\circ \in \{=, \neq\}$, such that $\mathcal{L}$ contains $(x' \circ y')$ for some $x' \in X_i, y' \in X_j$. From Observations 34 and 35, we easily obtain that $(\phi_A \wedge \bigwedge \mathcal{L}_A)$ is satisfiable by $F_{\phi,\phi_A}(a_\phi)$. Also, if $(\phi_A \wedge \bigwedge \mathcal{L}_A \wedge \neg L_A(x_A, y_A))$ was satisfiable, then by Observation 35 we would have that $(\phi \wedge \bigwedge \mathcal{L} \wedge \neg L(x, y))$ is satisfiable. It contradicts the assumption and completes the proof of the left-to-right implication.

Suppose now that there is $\mathcal{L}_A$ such that both $(\phi_A \wedge \bigwedge \mathcal{L}_A)$ is satisfiable by some $a_{\phi_A} : \mathrm{Var}(\phi)/\sim\to \mathbb{N}$ and $(\phi_A \wedge \bigwedge \mathcal{L}_A \wedge \neg L(x_A, y_A))$ is not satisfiable. We set $\mathcal{L}$ to contain all literals of the form $(v \circ z)$, where $\circ \in \{=, \neq\}$, such that $v \in X_i \cap V$ and $z \in X_j \cap V$ and $X_i \circ X_j$ is in $\mathcal{L}_A$. By Observation 35, we have that $(\phi \wedge \mathcal{L})$ is satisfiable by $F_{\phi_A,\phi}(a_{\phi_A})$. On the other hand, if $(\phi \wedge \bigwedge \mathcal{L} \wedge \neg L(x, y))$ was satisfiable by some a_ϕ, then by Observations 34 and 35, we would have that $(\phi_A \wedge \bigwedge \mathcal{L}_A \wedge \neg L_A(x_A, y_A))$ is satisfiable, which contradicts the assumption and completes the proof of the lemma. ◀

D Proof of Proposition 29

Proof of Proposition 29. By Lemma 28, it is enough to show that the algorithm returns TRUE if and only if $(\phi_A, V_A, L_A(x_A, y_A)) \in \mathrm{ABD}_{\mathrm{no\_eq}}$.

We will first show the proof of the right-to-left implication. If $L_A(x_A, y_A)$ is $(x_A = y_A)$, then the algorithm returns TRUE if ϕ_A is satisfiable and either x_A and y_A are the same variable, or $\{x_A, y_A\} \subseteq V_A$. In the first case an empty set of literals works as an explanation, while in the other, we can take $\mathcal{L}$ equal to $(x_A = y_A)$. If $L_A(x_A, y_A)$ is $(x_A \neq y_A)$, then the algorithm may return TRUE in lines 10, 11 and 12. In line 10, we set $\mathcal{L}$ to $\{x_A \neq y_A\}$. In line 11 to $\bigcup_{i\in[k]}\{x_i = y_i\} \cup \{z = w\}$, while in line 12 to $\bigcup_{i\in[k]}\{x_i = y_i\}$.

We now turn to the left-to-right implication. Suppose that there is a set of literals $\mathcal{L}$ such that both Points 1 and 2 in Definition 27 hold. Consider a formula ψ equal to $(\phi_A \wedge \bigwedge \mathcal{L})$. Let $X = \{X_1, \ldots, X_k\}$ be a partition of $\mathrm{Var}(\psi)$ such that ψ entails $(s = t)$ if and only if there exists $i \in [k]$ such that both s and t are in X_i. Then, for $i \in [k]$, we choose one element s_i from every X_i to be a representative of all elements in X_i. Then, in all disjunctions of disequalities, we first replace all variables with their representatives, and then remove all disequalities of the form $(s_i \neq s_i)$. Since all these transformations preserve the satisfiability of the formula, in the end we get no empty clauses. Denote the formula obtained in this way by ψ'.

Consider first the case where $L_A(x_A, y_A)$ is $(x_A = y_A)$. The case where x_A and y_A are the same variable is handled by the procedure in line 4. Thus we can assume that they are different. Since $(\phi_A \wedge \bigwedge \mathcal{L})$ entails $(x_A = y_A)$, and ϕ_A does not contain equalities, the formula $(x_A = y_A)$ must be entailed by $\bigwedge \mathcal{L}$. Indeed, suppose this is not the case, then there is an assignment $a : Var(\phi_A) \to \mathbb{N}$ satisfying $(\bigwedge \mathcal{L} \wedge x_A \neq y_A)$. Let $a' : Var(\phi_A) \to \mathbb{N}$ be a satisfying assignment to $(\phi_A \wedge \bigwedge \mathcal{L})$. We claim that $b(a, a')$ where $b : \mathbb{N}^2 \to \mathbb{N}$ is a binary injective operation satisfies $(\phi_A \wedge \bigwedge \mathcal{L} \wedge x_A \neq y_A)$. It clearly satisfies $(x_A \neq y_A)$, it satisfies

$\bigwedge \mathcal{L}$ since it is Horn. To see that $b(a, a')$ satisfies ϕ_A we use the following property of a negative equality formula ϕ: let $a, a' : \mathrm{Var}(\phi) \to \mathbb{N}$ such that a' satisfies ϕ and b be a binary injection, then $b(a, a')$ satisfies ϕ. Thus, we proved that $(x_A = y_A)$ must be entailed by $\bigwedge \mathcal{L}$. Hence x_A and y_A are in V_A. Since also $\phi_A \wedge (x_A = y_A)$ is satisfiable, the procedure returns TRUE in line 5.

Assume now that $L_A(x_A, y_A)$ is $(x_A \neq y_A)$. Since $(\phi_A \wedge L_A(x_A, y_A))$ is satisfiable, we have that x_A and y_A are in different blocks X_a, and X_b, respectively, of X. Assume without loss of generality that they are the representatives of their own blocks. Observe that ψ' contains a clause of the form $(x_A \neq y_A)$: otherwise $(\psi \wedge x_A = y_A)$ would be satisfiable by the assignment $a_X : \mathrm{Var}(\psi) \to \mathbb{N}$ sending all variables from X_i where $i \neq b$ to i, and all variables from X_b to a, i.e., no explanation would exist. So, we can assume that ψ' contains $(x_A \neq y_A)$. In this case either (i) x_A and y_A are both in V, or (ii) there is $z \in \{x_A, y_A\}$ which is in V, and $v \in \{x_A, y_A\} \setminus \{z\}$ which is not, and ϕ contains a clause equivalent to $(x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k \vee w \neq v)$ such that for every i both x_i and y_i as well as z and w are in the same block of X, or (iii) both x_A, y_A are not in V and ϕ contains a clause equivalent to $(x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k \vee x_A \neq y_A)$ such that for every i both x_i and y_i are in the same block of X. Observe that cases (i), (ii), and (iii) are handled by the procedure in lines 10, 11, and 12, respectively. ◀

A New Type Assignment for Strongly Normalizable Terms

Rick Statman

Carnegie Mellon University
Department of Mathematical Sciences
Pittsburgh, PA 15213
(statman@cs.cmu.edu)

Abstract

We consider an operator definable in the intuitionistic theory of monadic predicates and we axiomatize some of its properties in a definitional extension of that monadic logic. The axiomatization lends itself to a natural deduction formulation to which the Curry-Howard isomorphism can be applied. The resulting Church style type system has the property that an untyped term is typable if and only if it is strongly normalizable.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases lambda calculus

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.634

1 Introduction

Intersection types [4] are very interesting especially in their use for proving untyped terms to be strongly normalizable [6]. However, we view them only as types, and the Curry-Howard isomorphism does not seem to apply. Here we would like to extend the *formulae as types* direction of Curry-Howard to include all strongly normalizable terms. We shall do this by considering a definitional extension of a very weak version of intuitionistic monadic logic. Our notion of typing appears quite different from the clever application of Curry-Howard to the derivations of intersection types for untyped terms in [2]; we do no linearization of untyped terms.

2 Intuitionistic Monadic Logic

We consider the first-order language of intuitionistic monadic predicate logic in the negative fragment. The language consists of two individual constants
$0, 1$
and an arbitrary selection of monadic predicates R. In addition, we shall have two other distinguished monadic predicates
P, Q
that play a special role and remain mostly hidden. We have the connective, $\rightarrow$, the universal quantifier, $\wedge$, and a symbol for falsehood, @. We shall assume that $P0, Q1$ and P and Q are disjoint; that is, $\wedge x(Px \rightarrow (Qx \rightarrow @))$. As usual, we set $\sim F := F \rightarrow @$.

We define a certain definitional extension of our language as follows. Introduce a new connective/relation symbol D which takes a single individual and two formula arguments, and which is defined by $DxFG := (Px \rightarrow F)\ \&\ (Qx \rightarrow G)$.
Indeed, this is the only way that P and Q enter into our discussion.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 634–652

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

D satisfies many interesting properties. Of these, the equivalences

(a)	$D0FG$	$\leftrightarrow$	F	
(b)	$D1FG$	$\leftrightarrow$	G	
(c)	$Dt(F \to G)(H \to K)$	$\leftrightarrow$	$DtFH \to DtGK$	
(d)	$Dt(\wedge yF)G$	$\leftrightarrow$	$\wedge y\ DtFG$	y not free in G, t
(e)	$DtF(\wedge yG)$	$\leftrightarrow$	$\wedge yDtFG$	y not free in F, t

are the most important. They can be verified as follows

(a) Assume F, then $P0 \to F$ and $Q0 \to G$ since $\sim Q0$.
Conversely, assume $D0FG$. Then $P0 \to F$ so F since $P0$.
(b) Similar to (a)
(c) Assume $Dt(F \to G)(H \to K)$. Now assume $DtFH$. To show $DtGK$ assume Pt. Then, since $DtFH$ we get F and since $Dt(F \to G)(H \to K)$ we get $F \to G$. Thus G. So $Pt \to G$. Now assume Qt. Similarly, we get K. Thus, $Qt \to K$. Conversely, assume $DtFH \to DtGK$. To show $Dt(F \to G)(H \to K)$ assume Pt and F. Then $\sim Qt$ so $DtFH$;. Thus, $DtGK$ and hence since Pt, G. Similarly for $H \to K$.
(d) Assume $Dt(\wedge xF)G$. Let x be given. To show $DtFG$ assume Pt. Then $\wedge xF$, in particular, F. Thus, $Pt \to F$. Now assume Qt. By hypothesis G. Thus, $Qt \to G$. But x was arbitrary; thus, $\wedge x(Pt \to F$ and $Qt \to G)$. Conversely, suppose $\wedge x\ DtFG$. To show $Dt(\wedge xF)G$ assume Pt. Let x be given. We have, by assumption, $Pt \to F$ so F. Thus $\wedge xF$. We already have $Qt \to G$. Similarly for the other half.

The above equivalences would be shared by D with any *discriminator*. Discriminators have been extensively studied, and we refer the reader to Bloom & Tindell [3]. Fortunately, the properties above do not depend on the decidability of P and Q, their coverage, nor on their complementarity. For example, in any (Kripke) model satisfying $\wedge x\ (DxFF \leftrightarrow F)$ if the model fails to satisfy $\wedge x\ (Px \vee Qx)$ then the model satisfies $\sim\sim F$. Unfortunately, the five above are not complete in our context. For example,

$$Dt(DrFG)(DrHK) \leftrightarrow Dr(DtFH)(DtGK)$$

is valid but not derivable from the five. This can be seen as an exercise after the next section. There are more; indeed, the set of all valid equivalences is undecidable. If $A[S]$ is any monadic formula on the monadic predicate S then

$$A[S] \text{ is intuitionistically valid } \Leftrightarrow$$

$A[\lambda xDx(Rx)(Rx)] \leftrightarrow (R0 \to R0)$ is a valid equivalence. In particular, Kripke model M for S can be extended by setting $R = S$, $P = M - \{1\}$ and $Q = \{1\}$. Thus, we can apply the theorem of Maslow, Mints, and Orevkov [5].

3 Natural Deduction and Rewrite Rules

From now on, we consider only the restricted language without $@, P$, and Q, and with D as a primitive symbol. The above equivalences can be formulated as reduction rules;

$(0)D0FG$	$\rightarrowtail$	F	
$(1)D1FG$	$\rightarrowtail$	G	
$(\rightarrow)Dt(F \rightarrow G)(H \rightarrow K)$	$\rightarrowtail$	$(DtFH) \rightarrow (DtGK)$	
$(\wedge)Dt(\wedge uF)G$	$\rightarrowtail$	$\wedge uDtFG$	u not free in G or t
$(\wedge)DtF(\wedge vG)$	$\rightarrowtail$	$\wedge v\ DtFG$	v not free in F or t
$(\$) \wedge uF$	$\rightarrowtail$	F	u not free in F
$(\$\$) \wedge u \wedge vF$	$\rightarrowtail$	$\wedge v \wedge uF$	

Here ($) and ($$) make for a smoother theory. The congruence generated by $\rightarrowtail$ is called formula conversion (conv.).

