24th International Symposium on Temporal Representation and Reasoning

TIME 2017, October 16–18, 2017, Mons, Belgium

Edited by

Sven Schewe
Thomas Schneider
Jef Wijsen

LIPICS

Editors

Sven Schewe	Thomas Schneider	Jef Wijsen
University of Liverpool	University of Bremen	University of Mons
UK	Germany	Belgium
`sven.schewe@liverpool.ac.uk`	`tschneider@cs.uni-bremen.de`	`jef.wijsen@umons.ac.be`

ACM Classification 1998
C.2.4 Distributed Systems, D.2.2 Design Tools and Techniques, D.2.4 Software/Program Verification, D.4.6 Security and Protection, F.1.1 Models of Computation, F.3.1 Specifying and Verifying and Reasoning about Programs, F.4.1 Mathematical Logic, G.1.3 [Numerical Linear Algebra] Linear Systems, G.2.2 Graph Theory, H.2 Database Management, I.2.2 Automatic Programming, I.2.4 Knowledge Representation Formalisms and Methods, I.2.7 Natural Language Processing, I.2.8 Problem Solving, Control Methods, and Search, I.6.4 Model Validation and Analysis, J.2 [Physical Sciences and Engineering] Archaeology

ISBN 978-3-95977-052-1

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/978-3-95977-052-1.

Publication date
October, 2017

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.TIME.2017.0

LIPIcs – Leibniz International Proceedings in Informatics

ISSN 1868-8969

http://www.dagstuhl.de/lipics

Contents

Preface

The International Symposium on Temporal Representation and Reasoning (TIME) is a well-established symposium series which brings together researchers interested in reasoning about temporal aspects of information in all areas of computer science. The symposium has a wide remit and is devoted to both theoretical aspects and well-founded applications. One of the key aspects of the symposium is its interdisciplinarity, with attendees from different areas such as artificial intelligence, database management, logic and verification, and beyond. The 24th edition of the symposium (TIME 2017) was held from 16 to 18 October 2017 in the city of Mons, Belgium, hosted by the University of Mons.

Following the call for papers of TIME 2017, a total of 48 abstracts were submitted. Some abstract submissions did not lead to a subsequent full paper submission. Eventually, a total of 36 full papers were submitted. Each submitted paper was reviewed by at least three members of the program committee and the reviews were followed by an additional discussion to select among those papers. The members of the program committee and the external reviewers did an excellent job that enabled a high-quality selection process, and we thank them for their commitment and dedication. In the end, 20 papers were selected for publication in the proceedings and presentation at the symposium. In addition to the contributed talks, this year's program featured three invited speakers: Alessandro Artale (Free University of Bozen-Bolzano, Italy), Javier Esparza (Technical University of Munich, Germany), and Sheila McIlraith (University of Toronto, Canada). We are delighted that they were able to accept our invitation, and grateful for their contribution.

These are the first TIME proceedings published in the Dagstuhl/LIPIcs series. We would like to thank Dr. Marc Herbstritt and the LIPIcs team for all the help and support. Finally, we would like to thank the following organizations for sponsoring the event: F.R.S.-FNRS, Université de Mons, COMPLEXYS, and INFORTECH.

Sven Schewe
Thomas Schneider
Jef Wijsen

Organization

Program Committee Chairs

- Sven Schewe (University of Liverpool, UK)
- Thomas Schneider (University of Bremen, Germany)
- Jef Wijsen (University of Mons, Belgium)

Program Committee

- Johann Eder (Alpen Adria Universität Klagenfurt, Austria)
- Fabio Grandi (University of Bologna, Italy)
- Ernst Moritz Hahn (State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China)
- Michael R. Hansen (Technical University of Denmark, Denmark)
- Fredrik Heintz (Linköping University, Sweden)
- Marcin Jurdziński (University of Warwick, UK)
- Roman Kontchakov (Birkbeck, University of London, UK)
- Jan Křetínský (Technical University of Munich, Germany)
- Bart Kuijpers (Hasselt University, Belgium)
- Martin Lange (University of Kassel, Germany)
- Martin Leucker (University of Lübeck, Germany)
- Cláudia Nalon (University of Brasília, Brazil)
- Andrea Orlandini (National Research Council of Italy ISTC-CNR, Italy)
- Doron Peled (Bar Ilan University, Israel)
- Roberto Posenato (Università degli Studi di Verona, Italy)
- Jean-François Raskin (Université Libre de Bruxelles, Belgium)
- Peter Z. Revesz (University of Nebraska-Lincoln, USA)
- Mark Reynolds (The Univeristy of Western Australia, Australia)
- Sven Schewe (University of Liverpool, UK)
- Renate A. Schmidt (University of Manchester, UK)
- Thomas Schneider (University of Bremen, Germany)
- Paolo Terenziani (University Piemonte Orientale, Italy)
- David Toman (University of Waterloo, Canada)
- Kristian Torp (Aalborg University, Denmark)
- Alejandro A. Vaisman (Instituto Tecnológico de Buenos Aires, Argentina)
- Jef Wijsen (University of Mons, Belgium)
- Stefan Wölfl (University of Freiburg, Germany)
- Martin Zimmermann (Saarland University, Germany)

Local Organizers

- Alexandre Decan (University of Mons, Belgium)
- Fanny Lallemand (University of Mons, Belgium)
- Jef Wijsen (University of Mons, Belgium)

External Reviewers

List of Authors

Antoine Amarilli
LTCI, Télécom ParisTech, Université Paris-Saclay
France
`antoine.amarilli@telecom-paristech.fr`

Alessandro Artale
Free University of Bozen-Bolzano
Italy
`artale@inf.unibz.it`

Mouhamadou Lamine Ba
University Alioune Diop of Bambey
Senegal
`mouhamadoulamine.ba@uadb.edu.sn`

Jonathan Behaegel
University of Nice Sophia Antipolis
France
`behaegel@i3s.unice.fr`

Laura Bozzelli
University “Federico II” of Naples
Italy
`lr.bozzelli@gmail.com`

Thomas Brihaye
Université de Mons
Belgium
`thomas.brihaye@umons.ac.be`

Massimo Cairo
University of Trento
Italy
`massimo.cairo@unitn.it`

Alexandre Caminada
Université de Technologie de Belfort-Montbéliard
France
`alexandre.caminada@utbm.fr`

Carlo Combi
University of Verona
Italy
`carlo.combi@univr.it`

Jean-Paul Comet
University of Nice Sophia Antipolis
France
`comet@unice.fr`

Carlo Comin
University of Verona
Italy
`carlo.comin@univr.it`

Jean-François Condotta
Artois University
France
`condotta@cril.fr`

David de Frutos-Escrig
Universidad Complutense de Madrid
Spain
`defrutos@sip.ucm.es`

Dario Della Monica
Universidad Complutense de Madrid
Spain
and
University “Federico II” of Naples
Italy
`dario.dellamonica@unina.it`

Daniel Deutch
Tel Aviv university
Israel
`danielde@post.tau.ac.il`

Javier Esparza
Technische Universität München
Germany
`esparza@in.tum.de`

Mathias Etcheverry
Universidad de la República
Uruguay
`mathiase@fing.edu.uy`

Maxime Folschette
University of Nantes / LS2N
France
`comet@unice.fr`

Gilles Geeraerts
Université libre de Bruxelles
Belgium
gigeerae@ulb.ac.be

Valentin Goranko
University of Stockholm
Sweden
valentin.goranko@philosophy.su.se

Fabio Grandi
University of Bologna
Italy
fabio.grandi@unibo.it

Michael Gruninger
University of Toronto
Canada
gruninger@mie.utoronto.ca

Hsi-Ming Ho
Université de Mons
Belgium
hsimho@gmail.com

Luke Hunsberger
Vassar College
USA
hunsberger@vassar.edu

Daniel Kernberger
University of Kassel
Germany
daniel.kernberger@uni-kassel.de

Marie Kiermeier
Ludwig-Maximilians-Universität München
Germany
marie.kiermeier@ifi.lmu.de

Roman Kontchakov
Birkbeck, University of London
UK
roman@dcs.bbk.ac.uk

Alisa Kovtunova
Free University of Bozen-Bolzano
Italy
alisa.kovtunova@inf.unibz.it

Antti Kuusisto
University of Bremen
Germany
kuusisto@uni-bremen.de

Martin Lange
University of Kassel
Germany
martin.lange@uni-kassel.de

Eythan Levy
Tel-Aviv University
Israel
eythan.levy@gmail.com

Zhuojun Li
University of Toronto
Canada
zhuojun.li@mail.utoronto.ca

Federica Mandreoli
University of Modena e Reggio Emilia
Italy
federica.mandreoli@unimo.it

Riccardo Martoglia
University of Modena e Reggio Emilia
Italy
riccardo.martoglia@unimo.it

Sheila A. McIlraith
University of Toronto
Canada
sheila@cs.toronto.edu

Laurent Moalic
Université de Technologie de
Belfort-Montbéliard
France
laurent.moalic@utbm.fr

Benjamin Monmege
Aix-Marseille Université, CNRS, LIF
France
benjamin.monmege@lif.univ-mrs.fr

Angelo Montanari
University of Udine
Italy
angelo.montanari@uniud.it

Aniello Murano
University “Federico II” of Naples
Italy
murano@na.infn.it

Flemming Nielson
Technical University of Denmark
Denmark
fnie@dtu.dk

Hanne Riis Nielson
Technical University of Denmark
Denmark
hrni@dtu.dk

Anastasia Paparrizou
Artois University
France
paparrizou@cril.fr

Wilma Penzo
University of Bologna
Italy
wilma.penzo@unibo.it

Giuseppe Perelli
University of Oxford
UK
giuseppe.perelli@cs.ox.ac.uk

Frédéric Pluquet
École Supérieure d'Informatique (HE2B-ESI), Brussels
Belgium
fpluquet@he2b.be

Roberto Posenato
University of Verona
Italy
roberto.posenato@univr.it

Romeo Rizzi
University of Verona
Italy
romeo.rizzi@univr.it

Raine Rönnholm
University of Tampere
Finland
raine.ronnholm@uta.fi

Vladislav Ryzhikov
Free University of Bozen-Bolzano
Italy
ryzhikov@inf.unibz.it

Sven Schewe
University of Liverpool
UK
sven.schewe@liverpool.ac.uk

Thomas Schneider
University of Bremen
Germany
tschneider@cs.uni-bremen.de

Guido Sciavicco
University of Ferrara
Italy
guido.sciavicco@unife.it

Pierre Senellart
DI ENS, ENS, CNRS, PSL Research University; Inria Paris
France
pierre@senellart.com

Michael Sioutis
Örebro University
Sweden
michael.sioutis@oru.se

Loredana Sorrentino
University "Federico II" of Naples
Italy
loredana.sorrentino@unina.it

Etienne Thuillier
Université de Technologie de Belfort-Montbéliard
France
etienne.thuillier@utbm.fr

Panagiotis Vasilikos
Technical University of Denmark
Denmark
panva@dtu.dk

Przemyslaw Andrzej Walega
University of Warsaw
Poland
p.a.walega@gmail.com

Martin Werner
Leibniz Universität Hannover
Germany
martin.werner@ikg.uni-hannover.de

Jef Wijsen
University of Mons
Belgium
jef.wijsen@umons.ac.be

Frank Wolter
University of Liverpool
UK
wolter@liverpool.ac.uk

Dina Wonsever
Universidad de la República
Uruguay
wonsever@fing.edu.uy

Michael Zakharyaschev
Birkbeck, University of London
UK
michael@dcs.bbk.ac.uk

Matteo Zavatteri
University of Verona
Italy
matteo.zavatteri@univr.it

Ontology-Mediated Query Answering over Temporal Data: A Survey

Alessandro Artale[1], **Roman Kontchakov**[2], **Alisa Kovtunova**[3], **Vladislav Ryzhikov**[4], **Frank Wolter**[5], **and Michael Zakharyaschev**[6]

1 **KRDB Research Centre, Free University of Bozen-Bolzano, Bolzano, Italy**
artale@inf.unibz.it

2 **Department of Computer Science and Information Systems, Birkbeck, University of London, London, UK**
roman@dcs.bbk.ac.uk

3 **KRDB Research Centre, Free University of Bozen-Bolzano, Bolzano, Italy**
alisa.kovtunova@inf.unibz.it

4 **KRDB Research Centre, Free University of Bozen-Bolzano, Bolzano, Italy**
ryzhikov@inf.unibz.it

5 **Department of Computer Science, University of Liverpool, Liverpool, UK**
wolter@liverpool.ac.uk

6 **Department of Computer Science and Information Systems, Birkbeck, University of London, London, UK**
michael@dcs.bbk.ac.uk

Abstract

We discuss the use of various temporal knowledge representation formalisms for ontology-mediated query answering over temporal data. In particular, we analyse ontology and query languages based on the linear temporal logic *LTL*, the multi-dimensional Halpern-Shoham interval temporal logic $\mathcal{HS}_n$, as well as the metric temporal logic *MTL*. Our main focus is on the data complexity of answering temporal ontology-mediated queries and their rewritability into standard first-order and datalog queries.

1998 ACM Subject Classification I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases Description Logic, Temporal Logic, Ontology Mediated Query Answering, Data Complexity

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.1

Category Invited Talk

1 Introduction

This paper is a survey of recent developments in applying temporal logics for ontology-mediated query answering over temporal data.

Ontology-based data access (OBDA) [73] has recently become one of the most successful applications of description logics (DLs). The chief aim of OBDA is to facilitate access to possibly heterogeneous, distributed and incomplete data for non-IT-expert users. To illustrate, suppose that such a user wants to query some data sources $\mathcal{D}$. Under the OBDA paradigm, the user does not have to know the schemas of $\mathcal{D}$ (that is, how the data is organised). Instead, the user is given an ontology $\mathcal{O}$ describing the domain of their interest in familiar and standard terms that can be used directly to formulate the desired queries $\boldsymbol{q}(\boldsymbol{x})$ in, say, the query language SPARQL, possibly with the help of a graphical tool. The OBDA system relies on a (GAV) mapping $\mathcal{M}$, relating the terms in $\mathcal{O}$ with the schemas of $\mathcal{D}$ (and produced by an IT expert), to find tuples $\boldsymbol{a}$ from $\mathcal{D}$ such that $\mathcal{O}, \mathcal{M}(\mathcal{D}) \models \boldsymbol{q}(\boldsymbol{a})$, where $\mathcal{M}(\mathcal{D})$ is the result of applying $\mathcal{M}$ to $\mathcal{D}$. Depending on the language of the *ontology-mediated query* (OMQ) $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$, this can sometimes be done by rewriting $\boldsymbol{Q}$ to a first-order (FO) or datalog query $\boldsymbol{q}'(\boldsymbol{x})$ that can be executed over any given data instance $\mathcal{D}$ directly by conventional data management systems. For example, FO-rewritings always exist if $\mathcal{O}$ is an *OWL 2 QL* ontology (based on the *DL-Lite* family of DLs) and $\boldsymbol{q}(\boldsymbol{x})$ is a conjunctive query (CQ) [37, 6], while datalog rewritings can be constructed for OMQs with ontologies in *OWL 2 EL* (based on the $\mathcal{EL}$ family of DLs) and CQs [78]. For recent applications of OBDA, the reader can consult [5, 22, 47, 36, 38, 77, 82].

The W3C standard ontology languages *OWL 2 QL* and *OWL 2 EL* mentioned above were designed to represent knowledge about static domains and are not suitable when the data and the terms the user is interested in are essentially temporal. Suppose, for example, that the data comprises sensor readings from some industrial installations, say, gas turbines, or from weather stations across a country, and that the user – a service engineer or, respectively, a meteorologist – is interested in detecting events such as

- *active power trip*, which happens when the active power of a turbine was above 1.5MW for a period of at least 10 seconds, maximum 3 seconds after which there was a period of at least one minute where the active power was below 0.15MW; or
- *blizzard*, which happens when severe snowstorms with low temperatures and strong winds last for at least three hours.

To be able to represent these concepts, an ontology language clearly requires various temporal constructs that have been studied in the context of temporal representation and reasoning [44, 45, 41].

Combinations of DLs with temporal formalisms have been widely investigated since the pioneering work of Schmiedel [81] and Schild [80] in the early 1990s; we refer the reader to [45, 21, 7, 62] for surveys and [71, 10, 51, 52, 50, 14] for more recent developments. However, the main reasoning task targeted in this line of research was concept satisfiability rather than query answering and the general aim was to probe various combinations of temporal and DL constructs that ensure decidability of concept satisfiability with acceptable combined complexity.

In the context of answering OMQs, our main concern is their FO- or datalog-rewritability, and the data complexity of query evaluation, where the given OMQ is regarded to be fixed while the data varies. Thus, in this survey we focus on temporal data modelling and algorithmic properties of OMQ answering and do not discuss in any detail advances in temporal DLs not related to query answering. The plan for this paper is as follows. We distinguish three temporal data models and the corresponding languages for ontologies and queries: the discrete point-based approach where time is discrete and each fact comes with a time-point in which it holds true, the more general interval-based approach where facts are stamped with the interval in which they are true, and finally, a model based on a dense flow of time where the focus is on modelling and querying metric temporal properties. In Sections 2–4, we discuss the state of the art in point-based ontology-mediated query answering. The languages considered range from the full two-sorted FOL to time-centric languages based of *LTL*, domain-centric languages based on DLs, and combinations of both. In Section 5, we consider ontology-mediated query answering over interval-based models focussing on Halpern and Shoham's modal logic for time intervals. In Section 6, we discuss dense time and how a combination of datalog and metric operators can be used to model metric knowledge and support ontology-mediated querying in this case. We close in Section 7 with a discussion of practical issues in temporal ontology-mediated querying and a recent implementation.

2 Point-Based Temporal Ontology-Mediated Querying

Suppose we have a database on submission, acceptance and publication of papers in the area of computer science collected from various sources on the web and elsewhere. For instance, the database may contain the facts

$$\begin{array}{ll} \mathsf{underSubmissionTo}(a, \mathrm{JACM}, \mathrm{Feb2016}), & \mathsf{UnderSubmission}(b, \mathrm{Jan2016}), \\ \mathsf{acceptedIn}(c, \mathrm{JACM}, \mathrm{July2016}), & \mathsf{Published}(c, \mathrm{Oct2016}), \\ \mathsf{authorOf}(\mathrm{Bob}, c, \mathrm{May2014}) & \end{array}$$

stating that paper a was under submission to JACM in February 2016; paper b was under submission in January 2016 (to an unknown journal); paper c, authored by Bob in May 2014, was recorded as accepted by JACM in July 2016, and published (in some venue) in October 2016. Observe that the predicates in the snippet above have a timestamp as their last argument (e.g., Oct2016) and either one or two domain arguments (e.g., a, JACM). Following the description logic (DL) tradition, we call predicates with one domain argument *concepts* (e.g., $\mathsf{Published}$) and predicates with two domain arguments *roles* (e.g., $\mathsf{authorOf}$). A finite set of timestamped facts such as the snippet above is called a temporal ABox. In general, a *temporal ABox*, denoted $\mathcal{A}$, consists of assertions of the form

$$A_k(a_i, n), \qquad P_k(a_i, a_j, n),$$

where A_k is a concept name, P_k a role name, a_i and a_j individual names, and $n \in \mathbb{Z}$ a timestamp.

Now, we introduce models for temporal ABoxes. Let $T \subseteq \mathbb{Z}$ be a (possibly infinite) interval. A T-*interpretation* $\mathcal{I}$ is a structure

$$\mathcal{I} = \big((T, <),\ \Delta^{\mathcal{I}},\ P_1^{\mathcal{I}}, P_2^{\mathcal{I}}, \dots, A_1^{\mathcal{I}}, A_2^{\mathcal{I}} \dots, a_1^{\mathcal{I}}, a_2^{\mathcal{I}}, \dots\big)$$

such that $<$ is the standard linear order on $\mathbb{Z}$ restricted to T, $\Delta^{\mathcal{I}} \neq \emptyset$ is the interpretation domain, $P_k^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \times T$ and $A_k^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times T$, for each k, and $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, for each i (we assume rigid, or time-independent, interpretation of individual names) and $a_i^{\mathcal{I}} \neq a_j^{\mathcal{I}}$ whenever $i \neq j$ (thus, we make the *unique name assumption*). Note that the domain $\Delta^{\mathcal{I}}$ is time-independent. Time-dependent domains can be modelled using an ‘existence predicate’; we refer the reader to [45, 44, 31] and references therein for a discussion of relevant domain assumptions in the literature on modal and temporal logic. For a temporal ABox $\mathcal{A}$, we say that a T-interpretation $\mathcal{I}$ *satisfies* $\mathcal{A}$ or that $\mathcal{I}$ is a *model* of $\mathcal{A}$ if T contains all timestamps n that occur in $\mathcal{A}$ and

$$(a_i^{\mathcal{I}}, n) \in A_k^{\mathcal{I}}, \quad \text{for all } A_k(a_i, n) \in \mathcal{A}, \qquad \text{and} \qquad (a_i^{\mathcal{I}}, a_j^{\mathcal{I}}, n) \in P_k^{\mathcal{I}}, \quad \text{for all } P_k(a_i, a_j, n) \in \mathcal{A}.$$

Let $\min \mathcal{A}$ and $\max \mathcal{A}$ be the minimal and, respectively, maximal integers occurring in $\mathcal{A}$. We assume without loss of generality that $\min \mathcal{A} = 0$. In what follows, we shall mostly be working with $\mathbb{Z}$-interpretations satisfying $\mathcal{A}$ (called $\mathbb{Z}$-*models of* $\mathcal{A}$), $\mathbb{N}$-interpretations satisfying $\mathcal{A}$ (called $\mathbb{N}$-*models of* $\mathcal{A}$) and $[\min \mathcal{A}, \max \mathcal{A}]$-interpretations satisfying $\mathcal{A}$ (called *ABox-fitting models of* $\mathcal{A}$).

The models $\mathcal{I}$ of a temporal ABox $\mathcal{A}$ reflect the *open-world assumption* underpinning ontology-mediated query answering: rather than assuming that the ABox contains all relevant

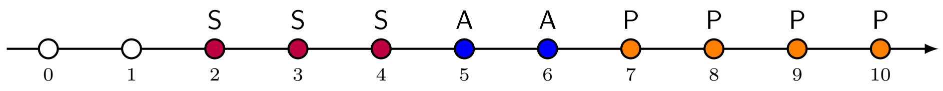


Figure 1 A typical timeline for a publication in Example 1: S, A and P stand for Submitted, Accepted and Published, respectively.

domain individuals and time points, one admits additional domain individuals and time points that might be required to satisfy domain knowledge. Thus, $\mathbb{Z}$-models reflect the common sense view of time as being infinite in the past and the future. $\mathbb{N}$-models and ABox-fitting models reflect a more pragmatic approach and assume that the time points not used as timestamps (or are before/after any timestamped data) are irrelevant for querying the data.

We next introduce the ontology and query languages that have been proposed for ontology-mediated querying of point-based temporal data. Most of these languages can be regarded as fragments of the *two-sorted first-order language* 2-FOL($<$) [84] constructed from atoms $A_k(x,t)$, $P_k(x,y,t)$, $t_1 < t_2$, and $t_1 = t_2$, where A_k is a concept name, P_k a role name, x and y are *domain variables* ranging over the interpretation domain $\Delta^{\mathcal{I}}$, and t, t_1 and t_2 are *temporal variables* ranging over the time instants in T. For any 2-FOL($<$)-formula φ, any T-interpretation $\mathcal{I}$, and any assignments $\mathfrak{d}$ of elements of $\Delta^{\mathcal{I}}$ to the domain variables and $\mathfrak{t}$ of elements of T to the temporal variables, we define the *truth-relation* $\mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} \varphi$ by induction as follows:

$$\begin{aligned}
\mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} A_k(x,t) &\text{ iff } (\mathfrak{d}(x), \mathfrak{t}(t)) \in A_k^{\mathcal{I}}, & \mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} \top,\\
\mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} P_k(x,y,t) &\text{ iff } (\mathfrak{d}(x), \mathfrak{d}(y), \mathfrak{t}(t)) \in P_k^{\mathcal{I}}, & \mathcal{I} \not\models^{\mathfrak{d},\mathfrak{t}} \bot,\\
\mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} t_1 < t_2 &\text{ iff } \mathfrak{t}(t_1) < \mathfrak{t}(t_2), & \mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} \neg\varphi \text{ iff } \mathcal{I} \not\models^{\mathfrak{d},\mathfrak{t}} \varphi,\\
\mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} t_1 = t_2 &\text{ iff } \mathfrak{t}(t_1) = \mathfrak{t}(t_2), & \mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} \varphi_1 \wedge \varphi_2 \text{ iff } \mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} \varphi_1 \text{ and } \mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} \varphi_2,\\
\mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} \forall x\, \varphi &\text{ iff } \mathcal{I} \models^{\mathfrak{d}',\mathfrak{t}} \varphi, \text{ for all } \mathfrak{d}' \text{ that differ from } \mathfrak{d} \text{ only on } x,\\
\mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} \forall t\, \varphi &\text{ iff } \mathcal{I} \models^{\mathfrak{d},\mathfrak{t}'} \varphi, \text{ for all } \mathfrak{t}' \text{ that differ from } \mathfrak{t} \text{ only on } t;
\end{aligned}$$

other first-order connectives and quantifiers such as $\rightarrow$, $\leftrightarrow$, $\exists$ are defined in the standard way. By an *ontology*, $\mathcal{O}$, we mean a set of 2-FOL($<$)-sentences. We say that $\mathcal{I}$ *satisfies* an ontology $\mathcal{O}$ or $\mathcal{I}$ is a *model* of $\mathcal{O}$ if $\mathcal{I} \models \varphi$, for each $\varphi \in \mathcal{O}$ (since the ontology contains only sentences, the assignments are irrelevant).