$D - \{(0), (1)\}$ is denoted $D-$. The following are easily verified.

Facts:

(i) $D-$ reductions satisfy the weak-diamond property.
(ii) A given formula $D-$ reduces to only finitely many others.
(iii) $D-$ reduction has the strong diamond property, modulo ($$).
There is an obvious notion of residual for reductions. Residuals, if they exist, are unique and the corresponding reductions commute modulo the order of $\wedge$'s.
(iv) $D-$ is Church-Rosser.
(v) $(0) + (1)$ has unique normal forms.
(vi) There is a *strip lemma* for $(0)+(1)$ -reduction over D; If $F \twoheadleftarrow G$ in general, and $G \rightarrowtail\!\!\!\!\twoheadrightarrow H$ by $(0) + (1)$ then $H \rightarrowtail\!\!\!\!\twoheadrightarrow K$ in general and $F \rightarrowtail\!\!\!\!\twoheadrightarrow K$ by $(0) + (1)$.
(vii) $\rightarrowtail$ is Church-Rosser.
(viii) There is a *standardization theorem* for $(\wedge)$;
If $F \rightarrowtail\!\!\!\!\twoheadrightarrow \wedge yG$ then there exists H such that $F \rightarrowtail\!\!\!\!\twoheadrightarrow \wedge yH$ by $(\wedge)$ and ($$) alone and $H \rightarrowtail\!\!\!\!\twoheadrightarrow G$.
(ix) A formula F in $\rightarrowtail$ normal form has the properties

(a) F does not contain $D0HK$ or $D1HK$

(b) If F contains DtF_0F_1 then if F_i is not atomic then F_i has the form $H \rightarrow K$ and F_{1-i} is atomic.

(i) $\rightarrowtail$ normal forms are unique up to ($$).

If we have a formula F with no variable both free and bound and no variable bound twice (alpha normal form) and we require that in ($) expansions u is new, then in any conversion F conv. G each quantifier $\wedge v$ has at most one descendant in G. The *replacement* of $\wedge v$ in F by t in this conversion is the result of substituting t for every occurrence of v and omitting $\wedge v$. The result is a valid conversion when redundant steps are omitted.

Now we have the natural deduction rules for intuitionistic monadic logic with D

$$
\begin{array}{ll}
 & \begin{array}{c} / \\ F \\ / \\ . \\ . \\ . \\ G \end{array} \\
(\to I) & \hline \\
 & F \to G
\end{array}
$$

$$
\begin{array}{lll}
 & \begin{array}{c} . \\ . \\ . \\ F \to G \end{array} & \begin{array}{c} . \\ . \\ . \\ F \end{array} \\
(\to E) & \hline \\
 & G
\end{array}
$$

$$
\begin{array}{ll}
 & \begin{array}{c} . \\ . \\ . \\ F \end{array} \\
(\wedge I) & \hline \\
 & \wedge v F
\end{array}
\qquad v \text{ (eigenvariable) not free in any assumption}
$$

$$
\begin{array}{ll}
 & \begin{array}{c} . \\ . \\ . \\ \wedge v F \end{array} \\
(\wedge E) & \hline \\
 & [t/v]F
\end{array}
\qquad t \text{ free for } v \text{ in } F \text{ (i.e. if } t \text{ is a variable it should not become bound by substitution)}
$$

$$
\begin{array}{ll}
 & \begin{array}{c} . \\ . \\ . \\ F \end{array} \\
(\text{conv.}) & \hline \\
 & G
\end{array}
\qquad F \text{ conv. } G
$$

These rules will be transformed into typing rules for untyped lambda terms.

4 Typing Rules

Although we are essentially using Church typing, two deviations from normal conventions will be employed. First, as already mentioned, we adopt Curry style derivations. This is merely a notational convenience. In a second notational convenience, we dispense with

lambda abstraction over object variables appearing in types normally used as a coercion into universally quantified types. Instead, we could distinguish the notion of a *typed sub-term* from an untyped one, so an untyped sub-term begins a nested set of typed ones; but, in practice, we shall just refer to a particular point in the typing derivation tree. Nevertheless, each typed sub-term has a unique type. The typing rules are the following.

$$
\begin{array}{lll}
 & \begin{array}{c} / \\ x : F \\ / \\ . \\ . \\ . \\ X : G \end{array} & \\
(\to I) & \hline & \\
 & \lambda x X \quad : \quad F \to G & \\
 & \begin{array}{cc} . & . \\ . & . \\ . & . \end{array} & \\
(\to E) & \hline X : F \to G \qquad Y : F & \\
 & (XY) : G & \\
 & \begin{array}{c} . \\ . \\ . \end{array} & \\
(\wedge I) & \hline X : F & \\
 & X : \wedge v F & v \text{ (eigenvariable) not free in the type of any free variable of } X \\
 & \begin{array}{c} . \\ . \\ . \end{array} & \\
(\wedge E) & \hline X : \wedge v F & t \text{ free for } v \text{ in } F \\
 & X : [t/v]F & \\
 & \begin{array}{c} . \\ . \\ . \end{array} & \\
(\text{conv.}) & \hline X : F & F \text{ conv. } G \\
 & X : G &
\end{array}
$$

where x can occur in an assumption $x : F$ with at most one F. These rules are to be understood in the obvious way. Each derivation of $X : G$, for untyped X, corresponds to a Church typing of X with G, and the free variables of X with the types assigned to them in the assumptions.

5 Reductions of Derivations

With each conversion, F (conv.) G, a pair of reductions

$$F \rightarrowtail H \leftarrowtail G$$

can be associated. We can always assume that H has only 0 and 1 in atomic sub-formulae $R0$ or $R1$, and no $D0KK'$ or $D1KK'$, and by the strip lemma, each reduction begins with $(0), (1)$ reductions and proceeds afterwards with none. Two conversions

$$F \text{ conv. } G \text{ conv. } H$$

in a row can be transformed into a single one

$$F \text{ conv. } H$$

We now define the notion of a "derivation reduction" in 4 parts.

(1) In three successive inferences $(\wedge I)$, (conv.), $(\wedge E)$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\cdot\\\cdot\\\cdot\\F\end{array}}{\wedge v \quad F}}{\wedge u \quad G}}{[t/u] \quad G}$$

either $\wedge u$ is a descendant of $\wedge v$ or it is not. In the first case, omitting trivial pairs of ($$)'s we have

$$\dfrac{[t/v] \quad \dfrac{\begin{array}{c}\cdot\\\cdot\\\cdot\\[t/v] \quad F\end{array}}{[t/u] \quad G}}{}$$

In the latter case, there is the trivial case that $\wedge v$ is omitted by ($), and we have

$$\dfrac{\dfrac{\begin{array}{c}\cdot\\\cdot\\\cdot\\F\end{array}}{\wedge u\ G}}{[t/u]\ G} \qquad \text{(conv.)}$$

Otherwise, in case $\wedge u$ is omitted by (\$), we have

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ F\end{array}}{\wedge v\ F}}{[t/u]\quad G}\ (\text{conv.})$$

Finally, we have

$\wedge v\ F$ (conv.) $\wedge v\ \wedge u\ H(v,u)$ by $(\wedge)$ and (\$\$)
$\wedge v\ \wedge u\ H(v,u) \rightarrowtail\!\!\!\!\twoheadrightarrow \wedge v\ \wedge u\ K(v,u)$
$\wedge u\ \wedge v\ K(v,u) \twoheadleftarrow\!\!\!\!\leftarrowtail \wedge u\ \wedge v\ L(u,v)$
$\wedge u\ \wedge v\ L(u,v) \twoheadleftarrow\!\!\!\!\leftarrowtail \wedge u\ G$ by $(\wedge)$ and (\$)

and so

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ F\end{array}}{\wedge u\ H(v,u)}}{H(v,t)}}{L(t,v)}}{\wedge v\ L(t,v)}}{[t/u]G}$$

In this manner the three successive inferences are reduced to either

(conv.)
(conv.), $(\wedge E)$
$(\wedge I)$, (conv.) or
$(\wedge E)$, (conv.), $(\wedge I)$

(2) In three successive inferences $(\rightarrow I)$, (conv.), $(\wedge E)$;

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}/\\F\\/\\\cdot\\\cdot\\\cdot\\G\end{array}}{F \to G}}{\wedge u\ H}}{[t/u]H}$$

omitting trivial pairs of ($$)'s, there exist H', F', G' such that

$$\begin{array}{lllll} \wedge uH \rightarrowtail\!\!\!\!\twoheadrightarrow \wedge uH' \rightarrowtail H' \text{ by } (\$) & & & & \\ H' & \rightarrowtail\!\!\!\!\twoheadrightarrow & (F' \to G') & \twoheadleftarrow\!\!\!\!\prec & (F \to G) \\ H' & \twoheadleftarrow\!\!\!\!\prec & [t/u]H & & \end{array}$$

and so the three successive inferences can be reduced to an $(\to I)$, (conv.).

(3) In three successive inferences $(\wedge I)$, (conv.), $(\to E)$;

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}/\\F\\/\\\cdot\\\cdot\\\cdot\\H\end{array}}{\wedge u\ H}}{F \to G} \qquad \begin{array}{c}\cdot\\\cdot\\\cdot\\F\end{array}}{G}$$

omitting trivial pairs of ($$)'s, there exist H', F', G' such that

$$\begin{array}{lllllll} \wedge uH & \rightarrowtail\!\!\!\!\twoheadrightarrow & \wedge uH' & \rightarrowtail & H' & \text{by } (\$) \\ H' & \rightarrowtail\!\!\!\!\twoheadrightarrow & (F' \to G') & \twoheadleftarrow\!\!\!\!\prec & (F \to G) & \\ H' & \twoheadleftarrow\!\!\!\!\prec & H & & & \end{array}$$

and so the three successive inferences can be reduced to an (conv.) $(\to E)$.

(4) In three successive inferences $(\to I)$, (conv.), $(\to E)$;

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}/\\F\\/\\\cdot\\\cdot\\\cdot\\G\end{array}}{F \to G} \qquad \begin{array}{c}\cdot\\\cdot\end{array}}{H \to K} \qquad \begin{array}{c}\cdot\\H\end{array}}{K}$$

we have F conv. H and G conv. K. These three successive inferences can be reduced to

$$\dfrac{\dfrac{\begin{array}{c}\cdot\\\cdot\\\cdot\\H\end{array}}{F} \qquad \begin{array}{c}\cdot\\\cdot\\\cdot\\G\end{array}}{K}$$

which uses only (conv.).

A *segment* in a derivation is an alternating sequence of $(\wedge I)$, or $(\wedge E)$ inferences and conversions. Thus, in a segment, we can assume that no $(\wedge I)$ precedes an $(\wedge E)$ by applying suitable reductions (1)-(3). Thus, if a segment begins and ends in a formula beginning with $\to$, it is simply a conversion. When employing the rules as typing rules, applying reductions (1)-(3) to segments does not alter the untyped term being typed.

6 The Main Result

We shall now prove that an untyped term is strongly normalizable if, and only if, it has a type in our system.

▶ **Lemma 1.** *Suppose that we have typings $X : F$ and $X : G$ of the untyped X. Then there exists a typing $X : DvFG$ for v new. Moreover, if, for the free variable x, we have $x : H$ in $X : F$ and $x : K$ in $X : G$ then $x : DvHK$ in $X : DvFG$.*

▶ **Lemma 2.** *A normal untyped term X has a typing $X : F$.*

▶ **Lemma 3.** *Suppose that x occurs in X and $[Y/x]X$ has a typing*

$$[Y/x]X : F.$$

Then there is a typing $(\lambda xX)Y : F$, where the free variables of X may have new types.

▶ Proposition 1. If X is strongly normalizable then for some F we have a typing $X : F$.

Proof. Now suppose that X is strongly normalizable. We show that X has a typing by induction on the reduction tree of X with a subsidiary induction on the length of X. This is really induction on Barendregts's perpetual reduction strategy beginning with X ([1] pg. 334). We can write $X :=$

$$\lambda x_1 \dots x_r \begin{cases} x_i & \\ & X_1 \dots X_s \\ (\lambda x.X_0) & \end{cases}$$

Case 1: $r > 0$. Then the induction hypothesis on length can be applied directly.

Case 2: $r = 0$ and there is no head redex. This is just like the case of normal terms.

Case 3: $r = 0$ and X has a head redex. We distinguish two subcases.

Subcase 1: x is not free in X_0. Now both X_1 and $X_0X_2 \dots X_s$ have shorter reduction trees than X. Thus, by induction hypothesis, both have typings with $X_1 : G$. We may adjust the typings of the free variables in X_1 and $X_0X_2 \dots X_s$ so that they match as in the case of normal terms. Thus, we have a typing of X with $x : G$.

Subcase 2: x appears free in X_0. Now the reduction tree of

$$([X_1/x]X_0)X_2 \dots X_s$$

is smaller than that of X so the induction hypothesis applies and this term has a typing. Now we can apply Lemma 3 and adjust the types of the free variables in $([X_1/x]X_0)X_2 \dots X_2$. ◀

▶ **Lemma 4.** *If $X : G$ is strongly normalizable with $y : F$ and $Y : F$ is strongly normalizable then $[Y/y]X : G$ is strongly normalizable.*

▶ Proposition 2. If the untyped term X has a typing $X : F$ then X is strongly normalizable.

Proof. By induction on X where we again write $X :=$

$$\lambda x_1 \dots x_r \begin{cases} x_i & \\ & X_1 \dots X_s \\ (\lambda x X_0). & \end{cases}$$

Lemma 4 prevails. ◀

Acknowledgement. The author would like to thank refree 3 (the expert) for his many thoughtful and usefull suggestions. There was neither time nor space to include them all in the version of this paper.

References

1 H. P. Barendegt, "The Lambda Calculus", North Holland, 1984.
2 A. Bucciarelli, Piperno, A., and Salvo, I., Intersection types and lambda definability, MSCS03 **13** (1), 2003, pp. 15-53.
3 S. Bloom and R. Tindell, Varieties of "if-then-else", *SICOMP* **12** (4), 1983.

4 M. Coppo and M. Dezani, A new type assignment for lambda terms, *Archiv fur Math. Logik*, **19**, 1978.

5 S. Ghilezan, Strong normalization and typability with intersection types, *Notre Dame Journal of Formal Logic*, **37** (1), 1996.

6 S. A. Maslow, G. E. Mints, and V. P. Orevkov, Unsolvability in the constructive predicate calculus, *Soviet Math. Doklady* **4** 1963, pp. 1365-1367.

7 G. Pottinger, "A type assignment for strongly normalizable lambda terms", Curry Festschrift, 1980.

8 M. Sorenson and P. Urzyczyn, Lectures on the Curry-Howard Isomorphism Manuscript, 1998.

A Proof of Lemma 1

Proof. Lemma 1. By induction on X.

Basis: $X = x$. The typings $X : F$ and $X : G$ are both segments, which we can assume are the same length by adding trivial conversions. In addition, we can assume that all the variables which occur bound in one typing are distinct from the variables which at some point occur free in the other. Then we can simulate both typings in a typing by $DvFG$ as follows.

From $X : F$ to $X : DvFG$

$$\begin{array}{ccc} \dfrac{x \;:\; H}{x \;:\; \wedge uH} & \mapsto & \dfrac{x : DvHK}{\dfrac{x : \wedge uDvHK}{x : Dv(\wedge uH)K}} \\[2ex] \dfrac{x : \wedge uH}{x : [t/u]H} & \mapsto & \dfrac{x : Dv(\wedge uH)K}{\dfrac{x : \wedge u\ DvHK}{x : [t/u]DvHK}} \end{array}$$

and similarly for from $X : G$ to $X : DvFG$.

Induction step:

Case 1: $X = (YZ)$

In $X : F$ we have

$$\dfrac{\begin{array}{ll} Y & : H \\ & \cdot \\ & \cdot \text{ (segment)} \\ & \cdot \\ Y & : K \to L \end{array} \qquad \begin{array}{ll} Z & : M \\ & \cdot \\ & \cdot \text{ (segment)} \\ & \cdot \\ Z & : K \end{array}}{\begin{array}{ll} (YZ) & : L \\ & \cdot \\ & \cdot \text{ (segment)} \\ & \cdot \\ (YZ) & : F \end{array}}$$

and in $X : G$ we have

$$\begin{array}{llll}
 & \cdot & & \cdot \\
 & \cdot & & \cdot \\
 & \cdot & & \cdot \\
Y & : H' & Z & : M' \\
 & \cdot & & \cdot \\
 & \cdot \text{ (segment)} & & \cdot \text{ (segment)} \\
 & \cdot & & \cdot \\
Y & : K' \to L' & Z & : K' \\
\hline
 & & (YZ) & : L' \\
 & & & \cdot \\
 & & & \cdot \text{ (segment)} \\
 & & & \cdot \\
 & & (YZ) & : G
\end{array}$$

The segments can be simulated as in the basis case and this arrives at

$$\dfrac{\begin{array}{ll}
\cdot & \cdot \\
\cdot & \cdot \\
\cdot & \cdot \\
Y : DvHH' & z : DvMM' \\
\cdot & \cdot \\
\cdot \text{ (segment)} & \cdot \text{ (segment)} \\
\cdot & \cdot \\
\underline{Y : Dv(K \to L)(K' \to L')} \quad \text{(conv.)} & \cdot \\
\\
Y : DvKK' \to DvLL' & Z : DvKK'
\end{array}}{(YZ) : DvLL'}$$

and the final segment can be, again, simulated as in the basis case.

Case 2: $X = \lambda yY$.

In $X : F$ we have

$$\dfrac{\begin{array}{c}
/ \\
y : H \\
/ \\
\cdot \\
\cdot \\
\cdot \\
Y : K
\end{array}}{\begin{array}{c}
\lambda yY : H \to K \\
\cdot \\
\cdot \text{ (segment)} \\
\cdot \\
X : F
\end{array}}$$

In $X : G$ we have

$$\dfrac{\begin{array}{c} / \\ y : H' \\ / \\ . \\ . \\ . \\ Y : K' \end{array}}{\begin{array}{c} \lambda y Y : H' \to K' \\ . \\ . \text{ (segment)} \\ . \\ X : G \end{array}}$$

and so we have the simulation

$$\dfrac{\begin{array}{rcl} & / & \\ y & : & DvHH' \\ & / & \\ & . & \\ & . & \\ & . & \\ Y & : & DvKK' \end{array}}{\begin{array}{rcl} \lambda y Y & : & DvHH' \to DvKK' \end{array}}$$

$$\dfrac{}{X \quad : \quad Dv(H \to K)(H' \to K')} \text{ (conv.)}$$

and the final segment can be, again, simulated as in the basis case. ◀

As usual we say that an untyped term is *strongly normalizable* if every beta reduction sequence terminates.

B Proof of Lemma 4

Proof. Lemma 4: By induction where we let

$$\begin{array}{rcl} k & = & \text{the length of any } \rightarrowtail \text{ normal form of } F, \\ l & = & \text{the size of the reduction tree of } Y, \\ m & = & \text{the size of the reduction tree of } X, \\ n & = & \text{the length of } X \end{array}$$

and we order the 4-tuples (k, l, m, n) lexicographically. As before we write $X :=$

$$\lambda x_1 \dots x_r \begin{cases} x_i & \\ & X_1 \dots X_2 \\ (\lambda x X_0) & \end{cases}$$

Case 1: $r > 0$. In this case the result follows from the induction hypothesis applied to n.

Case 2: $r = 0$ and x_i is not the free variable y. In this case the result follows by the induction hypothesis applied to the X_j in place of X.

Case 3: $r = 0$ and X has the head redex $(\lambda x X_0)X_1$. In this case the typing of X has the form

$$
\begin{array}{lll}
 & / & \\
 & x : H,\ y : F & \\
 & / & \\
 & \vdots & \\
**** & X_0 : J & \\
\hline
 & & (\to I) \\
 & \lambda x.X_0 : H \to J \qquad y : F & \\
 & \cdot \ \text{segment} \qquad\quad \cdot & \\
 & \cdot \qquad\quad \cdot & \\
 & \lambda x.X_0 : K \to L \qquad X_1 : K & \\
\hline
 & (\lambda x.X_0)X_1 : L & \\
 & \vdots & \\
 & \cdot \ \text{(segment)} & \\
 & \vdots & \\
 & (\lambda x.X_0)X_1 : L & \\
 & \vdots & \\
 & X : G \quad . &
\end{array}
$$

By the remark before Lemma 1 we may assume that the segment from $H \to J$ to $K \to L$ is actually a conversion which factors into a conversion of H to K followed by a conversion of J to L. We note here that no eigenvariable in either segment displayed above can occur in F. We distinguish two subcases.

Subcase 1: x is not free in X_0. By induction hypothesis on n, $[Y/y]X_1$ is strongly normalizable. In addition,

$$
\begin{array}{lll}
y : F & & \\
\vdots & & \\
X_0 : J & & \\
\hline
 & & (\text{conv.}) \\
X_0 : M & & \\
\vdots & & \\
X_0 X_2 \ldots X_s : G. & &
\end{array}
$$

is a typing of the strongly normalizable $X_0X_2 \dots X_s$ for which the induction hypothesis applies to m. Thus,

$$[Y/y](X_0X_2 \dots X_s)$$

is strongly normalizable. But then Barendregt's perpetual strategy terminates when applied to $[Y/y]X$, so $[Y/y]X$ is strongly normalizable.

Subcase 2: x is free in X_0. Now

$$\begin{array}{rcl} y & : & F \\ & \vdots & \\ X_1 & : & K \end{array} \quad \text{(conv.)}$$

$$\begin{array}{rcl} X_1 & : & H, \quad y : F \\ & \vdots & \\ [X_1/x]X_0 & : & J \end{array} \quad \text{(conv.)}$$

$$\begin{array}{rcl} [X_1/x]X_0 & : & M \\ & \vdots & \\ ([X_1/x]X_0)X_2 \dots X_s & : & G \end{array}$$

is a typing of $([X_1/x]X_0)X_2 \dots X_s$ and the induction hypothesis applies to m. Thus, $([X_1/x]X_0)X_2 \dots X_s$ is strongly normalizable and Barendregt's perpetual strategy terminates when applied to $[Y/y]X$. Thus, $[Y/y]X$ is strongly normalizable.

Case 4: $r = 0$ and $x_i = y$ is free in X. Let

$$Y = \lambda y_1 \dots y_t \begin{cases} y_k \\ \\ \lambda z.Z \end{cases} Y_1 \dots Y_q$$

Subcase i: $t = 0$ and Y has no head redex. Then $[Y/y]X$ is strongly normalizable by the induction hypothesis for n applied to the terms $[Y/y]X_j$ and the assumption that Y is strongly normalizable applied to the terms Y_j.

Subcase ii: $t = 0$ and Y has a head redex. By induction hypothesis for n applied to the $[Y/y]X_j$ we have that

$$x[Y/y]X_1 \dots [Y/y]X_s$$

is strongly normalizable and we can apply the induction hypothesis for ℓ to

$$[([Y_1/z]Z)Y_2 \dots Y_q/x](x[Y/y]X_1 \dots [Y/y]X_s)$$

where $x : F$.