► **Example 1.** Consider a simple temporal ontology about research papers (as above) with role names publishedIn, acceptedIn and underSubmissionTo. We state that the domains of the three roles are mutually disjoint using axioms such as

$$\forall t \forall x \forall y_1 \forall y_2 \big(\mathsf{publishedIn}(x, y_1, t) \wedge \mathsf{acceptedIn}(x, y_2, t) \rightarrow \bot\big). \tag{1}$$

Basic temporal dependencies can be formulated as follows:

$$\forall t \forall x \forall y \big(\mathsf{publishedIn}(x,y,t) \rightarrow \forall s \big((s > t) \rightarrow \mathsf{publishedIn}(x,y,s)\big)\big), \tag{2}$$

$$\begin{aligned}\forall t \forall x \forall y \big(\mathsf{publishedIn}(x,y,t) \rightarrow \exists s \big((s < t) \wedge \mathsf{acceptedIn}(x,y,s) \wedge{}\\ \exists s'((s < s') \wedge \mathsf{publishedIn}(x,y,s') \wedge \neg\exists s'' \big((s < s'') \wedge (s'' < s')\big)\big)\big)\big),\end{aligned} \tag{3}$$

$$\begin{aligned}\forall t \forall x \forall y \big(\exists s' \left((s' < t) \wedge \mathsf{acceptedIn}(x,y,s')\right) \wedge{}\\ \exists s'' \left((t < s'') \wedge \mathsf{acceptedIn}(x,y,s'')\right) \rightarrow \mathsf{acceptedIn}(x,y,t)\big),\end{aligned} \tag{4}$$

an analogue of (3) for $\mathsf{acceptedIn}$ and $\mathsf{underSubmissionTo}$ and the convexity axiom (4) for $\mathsf{underSubmissionTo}$. The temporal ABox does not always use these role names, but rather integrates information from various data sources. For example, for a paper to be published it is necessary and sufficient that it is published in some venue (even if the publication venue is unknown). So, we use concept names $\mathsf{Published}$, $\mathsf{Accepted}$ and $\mathsf{UnderSubmission}$ to refer to all published, accepted and submitted papers, respectively: e.g.,

$$\forall t \forall x \, \big(\mathsf{Published}(x,t) \leftrightarrow \exists y \, \mathsf{publishedIn}(x,y,t)\big) \tag{5}$$

and similarly for $\mathsf{acceptedIn}$ and $\mathsf{underSubmissionTo}$. It follows from these axioms, in particular, that $\mathsf{Submitted}$, $\mathsf{Accepted}$ and $\mathsf{Published}$ form consecutive intervals as depicted in Fig. 1.

A 2-FOL($<$) *ontology-mediated query* (OMQ) is a pair $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{t}) = (\mathcal{O}, \boldsymbol{q}(\boldsymbol{x},\boldsymbol{t}))$, where $\mathcal{O}$ is an ontology and $\boldsymbol{q}(\boldsymbol{x},\boldsymbol{t})$ a 2-FOL($<$)-formula with free domain variables $\boldsymbol{x}$ and free temporal variables $\boldsymbol{t}$. We call $\boldsymbol{q}(\boldsymbol{x},\boldsymbol{t})$ a *query* and $\boldsymbol{x}$, $\boldsymbol{t}$ its *answer variables.* Given a temporal ABox $\mathcal{A}$, a model $\mathcal{I}$ of $\mathcal{A}$, a tuple $\boldsymbol{a}$ of individual names in $\mathcal{A}$ of the same length as $\boldsymbol{x}$, and a tuple $\boldsymbol{n}$ of time points in $\mathcal{A}$ of the same length as $\boldsymbol{t}$, we write $\mathcal{I} \models \boldsymbol{q}(\boldsymbol{a},\boldsymbol{n})$ if $\mathcal{I} \models^{\mathfrak{d},\mathfrak{t}} \boldsymbol{q}(\boldsymbol{x},\boldsymbol{t})$, for the assignments $\mathfrak{d}\colon \boldsymbol{x} \mapsto \boldsymbol{a}$ and $\mathfrak{t}\colon \boldsymbol{t} \mapsto \boldsymbol{n}$. Let $T \in \{\mathbb{Z}, \mathbb{N}, [\min \mathcal{A}, \max \mathcal{A}]\}$. We say that the tuple $(\boldsymbol{a},\boldsymbol{n})$ is a *certain answer to* $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q}(\boldsymbol{x},\boldsymbol{t}))$ *over* $\mathcal{A}$ *and* T and write $\mathcal{O},\mathcal{A} \models_T \boldsymbol{q}(\boldsymbol{a},\boldsymbol{n})$ if

$$\mathcal{I} \models \boldsymbol{q}(\boldsymbol{a},\boldsymbol{n}) \text{ for all } T\text{-models } \mathcal{I} \text{ of } \mathcal{O} \text{ and } \mathcal{A}.$$

▶ **Example 2.** In the context of Example 1, we now assume that the unit of time is one month. Then we can formulate the following queries.

- Find all accepted papers and their acceptance dates such that the paper was under submission for at least a year:

$$\boldsymbol{q}(x,t) = \mathsf{Accepted}(x,t) \land \mathsf{UnderSubmission}(x,t-1) \land \mathsf{UnderSubmission}(x,t-13). \tag{6}$$

  Since the flow of time is discrete, any formula of the form $P(\boldsymbol{x}, t-1)$ is simply an abbreviation for $\exists t' \, \big[P(\boldsymbol{x},t') \land (t' < t) \land \neg\exists t'' \, \big((t' < t'') \land (t'' < t)\big)\big]$; $\mathsf{UnderSubmission}(x,t-13)$ can be defined similarly.
- Papers that were published within two months after acceptance but had been under submission for three years:

$$\boldsymbol{q}(x,t) = \exists s \, \big((s < t) \land \mathsf{Accepted}(x,s) \land \mathsf{UnderSubmission}(x,s-1) \land \\ \mathsf{Published}(x,s+2) \land \mathsf{UnderSubmission}(x,s-37)\big). \tag{7}$$

- Authors of papers that were submitted more than two years ago but have not been accepted yet:

$$\boldsymbol{q}(x,t) = \exists y \, \big(\mathsf{authorOf}(x,y,t) \land \mathsf{UnderSubmission}(y,t-24) \land \mathsf{UnderSubmission}(y,t)\big). \tag{8}$$

  Recall that $\mathsf{UnderSubmission}$ is disjoint with $\mathsf{Accepted}$ and can only occur before the paper is eventually accepted.

Note that 2-FOL($<$)-formulas as we defined them do not use individual constants. This assumption is for simplicity only; it is straightforward to extend the syntax and semantics of temporal ontologies and queries to 2-FOL($<$) with individual constants.

Given two fragments $\mathcal{L}$ and $\mathcal{Q}$ of 2-FOL($<$), we denote by $(\mathcal{L},\mathcal{Q})$ the *class of ontology-mediated queries* $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{t}) = (\mathcal{O}, \boldsymbol{q}(\boldsymbol{x},\boldsymbol{t}))$ such that $\mathcal{O}$ is formulated in $\mathcal{L}$ and $\boldsymbol{q}(\boldsymbol{x},\boldsymbol{t})$ in $\mathcal{Q}$. Let T

be any of $\mathbb{Z}$, $\mathbb{N}$ or $[\min \mathcal{A}, \max \mathcal{A}]$. By $(\mathcal{L}, \mathcal{Q})$-*OMQ evaluation over* T we understand the problem of deciding, for a given $(\mathcal{L}, \mathcal{Q})$-OMQ $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{t}) = (\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}, \boldsymbol{t}))$, a temporal ABox $\mathcal{A}$ and tuples $\boldsymbol{a}$ and $\boldsymbol{n}$ in $\mathcal{A}$ of the same length as $\boldsymbol{x}$ and $\boldsymbol{t}$, whether $\mathcal{O}, \mathcal{A} \models_T \boldsymbol{q}(\boldsymbol{a}, \boldsymbol{n})$. The *combined complexity* of $(\mathcal{L}, \mathcal{Q})$-OMQ evaluation over T is defined as the computational complexity of the above problem. As the queries and ontologies are mostly much smaller than the ABox $\mathcal{A}$, combined complexity is often misleading as a measure of the resources needed for query evaluation [85]. An alternative and often more appropriate complexity measure is the *data complexity* of $(\mathcal{L}, \mathcal{Q})$-OMQ evaluation over T, that is the complexity of deciding, for *fixed* $\mathcal{O}$ in $\mathcal{L}$ and $\boldsymbol{q}(\boldsymbol{x}, \boldsymbol{t})$ in $\mathcal{Q}$, whether $\mathcal{O}, \mathcal{A} \models_T \boldsymbol{q}(\boldsymbol{a}, \boldsymbol{n})$ for any given ABox $\mathcal{A}$ and tuples $\boldsymbol{a}$ and $\boldsymbol{n}$.

The data complexity of OMQ evaluation is closely related to the equivalent rewritability of OMQs into standard query languages. With any temporal ABox $\mathcal{A}$ we associate a $[\min \mathcal{A}, \max \mathcal{A}]$-interpretation

$$\mathcal{I}_{\mathcal{A}} = \big(([\min \mathcal{A}, \max \mathcal{A}], <), \Delta^{\mathcal{I}_{\mathcal{A}}}, P_1^{\mathcal{I}_{\mathcal{A}}}, P_2^{\mathcal{I}_{\mathcal{A}}}, \dots, A_1^{\mathcal{I}_{\mathcal{A}}}, A_2^{\mathcal{I}_{\mathcal{A}}}, \dots, a_1^{\mathcal{I}_{\mathcal{A}}}, a_2^{\mathcal{I}_{\mathcal{A}}}, \dots\big),$$

where $\Delta^{\mathcal{I}_{\mathcal{A}}}$ is the set of individual names in $\mathcal{A}$, $a_i^{\mathcal{I}_{\mathcal{A}}} = a_i$ for all i, and

$$A_k^{\mathcal{I}_{\mathcal{A}}} = \{(a_i, n) \mid A_k(a_i, n) \in \mathcal{A}\} \quad \text{and} \quad P_k^{\mathcal{I}_{\mathcal{A}}} = \{(a_i, a_j, n) \mid P_k(a_i, a_j, n) \in \mathcal{A}\}, \quad \text{for all } k.$$

Now, let $\mathcal{Q}'$ be any query language over $[\min \mathcal{A}, \max \mathcal{A}]$-interpretations, for example, 2-FOL($<$) itself, a fragment of 2-FOL($<$) or even its extension. We say that $(\mathcal{L}, \mathcal{Q})$-OMQs are $\mathcal{Q}'$-*rewritable over* T if, for every OMQ $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}, \boldsymbol{t}))$ in $(\mathcal{L}, \mathcal{Q})$, there exists $\boldsymbol{q}'(\boldsymbol{x}, \boldsymbol{t})$ in $\mathcal{Q}'$ such that, for every temporal ABox $\mathcal{A}$ that has a common model with $\mathcal{O}$, the following equivalence holds for all tuples $\boldsymbol{a}$ and $\boldsymbol{n}$ in $\mathcal{A}$ of appropriate length:

$$\mathcal{O}, \mathcal{A} \models_T \boldsymbol{q}(\boldsymbol{a}, \boldsymbol{n}) \qquad \text{iff} \qquad \mathcal{I}_{\mathcal{A}} \models \boldsymbol{q}'(\boldsymbol{a}, \boldsymbol{n}).$$

If $\mathcal{Q}'$ is 2-FOL($<$) over T, then $\mathcal{I}_{\mathcal{A}} \models \boldsymbol{q}'(\boldsymbol{a}, \boldsymbol{n})$ is the standard database query evaluation problem for temporal ABoxes and 2-FOL($<$) queries, which is known to be PSPACE-complete for combined complexity; see, e.g., [61]; if, however, one fixes the query and thus considers the data complexity, then this problem is in AC^0, the class of languages computable by bounded-depth polynomial-size circuits with unary NOT-gates and unbounded fan-in AND- and OR-gates.