Subcase iii: $t > 0$. For easier notation let $Y = \lambda z.Z$. In this case the typing of X has the form

$$\dfrac{\dfrac{\begin{array}{l} y : F \\ \vdots \\ \cdot \quad \text{(segment)} \\ \vdots \\ y : H \to K \end{array} \qquad \begin{array}{l} y : F \\ \vdots \\ \cdot \\ \vdots \\ X_1 : H \end{array}}{yX_1 : K}}{\begin{array}{l} yX_1 : J \\ \vdots \\ X : G \end{array}} \; (\text{conv.}) \quad , \quad y : F$$

and the typing of Y has the form

$$\dfrac{\begin{array}{l} \not / \\ Z : L \\ \not / \\ \vdots \\ Z : M \end{array}}{\begin{array}{l} \lambda z.Z : L \to M \\ \vdots \\ \cdot \quad \text{(segment)} \\ \vdots \\ \lambda z.Z : F \end{array}} \; (\to I)$$

Now the segment

$$\begin{array}{l} \lambda z.Z : L \to M \\ \vdots \\ \cdot \quad \text{(segment)} \\ \vdots \\ \lambda z.Z : F \\ \vdots \\ \cdot \quad \text{(segment)} \\ \vdots \\ \lambda z.Z : H \to K \end{array}$$

reduces to a conversion by the remark preceeding Lemma 1 and we have the typings

$$\dfrac{\begin{array}{c} Z : H \end{array}}{\begin{array}{c} z : L' \\ \vdots \\ Z : M' \end{array}} \quad (\text{conv.})$$

$$Z : J$$

for suitable instances L' of L and M' of M and

$$\dfrac{\begin{array}{c} / \\ Z : L \\ / \\ \vdots \\ Z : M \end{array}}{\lambda z.Z : L \to M} \quad (\to I)$$

$$\begin{array}{rcl} \lambda z.Z & : & L \to M \\ & \vdots & \\ & \cdot & \text{(segment)} \\ & \vdots & \\ \lambda z Z & : & F \\ & \vdots & \\ & \vdots & \\ & \vdots & \\ [Y/y]X_1 & : & H \end{array}$$

and

$$\dfrac{\begin{array}{c} / \\ Z : L \\ / \\ \vdots \\ Z : M \end{array}}{\lambda z.Z : L \to M} \quad (\to I)$$

$$\begin{array}{rcl} \lambda z.Z & : & L \to M \\ & \vdots & \\ & \cdot & \text{(segment)} \\ & \vdots & \\ x : J, \quad Y & : & F \\ \vdots & & \\ \vdots & & \\ x([Y/y]X_2)\ldots([Y/y]X_s) & : & G \end{array}$$

Thus, by induction hypothesis for n, $x([Y/y]X_2)\ldots([Y/y]X)s$, is strongly normalizable. By induction hypothesis for n, $[Y/y]X_1$ is strongly normalizable. Now the length of any $\rightarrowtail$ normal form of H is less than that of F since H conv. L' and $L \to M$ conv. F. Thus, by induction hypothesis for k

$$[[Y/y]X_1/z]Z$$

is strongly normalizable. In addition, the length of any $\rightarrowtail$ normal form of J is less than that of F since H conv. L' and $L \to M$ conv. F. Thus, by induction hypothesis for k

$$([[Y/y]X_1/z]Z)([Y/y]X_2)\ldots([Y/y]X_s)$$

is strongly normalizable. Hence, Barendregt's perpetual strategy terminates for $[Y/y]X$ and it is strongly normalizable. ◀

Semantics of Intensional Type Theory extended with Decidable Equational Theories *

Qian Wang[1,3,4,5,6] and Bruno Barras[1,2]

1 **Laboratoire d'Informatique de L'École Polytechnique**
2 **INRIA Saclay – Île de France**
3 **Key Laboratory for Information System Security, Ministry of Education**
4 **Tsinghua National Laboratory for Information Science and Technology (TNList)**
5 **School of Software, Tsinghua University**
6 **Department of Computer Science and Technology, Tsinghua University**

Abstract

Incorporating extensional equality into a dependent intensional type system such as the Calculus of Constructions (CC) provides with stronger type-checking capabilities and makes the proof development closer to intuition. Since strong forms of extensionality generally leads to undecidable type-checking, it seems a reasonable trade-off to extend intensional equality with a decidable first-order theory, as experimented in earlier work on CoqMTU and its implementation CoqMT.

In this work, CoqMTU is extended with strong eliminations. The meta-theoretical study, particularly the part relying on semantic arguments, is more complex. A set-theoretical model of the equational theory is the key ingredient to derive the logical consistency of the formalism. Strong normalization, the main lemma from which type-decidability follows, is proved by attaching realizability information to the values of the model.

The approach we have followed is to first consider an abstract notion of first-order equational theory, and then instantiate it with a particular instance, Presburger Arithmetic. These results have been formalized using Coq.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Calculus of Constructions, Extensional Type Theory, Intensional Type Theory, Model, Meta-theory, Consistency, Strong Normalization, Presburger Arithmetic

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.653

1 Introduction

Most proof assistants, such as Coq [19], implement intensional type theory because extensional type theory is usually undecidable. Coq implements ECIC, a type theory that extends CC [8] with two more features: inductive types in the style of the Calculus of Inductive Constructions (CIC) [12], and a predicative hierarchy of universes, in the style of the Extended Calculus of Constructions (ECC) [10].

The purely intensional version of type-theory may become awkward when it comes to programming with dependent types. In the well-known examples of "vectors", one has the type $\mathsf{Vect}(n)$ of lists of length n, a concatenation function @ such that $v_1@v_2$ has length

* Supported by National Science Foundation of China grant 61272002, Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation 2011-9, Major Research plan of the National Natural Science Foundation of China grant 91218302, National Basic Research Program of China (973 Program) grant 2010CB328003.

$n_1 + n_2$ whenever v_i has length n_i. But showing that $v@\mathsf{nil} = v$ is not possible because it is not even a well-typed statement: lengths $n + 0$ and n are not identified because $+$ is defined recursively on the first argument (n here) which is not in the form of 0 or successor.

Several works tried to fix this problem by either adding rewrite rules to increase the computing ability [5] or including in extensional equalities [14, 15, 11]. Unfortunately, such solutions have never been implemented as a proof assistant until CoqMT [17] which is evolved from [6, 7, 16] and further generalized by CoqMTU [4]. The idea is a trade-off between decidability and type-checking capabilities, only allowing decidable extensional theory and automatically checking the theory equality by a decision procedure. Though this solution looks nice, the meta-theory of CoqMTU is not well understood yet: confluence and subject reduction of the full calculus are proved, but strong normalization (SN) and consistency are only proved in absence of strong elimination.

Our first contribution, shown in Section 4, is that formalizing a new schema (CCT) incorporating an abstract decidable theory into CC family, which is not only ECIC implemented by Coq, but a further extension of it. The abstract theory can be instantiated by any concrete one of interest if it can be represented in CCT and satisfies special assumptions to ensure the main meta-theoretical properties. CCT captures many calculus, but we provide a uniform way of establishing the meta-theoretical properties proved semantically.

Our second contribution, shown in Section 5, is about proving the key properties of CCT: consistency and strong normalization (with strong elimination), the basis for proving the meta-theory of other calculus extending CC and admitting a decidable theory. As often, this requires a set-theoretical model. We have based our study on the work of the second author [3, 2], that provides a modular framework for modeling a wide range of type theories from CC to the formalism of Coq. In this article, we show that this framework can be reused and that it accommodates the extra features of CCT.

To implement CICUT as a proof assistant, we must investigate its syntactic properties, among which only Church-Rosser can not be proved in the usual way, because we do not embed extensional equations into ι-reduction as CoqMTU does. Nevertheless, it is not a disaster because we believe Church-Rosser still hold and will study the whole syntactic properties of CICUT in the future.

Finally, our last contribution, shown in Section 6, is to show that Presburger Arithmetic fits perfectly in this abstract notion of decidable first-order theory, which also demonstrates our abstraction philosophy works well.

In the following sections, most of the proofs will not be shown in the paper due to the page limit. They have been done in the Coq development and available to whom are interested[1]. Many similar notations are overloaded to avoid ambiguity, such as λ is overloaded by $\dot{\lambda}$ and $\check{\lambda}$ to represent abstractions of set and pure λ-term respectively. We will detail when we need. The notations without overloading are Coq primitives, such as $\forall$ for universal quantification because the whole work is formally done in Coq. To make the concepts easier to understand, we use $\to$ for function types only, and use $\Rightarrow$ for implication. We also try to hide De Bruijn index, though it is heavily used in the development.

2 CICUT

Since the focus of this paper is the semantic meta-theory, the syntax of CICUT will not be exhaustively introduced here. The core of CICUT is almost the same as that of the CC.

[1] The development is available at `https://github.com/superwalter/SETheory-dev`

$$\frac{}{\Gamma \vdash} \quad \frac{\Gamma \vdash T : s}{\Gamma\,[x : T] \vdash} \quad \frac{\Gamma \vdash, \quad \Gamma(n) = (x : T)}{\Gamma \vdash n :\uparrow^{n+1} T} \quad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{Prop} : \mathsf{Kind}}$$

$$\frac{\Gamma \vdash T : s_1, \quad \Gamma\,[x : T] \vdash U : s_2}{\Gamma \vdash \Pi x : T.U : s_2} \quad \frac{\Gamma\,[x : T] \vdash M : U}{\Gamma \vdash \lambda x : T.M : \Pi x : T.U}$$

$$\frac{\Gamma \vdash M : \Pi x : T.U, \quad \Gamma \vdash N : T}{\Gamma \vdash M\,N : U[x \backslash N]} \quad \frac{\Gamma \vdash M : T, \quad \Gamma \vdash T = T' : s}{\Gamma \vdash M : T'}$$

$$\frac{\Gamma \vdash M : T}{\Gamma \vdash M = M : T} \quad \frac{\Gamma \vdash M = M' : T}{\Gamma \vdash M' = M : T} \quad \frac{\Gamma \vdash M_1 = M_2 : T, \quad \Gamma \vdash M_2 = M_3 : T}{\Gamma \vdash M_1 = M_3 : T}$$

$$\frac{\Gamma \vdash T = T' : s_1, \quad \Gamma\,[x : T] \vdash U = U' : s_2}{\Gamma \vdash \Pi x : T.U = \Pi x : T'.U' : s_2} \quad \frac{\Gamma \vdash T = T' : s_1, \quad \Gamma\,[x : T] \vdash M = M' : U}{\Gamma \vdash \lambda x : T.M = \lambda x : T'.M' : \Pi x : T.U}$$

$$\frac{\Gamma \vdash M = M' : \Pi x : T.U, \quad \Gamma \vdash N = N' : T}{\Gamma \vdash M\,N = M'\,N' : U[x \backslash N]} \quad \frac{\Gamma\,[x : T] \vdash M = M' : U, \quad \Gamma \vdash N = N' : T}{\Gamma \vdash (\lambda x : T.M)\,N = M'[x \backslash N'] : U[x \backslash N]}$$

Figure 1 Calculus of Constructions, judgmental presentation.

Figure 1 shows a judgmental presentation of the typing rules of CC, it is different from the usual one by replacing an equivalence relation on untyped λ-terms by equality judgments. $\Gamma \vdash M = M' : T$ expresses that M is equal to M', both being of type T in context Γ. [13] shows these two representations are equivalent, and we will extend this conclusion for all of the extensions considered in this article, leaving the proof to the future.

The main novel feature of CICUT and CoqMTU in the syntax of terms is the embedding of a type of first-order terms. Regarding the typing rules, the main difference is the extension of the definitional equality with a decidable theory $\sim_{\mathcal{T}}$ on these first-order terms. See [4] for a more detailed presentation. Let us just give the extra inference rules (besides Presburger arithmetic axioms) that need to be considered to have CICUT instantiated with Presburger arithmetic. It introduces three canonical constants and a defined symbol (Rec):

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{nat} : \mathsf{Kind}} \quad \frac{\Gamma \vdash}{\Gamma \vdash 0 : \mathsf{nat}} \quad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{S} : \mathsf{nat} \to \mathsf{nat}}$$

$$\frac{\Gamma \vdash P : \mathsf{nat} \to s, \quad \Gamma \vdash N : \mathsf{nat}, \quad \Gamma \vdash M_0 : P\,0, \quad \Gamma \vdash M_1 : \Pi n : \mathsf{nat}.\, P\,n \to P\,(\mathsf{S}\,n)}{\Gamma \vdash \mathsf{Rec}(P, N, M_0, M_1) : P\,N}$$

$$\frac{\Gamma \vdash \quad M \sim_{\mathcal{T}} N}{\Gamma \vdash M = N : \mathcal{T}} \quad \frac{N \sim_{\mathcal{T}} 0}{\mathsf{Rec}(P, N, M_0, M_1) \sim_{\mathcal{T}} M_0}$$

$$\frac{N \sim_{\mathcal{T}} \mathsf{S}\,N'}{\mathsf{Rec}(P, N, M_0, M_1) \sim_{\mathcal{T}} M_1\,N'\,\mathsf{Rec}(P, N', M_0, M_1)}$$

The main difference between CICUT and CoqMTU is the abandon of incorporating definitional equalities into reductions and restore the strong elimination, which is witnessed

by the fact that the eliminator of natural numbers (Rec) can be used with a term P belonging to any sort s. By contrast, weak eliminations are obtained by restricting s to the Prop, the sort of propositions. In other words, weak elimination provides the induction principle of the natural numbers, but not the possibility to define functions by structural induction.

3 Extensible set-theoretical realizability model of CC

This section gives a short introduction to a general method used to build consistency and strong normalization models of a wide range of type-theories with dependent types. See [2] for more details.

Consistency of such formalisms is often achieved by providing a set-theoretical model, that interprets terms and types as sets. The judgment that a term has a given type is interpreted by the proposition that the term is interpreted by a member of the interpretation of the type. The soundness of such a model implies the consistency of the formalism, as soon as one type (representing the absurd proposition) is interpreted by the empty set.

Strong normalization (SN) is the property that any well-typed term of a type system cannot be reduced ad infinitum. It often captures the logical strength of the type system seen as a logical formalism. This is why it generally requires a particular model construction. Types are not mere sets of values anymore, but they should also be interpreted as sets of strongly normalizing λ-terms, that represent the possible terms that have this type. The latter are often called "realizers". The soundness of the model shall also require that the term can be interpreted by a realizer of its type. SN is thus a consequence of the construction of such a realizability model.