Of course, the OMQ evaluation problem for the full 2-FOL($<$) is undecidable, and it is one of the main problems of temporal ontology-mediated query answering to design useful ontology and query languages for which the query evaluation problem is decidable or, even better, feasible in practice. The latter requirement is typically interpreted as being at least in PTIME in data complexity, but to query very large data and employ existing query engines PTIME query evaluation often is not sufficient, and one aims at rewritability into first-order logic (AC^0 data complexity). In the following two sections, we discuss a few known approaches to this problem.

3 Queries Mediated by Domain- or Time-centric Ontologies

An important way of obtaining temporal ontology languages from 2-FOL is simply omitting (non-trivial) quantification over one of its two sorts. Thus, intuitively, if we disallow all but a single outermost universal quantifier for a temporal variable, then we obtain a 'domain-centric' ontology language, in which one can define a time-independent model of the domain; and if we disallow all but a sequence of outermost universal quantifiers for domain variables, then we obtain a 'time-centric' ontology language using which one can define a propositional temporal model. Both approaches have been investigated, and, in the rest of the section, we shall provide a summary of the obtained results. If a model representing both temporal and domain knowledge is needed, we have to carefully define the interaction between the domain and time quantifiers.

3.1 Domain-Centric Ontology Languages

It is straightforward to restrict 2-FOL($<$) in such a way that it only defines non-temporal properties: such an ontology would consist of 2-FOL($<$) sentences $\forall t\, \varphi(t)$, where φ does not contain quantifiers over temporal variables. Of course, query evaluation is still undecidable, and so further restrictions of its expressive power are needed. In standard, non-temporal, ontology-mediated query answering, description logics are the most popular fragments of first-order logic used to define ontologies. Here, we introduce three basic families of DLs that have been important in the context of OMQ answering, and from which many others can be derived in a straightforward way. Namely, we introduce the basic expressive DL $\mathcal{ALC}$ [19] and the lightweight DLs *DL-Lite* [37, 6] and $\mathcal{EL}$ [18]. In $\mathcal{ALC}$, *concepts* C are constructed using the grammar

$$C \quad ::= \quad \top \quad | \quad A_k \quad | \quad \neg C \quad | \quad C_1 \sqcap C_2 \quad | \quad \exists P_k.C.$$

An $\mathcal{ALC}$ *TBox* (*ontology*) is a finite set of *concept inclusions* $C_1 \sqsubseteq C_2$, where C_1 and C_2 are $\mathcal{ALC}$ concepts. Concepts in the fragment $\mathcal{EL}$ of $\mathcal{ALC}$ are $\mathcal{ALC}$ concepts without occurrences of negation $\neg$. An $\mathcal{EL}$ TBox is a finite set of concept inclusions $C_1 \sqsubseteq C_2$, where C_1 and C_2 are $\mathcal{EL}$ concepts. In *DL-Lite*, *basic concepts* B and *roles* R are constructed using the grammar

$$\begin{aligned} B \quad &::= \quad \top \quad | \quad A_k \quad | \quad \exists R.\top, \\ R \quad &::= \quad P_k \quad | \quad P_k^-. \end{aligned}$$

A *DL-Lite*$^{\mathcal{H}}_{core}$ TBox is a finite set of concept and role inclusions of the form

$$\begin{aligned} &B_1 \sqsubseteq B_2, \qquad\qquad && B_1 \sqcap B_2 \sqsubseteq \bot, \\ &R_1 \sqsubseteq R_2, && R_1 \sqcap R_2 \sqsubseteq \bot, \end{aligned}$$

where B_1 and B_2 are basic concepts and R_1 and R_2 are roles. Concept and role inclusions of the second type are also called *disjointness axioms*. In *DL-Lite*$^{\mathcal{H}}_{horn}$, one can also form intersections of basic concepts:

$$B_1 \sqcap \dots \sqcap B_k \sqsubseteq B, \qquad\qquad B_1 \sqcap \dots \sqcap B_k \sqsubseteq \bot;$$

in *DL-Lite*$^{\mathcal{H}}_{krom}$, one can use negation (but still any concept inclusion contains only two concepts): that is, concept inclusions are of the form $B_1 \sqsubseteq B_2$, $B_1 \sqcap B_2 \sqsubseteq \bot$ and $\top \sqsubseteq B_1 \sqcup B_2$; finally, in *DL-Lite*$^{\mathcal{H}}_{bool}$ one can use both conjunction and negation resulting in concept inclusions of the form $D_1 \sqsubseteq D_2$, where the D_i are defined using the rule

$$D \quad ::= \quad B \quad | \quad \neg D \quad | \quad D_1 \sqcap D_2.$$

(We will assume without loss of generality that the concept inclusions in *DL-Lite*$^{\mathcal{H}}_{bool}$ are given in normal form: $B_1 \sqcap \dots \sqcap B_k \sqsubseteq B_1' \sqcup \dots \sqcup B_n'$; as usual, we assume that the empty union is $\bot$ and the empty intersection is $\top$). All of the above languages in the DL-Lite family contain role inclusions, and the fragment of *DL-Lite*$^{\mathcal{H}}_{c}$ without role inclusions is denoted by

DL-Lite$_{\boldsymbol{c}}$. Two concept (or role) inclusions $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$ are often abbreviated as $C_1 \equiv C_2$ and are called a concept (respectively, role) *equivalence axiom*.

The DLs introduced above can be regarded as fragments of first-order logic (with a single sort). In the temporal setting, every (basic) concept C can be translated (using the so-called *standard translation*) to a 2-FOL(<)-formula $C^\sharp(x,t)$ with one free domain variable x and a single temporal variable t, and every role R to a 2-FOL(<)-formula $R^\sharp(x,y,t)$ with two domain variables x, y and a single temporal variable t:

$$\begin{aligned}
(A_k)^\sharp(x,t) &= A_k(x,t), & (\neg C)^\sharp(x,t) &= \neg C^\sharp(x,t),\\
(P_k)^\sharp(x,y,t) &= P_k(x,y,t), & (C_1 \sqcap C_2)^\sharp(x,t) &= C_1^\sharp(x,t) \land C_2^\sharp(x,t),\\
(P_k^-)^\sharp(x,y,t) &= P_k(y,x,t), & (\exists R.C)^\sharp(x,t) &= \exists y\,\big(R^\sharp(x,y,t) \land C^\sharp(y,t)\big).
\end{aligned}$$

Every concept inclusion $C_1 \sqsubseteq C_2$ is then translated as $\forall t \forall x\,\big(C_1^\sharp(x,t) \to C_2^\sharp(x,t)\big)$ and every role inclusion $R_1 \sqsubseteq R_2$ as $\forall t\forall x\forall y\,\big(R_1^\sharp(x,y,t) \to R_2^\sharp(x,y,t)\big)$. Thus, we obtain a first important type of temporal ontologies by demanding that its (essentially atemporal) concept and role inclusions hold true at *every time point*. More formally, given a T-interpretation $\mathcal{I}$ and $n \in T$, we can define an *n-slice* $\mathcal{I}(n)$ of $\mathcal{I}$ by taking the standard Tarski-style interpretation for the respective DL:

$$\mathcal{I}(n) \;=\; \big(\Delta^{\mathcal{I}}, P_1^{\mathcal{I}(n)}, P_2^{\mathcal{I}(n)}, \dots, A_1^{\mathcal{I}(n)}, A_2^{\mathcal{I}(n)}, \dots, a_1^{\mathcal{I}(n)}, a_2^{\mathcal{I}(n)}, \dots\big),$$

where $a_i^{\mathcal{I}(n)} = a_i^{\mathcal{I}}$, for all i, and

$$P_k^{\mathcal{I}(n)} = \{(u,v) \mid (u,v,n) \in P_k^{\mathcal{I}}\} \quad\text{and}\quad A_k^{\mathcal{I}(n)} = \{u \mid (u,n) \in A_k^{\mathcal{I}}\}, \quad \text{for all } k.$$

It follows that $\mathcal{I}$ is a T-model of an ontology $\mathcal{O}$ iff each of the concept and role inclusions in the ontology is satisfied in each of the slices $\mathcal{I}(n)$, for $n \in T$.

▶ **Example 3.** In a domain-centric ontology language in the context of Example 1, we can express (1) by using a concept disjointness axiom and (5) using a concept equivalence axiom:

$$\exists \mathsf{publishedIn}.\top \sqcap \exists \mathsf{acceptedIn}.\top \sqsubseteq \bot, \tag{1'}$$

$$\mathsf{Published} \equiv \exists \mathsf{publishedIn}.\top. \tag{5'}$$

It is to be noted that, in the languages just introduced, one cannot represent or reason about any dependencies between the interpretations $\mathcal{I}(n)$ and $\mathcal{I}(m)$ for distinct time points n and m. Examples of such dependencies are sentences (2)–(4) in Example 1. The extension of any ontology language $\mathcal{L}$ with the option to say that a concept name A is time-independent (that is, $A^{\mathcal{I}(n)} = A^{\mathcal{I}(m)}$ for all time points n, m) is called $\mathcal{L}$ *with rigid concepts*. The extension of $\mathcal{L}$ with *rigid roles* is defined analogously. In Example 1, $\mathsf{authorOf}$ could be a rigid role. Of course, if the language has rigid roles, then rigid concepts can be 'simulated' by considering domains of rigid roles: if role R is rigid, then the equivalence axiom $C \equiv \exists R$ ensures that concept C is also rigid.

Baader et al. [20] proposed domain-centric languages. They introduced $\mathcal{ALC}$*-LTL* as the language of $\mathcal{ALC}$ axioms (concept inclusions, or GCI as they are often called in description logic) and ABox assertions with Boolean connectives and temporal operators applied to them. For example, the temporalised axiom $\Diamond_F \Box_F(\mathsf{USCitizen} \sqsubseteq \exists \mathsf{insuredBy}.\mathsf{Insurer})$ says that there is a future time point, from which on every US citizen will always have a health insurance. It turns out that without rigid symbols the two components – the domain and the time – have very little interaction, and so in order to check whether a given formula φ in $\mathcal{ALC}$*-LTL*

Table 1 Complexity of the satisfiability problem over $\mathbb{N}$.

language	combined complexity		
	no rigid symbols	rigid concepts only	rigid roles & concepts
$\mathcal{ALC}$-LTL [20]	ExpTime	NExpTime	2ExpTime
—"— global GCIs [20]	ExpTime	ExpTime	2ExpTime
$\mathcal{EL}$-LTL [28]	PSpace	NExpTime	NExpTime
—"— global GCIs [28]	PSpace	PSpace	PSpace

is satisfiable, one can check whether (1) the *propositional abstraction* of φ (the result of replacing DL axioms with propositional variables) is satisfiable and (2) the satisfying model yields consistent sets of DL axioms. As a result, the complexity is usually the maximum of the complexities of the two components; see Table 1. Rigid concepts and/or rigid roles make the interaction stronger and require additional global guessing/bookkeeping but the propositional abstraction technique is still applicable. The second set of results in Table 1 refers to the fragment of $\mathcal{ALC}$*-LTL* in which Boolean connectives and temporal operators can be applied only to ABox assertions but $\mathcal{ALC}$ axioms hold globally in all models (in precisely the same way as we defined in the standard translation above). Such a restriction dramatically reduces the complexity for the logic $\mathcal{EL}$*-LTL* [28].

Note also that the Semantic Web community has developed a variety of extensions of RDF/S and OWL with validity time [64, 74, 48]. The focus of this direction of research is on representing and querying timestamped RDF triples or OWL axioms.