The main ingredient in this model construction is the notion of saturated sets due to Tait [18] (but Girard's reducibility candidates serve the same purpose). They are sets of strongly normalizing λ-terms such that the type constructors (such as the arrow type or intersection) can be interpreted.

SN of CC and CIC in presence of weak elimination can be proved in this setting. However, to take care of strong elimination, we need a more subtle definition of realizers: a saturated set for the value of each type is not enough, each member of the value of a type should have its own saturated set of realizers (see the R function below). This is closely related to the notion of Λ-sets introduced by Altenkirch [1].

In the remainder of this section, we show how the model construction can be carried out, provided we can implement the two signatures below: the first one gathers what is required to build a set-theoretical model (the set-denotation, symbols will be overloaded with a $\dot{}$); the second one corresponds to the extra requirements to have a full realizability model (the term-denotation, symbols will be overloaded with a $\check{}$), from which SN will follow.

3.1 Abstract Parameterized Models

The first abstract model is designed to provide a uniform structure of set denotations to all the models, hiding the implementation until instantiation. It contains the set-denotation for all closed terms in the model together with the properties to ensure soundness. It uses a higher-order presentation: binding constructions are represented using functional arguments.

▶ **Definition 1** (Abstract model)**.** The model has the following signature:

$$\mathsf{X} : \mathit{Type} \qquad \dot{\in} : \mathsf{X} \to \mathsf{X} \to \mathit{Prop} \qquad \doteq : \mathsf{X} \to \mathsf{X} \to \mathit{Prop} \qquad \bigstar : \mathsf{X}$$

$$\dot{@} : \mathsf{X} \to \mathsf{X} \to \mathsf{X} \qquad \dot{\lambda} : \mathsf{X} \to (\mathsf{X} \to \mathsf{X}) \to \mathsf{X} \qquad \dot{\Pi} : \mathsf{X} \to (\mathsf{X} \to \mathsf{X}) \to \mathsf{X}$$

satisfying certain properties (the followings are some examples):

$$\frac{N \dot{\in} A}{(\dot{\lambda} x \dot{\in} A.f) \dot{@} N \doteq F(N)}\ [\beta] \qquad \frac{\forall x \dot{\in} A.F(x) \dot{\in} \bigstar}{(\dot{\Pi} x \dot{\in} A.F) \dot{\in} \bigstar}\ [\textsc{Imp}] \qquad \frac{\forall x \dot{\in} A.(f(x)) \dot{\in} (F(x))}{(\dot{\lambda} x \dot{\in} A.f) \dot{\in} (\dot{\Pi} x \dot{\in} A.F)}\ [\Pi\text{-I}]$$

where $\dot{\lambda} x \dot{\in} A.f$ stands for $\dot{\lambda}(A, x \mapsto f(x))$ and $\dot{\Pi} x \dot{\in} A.B$ for $\dot{\Pi}(A, x \mapsto B(x))$.

X is the type of values, that can be seen as a kind of set-theory since we assume we have relations $\dot{\in}$ and $\doteq$ (equality and membership). $\bigstar$ is the set of all the values. Other symbols and related properties are specific to CC, hence look similar to their counterpart in CC.

The second abstract model is an supplement of the first one when upgrading a consistency model to a SN model. The key parameter is the function R that takes a type and one of its elements, and returns the saturated set formed by the realizers of this element.

▶ **Definition 2** (Abstract supplement of SN model)**.** This model aims at building a saturated set for each type and indicates that each proposition is inhabited.

$$✠ : \mathsf{X} \qquad ✠ \in \dot{\Pi}(P \dot{\in} \bigstar).P \qquad \mathsf{R} : \mathsf{X} \to \mathsf{X} \to \mathsf{SAT}$$

$$\mathsf{R}(\dot{\Pi} x \dot{\in} A.B, f) = \bigcap_{x \dot{\in} A} \mathsf{R}(A, x) \overset{sat}{\to} \mathsf{R}(B(x), f \dot{@} x) \qquad \mathsf{R}(\bigstar, P) = \mathsf{SN}$$

where SAT and SN are the sets of all saturated sets and all SN pure λ-terms respectively. $\bigcap$ and $\overset{sat}{\to}$ are standard set intersection and product on saturated sets.

The instance ✠ (the *daimon*) ensures that any type is inhabited including *False*, such that SN is guaranteed in arbitrary type context. Consequently, consistency can not be proved as in the consistency model, but as a matter of fact, it still can be deduced as will be shown in Section 5.

3.2 Main Model Construction

The main model is called M, consisting in giving an account of all syntactic entities (terms, judgments, derivations) based on an instance of the signatures above. The denotations of open terms depend on the valuations of free variables, thus a term is encoded as a pair of functions each taking a valuation ($\mathsf{N} \to \mathsf{X}$ for set-denotation, or $\mathsf{N} \to \Lambda$ for term-denotation) as a parameter returning a set or a pure λ-term as denotation, with some requirements to ensure consistency. Since de Bruijn indices are used, the arguments of valuations are natural numbers. Λ is the type of pure λ-terms.

Due to space constraints, we do not expose the full details of how sorts are dealt with. In the following, we will only consider the sort of propositions. See the formal development or [2] for an exact account. And the symbols of M will be overloaded with a $\tilde{}$.

▶ **Definition 3** (Pseudo-terms)**.** A *term* is a pair of a set and term denotation:

$$\mathsf{Term} \triangleq \{f : (\mathsf{N} \to \mathsf{X}) \to \mathsf{X}\} \times \{g : (\mathsf{N} \to \Lambda) \to \Lambda \mid \mathsf{sub}(g) \wedge \mathsf{lift}(g)\}$$

where $\mathsf{sub}(g)$ and $\mathsf{lift}(g)$ assert that g commutes with substitution and relocation.

We use $\mathsf{Val}(t)_i \triangleq f(i)$ and $\mathsf{Tm}(t)_j \triangleq g(j)$ to denote the set-denotation and term-denotation of term t with certain valuations i and j.

▶ **Definition 4** (Explicit substitution)**.** An *explicit substitution* is a pair:

$$\mathsf{Esub} \triangleq \{f : (\mathsf{N} \to \mathsf{X}) \to (\mathsf{N} \to \mathsf{X})\} \times \{g : (\mathsf{N} \to \Lambda) \to (\mathsf{N} \to \Lambda) | \mathsf{El}(g), \mathsf{Es}(g)\}$$

where $\mathsf{El}(g)$ and $\mathsf{Es}(g)$ are commutation properties similar to the previous definition.

We use $\sigma(i)$ and $\sigma(j)$ to represent $f(i)$ and $g(j)$ for an explicit substitution σ.

▶ **Definition 5** (Term constructors)**.** *Constructors* in M are encoded as:

$$\begin{array}{rcll}
\mathsf{Prop} & \triangleq & \langle i \mapsto \bigstar, & j \mapsto \mathrm{K} \rangle \\
\tilde{n} & \triangleq & \langle (i \mapsto i(n)), & j \mapsto j(n) \rangle \\
M \tilde{@} N & \triangleq & \langle i \mapsto \mathsf{Val}(m)_i \dot{@} \mathsf{Val}(N)_i, & j \mapsto \mathsf{Tm}(m)_j \check{@} \mathsf{Tm}(N)_j \rangle \\
\tilde{\lambda} A.t & \triangleq & \langle i \mapsto \dot{\lambda} x \dot{\in} \mathsf{Val}(A)_i . \mathsf{Val}(t)_{i'}, & j \mapsto \mathrm{K} \check{@} (\check{\lambda} x . \mathsf{Tm}(t)_{j'}) \check{@} \mathsf{Tm}(A)_j \rangle \\
\tilde{\Pi} A.B & \triangleq & \langle i \mapsto \dot{\Pi} x \dot{\in} \mathsf{Val}(A)_i . \mathsf{Val}(B)_{i'}, & j \mapsto \mathrm{K} \check{@} \mathsf{Tm}(A)_j \check{@} \check{\lambda} x . \mathsf{Tm}(B)_{j'} \rangle
\end{array}$$

where K is the combinator $\check{\lambda} x . \check{\lambda} y . x$, i' is the function such that $i'(0) = x$ and $i'(n+1) = i(n)$, and j' is defined as $j'(0) = j(0)$ and $j'(n+1) = \check{\uparrow}^1 j(n)$.

Note that we just show the pair omitting the trivial proof of the assertions. The K combinator used in the second component of $\tilde{\lambda}$ and $\tilde{\Pi}$ allows to simulate CC-reductions occurring in types. Many properties such as $\mathsf{Val}(M \tilde{@} N)_i = \mathsf{Val}(M)_i \dot{@} \mathsf{Val}(N)_i$ follow straightforwardly.

Similarly, we define several instances of explicit substitution, identity and cons:

$$\mathsf{id} \triangleq \langle i \mapsto i, j \mapsto j \rangle \qquad \sigma \cdot M \triangleq \langle i \mapsto \mathsf{Val}(M)_i \odot \sigma(i), j \mapsto \mathsf{Tm}(M)_j \odot \sigma(j) \rangle$$

where $i \mapsto x \odot \sigma \triangleq [0 \mapsto x, \ldots, n+1 \mapsto \sigma(n)] (n \geq 0)$.

An *explicit substitution* σ applied to a term M is defined as:

$$M[\sigma] \triangleq \langle i \mapsto \mathsf{Val}(M)_{\sigma(i)}, j \mapsto \mathsf{Tm}(M)_{\sigma(j)} \rangle .$$

▶ **Definition 6** (Judgment)**.** A *type judgment*, denoted by $M : T$, holds for valuations i j iff:

$$[M : T]_{i,j} \triangleq \mathsf{Tm}(M)_j \Vdash_{\mathsf{Val}(T)_i} \mathsf{Val}(M)_i$$

where $t \Vdash_T x$ stands for $x \dot{\in} T \wedge t \check{\in} \mathsf{R}(T, x)$, which reads as "$t$ realizes x of type T".

▶ **Definition 7** (Semantics of context)**.** The denotation of a M-context Γ is a set of pairs of valuations defined as :

$$[\Gamma] = \{(i,j) \mid \forall n . [n : \tilde{\uparrow}^{n+1} \Gamma(n)]_{i,j}\}$$

We will write membership of (i,j) to $[\Gamma]$ as $(i,j) \tilde{\in} [\Gamma]$.

▶ **Definition 8** (Judgments with context)**.** There are four kinds of judgments:

$$\begin{array}{rcll}
\tilde{\vdash} \Gamma & \triangleq & \exists (i,j) . (i,j) \tilde{\in} [\Gamma] & \text{(Well-founded Judgment)} \\
\Gamma \tilde{\vdash} M \tilde{\in} T & \triangleq & \forall (i,j) \tilde{\in} [\Gamma] . [M : T]_{i,j} & \text{(Typing Judgment)} \\
\Gamma \tilde{\vdash} M \doteq N & \triangleq & \forall (i,j) \tilde{\in} [\Gamma] . \mathsf{Val}(M)_i \doteq \mathsf{Val}(N)_i & \text{(Equality Judgment)} \\
\Gamma_1 \tilde{\vdash} \sigma \triangleright \Gamma_2 & \triangleq & \forall (i,j) \tilde{\in} [\Gamma_1], (\sigma(i), \sigma(j)) \tilde{\in} [\Gamma_2] & \text{(Explicit Substitution Judgment)}
\end{array}$$

To express strong normalization, we need to express the notion of reduction between pseudo-terms. This can be based on the set-denotation since the $[\beta]$ parameter of Def. 1 assigns the same set to β-equivalent pseudo-terms. A pseudo-term reduces to another if the term-denotation of the former reduces to that of the latter, whatever the valuation:

▶ **Definition 9.** The *pseudo-reduction* (one step or more) in M is defined as:

$$M \tilde{\to} M' \triangleq \forall j. \mathsf{Tm}(M)_j \check{\to}^+ \mathsf{Tm}(M')_j$$

A pseudo-term is said strongly normalizing if there is no infinite chain of pseudo-reduction starting from it. The abstract SN lemma can be proved now:

▶ **Lemma 10** (Abstract SN). *For any well-typed term t in a valid context Γ, t is SN.*

$$\tilde{\vdash} \Gamma \ \wedge \ \Gamma \tilde{\vdash} t \tilde{\in} T \ \Rightarrow \ SN(t)$$

Proof. By $\tilde{\vdash} \Gamma$, we have $\exists (p,q) \tilde{\in} [\Gamma]$. By $\Gamma \tilde{\vdash} t \tilde{\in} T$, we have $\mathsf{Tm}(t)_j \check{\in} \mathsf{R}(\mathsf{Val}(T)_i, \mathsf{Val}(t)_i)$ for all $(i,j) \tilde{\in} [\Gamma]$. Since $\mathsf{R}(\mathsf{Val}(T)_i, \mathsf{Val}(t)_i)$ is a saturated set, we have $\mathsf{Tm}(t)_q$ is SN by applying (p,q). Any reduction from t can be simulated by a reduction from $\mathsf{Tm}(t)_q$, hence t is SN as a consequence of $\mathsf{Tm}(t)_q$ is SN. ◀

3.3 Soundness of the Main Model

There are four steps to investigate the meta-theories of CC using the model above. The first is to prove the existence of a model, that is, there are instances of the abstract models.