3.2 Time-Centric OMQs

One of the main differences between description logics and first-order logic is that the former do not use individual variables. Instead, description logic constructors such as existential restrictions express certain quantifier patterns. The situation in reasoning about time is similar: instead of representing explicitly the temporal precedence relation $<$ using individual variables, one employs temporal operators encoding certain natural language patterns. A very well studied language based on temporal operators is *linear-time temporal logic* (*LTL*) [72]. In contrast to the description logics introduced above, which are much weaker than first-order logic, *LTL* with operators $\mathcal{S}$ ('since') and $\mathcal{U}$ ('until') has exactly the same expressive power as the corresponding *time-fragment* of 2-FOL($<$) (Kamp's theorem); see, e.g., [44, 75]. We now introduce the *LTL* extensions of the concept and role grammars defined above:

$$\begin{array}{lcllllllllll} D & ::= & C & \mid & \bigcirc_P D & \mid & \bigcirc_F D & \mid & D_1 \mathcal{S} D_2 & \mid & D_1 \mathcal{U} D_2, \\ S & ::= & R & \mid & \bigcirc_P S & \mid & \bigcirc_F S & \mid & S_1 \mathcal{S} S_2 & \mid & S_1 \mathcal{U} D_2. \end{array}$$

We will also use common abbreviations: for example, $\Diamond_P D = \top \mathcal{S} D$ ('sometime in the past' for concepts) and $\Box_F S = \neg(\top \mathcal{U} \neg S)$ ('always in the future' for roles). The standard translation of concepts can be extended to temporalised concepts as follows:

$$\begin{aligned} (\bigcirc_P D)^\sharp(x,t) &= D^\sharp(x,t-1), \\ (\bigcirc_F D)^\sharp(x,t) &= D^\sharp(x,t+1), \\ (D_1 \mathcal{S} D_2)^\sharp(x,t) &= \exists t_1\big((t_1 < t) \ \wedge \ D_2^\sharp(x,t_1) \ \wedge \ \forall t_2\big((t_1 < t_2) \wedge (t_2 < t) \to D_1^\sharp(x,t_2)\big)\big), \\ (D_1 \mathcal{U} D_2)^\sharp(x,t) &= \exists t_1\big((t < t_1) \ \wedge \ D_2^\sharp(x,t_1) \ \wedge \ \forall t_2\big((t < t_2) \wedge (t_2 < t_1) \to D_1^\sharp(x,t_2)\big)\big). \end{aligned}$$

Temporalised roles are translated into 2-FOL($<$) similarly (but two domain variables are used rather than one). Recall that $(t-1)$ is a shortcut for t' that satisfies the condition $(t' < t) \land \neg\exists t'' \big((t' < t'') \land (t'' < t)\big)$. So, under the strict interpretation of $\mathcal{U}$ and $\mathcal{S}$, the temporal operators $\bigcirc_F$ ('next time') and $\bigcirc_P$ ('previous time') could be equivalently defined as $\bigcirc_F D = \bot\, \mathcal{U}\, D$ and $\bigcirc_P D = \bot\, \mathcal{S}\, D$, respectively.

► **Example 4.** In the context of Example 1, we can represent the 'concept' analogues of sentences (1)– (4) as follows:

$$\mathsf{Published} \sqcap \mathsf{Accepted} \sqsubseteq \bot, \tag{1''}$$
$$\mathsf{Published} \sqsubseteq \Box_F \mathsf{Published}, \tag{2''}$$
$$\mathsf{Published} \sqsubseteq \Diamond_P (\mathsf{Accepted} \sqcap \bigcirc_F \mathsf{Published}), \tag{3''}$$
$$\Diamond_P \mathsf{Accepted} \sqcap \Diamond_F \mathsf{Accepted} \sqsubseteq \mathsf{Accepted}. \tag{4''}$$

Rigid concepts and roles can be defined in the language introduced above be using inclusions of the form $C \sqsubseteq \Box_F \Box_P C$ or $C \equiv \bigcirc_F C$.

If no relational knowledge is needed for the domain and the focus is on temporal aspects (as in Example 4), then it suffices to work with ontologies that represent the behaviour of individual domain elements without formalising any interaction between them. So, in the remainder of Section 3.2, we concentrate on the concept-only ontology languages and, in Section 4, we show how these results can be extended to the full setting (under certain restrictions).

In order to present the fine-grained analysis of the complexity of OMQ evaluation, we assume that our ontologies are given in a certain normal form. More precisely, it is known that any *LTL* formula can be transformed into a polynomial-size *LTL* formula in *separated normal form* (SNF) [42] that has the same models (if restricted to the original vocabulary – the transformation requires introduction of auxiliary names, but the result is a conservative extension, which preserves the models restricted to the original vocabulary). The formulas in SNF are conjunctions of global and initial *temporal clauses* that only use the operators $\bigcirc_P$, $\bigcirc_F$, $\Box_P$ and $\Box_F$. So, we consider the *time-centric* ontology language $LTL^{\Box\bigcirc}_{bool}$ with concept inclusions of the form

$$L_1 \sqcap \dots \sqcap L_k \sqsubseteq L'_1 \sqcup \dots \sqcup L'_n,$$

where the L_i and L'_i are concept names possibly prefixed by unary temporal operators $\bigcirc_P$, $\bigcirc_F$, $\Box_P$ or $\Box_F$. We also define the *core*, *krom* and *horn* fragments of $LTL^{\Box\bigcirc}_{bool}$, where the temporal clauses are restricted to

$$L_1 \sqsubseteq L_2, \qquad L_1 \sqcap L_2 \sqsubseteq \bot, \tag{core}$$
$$L_1 \sqsubseteq L_2, \qquad L_1 \sqcap L_2 \sqsubseteq \bot, \qquad \top \sqsubseteq L_1 \sqcup L_2, \tag{krom}$$
$$L_1 \sqcap \dots \sqcap L_k \sqsubseteq L, \qquad L_1 \sqcap \dots \sqcap L_k \sqsubseteq \bot, \tag{horn}$$

respectively, and the $\bigcirc$- and $\Box$-fragments $LTL^{\bigcirc}_{\boldsymbol{c}}$ and $LTL^{\Box}_{\boldsymbol{c}}$, where only $\bigcirc_P/\bigcirc_F$ and $\Box_P/\Box_F$ operators can be applied. It can be seen that the sub-Boolean fragments of $LTL^{\Box\bigcirc}_{bool}$ are in fact the concept-only counterparts of the respective fragments of *DL-Lite*. Recall that the satisfiability problem is PSpace-complete for $LTL^{\Box\bigcirc}_{bool}$- and $LTL^{\Box\bigcirc}_{horn}$-formulas, NP-complete for $LTL^{\Box\bigcirc}_{krom}$- and $LTL^{\Box}_{krom}$-formulas, and NLogSpace-complete for $LTL^{\bigcirc}_{core}$-, $LTL^{\bigcirc}_{krom}$- and $LTL^{\Box}_{core}$-formulas [9].

Let $\mathcal{O}$ be an ontology in a time-centric language. An *atomic LTL-OMQ* is a pair of the form $(\mathcal{O}, A_k(x,t))$, where A_k is a concept name. We also consider a larger class of OMQs

Table 2 Data complexity and rewritability of *LTL*-OMQs over $\mathbb{Z}$.

	atomic			positive		
$\boldsymbol{c}$	$LTL_{\boldsymbol{c}}^{\Box}$	$LTL_{\boldsymbol{c}}^{\bigcirc}$	$LTL_{\boldsymbol{c}}^{\Box\bigcirc}$	$LTL_{\boldsymbol{c}}^{\Box}$	$LTL_{\boldsymbol{c}}^{\bigcirc}$	$LTL_{\boldsymbol{c}}^{\Box\bigcirc}$
bool	2-FOL($<$)	MSO($<$)	MSO($<$)	MSO($<$)	MSO($<$)	MSO($<$)
krom	2-FOL($<$)	2-FOL($<,+$)	MSO($<$)*	MSO($<$)	MSO($<$)	MSO($<$)
horn	2-FOL($<$)	MSO($<$)	MSO($<$)	2-FOL($<$)	MSO($<$)	MSO($<$)
core	2-FOL($<$)	2-FOL($<,+$)	MSO($<$)*	2-FOL($<$)	2-FOL($<,+$)	MSO($<$)*

*It is still open whether these can be improved to 2-FOL($<,+$); all other results in the table are optimal: in particular, MSO($<$) means NC^1-hardness for data complexity and so, no 2-FOL($<,+$)-rewritability.

based on *positive temporal concepts* $\varkappa$, which are defined by the following grammar:

$$\varkappa \quad ::= \quad \bot \quad | \quad \top \quad | \quad A_k \quad | \quad \varkappa_1 \sqcap \varkappa_2 \quad | \quad \varkappa_1 \sqcup \varkappa_2 \quad | \\ \Box_P \varkappa \quad | \quad \Box_F \varkappa \quad | \quad \varkappa_1 \mathcal{S} \varkappa_2 \quad | \quad \varkappa_1 \mathcal{U} \varkappa_2.$$

Observe that operators $\bigcirc_P$, $\bigcirc_F$, $\Diamond_P$ and $\Diamond_F$ can be used in positive temporal concepts as abbreviations. A *positive LTL-OMQ* is a pair of the form $(\mathcal{O}, \varkappa(x,t))$. It is to be noted that, unlike the ontology language, where we used a normal form, one cannot eliminate the binary temporal operators $\mathcal{S}$ and $\mathcal{U}$ (and the 'sometime in the past/future' operators $\Diamond_P/\Diamond_F$).

In the context of *LTL*-OMQs, two types of $\mathcal{Q}'$-rewritability are of interest for the target language $\mathcal{Q}'$: 2-FOL($<$) and 2-FOL($<,+$). The second language extends 2-FOL($<$) with the ternary numeric predicate PLUS that is interpreted in the two-sorted structure $\mathcal{I}_\mathcal{A}$ (defined in Section 2) as follows:

$$\mathcal{I}_\mathcal{A} \models \textsc{plus}(n, n_1, n_2) \quad \text{iff} \quad n = n_1 + n_2, \quad \text{for } n, n_1, n_2 \in [\min \mathcal{A}, \max \mathcal{A}].$$

Observe that even though we can express terms such as $t+n$, for a fixed $n \in \mathbb{Z}$, in 2-FOL($<$), terms of the form $t+s$ are not expressible in 2-FOL($<$). Evaluation of 2-FOL($<,+$)-formulas is known to be in LogTime-uniform AC^0 for data complexity [56] (recall that AC^0 is the class of languages computable by bounded-depth polynomial-size circuits with unary NOT-gates and unbounded fan-in AND- and OR-gates).

▶ **Example 5.** Let $\mathcal{O}$ be an ontology with the following two axioms:

$$\bigcirc_P A \sqsubseteq B, \qquad \bigcirc_P B \sqsubseteq A. \tag{9}$$

Consider the positive *LTL*-OMQ $\boldsymbol{Q}(x,t) = (\mathcal{O}, \bigcirc_F \bigcirc_F B(x,t))$ and ABox $\mathcal{A} = \{A(a,0), C(a,1)\}$. We have $(a^\mathcal{I}, 2n+1) \in B^\mathcal{I}$, for any $n \geq 0$ and any $\mathbb{N}$- or $\mathbb{Z}$-model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$. It follows that $(a,1)$ is the only certain answer to $\boldsymbol{Q}(x,t)$ because only 1 of all odd numbers is within the interval between $\min \mathcal{A}$ and $\max \mathcal{A}$ (note, however, that the relevant B is true at moment 3).

▶ **Example 6.** Consider now the atomic *LTL*-OMQ $\boldsymbol{Q}(x,t) = (\mathcal{O}, A(x,t))$ with the same $\mathcal{O}$ defined by (9). It is not hard to see that $(a,n) \in A^\mathcal{I}$ for any $\mathbb{Z}$-model $\mathcal{I}$ of $\mathcal{O}$ and a given temporal ABox $\mathcal{A}$ iff t is either at an even distance $t-s$ from some $A(a,s) \in \mathcal{A}$ or at an odd distance $t-s$ from some $B(a,s) \in \mathcal{A}$. Thus, the following formula

$$\exists s, n, k, k' \left[\big(A(x,s) \land \textsc{plus}(k,n,n) \land \textsc{plus}(t,s,k) \big) \lor \\ \big(B(x,s) \land \textsc{plus}(k,n,n) \land \textsc{plus}(k',k,1) \land \textsc{plus}(t,s,k') \big) \right]$$

is a 2-FOL($<,+$)-rewriting of $\boldsymbol{Q}(x,t)$, where, for example, $\textsc{plus}(k,n,n)$ means $k=2n$. Note that s, n, k and all other quantified variables range between $0=\min\mathcal{A}$ and $\max\mathcal{A}$ in any $\mathcal{I}_{\mathcal{A}}$; in particular, $t\geq s$. Finally, observe that $\boldsymbol{Q}(x,t)$ is *not* 2-FOL($<$)-rewritable since properties such as 't is even' are not definable by 2-FOL($<$)-formulas [61].

▶ **Example 7.** Next, instead of just checking whether the distance is even or odd, we devise an ontology that checks whether the number of certain symbols in a given interval is even or odd. More precisely, consider the atomic *LTL*-OMQ $\boldsymbol{Q}(x,t)=(\mathcal{O},B_0(x,t))$, where $\mathcal{O}$ consists of concept inclusions

$$\bigcirc_{\!F}B_k\sqcap A_0\sqsubseteq B_k \qquad \text{and} \qquad \bigcirc_{\!F}B_k\sqcap A_1\sqsubseteq B_{1-k}, \qquad \text{for}\ \ k=0,1.$$

Informally, each occurrence of A_0 in the ABox keeps the same subscript k in B_k and each occurrence of A_1 flips the subscript over by replacing B_0 with B_1 and the other way round. So, for any word $\boldsymbol{e}=(e_0,\dots,e_{n-1})\in\{0,1\}^n$, let $\mathcal{A}_{\boldsymbol{e}}=\{B_0(a,n)\}\cup\{A_{e_i}(a,i)\mid 0\leq i<n\}$. It is not hard to check that $(a,0)$ is a certain answer to $\boldsymbol{Q}(x,t)$ over $\mathcal{A}_{\boldsymbol{e}}$ iff the number of 1s in $\boldsymbol{e}$ is even (Parity). As Parity is not in AC^0 [43], $\boldsymbol{Q}(x,t)$ is not 2-FOL-rewritable even if *arbitrary* numeric predicates (not only $\textsc{plus}$) are allowed in rewritings.

On the other hand, Parity is a regular language, and so belongs to NC$^1\supsetneq$ AC^0, the class of languages recognisable by logarithmic-depth circuits with unary NOT-gates and fan-in two AND- and OR-gates. Recall also that (i) regular languages coincide with those definable by *monadic second-order* (MSO) formulas built from atoms of the form $A(t)$ and $t<t'$ using the Booleans, first-order quantifiers $\forall t$ and $\exists t$, and second-order quantifiers $\forall A$ and $\exists A$ [35], and that (ii) MSO($<$)-formulas can encode the semantics of propositional temporal logic see, e.g., [46]. Thus, all $\mathit{LTL}^{\Box\bigcirc}_{bool}$ OMQs are MSO($<$)-rewritable, and so answering such OMQs is in NC^1 for data complexity.[1] On the other hand, in many cases we can construct 2-FOL($<,+$)- or even 2-FOL($<$)-rewritings; for details on the results, see Table 2 [8].

3.3 Query Answering with Domain-Centric Ontologies

Even without a temporal dimension, first-order logic is too expressive for effective ontology-mediated query answering. Instead, research has been focussed on ontologies in description logic and on queries in small fragments of FO. Most popular are *conjunctive queries* (CQs), defined by the grammar

$$\varphi \quad ::= \quad A_k(x) \quad | \quad P_k(x_1,x_2) \quad | \quad \varphi_1\wedge\varphi_2 \quad | \quad \exists x\,\varphi,$$

and disjunctions of CQs called *unions of conjunctive queries* (UCQs). Thus, CQs are (equivalent to) conjunctions of atoms in which some variables are existentially quantified. We briefly discuss basic results on the rewritability and data complexity of ontology-mediated queries in the atemporal classes of OMQs ($\mathit{DL\text{-}Lite}^{\mathcal{H}}_{horn}$, CQ), ($\mathcal{EL}$, CQ), and ($\mathcal{ALC}$, CQ). We assume for simplicity that ABoxes consist of facts without time stamps and interpret the description logics in the corresponding single sorted interpretations. Then OMQs in ($\mathit{DL\text{-}Lite}^{\mathcal{H}}_{horn}$, CQ) are always rewritable into UCQs, a fact which was the main motivation for the introduction of the *DL-Lite* family [37, 6].

▶ **Example 8.** For $\mathcal{O}=\{\exists\mathsf{publishedIn}.\top\sqsubseteq\mathsf{Published}\}$ and the CQ $\boldsymbol{q}(x)=\mathsf{Published}(x)$, a rewriting of $(\mathcal{O},\boldsymbol{q}(x))$ is given by $\boldsymbol{q}'(x)=\exists y\,\mathsf{publishedIn}(x,y)\vee\mathsf{Published}(x)$. Intuitively, $\boldsymbol{q}'(x)$ is the disjunction over all possible 'reasons' (according to $\mathcal{O}$) for x to be published.

[1] By NC^1 we mean the uniform NC^1, which coincides with ALogTime.

It follows that answering OMQs from ($DL\text{-}Lite^{\mathcal{H}}_{horn}$, CQ) is in AC^0 fot data complexity. In contrast, not all OMQs in $(\mathcal{EL}, \mathrm{CQ})$ are rewritable into UCQs or even first-order logic.

▶ **Example 9.** For $\mathcal{O} = \{\, \exists\mathsf{refersTo.Publication} \sqsubseteq \mathsf{Publication} \,\}$ and CQ $\boldsymbol{q}(x) = \mathsf{Publication}(x)$, one can readily see that $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}(a)$ iff there is a path from a to some individual b such that $\mathsf{Publication}(b) \in \mathcal{A}$ along the $\mathsf{refersTo}$-relation in $\mathcal{A}$. Since reachability cannot be expressed in first-order logic, there is no rewriting of $(\mathcal{O}, \boldsymbol{q}(x))$ in first-order logic.

It can be shown, however, that every OMQ in $(\mathcal{EL}, \mathrm{CQ})$ is rewritable into a datalog program, and so CQ evaluation is in PTIME for data complexity [78]. For OMQs in $(\mathcal{ALC}, \mathrm{CQ})$, the situation is even worse: in this case, OMQ answering can be coNP-hard for data complexity [55]. Bienvenu et al. [24] give a partial classification of OMQs in $(\mathcal{ALC}, \mathrm{CQ})$ into those in PTIME and those that are coNP-hard for data complexity.

We now return to querying temporal data under the assumption that the ontology is domain-centric. Querying in this framework has mainly been investigated in the context of ontology-based monitoring of dynamic systems [15, 13, 17]. Suppose that timestamped data is collected while monitoring a system. The data collected at each time point n forms a sequence $(\mathcal{A}(i))_{0 \le i \le n}$ of ABoxes $\mathcal{A}(i)$ such that every $\mathcal{A}(i)$ contains assertions of the form $A_k(a, i)$ and $P_k(a, b, i)$. Thus, the temporal ABox $\mathcal{A}$ is given as $\mathcal{A} = \bigcup_{0 \le i \le n} \mathcal{A}(i)$, where $\min \mathcal{A} = 0$ and $\max \mathcal{A} = n$ is the current time point. Ontology-mediated queries are used to detect whether an event of interest has occured in $\mathcal{A}$ up to the time point n. The following example illustrates this scenario.

▶ **Example 10.** Suppose that a temporal ABox $\mathcal{A}$ maintained by a journal editor contains data about the submission, reviewing, acceptance and publication of articles. Thus, similarly to the temporal ABox introduced above it contains assertions stating whether an article is under submission, has been accepted, has been published, and so on. A monitoring query of interest might be query (8) from Example 2: find the authors of papers that were submitted more than two years ago but have not been accepted yet.

To query temporal data under domain-centric ontologies, CQs have been extended to temporalised CQs in which LTL operators can be applied to CQs. Thus, queries in *LTL*-CQ are defined by the following grammar:

$$\psi \quad ::= \quad \varphi \quad | \quad \neg\psi \quad | \quad \psi_1 \wedge \psi_2 \quad | \quad \psi_1 \,\mathcal{S}\, \psi_2 \quad | \quad \psi_1 \,\mathcal{U}\, \psi_2,$$

where φ is a CQ. Note that disjunction and the temporal connectives $\bigcirc_P$, $\bigcirc_F$, $\Box_P$, $\Box_F$, $\Diamond_P$, and $\Diamond_F$ can be used as abbreviations in *LTL*-CQs. Thus, *LTL*-CQ extends the set of queries in *LTL*-OMQs from Section 3.2 by admitting negation and applying *LTL* connectives to CQs rather than atomic queries; however, negation can only be applied to a formula all of whose free variables are answer variables of the query. Observe that *LTL*-CQs do not contain temporal variables. To evaluate an *LTL*-CQ, the user chooses a time point n for evaluation, typically the last time point of the temporal ABox representing the dynamic system to be monitored. Formally, *LTL*-CQs ψ can be translated into 2-FOL(<) formulas with a single temporal variable t and any number of domain variables as follows (we only give the translation for CQs, the extension to general *LTL*-CQs is defined in the same way as the extension of $\cdot^\sharp$ to temporalised concepts in Section 3.2):

$$(A_k(x))^\flat = A_k(x,t), \qquad (P_k(x,y))^\flat = P_k(x,y,t),$$
$$(\varphi_1 \wedge \varphi_2)^\flat = \varphi_1^\flat \wedge \varphi_2^\flat, \qquad (\exists x\, \varphi)^\flat = \exists x\, \varphi^\flat.$$

Table 3 Combined and data complexity of LTL-CQ answering with various DLs over $\mathbb{N}$.

DL	combined/data complexity		
	no rigid	rigid concepts	rigid concepts & roles
$\textit{DL-Lite}^{[\mathcal{H}}_{core\mid horn}$ [27]	PSPACE/NC^1		
$\textit{DL-Lite}_{krom\mid bool}$ [27]	EXPTIME/coNP	coNEXPTIME/coNP	2EXPTIME/in EXPTIME
$\textit{DL-Lite}^{\mathcal{H}}_{krom\mid bool}$ [27]	2EXPTIME/coNP		2EXPTIME/in EXPTIME
$\mathcal{EL}$ [28]	PSPACE/PTIME	PSPACE/coNP	coNEXPTIME/coNP
$\mathcal{ALC}$ [15]	EXPTIME/coNP	coNEXPTIME/coNP	2EXPTIME/in EXPTIME

Now, given a DL ontology $\mathcal{O}$, a temporal ABox $\mathcal{A}$, a time point $n \in [\min \mathcal{A}, \max \mathcal{A}]$, an LTL-CQ $\psi(\boldsymbol{x})$, a tuple $\boldsymbol{a}$ in $\mathcal{A}$, and $T \in \{\mathbb{Z}, \mathbb{N}, [\min \mathcal{A}, \max \mathcal{A}]\}$, one is interested in whether $(\boldsymbol{a}, n)$ is a certain answer to $(\mathcal{O}, \psi^\flat(\boldsymbol{x}, t))$ over $\mathcal{A}$ and T, or $\mathcal{O}, \mathcal{A} \models_T \psi^\flat(\boldsymbol{a}, n)$ in symbols.

▶ **Example 11.** Query (8) cannot be expressed in LTL-CQ. Its natural formalisation using temporal operators is the following

$$\boldsymbol{q}(x) = \exists y\big(\mathsf{authorOf}(x,y) \land \mathsf{UnderSubmission}(y) \land \bigcirc_P^{24}\mathsf{UnderSubmission}(y)\big),$$

but the quantifier $\exists y$ is applied to a temporalised formula which is not allowed in LTL-CQ. By regarding y as an answer variable and considering instead

$$\boldsymbol{q}(x,y) = \mathsf{authorOf}(x,y) \land \mathsf{UnderSubmission}(y) \land \bigcirc_P^{24}\mathsf{UnderSubmission}(y),$$

one obtains an LTL-CQ. Query (6) from Example 2 can be formulated as an LTL-CQ as follows:

$$\boldsymbol{q}(x) = \mathsf{Accepted}(x) \land \bigcirc_P\mathsf{UnderSubmission}(x) \land \bigcirc_P^{13}\mathsf{UnderSubmission}(x).$$

Table 3 summarises the known results [27, 28, 15] on the data and combined complexity of LTL-CQ evaluation mediated by domain-centric DL ontologies for $\mathbb{N}$-models (thus, for the evaluation problem $\mathcal{O}, \mathcal{A} \models_{\mathbb{N}} \psi^\flat(\boldsymbol{a}, n)$). It is not difficult to show the same upper bounds for ABox-fitting models, and we conjecture that the same lower bounds hold for ABox-fitting models as well. We also conjecture that the same results hold for $\mathbb{Z}$-models. The proofs generalise the propositional abstraction method employed in the analysis of the complexity of the satisfiability problem for $\mathcal{ALC}$-LTL ontologies. In fact, since LTL-CQs are closed under negation, the upper bounds in Table 1 can be proved by a straightforward reduction using the upper bounds for combined complexity in Table 3. It is of interest to observe that even for basic *DL-Lite* dialects, and without rigid concepts and roles one does not obtain FO-rewritability (because the problem is NC^1-hard), which is caused by negation in LTL-CQs. In contrast, query evaluation for $\mathcal{EL}$ without right concept and roles is still in PTIME in data complexity.

The complexity landscape presented in Table 3, has been further extended to more expressive description logics, in particular, containing subroles and transitive roles: the results for those cases are essentially the same as for $\mathcal{ALC}$ [16, 17].

The rewritability properties of OMQs using LTL-CQs are investigated by Borgwardt et al. [25, 26], where the focus is on query evaluation for ABox-fitting models. If no negation is present in an LTL-CQs $\boldsymbol{q}(\boldsymbol{x})$ and the ontology $\mathcal{O}$ is in $\textit{DL-Lite}_{core}$ without rigid symbols, then LTL-CQ rewriting $\boldsymbol{q}'(\boldsymbol{x})$ of $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$ can be obtained by simply replacing any non-temporal CQ $\varphi(\boldsymbol{x})$ in $\boldsymbol{q}(\boldsymbol{x})$ by the UCQ-rewriting of the non-temporal OMQ $(\mathcal{O}, \varphi(\boldsymbol{x}))$. A general transfer theory is developed [26] with the aim of showing that for a large class of domain-centric ontology languages rewritability (as well as combined rewritability [57]) is preserved under moving from non-temporal queries (such as CQs) to temporalised queries (such as *LTL*-CQs without negation).