We can show that the abstract models can be instantiated in intuitionistic Zermelo-Frankel set theory with replacement (IZF_R). Types are encoded as a couples formed of a set of values and a function from this set towards saturated set.

All propositions have the same set-denotation $\{\emptyset\}$, and associate a saturated set to $\emptyset$. So the type of all propositions is defined as

$$\bigstar \triangleq (\{(\emptyset, \_ \mapsto S) | S \check{\in} \mathsf{SAT}\}, \_ \mapsto \mathsf{SN}).$$

Dependent product is based on an alternative encoding of functions due to Aczel (see [2] for details), in order to interpret the impredicativity of Prop. The term-denotation of products is fixed by the abstract model. Finally, the daimon ✠ has to be taken as $\emptyset$.

The second step is to prove that the model interprets CC correctly. Syntax maps to semantic terms in the model straightforwardly.

▶ **Theorem 11** (Soundness). *If $\Gamma \vdash t : T$ (in CC, see Fig. 1), then $\tilde{\Gamma} \tilde{\vdash} \tilde{t} \tilde{\in} \tilde{T}$ (in M).*

The third step is the prove the consistency and SN in M. Consistency can now be proved independently from strong normalization:

▶ **Theorem 12** (Consistency). *The model M is consistent.*

$$\forall M, \neg([\,] \tilde{\vdash} M \tilde{\in} (\tilde{\Pi} P : \mathsf{Prop}.P))$$

Proof. Assume there is closed proof t of $False \triangleq \tilde{\Pi} \mathsf{Prop}.\tilde{0}$ in the model. Then, the term interpretation of the proof t should be closed by commutation with substitution. By the properties of $\tilde{\Pi}$, $\mathsf{Tm}(t)_j \check{@} u$, where u and j are respectively a closed term and a closed valuation, should be in all saturated sets. As a consequence, it must contain free variables which must be in $\mathsf{Tm}(t)_j$ since u is closed. Since both t and j are closed, $\mathsf{Tm}(t)_j$ should be closed, a contradiction. ◀

▶ **Theorem 13** (Strong normalization). *Well typed M-terms are SN.*

Finally, by soundness result, consistency and SN of CC is an immediate consequence of the consistency and SN of M.

4 A Sound Model of Abstract First Order Theory

Compared with sticking to some specific first-order theory, we are more interested in investigating an abstract model ($\mathsf{M_T}$) such that the abstract meta-theoretical results established in that model can apply to any of its instance.

There are three principles to design such an abstract model. Firstly, this model should capture as many theories as possible which leads to an *abstract expression* of its signature and axioms. Secondly, this model should provide enough evidence to ensure only the valid theories are included : including in such a theory would not break the meta-theories established in M. The evidence can be taken as abstract property about the abstract expressions, which we call *assumption*. At last, we want to carry out the tough work as much as possible, such that it is not so hard to instantiate this model later for any allowable specific theory. There should be as few assumptions as possible and each assumption should be the familiar concept. Many decidable theories are first order, hence we restrict us to first-order theories.

To prove the soundness, we also need to abstractly formalize the syntax of the theory as well as the interpretation rules following the abstraction schema of $\mathsf{M_T}$. There are two kinds of abstraction: abstract expressions and assumptions. When instantiating, all abstract expressions and assumptions should be specified and proved respectively.

Following Coq's convention, environment **Parameter** and **Axiom** are used for abstract expression and assumption respectively. Environment **Definition** and **Lemma** are only used for concrete definition and property. Since $\mathsf{M_T}$ is an extension of M, the symbols in $\mathsf{M_T}$ are also reloaded by $\tilde{}$. Syntactic symbols are overloaded by $\hat{}$ for formulas and $\bar{}$ for terms.

4.1 Syntax of the Abstract Theory

Classically, first-order theory consists in four parts: the signature, the formulas, the axioms and the inference rules.

▶ **Parameter 14** (Signature)**.** The domain and the operations of signature are:

$$\mathsf{T} : \mathsf{Set}; \quad \bar{\uparrow} : \mathsf{T} \to \mathsf{N} \to \mathsf{N} \to \mathsf{T}; \quad \bar{\Theta} : \mathsf{T} \to \mathsf{T} \to \mathsf{N} \to \mathsf{T}; \quad \bar{\Phi} : \mathsf{T} \to \mathsf{N} \to [..\mathsf{N}..]$$

where Set is the Coq's primitive type, $[..\mathsf{N}..]$ is the list of natural numbers. $\bar{\Theta}_k^u t$ is the substitution operation on t, $\bar{\Phi}^k t$ the function collecting the free variables in t, and $\bar{\uparrow}_k^n t$ the operation relocating the free variables in t. The latter three operations are indexed by some number k corresponding to the depth at which the operator is applied.

▶ **Definition 15** (Formulas)**.** Formulas are defined inductively upon signature:

$$\mathsf{F} ::= \hat{\top} \mid \hat{\bot} \mid f \hat{\Rightarrow} g \mid f \hat{\vee} g \mid f \hat{\wedge} g \mid t \sim_{\mathcal{T}} u \mid \hat{\neg} f \mid \hat{\forall} f \mid \hat{\exists} f$$

where f and g are formulas, and t and u are first-order terms.

Note that equality is the only predicate we are interested in. We shall (again) use $\hat{\Phi}^k f$, $\hat{\uparrow}_k^n f$ and $\hat{\Theta}_k^N f$ to denote respectively the set of free variables, the relocation of free variables and the substitution operating on formulas.

▶ **Definition 16** (Context)**.** The *context* (notated as C) is a list of declarations corresponding to either a term variable or an assumption:

$$\mathsf{C} \triangleq [..(\mathsf{T} + \{f : \mathsf{F}\})..].$$

Since theories considered here are first-order, a well-formed formula should contain term variables only.

▶ **Definition 17** (Well-formed term and formula)**.** Given a context H,

$$\mathsf{Wf_t}(H,t) \triangleq \forall n \in_l \bar{\Phi}t, H(n) = \mathsf{T}; \quad \mathsf{Wf_f}(H,g) \triangleq \forall n \in_l \hat{\Phi}g, H(n) = \mathsf{T}$$

A valid formula should be a well-formed formula justified from the axioms by a succession of judgments. We define first the (abstract) notion of axiom, which should be well-formed formulas. Inference rules for generating theorems from the axioms come next. They are defined here in terms of introduction rules, elimination rules and judgmental rules.

▶ **Parameter 18** (Axioms)**.** Axioms are Coq's predicates which judges special formulas in a context, further they should be well-formed :

$$\widehat{\mathsf{Ax}} : \mathsf{C} \to \mathsf{F} \to \mathit{Prop}; \quad \widehat{\mathsf{Ax}}(H,f) \Rightarrow \mathsf{Wf_f}(H,f)$$

▶ **Definition 19** (Derivation rules)**.** A derivation rule ($\hat{\vdash} : \mathsf{C} \to \mathsf{F} \to$ Prop) is a Coq's predicate with arity two defined inductively as followings:

$$\frac{H(n) = f, \quad \mathsf{Wf_f}(H, \hat{\uparrow}^{(S\ n)} f)}{H \hat{\vdash} \hat{\uparrow}^{(S\ n)} f} \quad \frac{\widehat{\mathsf{Ax}}(H,f)}{H \hat{\vdash} f} \quad \frac{}{H \hat{\vdash} \hat{\top}} \quad \frac{H \hat{\vdash} \bot, \quad \mathsf{Wf_f}(H,f)}{H \hat{\vdash} f}$$

$$\frac{H \hat{\vdash} f \hat{\rightharpoonup} \hat{\bot}}{H \hat{\vdash} \hat{\neg} f} \quad \frac{H \hat{\vdash} \hat{\neg} f}{H \hat{\vdash} f \hat{\rightharpoonup} \hat{\bot}} \quad \frac{H \hat{\vdash} f_1, \quad H \hat{\vdash} f_2}{H \hat{\vdash} f_1 \hat{\wedge} f_2} \quad \frac{H \hat{\vdash} f_1 \hat{\wedge} f_2}{H \hat{\vdash} f_1} \quad \frac{H \hat{\vdash} f_1 \hat{\wedge} f_2}{H \hat{\vdash} f_2}$$

$$\frac{H \hat{\vdash} f_1, \mathsf{Wf_f}(H, f_2)}{H \hat{\vdash} f_1 \hat{\vee} f_2} \quad \frac{H \hat{\vdash} f_2, \mathsf{Wf_f}(H, f_1)}{H \hat{\vdash} f_1 \hat{\vee} f_2} \quad \frac{H \hat{\vdash} f_1 \hat{\vee} f_2, (f_1 :: H) \hat{\vdash} \hat{\uparrow}^1 f_3, (f_1 :: H) \hat{\vdash} \hat{\uparrow}^1 f_3}{H \hat{\vdash} f_3}$$

$$\frac{\mathsf{Wf_f}(H, f_1), (f_1 :: H) \hat{\vdash} \hat{\uparrow}^1 f_2}{H \hat{\vdash} f_1 \hat{\rightharpoonup} f_2} \quad \frac{H \hat{\vdash} f_1 \hat{\rightharpoonup} f_2, H \hat{\vdash} f_1}{H \hat{\vdash} f_2} \quad \frac{(\mathsf{T} :: H) \hat{\vdash} f}{H \hat{\vdash} \hat{\forall} f} \quad \frac{H \hat{\vdash} \hat{\forall} f, \mathsf{Wf_t}(H,t)}{H \hat{\vdash} \hat{\Theta}^t f}$$

$$\frac{H \hat{\vdash} \hat{\Theta}^t f \quad \mathsf{Wf_t}(H,t)}{H \hat{\vdash} \hat{\exists} f} \quad \frac{H \hat{\vdash} \hat{\exists} f \quad (f :: \mathsf{T} :: H) \hat{\vdash} \hat{\uparrow}^2 g}{H \hat{\vdash} g}$$

Valid formulas (or theorems) are those formulas that can be derived by application of the above rules. Valid formulas should be well-formed:

▶ **Lemma 20.** *If* $H \hat{\vdash} f$*, then* $\mathsf{Wf_f}(H,f)$

4.2 The abstract model of the theory

$\mathsf{M_T}$ is another formalization of the theory using the material provided by M. The domain of first-order term is encoded by a constant $\mathcal{S} : \mathsf{Term}$ in $\mathsf{M_T}$. First-order terms also encoded by Term should satisfy the following assumptions:

▶ **Axiom 21.** *The following assumptions aims at ensuring the meta-theory:*

$$\forall \Gamma, \Gamma \tilde{\vdash} \mathcal{S} \tilde{\in} \mathsf{Kind}\ [1] \quad \tilde{\uparrow}^n_k \mathcal{S} \tilde{=} \mathcal{S} \wedge \tilde{\Theta}^N_k \mathcal{S} \tilde{=} \mathcal{S}\ [2]$$

$$\frac{\mathsf{Val}(x)_i \dot{\in} \mathsf{Val}(\mathcal{S})_i \quad \mathsf{Val}(y)_i \dot{\in} \mathsf{Val}(\mathcal{S})_i}{\mathsf{Val}(x)_i \dot{=} \mathsf{Val}(y)_i \vee \neg \mathsf{Val}(x)_i \dot{=} \mathsf{Val}(y)_i}\ [3]$$

$$\frac{\mathsf{Val}(x)_i \dot{\in} \mathcal{S} \quad \mathsf{Val}(y)_i \dot{\in} \mathcal{S} \quad \neg \mathsf{Val}(x)_i \dot{=} \mathsf{Val}(y)_i}{\exists P, P \dot{\in} (\dot{\Pi} x \dot{\in} \mathcal{S}.\bigstar) \wedge P \dot{@} \mathsf{Val}(x)_i \dot{=} \mathit{true} \wedge P \dot{@} \mathsf{Val}(x)_i \dot{=} \mathit{false}}\ [4]$$

where $\mathit{true} \triangleq \dot{\Pi} P \dot{\in} \bigstar . \dot{\Pi} p \dot{\in} P . P$ *and* $\mathit{false} \triangleq \dot{\Pi} P \dot{\in} \bigstar . P$.