Another major concern of research on the use of *LTL*-CQs in monitoring applications is the question whether it is possible to avoid storing the whole sequence $\mathcal{A}_0, \dots, \mathcal{A}_n$ to compute the certain answers to a given *LTL*-CQ at time-point n but instead keeping only a tail $\mathcal{A}_{n-b}, \dots, \mathcal{A}_n$ of the data. A variety of results in this direction have been obtained [26]. One approach is based on a classical separation result stating that, for every *LTL*-formula, there exists an *LTL*-formula without past-operators, which is equivalent to the original formula at the time-point 0 in $\mathbb{N}$-models [44].

Another approach is to add temporal connectives to the query language while keeping a standard atemporal ontology language [68, 69]. In the *streaming data* scenario, the relevant slices of the temporal data (i.e., the finite data history in the form of a sequence of ABoxes to be considered by the query) are specified with a *window* operator using a *sliding* parameter that determines the rate at which snapshots of the data are taken, and a *width* parameter that fixes the size on the window/history. This approach is realised in the Stream-Temporal Query Language STARQL [70]. Soylu et al. [83] have shown how the evaluation of STARQL queries is possible using standard SQL engines and report on the performance.

4 Combinations

In many cases, neither a domain nor a time-centric ontology language suffices, but some combination of them is needed. Designing combinations with good computational properties is notoriously difficult as the two-dimensional structure of temporal data makes it rather straightforward to encode the behaviour of Turing machines for even seemingly inexpressive languages. Thus, straightforward language combinations are often undecidable. In fact, only under rather intricate restrictions decidability is preserved [45, 54]. In this survey, our main concern is not decidability, but much stronger conditions such as tractability of OMQ evaluation and rewritability into 2-FOL. It should thus be clear that the interaction between temporal and DL constructors has to be pretty much restricted to obtain algorithmically well behaved combinations. In this section, we discuss three recent approaches to address this problem.

4.1 A 2-FOL(<)-Rewritable Temporal Extension of DL-Lite

Artale et al. [12] begin with the observation that, for any ontology $\mathcal{O}$, if one wants all OMQs based on $\mathcal{O}$ to be 2-FOL(<)-rewritable, then one has to ensure that $\mathcal{O}$ is materialisable (in the sense that, for any temporal ABox $\mathcal{A}$ consistent with $\mathcal{O}$, there exists a model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{O}$ that gives exactly the certain answers to any OMQ with $\mathcal{O}$). Equivalently, one requires that no disjunction of CQs is entailed if none of it disjuncts is entailed. This excludes the use of the temporal operators $\Diamond_F$ and $\Diamond_P$ on the right-hand side of concept inclusions, as illustrated by the following example.

▶ **Example 12.** Let $\mathcal{O} = \{A \sqsubseteq \Diamond_F B\}$. Consider the two-sorted CQs

$$\begin{aligned} \boldsymbol{q}_1(x,t) &= \exists t' \left((t < t') \wedge C(x,t') \wedge B(x,t')\right), \\ \boldsymbol{q}_2(x,t) &= \exists t' \left((t < t') \wedge C(x,t') \wedge \exists t'' \left((t' < t'') \wedge B(x,t'')\right)\right). \end{aligned}$$

For $\mathcal{A} = \{A(a,0), C(a,1), D(a,2)\}$, either B occurs together with C (for example, at moment 1), or B occurs after the moment 1, and so $\mathcal{O}, \mathcal{A} \models_{\mathbb{Z}} \boldsymbol{q}_1(a,0) \vee \boldsymbol{q}_2(a,0)$ but $\mathcal{O}, \mathcal{A} \not\models_{\mathbb{Z}} \boldsymbol{q}_i(a,0)$ for $i = 1, 2$. One can use an encoding of 2+2-SAT [79] to show that there is a two-sorted CQ $\boldsymbol{q}$ such that evaluating $(\mathcal{O}, \boldsymbol{q})$ over $\mathbb{Z}$-models is in fact coNP-hard for data complexity (and thus, there is no rewriting).

The following combination of *DL-Lite* and *LTL* is then suggested so that it avoids non-materialisability by not admitting any temporal operators $\Diamond_P$ and $\Diamond_F$ on the right-hand side of concept or role inclusions. *Basic concepts* B, *temporalised concepts* C, *roles* R and *temporalised roles* S are defined by the following grammar:

$$B \ ::= \ A_i \ \mid \ \exists R.\top, \qquad C \ ::= \ B \ \mid \ C_1 \sqcap C_2 \ \mid \ \Diamond_F C \ \mid \ \Diamond_P C,$$
$$R \ ::= \ P_i \ \mid \ P_i^-, \qquad S \ ::= \ R \ \mid \ S_1 \sqcap S_2 \ \mid \ \Diamond_F S \ \mid \ \Diamond_P S;$$

note that $\exists R.\top$ can only contain a basic role because temporalised roles can contain $\Diamond_P$ and $\Diamond_F$ (which are not allowed to occur on the right-hand side of concept inclusions). The *concept* and *role inclusions* in $\textit{DL-Lite}^{\text{lhs}\Diamond}_{\textit{horn}}$ are of the form

$$C \sqsubseteq B, \qquad S \sqsubseteq R.$$

A $\textit{DL-Lite}^{\text{lhs}\Diamond}_{\textit{horn}}$ ontology is a finite set of inclusions in $\textit{DL-Lite}^{\text{lhs}\Diamond}_{\textit{horn}}$. The following ontology illustrates expressiveness of the language.

▶ **Example 13.** In the context of Example 1, $\textit{DL-Lite}^{\text{lhs}\Diamond}_{\textit{horn}}$ can represent all 2-FOL(<)-sentences except (3):

$$\exists \mathsf{publishedIn}.\top \sqcap \exists \mathsf{acceptedIn}.\top \sqsubseteq \bot, \tag{1'}$$
$$\Diamond_P \mathsf{publishedIn} \sqsubseteq \mathsf{publishedIn}, \tag{2'}$$
$$\Diamond_P \mathsf{acceptedIn} \sqcap \Diamond_F \mathsf{acceptedIn} \sqsubseteq \mathsf{acceptedIn}. \tag{4'}$$

Note that (2′) and (4′) are role inclusions expressing convexity (also known as existential rigidity) of publishedIn and acceptedIn, respectively. We can also say that authorOf is a rigid role: $\Diamond_P \Diamond_F \mathsf{authorOf} \sqsubseteq \mathsf{authorOf}$.

As the query language we take the obvious extension of single-sorted CQs to two-sorted CQs, 2-CQ(<), defined by the following grammar:

$$\varphi \ ::= \ A_k(x,t) \ \mid \ P_k(x_1,x_2,t) \ \mid \ (t_1 < t_2) \ \mid \ (t_1 = t_2) \ \mid \ \varphi_1 \wedge \varphi_2 \ \mid \ \exists x\, \varphi \ \mid \ \exists t\, \varphi.$$

The query language 2-CQ(<) is rather expressive allowing an arbitrary nesting of domain and temporal quantifiers as illustrated by the following example.

▶ **Example 14.** Assuming that authorOf is rigid and using the fact that UnderSubmission is convex, query (8) can now be expressed as follows (cf. Example 11):

$$\boldsymbol{q}(x,t) \ = \ \exists y \big(\mathsf{authorOf}(x,y,t) \wedge \exists t_1 \exists t_2 \dots \exists t_{24} \big((t_{24} < t_{23}) \wedge \dots \wedge (t_2 < t_1) \wedge (t_1 < t) \wedge \mathsf{UnderSubmission}(y, t_{24})\big) \wedge \mathsf{UnderSubmission}(y,t)\big),$$

On the other hand, unlike *LTL*-CQ, 2-CQ(<) does not allow the $\bigcirc_P, \bigcirc_F$ operators, and it is not known whether the addition of these operators to 2-CQ(<) will preserve rewritibility.

Using the fact that ontologies in $\textit{DL-Lite}^{\text{lhs}\Diamond}_{\textit{horn}}$ are materialisable, one can show that OMQs with $\textit{DL-Lite}^{\text{lhs}\Diamond}_{\textit{horn}}$ ontologies and two-sorted CQs are 2-FOL(<)-rewritable over $\mathbb{Z}$-models.

▶ **Example 15.** A 2-FOL(<)-rewriting for OMQ ($\{(4')\}$, $\mathsf{acceptedIn}(x,y,t)$) over $\mathbb{Z}$ is

$$\mathsf{acceptedIn}(x,y,t) \lor \\ \big[\exists t' \big((t' < t) \land \mathsf{acceptedIn}(x,y,t')\big) \land \exists t' \big((t' > t) \land \mathsf{acceptedIn}(x,y,t')\big)\big].$$

4.2 Towards a Classification for Temporal DL-Lite

A more systematic investigation into the data complexity and rewritability of OMQs based on temporal *DL-Lite* was launched by Artale et al. [8]; see also [59]. The considered languages are based on the time-centric ontology languages introduced in Section 3.2. Thus, in contrast to $\textit{DL-Lite}^{\mathrm{lhs}\Diamond}_{horn}$, the operators $\Diamond_P$ and $\Diamond_F$ do not occur explicitly in ontologies but, instead, the basic temporal operators are $\bigcirc_P$, $\bigcirc_F$, $\Box_P$, and $\Box_F$. Now, temporal operators can occur both on the left- and right-hand side of concept and role inclusions. Formally, *basic concepts* B, *temporalised concepts* C, *roles* R and *temporalised roles* S are defined by the following grammar:

$$B \ ::= \ A_i \ \mid \ \exists R.\top, \qquad C \ ::= \ B \ \mid \ \Box_F C \ \mid \ \Box_P C \ \mid \ \bigcirc_F C \ \mid \ \bigcirc_P C$$
$$R \ ::= \ P_i \ \mid \ P_i^-, \qquad S \ ::= \ R \ \mid \ \Box_F S \ \mid \ \Box_P S \ \mid \ \bigcirc_F S \ \mid \ \bigcirc_P S.$$

Concept and *role inclusions* in normal form are as follows:

$$C_1 \sqcap \dots \sqcap C_k \ \sqsubseteq \ C_1' \sqcup \dots \sqcup C_n \qquad \text{and} \qquad S_1 \sqcap \dots \sqcap S_k \ \sqsubseteq \ S_1' \sqcup \dots \sqcup S_n'.$$

The next example shows how concept and role inclusions in $\textit{DL-Lite}^{\mathrm{lhs}\Diamond}_{horn}$ can be expressed using the operators $\Box_P$ and $\Box_F$.

▶ **Example 16.** Role inclusion $(2')$ from Example 13 can equivalently be expressed using $\Box_F$:

$$\mathsf{publishedIn} \sqsubseteq \Box_F \mathsf{publishedIn}. \tag{2''}$$

Note that $\Diamond_P$ on the left-hand side is replaced by $\Box_F$ on the right-hand side. To express $(4')$, however, fresh role names $\mathsf{acceptedInF}$ and $\mathsf{acceptedInP}$ are required, and the following three role inclusions are, in fact, a model conservative extension of $(4')$:

$$\mathsf{acceptedIn} \sqsubseteq \Box_F \mathsf{acceptedInF}, \tag{$4_1''$}$$
$$\mathsf{acceptedIn} \sqsubseteq \Box_P \mathsf{acceptedInP}, \tag{$4_2''$}$$
$$\mathsf{acceptedInF} \sqcap \mathsf{acceptedInP} \sqsubseteq \mathsf{acceptedIn}. \tag{$4_3''$}$$

It is not difficult to generalise this argument to arbitrary concept and role inclusions in $\textit{DL-Lite}^{\mathrm{lhs}\Diamond}_{horn}$.

We classify ontologies depending on the shape of their inclusions and the temporal operators in them similarly to the fragments of $LTL^{\Box\bigcirc}_{bool}$ in Section 3.2. For $\boldsymbol{c} \in \{\mathit{bool}, \mathit{horn}, \mathit{krom}, \mathit{core}\}$ and $\boldsymbol{o} \in \{\Box, \bigcirc, \Box\bigcirc\}$, we denote by $\textit{DL-Lite}^{\boldsymbol{o}}_{\boldsymbol{c}}$ the ontology language whose (concept and role) inclusions have the shape specified by $\boldsymbol{c}$ (for example, the *core* fragments only contain inclusions and disjointness axioms between temporalised concepts/roles, whereas $\boldsymbol{c} = \mathit{horn}$ allows, in addition, intersection $\sqcap$ to be applied to concepts/roles) and only use the (future and past) operators indicated in $\boldsymbol{o}$ (for example, $\boldsymbol{o} = \Box$ means that only $\Box_F$ and $\Box_P$ can be used).