Assumption [1] and [2] assert that $\mathcal{S}$ is a closed object of type Kind. Assumption [3] asserts that equation of set-denotations of the first-order term should be decidable. The last assumption is included to ensure we could define the equality of the theory as Leibniz equality later. The idea is that there exists a predicate that discriminates between different values of $\mathcal{S}$.

We further assume that axioms actually hold in the model $\mathsf{M_T}$:

▶ **Axiom 22.** *Axioms are formulas interpreted by provable* M*-terms.*

The encoding of formulas is given by a standard impredicative encoding due to Girard [9].

▶ **Definition 23** (Formulas)**.** The formulas are defined impredicatively:

$$x\tilde{=}y \triangleq \tilde{\Pi}p : (\tilde{\Pi}x : \mathcal{S}.\mathsf{Prop}).\tilde{\Pi}t : (p\tilde{@}x).(p\tilde{@}y) \qquad \tilde{\neg}f \triangleq f\tilde{\to}\tilde{\bot} \qquad A\tilde{\to}B \triangleq \tilde{\Pi}x : A.B$$

$$\tilde{\bot} \triangleq \tilde{\Pi}P : \mathsf{Prop}.P \qquad A\tilde{\vee}B \triangleq \tilde{\Pi}P : \mathsf{Prop}.((A\tilde{\to}P)\tilde{\to}(B\tilde{\to}P)\tilde{\to}P)$$

$$\tilde{\top} \triangleq \tilde{\Pi}P : \mathsf{Prop}.P\tilde{\to}P \qquad A\tilde{\wedge}B \triangleq \tilde{\Pi}P : \mathsf{Prop}.((A\tilde{\to}B\tilde{\to}P)\tilde{\to}P)$$

$$\tilde{\forall}f \triangleq \tilde{\Pi}x : \mathcal{S}.f \qquad \tilde{\exists}f \triangleq \tilde{\Pi}P : \mathsf{Prop}.((\tilde{\Pi}n : \mathcal{S}.f[x/n])\tilde{\to}P)$$

We can prove similar properties to those in Definition 19, but in the format of typing judgments. Among the 26 properties, we only show the following by using Rule [3] and [4] of the previous Assumption 21.

▶ **Lemma 24.** *Equality of theory can be embedded into the typed equality.*

$$\frac{\mathsf{WFCE}(\Gamma) \quad \Gamma\tilde{\vdash}x\tilde{\in}\mathcal{S} \quad \Gamma\tilde{\vdash}y\tilde{\in}\mathcal{S} \quad \Gamma\tilde{\vdash}t\tilde{\in}(x\tilde{=}y)}{\Gamma\tilde{\vdash}x\dot{=}y}\ [\tilde{=}\text{-E}]$$

where $\mathsf{WFCE}(\Gamma)$*, standing for* well-formed closed environment*, is defined as:*

$$\forall(i,j')\tilde{\in}[\Gamma], \exists j, (i,j)\tilde{\in}[\Gamma] \wedge \mathsf{CPT}(j)$$

$\mathsf{CPT}(j)$ *stands for* Closed Pure Term*, means* j *always returns a closed term to any index.*

WFCE looks strange but necessary. In M, the value of any proposition is the singleton of empty, the distinction between true propositions and false propositions reflects in their realizers. False proposition contains open realizers only, while true proposition contains at least one close realizer. A well-formed context Γ can contain false propositions and of course false proposition $(x\tilde{=}y)$ is derivable from Γ indicating that x and y may have different values. That's why we need a constraint WFCE on the context showing that the term-valuation of each variable should be the closed to ensure that Γ does not contain false propositions.

4.3 Interpretation Rules and Soundness

Soundness of $\mathsf{M_T}$ can be proved by defining abstract interpretation rules.

▶ **Parameter 25** (Signature mapping)**.** There exists a function $\mathsf{I_T} : \mathsf{T} \to \mathsf{Term}$ which assigns a M-term for each first-order term.

▶ **Definition 26** (Formula mapping)**.** The function $\mathsf{I_F} : \mathsf{F} \to \mathsf{Term}$, which assigns a M-term to each formula, is defined as:

$$\mathsf{I_F}(x \sim_{\mathcal{T}} y) \triangleq \mathsf{I_T}(x)\tilde{=}\mathsf{I_T}(y) \qquad \mathsf{I_F}(\hat{\bot}) \triangleq \tilde{\bot} \qquad \mathsf{I_F}(\hat{\top}) \triangleq \tilde{\top}$$

$$\mathsf{I_F}(\hat{\neg}f) \triangleq \tilde{\neg}\mathsf{I_F}(f) \qquad \mathsf{I_F}(f\hat{\wedge}g) \triangleq \mathsf{I_F}(f)\tilde{\wedge}\mathsf{I_F}(g) \qquad \mathsf{I_F}(f\hat{\vee}g) \triangleq \mathsf{I_F}(f)\tilde{\vee}\mathsf{I_F}(g)$$

$$\mathsf{I_F}(f\hat{\to}g) \triangleq \mathsf{I_F}(f)\tilde{\to}\mathsf{I_F}(g) \qquad \mathsf{I_F}(\hat{\forall}f) \triangleq \tilde{\forall}\mathsf{I_F}(f) \qquad \mathsf{I_F}(\hat{\exists}f) \triangleq \tilde{\exists}\mathsf{I_F}(f)$$

The interpretation of the context is a combination of $\mathsf{I_T}$ and $\mathsf{I_F}$:

▶ **Definition 27.** The function $\mathsf{I}_\Gamma : \mathsf{C} \to [..\mathsf{Term}..]$ is defined as:

$$\mathsf{I}_\Gamma(e) \triangleq \begin{cases} [\,] & (e = [\,]) \\ f(x) :: \mathsf{I}_\Gamma(l) & (e = x :: l) \end{cases} \text{ and } f(x) = \begin{cases} \mathsf{I}_\Gamma(x) & ((x = f) : \mathsf{F})) \\ \mathcal{S} & (x = \mathsf{T}) \end{cases}$$

We now need to assume or prove that each one of these semantic mappings produces expressions of the expected type ($\mathcal{S}$ for terms and Prop for formulas). Omitting the trivial case of contexts, this yields:

▶ **Axiom 28.** $\mathsf{Wf_t}(\Gamma, t) \Rightarrow \mathsf{I}_\Gamma(\Gamma) \tilde{\vdash} \mathsf{I_T}(t) \tilde{\in} \mathcal{S}$

▶ **Lemma 29.** $\mathsf{Wf_f}(\Gamma, f) \Rightarrow \mathsf{I}_\Gamma(\Gamma) \tilde{\vdash} \mathsf{I_F}(f) \tilde{\in} \mathsf{Prop}$

All axioms are formulas provable in the model.

▶ **Axiom 30.** $\widehat{\mathsf{Ax}}(\Gamma, f) \Rightarrow \exists t, \mathsf{I}_\Gamma(\Gamma) \tilde{\vdash} t \tilde{\in} \mathsf{I_F}(f)$

Soundness of $\mathsf{M_T}$ is not a trivial result:

▶ **Theorem 31** (Soundness)**.** $\mathsf{M_T}$ *is sound, that is any derivable formula can be proved in* $\mathsf{M_T}$ *by induction on the syntactic derivation rules (Definition 19):*

$$\Gamma \hat{\vdash} P \Rightarrow \exists p, \mathsf{I}_\Gamma(\Gamma) \tilde{\vdash} p \tilde{\in} \mathsf{I_F}(P)$$

5 Soundness of CCT and CICUT

Having a sound model of the theory, the work remaining to ensure the meta-theory of CCT is to prove the soundness of the conversion rule extended with first-order equations.

The theorem is introduced step by step. When the original conversion checking fails to check $\Gamma \vdash A \simeq B$, a decision procedure $\mathsf{Preprocess}$ is called to check whether the theory can be used to check this equation. If the theory is applicable, $\mathsf{Preprocess}$ will refine the context and relocate the variables and the theory will check whether A' and B' are equal in the context Γ'. If any of the above steps fails, then it makes no difference to Coq. Hence, we can start the proof from the conditions that the equality is derivable in the theory:

$$\mathsf{Preprocess}(\Gamma, A, B) = (\Gamma', A', B')[1] \quad \text{and} \quad \Gamma' \hat{\vdash} A' \sim_{\mathcal{T}} B'[2]$$

The $\mathsf{Preprocess}$ function should maintain some invariants:

$$\mathsf{I}_\Gamma(\Gamma) \tilde{\vdash} \sigma \triangleright \mathsf{I}_\Gamma(\Gamma')\,[3]; \quad \mathsf{I_T}(A) = (\mathsf{I_T}(A'))[\sigma]\,[4]; \quad \mathsf{I_T}(B) = (\mathsf{I_T}(B'))[\sigma]\,[5]$$

By Lemma 20, any derivable formula should be a well-formed formula, particularly for equation in condition [2], its sub-term A' and B' should be well-formed in Γ'. Further, by Axiom 28, they should be interpreted in the scope $\mathcal{S}$:

$$\mathsf{I}_\Gamma(\Gamma') \tilde{\vdash} \mathsf{I_T}(A') \tilde{\in} \mathcal{S} \quad [6] \qquad \mathsf{I}_\Gamma(\Gamma') \tilde{\vdash} \mathsf{I_T}(B') \tilde{\in} \mathcal{S} \quad [7]$$

By the soundness theorem (Theorem 31) the interpretation of this equality should be inhabited in $\mathsf{M_T}$ (condition [8]). If Γ' does not contain false formula, then we have $\mathsf{WFCE}(\mathsf{I}_\Gamma(\Gamma'))$, hence the equality judgment can correctly interpret the equality in theory by Lemma 24 using conditions [6], [7] and [8]:

$$\exists t, \mathsf{I}_\Gamma(\Gamma') \tilde{\vdash} t \tilde{\in} (\mathsf{I_T}(A') \doteq \mathsf{I_T}(B'))\,[8] \qquad \mathsf{I}_\Gamma(\Gamma') \tilde{\vdash} \mathsf{I_T}(A') \doteq \mathsf{I_T}(B')\,[9]$$

Finally, by the substitution lemma applied to conditions [2], [3], [4] and [5], we derive:

$$\mathsf{I}_\Gamma(\Gamma) \tilde{\vdash} \mathsf{I}_\mathsf{T}(A) \doteq \mathsf{I}_\mathsf{T}(B)$$

Following the above analysis, the new conversion rule is justified by proving:

▶ **Theorem 32.** *Equal terms in theory have the same values in the model:*

$$\frac{\mathsf{WFCE}(\mathsf{I}_\Gamma(\Gamma')) \quad \Gamma \tilde{\vdash} \sigma \triangleright \mathsf{I}_\Gamma(\Gamma') \quad \Gamma' \hat{\vdash} A' \sim_\mathcal{T} B' \quad x = \mathsf{I}_\mathsf{T}(A')[\sigma] \quad y = \mathsf{I}_\mathsf{T}(B')[\sigma]}{\Gamma \tilde{\vdash} x \doteq y}$$

Since we require $\mathsf{WFCE}(\mathsf{I}_\Gamma(\Gamma'))$, not all first-order objects (terms and formulas) can be extracted from Γ. That means we may not extract satisfiable equations to Γ' and use these equations to check another equation as SMT solvers do. Nevertheless, it is still an significant improvement of conversion checking.

The sound model of CCT can be built now by adding a sound model for the abstract theory and proving the extended conversion rule. Further, since we no longer incorporate the equality of the theory into ι-reduction, we can absorb the models of inductive types and universes built by Barras to yield a sound model for CICUT. All the meta-theoretical properties that we have established for CC can be established for CCT and CICUT as well.

6 Example: Presburger Arithmetic

In this section, we take Presburger Arithmetic as an example to illustrate how to instantiate the above abstract setting for a specific theory.

Firstly, we define the signature and axioms of Presburger, and prove it correctly instantiates the abstract syntax of theory in Subsection 4.1.

6.1 Formalization of Presburger Arithmetic

The definition of the signature and axioms instantiate Parameter 14 and 18.

▶ **Definition 33** (Signature)**.** Presburger signature is defined inductively:

$$\mathsf{T} \triangleq \bar{n} | \mathsf{C0} | \mathsf{C1} | (x : \mathsf{T}) \bar{+} (y : \mathsf{T})$$

where $\bar{n}$ indicates the free variables, $\mathsf{C0}$ and $\mathsf{C1}$ are two constants representing zero and one respectively, $\bar{+}$ is the addition relation.

With this definition, variable relocation, substitution and free variable operations can be defined by recursion on the structure of T.

▶ **Definition 34** (Axioms)**.** An axiom is a special formula presented by a predicate in Coq taking a context and formula as arguments and checking whether this formula is an axiom:

$$\widehat{\mathsf{Ax}}(H, f) \triangleq \begin{cases} f = \hat{\forall} x. \hat{\neg}(\mathsf{C0} \sim_\mathcal{T} (x \bar{+} \mathsf{C1})) & \vee \\ f = \hat{\forall} xy. ((x \bar{+} \mathsf{C1} \sim_\mathcal{T} y \bar{+} \mathsf{C1}) \hat{\rightarrow} (x \sim_\mathcal{T} y)) & \vee \\ f = \hat{\forall} x. x \sim_\mathcal{T} (x \bar{+} \mathsf{C0}) & \vee \\ f = \hat{\forall} xy. ((x \bar{+} y) \bar{+} \mathsf{C1}) \sim_\mathcal{T} (x \bar{+} (y \bar{+} \mathsf{C1})) & \vee \\ \exists g, \mathsf{Wf}_\mathsf{f}(\mathsf{T} :: H, g) \wedge (f = g[\mathsf{C0}] \hat{\rightarrow} (\hat{\forall} n. g[n] \hat{\rightarrow} g[n \bar{+} \mathsf{C1}]) \hat{\rightarrow} (\hat{\forall} n. g[n])) \end{cases}$$

The assumption in Parameter 18 can be proved, because g is well-typed in $(\mathsf{T} :: H)$.

Secondly, we give a brief introduction to the formalization of natural number in [2]: useful definition (maybe abstract) and properties without proof. Then the M_T will be instantiated by the these definitions and properties. The abstract interpretation rules are instantiated accordingly which will not be detailed here.

6.2 Formalization of Natural Numbers

As mentioned in Section 3, the first step is to define the set-values of natural numbers (NAT), its constructors (ZERO and SUC) and eliminator (NREC) in set-theory. Proving that this setting interprets all of the axioms of Presburger arithmetic is straightforward.

The second step is to define the realizers (saturated set) of the type and its constructors. More details can be found in [2].

▶ **Definition 35** (Realizer of constructors)**.** The realizers of natural number are:

$$\mathsf{ZE} \triangleq \check{\lambda}x.\check{\lambda}f.x \qquad \mathsf{SU} \triangleq \check{\lambda}n.\check{\lambda}x.\check{\lambda}f.f\check{@}n\check{@}(n\check{@}x\check{@}f)$$

We then define the saturated set $\mathsf{cNAT}(k)$, the realizers of number k. The idea is to follow the impredicative definition of natural numbers, but we need the dependent version. So we take the least fixpoint of a function fNAT defined as follows:

▶ **Definition 36** (Saturated set associated to natural numbers)**.**

$$\mathsf{fNAT}(A,k) \triangleq \bigcap_{P:\mathsf{X}\to\mathsf{SAT}} P(\mathsf{ZERO}) \to \left[\bigcap_{n\in\mathsf{NAT}} A(n)\to P(n)\to P(\mathsf{SUCC}(n))\right] \to P(k)$$

$$\mathsf{cNAT}(k) \triangleq \bigcap_{A \text{ s.t. } \mathsf{fNAT}(A)\subseteq A} A$$

and these definitions satisfy following properties :

$$\frac{}{\mathsf{ZE}\check{\in}(\mathsf{cNAT}(\mathsf{ZERO}))} \qquad \frac{n\dot{\in}\mathsf{NAT} \quad t\check{\in}\mathsf{cNAT}(n)}{\mathsf{SU}\check{@}t\check{\in}\mathsf{cNAT}(\mathsf{SUCC}(n)))}$$

Then, natural numbers can be defined in M:

▶ **Definition 37** (Natural Number)**.** The type of natural number and its constructors are defined in M as follows:

$$\begin{aligned}
\mathsf{Nat} &\triangleq \langle i\mapsto(\mathsf{NAT},\mathsf{cNAT}), j\mapsto\mathrm{K}\rangle\\
\mathsf{Zero} &\triangleq \langle i\mapsto\mathsf{ZERO}, j\mapsto\mathsf{ZE}\rangle\\
\mathsf{Succ} &\triangleq \langle i\mapsto\mathsf{SUCC}, j\mapsto\mathsf{SU}\rangle\\
\mathsf{NatRec}(f,g,n) &\triangleq \left\langle \begin{array}{l} i\mapsto\mathsf{NREC}(\mathsf{Val}(f)_i,(n,y\mapsto\mathsf{Val}(g)_i\dot{@}n\dot{@}y),\mathsf{Val}(n)_i),\\ j\mapsto\mathsf{Tm}(n)_j\check{@}\mathsf{Tm}(f)_j\check{@}\mathsf{Tm}(g)_j \end{array} \right\rangle
\end{aligned}$$

▶ **Lemma 38.** *Typing rules of the natural numbers can be proved:*

$$\Gamma\tilde{\vdash}\mathsf{Zero}\tilde{\in}\mathsf{Nat} \quad \Gamma\tilde{\vdash}\mathsf{Succ}\tilde{\in}(\tilde{\Pi}n\tilde{\in}\mathsf{Nat}.\mathsf{Nat}) \quad \Gamma\tilde{\vdash}\mathsf{Nat}\tilde{\in}\mathsf{Kind}$$

$$\frac{\Gamma\tilde{\vdash}n\tilde{\in}\mathsf{Nat} \quad \Gamma\tilde{\vdash}f\tilde{\in}(P\tilde{@}\mathsf{Zero}) \quad \Gamma\tilde{\vdash}g\tilde{\in}(\tilde{\Pi}n\tilde{\in}\mathsf{Nat}.\tilde{\Pi}(P\tilde{@}n).P\tilde{@}(\mathsf{Succ}\tilde{@}n))}{\Gamma\tilde{\vdash}\mathsf{NatRec}(f,g,n)\tilde{\in}(P\tilde{@}n)}$$

6.3 Instantiation the Model of Abstract Theory by Presburger

The $\mathcal{S}$ in M_T is instantiated by Nat. Furthermore, we need to provide enough constructors to interpret Presburger Signature (Definition 33).

▶ **Definition 39** (Presburger Semantic)**.** The semantic of Presburger is:

$$\mathcal{S} \triangleq \mathsf{Nat} \quad \mathsf{Zero} \quad \mathsf{Succ}(\mathsf{Zero}) \quad \tilde{+} \triangleq \tilde{\lambda}(x\tilde{\in}\mathsf{Nat}).\tilde{\lambda}(y\tilde{\in}\mathsf{Nat}).\mathsf{NatRec}(x,\tilde{\lambda}(z\tilde{\in}\mathsf{Nat}).\mathsf{Succ},y)$$

By the induction scheme provided in the values of natural numbers, we can prove the last two assumptions in Axiom 21, while others are trivial according to the definition.

Instantiating Presburger axioms in M is just a translation work according to the interpretation rules, but proving them requires more efforts: the main task is to construct the proof term for each axiom. We have all the detailed proofs in our development to ensure:

▶ **Lemma 40.** M_T *instantiated by Presburger arithmetic is sound.*

Then, all the meta-theoretical properties holding in M_T preserve after instantiation by Presburger arithmetic, therefore we can conclude:

▶ **Lemma 41.** *Presburger arithmetic is a safe theory to be embedded, hence* CoqMT *is safe.*

Let us finish with an important remark. We have proven the strong normalization for CICUT instantiated with Presburger arithmetic, but the reduction considered includes β- and ι-reduction (the reduction of NatRec when its main argument is either 0 or successor), but not the reduction associated to the last two rules of Section 2. The main difficulty is that the latter is not sequential (both $\mathsf{S}(n)+m$ and $n+\mathsf{S}(m)$ reduce to a successor), while the β-reduction of λ-calculus is sequential. We thus cannot claim yet the decidability of type-checking for the presented version of CICUT.

7 Conclusion

In this paper, we give an abstract proof of consistency and SN to CCT, which extends CC family by incorporating extensional equalities from an abstract theory. Presburger arithmetic is proved to correctly instantiate the abstract theory which, on the one hand, ensures Presburger arithmetic is safe to be embedded, on the other hand demonstrates that our abstraction strategy works well for adding a new theory of interest.

Actually, this proof is applicable to a richer type system contains more features if each feature has a sound model and it does not have interference with other features, such as CICUT. That's our original motivation to do this research: improving the Coq by incorporating decidable extensional equalities.

We don't embed theory in the ι-reduction to ensure the consistency and SN, the side effect is that Church-Rosser can not be proved in the usual way, as well as the decidability of type checking (DoTC). However we have another way to prove DoTC, that's our next work : formalize the syntax of CCT or CICUT, and build a complete formal proof of all the properties required.

Acknowledgements. The authors would like to thank Jean-Pierre Jouannaud and Pierre-Yves Strub for many useful discussions. The authors would also like to thank Jean-Pierre Jouannaud and anonymous reviewers for their useful comments and language editing which have greatly improved the mansucript.

References

1 Thorsten Altenkirch. Proving strong normalization of cc by modifying realizability semantics. In Henk Barendregt and Tobias Nipkow, editors, *TYPES*, volume 806 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 1993.

2 Bruno Barras. Semantical investigations in intuitionistic set theory and type theories with inductive families. In preparation habilitation thesis, http://www.lix.polytechnique.fr/ barras/habilitation/.

3 Bruno Barras. Sets in coq, coq in sets. *Journal of Formalized Reasoning*, 3(1):29–48, 2010.

4 Bruno Barras, Jean-Pierre Jouannaud, Pierre-Yves Strub, and Qian Wang. Coqmtu: A higher-order type theory with a predicative hierarchy of universes parametrized by a decidable first-order theory. In *LICS*, pages 143–151. IEEE Computer Society, 2011.

5 Frédéric Blanqui. Inductive types in the calculus of algebraic constructions. In Martin Hofmann, editor, *TLCA*, volume 2701 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2003.

6 Frédéric Blanqui, Jean-Pierre Jouannaud, and Pierre-Yves Strub. From formal proofs to mathematical proofs: a safe, incremental way for building in first-order decision procedures. *CoRR*, abs/0804.3762, 2008.

7 Frédéric Blanqui, Jean-Pierre Jouannaud, and Pierre-Yves Strub. From formal proofs to mathematical proofs: A safe, incremental way for building in first-order decision procedures. In Giorgio Ausiello, Juhani Karhumäki, Giancarlo Mauri, and C.-H. Luke Ong, editors, *IFIP TCS*, volume 273 of *IFIP*, pages 349–365. Springer, 2008.

8 Thierry Coquand and Gérard P. Huet. The calculus of constructions. *Inf. Comput.*, 76(2/3):95–120, 1988.

9 J. L. Krivine. *Lambda-calculus, types and models*. Ellis Horwood, Upper Saddle River, NJ, USA, 1993.

10 Zhaohui Luo. Ecc, an extended calculus of constructions. In *LICS*, pages 386–395. IEEE Computer Society, 1989.

11 Nicolas Oury. Extensionality in the calculus of constructions. In Joe Hurd and Thomas F. Melham, editors, *TPHOLs*, volume 3603 of *Lecture Notes in Computer Science*, pages 278–293. Springer, 2005.

12 Christine Paulin-Mohring. Inductive definitions in the system coq - rules and properties. In Marc Bezem and Jan Friso Groote, editors, *TLCA*, volume 664 of *Lecture Notes in Computer Science*, pages 328–345. Springer, 1993.

13 Vincent Siles and Hugo Herbelin. Equality is typable in semi-full pure type systems. In *LICS*, pages 21–30. IEEE Computer Society, 2010.

14 Mark-Oliver Stehr. The open calculus of constructions (part i): An equational type theory with dependent types for programming, specification, and interactive theorem proving. *Fundam. Inform.*, 68(1-2):131–174, 2005.

15 Mark-Oliver Stehr. The open calculus of constructions (part ii): An equational type theory with dependent types for programming, specification, and interactive theorem proving. *Fundam. Inform.*, 68(3):249–288, 2005.

16 Pierre-Yves Strub. *Théories des Types et Procédures de Décisions*. These, Ecole Polytechnique X, July 2008.

17 Pierre-Yves Strub. Coq modulo theory. In Anuj Dawar and Helmut Veith, editors, *CSL*, volume 6247 of *Lecture Notes in Computer Science*, pages 529–543. Springer, 2010.

18 William Tait. A realizability interpretation of the theory of species. In Rohit Parikh, editor, *Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, chapter 7, pages 240–251–251. Springer Berlin / Heidelberg, Berlin, Heidelberg, 1975.

19 The Coq Development Team. The Coq Proof Assistant, Reference Manual, Version 8.4. Technical report, INRIA, Roquencourt, France, 2012.