The main ingredients of the query language are *positive temporal concepts* $\varkappa$ and *positive temporal roles* ϱ given by the grammars

$$\varkappa \ ::= \ \top \ \mid \ A_k \ \mid \ \exists R.\varkappa \ \mid \ \varkappa_1 \sqcap \varkappa_2 \ \mid \ \varkappa_1 \sqcup \varkappa_2 \ \mid \ \boldsymbol{op}_1\, \varkappa \ \mid \ \varkappa_1\, \boldsymbol{op}_2\, \varkappa_2,$$
$$\varrho \ ::= \ S \ \mid \ \varrho_1 \sqcap \varrho_2 \ \mid \ \varrho_1 \sqcup \varrho_2 \ \mid \ \boldsymbol{op}_1\, \varrho \ \mid \ \varrho_1\, \boldsymbol{op}_2\, \varrho_2,$$

Table 4 Data complexity and rewritability of positive OMQs over $\mathbb{Z}$.

	$\mathit{DL\text{-}Lite}_{c}^{\Box}$	$\mathit{DL\text{-}Lite}_{c}^{\bigcirc}$	$\mathit{DL\text{-}Lite}_{c}^{\Box\bigcirc}$
bool and *krom*		coNP-hard	
horn	NC^1-hard	NC^1-hard	
horn with monotone RIs	2-FOL($<$)		
core	2-FOL($<$)	2-FOL($<,+$)	?

where $\boldsymbol{op}_1 \in \{\bigcirc_F, \Diamond_F, \Box_F, \bigcirc_P, \Diamond_P, \Box_P\}$ and $\boldsymbol{op}_2 \in \{\mathcal{U}, \mathcal{S}\}$. Note that we can only use non-temporalised roles in $\exists R.\varkappa$. A *DL-Lite*$_{c}^{\boldsymbol{o}}$ *positive OMQ* is a pair of the form $\boldsymbol{Q}(x,t) = (\mathcal{O}, \varkappa(x,t))$ or $\boldsymbol{Q}(x,y,t) = (\mathcal{O}, \varrho(x,y,t))$, where $\mathcal{O}$ is a *DL-Lite*$_{c}^{\boldsymbol{o}}$ ontology, $\varkappa$ is a positive temporal concept and ϱ a positive temporal role (which can use all temporal operators, not necessarily only those in $\boldsymbol{o}$). If $\varkappa$ and ϱ are concept and role names, we refer to $\boldsymbol{Q}$ as an *atomic OMQ*.

Most of the data complexity and rewitability results reported in Table 4 are obtained by extending the constructions from *LTL*-OMQs. A surprising result here is that answering positive OMQ with *DL-Lite*$_{horn}^{\Box}$ ontologies turns out to be NC^1-hard (in contrast to $\mathit{LTL}_{horn}^{\Box}$, which is 2-FOL($<$)-rewritable). The class of *DL-Lite*$_{horn}^{\Box}$ ontologies with *monotone role inclusions* (the precise definition of which is too elaborate for this survey) includes, in particular, all *DL-Lite*$_{horn}^{\Box}$ ontologies whose role inclusions contain no $\Box_P$ and $\Box_F$ operators on the left-hand side. As demonstrated in Example 16, such ontologies are sufficient for encoding the language *DL-Lite*$_{horn}^{\mathrm{lhs}\Diamond}$ from Section 4.1. It is still an open problem whether OMQs with *DL-Lite*$_{core}^{\Box\bigcirc}$ are 2-FOL($<,+$)-rewritable or NC^1-hard.

Query (8) from Example 1 is expressible as a positive concept query if we assume that authorOf is a rigid role:

$$\boldsymbol{q}(x,t) \quad = \quad \exists \mathsf{authorOf}.(\mathsf{UnderSubmission} \sqcap \bigcirc_P^{24}\mathsf{UnderSubmission})(x,t)$$

(however, it is not expressible otherwise). In general, the query language of positive temporal concepts and roles is incomparable with 2-CQ($<$): the former, for example, allows union $\sqcup$ and $\Box_P/\Box_F$, but the latter contains not necessarily tree-shaped queries. It is still open whether the results of Table 4 hold for 2-CQ($<$) queries.

4.3 Temporal EL

The description logic $\mathcal{EL}$ is another tractable language, but since CQ answering in (atemporal) $\mathcal{EL}$ is PTime-complete, a more expressive than 2-FOL($<$) target language for rewritings in its temporal extension would be required. One candidate could be Datalog$_{1S}$, a decidable extension of Datalog with one unary *successor* function. Evaluating Datalog$_{1S}$ programs is known to be in ExpTime in combined complexity and PSpace-complete for data complexity [40].

Gutiérrez-Basulto et al. [49] considered a temporal extension $\mathcal{TEL}$ of $\mathcal{EL}$, in which concepts are defined by the following grammar:

$$C \quad ::= \quad A_k \quad | \quad \exists P_k.C \quad | \quad C_1 \sqcap C_2 \quad | \quad \Diamond_P C \quad | \quad \Diamond_F C \quad | \quad \bigcirc_P C \quad | \quad \bigcirc_F C.$$

(Note that $\mathcal{EL}$ has no role inverses, P_k^-.) Ontologies in $\mathcal{TEL}$ are finite sets of concept inclusions of the form $C_1 \sqsubseteq C_2$ (and contain no role inclusions). In terms of expressivity, observe that the 'concept' analogues ($2'$) and ($4'$) of sentences (2) and (4) in Example 1 belong to $\mathcal{TEL}$

(note that (1) strictly speaking does not belong to $\mathcal{TEL}$, but such an extension would be straightforward). Rigid concepts are also expressible in $\mathcal{TEL}$, and the language has rigid roles.

As the query language, Gutiérrez-Basulto et al. [49] chose atomic queries of the form $A(x,t)$. We mention, however, that queries (6), (7) and (8) can all be defined as $\mathcal{TEL}$-concepts in the ontology: for example,

$$\exists \mathsf{authorOf}.\big(\mathsf{UnderSubmission} \sqcap \bigcirc_P^{24}\mathsf{UnderSubmission}\big) \sqsubseteq Q, \qquad (8')$$

and then Q could be used as an atomic query.

Answering atomic OMQs in the full $\mathcal{TEL}$ turns out to be undecidable (here and below, all the results over $\mathbb{Z}$-models), but this is essentially due to the $\Diamond_P/\Diamond_F$ operators on the right-hand side of concept inclusions. In the fragment with only $\Diamond_P/\Diamond_F$ and only on the left-hand side of concept inclusions (like the language in Section 4.1), which is similar to the *inflationary* DATALOG$_{1S}$ [39], query answering is PTIME-complete for both data and combined complexity.

The fragment $\mathcal{TEL}^{\bigcirc}$ of $\mathcal{TEL}$ that uses only $\bigcirc_P/\bigcirc_F$ operators can express (as a model conservative extension) all axioms of the inflationary $\mathcal{TEL}$. For example, the concept analogue of (4′) (expressing convexity) can be encoded using two additional concept names and the following concept inclusions:

$$\mathsf{Accepted} \sqsubseteq \bigcirc_F \mathsf{AcceptedInF}, \qquad \mathsf{AcceptedInF} \sqsubseteq \bigcirc_F \mathsf{AcceptedInF},$$
$$\mathsf{Accepted} \sqsubseteq \bigcirc_P \mathsf{AcceptedInP}, \qquad \mathsf{AcceptedInP} \sqsubseteq \bigcirc_P \mathsf{AcceptedInP},$$
$$\mathsf{AcceptedInP} \sqcap \mathsf{AcceptedInF} \sqsubseteq \mathsf{AcceptedInF};$$

see also ($4''_1$)–($4''_3$). It is not known whether query answering in the full $\mathcal{TEL}^{\bigcirc}$ is decidable. However, it is PTIME-complete for data and PSPACE-complete for combined complexity in its sublanguage without rigid roles, and PSPACE-complete in data and in EXPTIME for combined complexity in the sublanguage where rigid roles can only occur on the left-hand side of concept inclusions. These results are proved by translating the query answering problem into DATALOG$_{1S}$. Moreover, acyclic $\mathcal{TEL}$-OMQs can be rewrtitten into 2-FOL($<,+$), and the evaluation problem for such OMQs is in PTIME in combined complexity. Making the ontology acyclic in one of the dimensions only (either time or DL), gives the following results: for temporally acyclic ontologies, which include all atemporal $\mathcal{EL}$ ontologies, it is PTIME-complete in both combined and data complexity; for DL-acyclic ontologies, OMQ answering is non-elementary for combined complexity but NC^1-complete for data complexity.

5 Interval-Based Temporal Ontology-Mediated Query Answering

In the ontology and query languages considered in Sections 2–4, time was assumed to be point-based and discrete. It is well-known, however, that both features may cause difficulties for modelling certain application domains.

We begin with the view of time as intervals, that is, sequences of points. The standard way of storing temporal information in databases is by attaching a validity time *interval* to tuples. For example, a relational table EmployeeSalaries with columns EmployeeID, MonthlySalary, FromTime and ToTime contains tuples such as (e007, £3000, 01/01/2008, 05/01/2014). The simplest and most intuitive way of representing this information in the point-based setting is to stipulate that such a tuple is a shorthand for the sequence of tuples

$$(\mathsf{e007}, \pounds 3000, 01/01/2008),\ (\mathsf{e007}, \pounds 3000, 02/01/2008), \dots, (\mathsf{e007}, \pounds 3000, 05/01/2014)$$

provided that it is known *a priori* that a day is the minimal unit of time required in the application. Such a conversion of intervals to points, performed explicitly or implicitly by a query-answering engine, is known to cause an exponential blow-up to the worst-case execution time (since timestamps are encoded in binary, see, e.g., [4]). At the same time, this conversion is not always sound. Consider, for instance, the tuple $(\mathsf{tb007}, 1500, 11{:}25, 11{:}29)$ from a table for a turbine performance monitoring system with columns TurbineID, AverageRotationSpeed, FromTime and ToTime. Clearly, the tuple $(\mathsf{tb007}, 1500, 11{:}27)$ would not make much sense since 1500 is the *average* rotation speed over the given interval. These examples suggest replacing the point-based setting with an interval-based view of time, where the truth-values of predicates are assigned to time intervals rather than points.

We discuss two interval-based temporal logics and related formalisms for ontology-mediated query answering.

5.1 Halpern-Shoham Interval Temporal Logic

In the interval temporal logic $\mathcal{HS}$ introduced by Halpern and Shoham [53], formulas are interpreted over the set of intervals of any given linear order. More precisely, let $\mathfrak{T} = (T, \leq)$ be a linear order, that is, $\leq$ is a reflexive, transitive, antisymmetric and connected binary relation on T. (As usual, $x < y$ is a shortcut for '$x \leq y$ and $x \neq y$'.) For example, the rationals $(\mathbb{Q}, \leq)$ and reals $(\mathbb{R}, \leq)$ are dense linear orders, while the integers $(\mathbb{Z}, \leq)$ and the natural numbers $(\mathbb{N}, \leq)$ are discrete ones. By an *interval in* $\mathfrak{T}$ we mean any ordered pair $\langle i, j\rangle$ such that $i \leq j$, and denote by $\mathsf{int}(\mathfrak{T})$ the set of all intervals in $\mathfrak{T}$. Note that $\mathsf{int}(\mathfrak{T})$ contains all the *punctual intervals* of the form $\langle i, i\rangle$, which is often referred to as the *non-strict* semantics. Under the *strict* semantics adopted by Allen [2], punctual intervals are disallowed.

Temporal ABoxes in the interval-based paradigm consist of assertions such as

$$A(a, \iota) \quad \text{and} \quad S(a, b, \iota)$$

saying that, respectively, $A(a)$ and $S(a, b)$ hold true at the interval $\iota \in \mathsf{int}(\mathfrak{T})$. For example, an ABox containing timetabling data of a summer school can have the assertions:

$$\mathsf{TutorialDay}(\text{'Semantic Web'}, \langle 07/26/2017\ 08{:}00, 07/26/2005\ 16{:}00\rangle),$$
$$\mathsf{LunchBreak}(\text{'Semantic Web'}, \langle 07/26/2017\ 11{:}30, 07/26/2005\ 12{:}30\rangle).$$

A *de facto* standard way of defining a language expressing statements (constraints) over intervals is by incorporating Allen's [2] interval relations defined as shown in Fig. 2.[2]

Since all of these relations are irreflexive, we refer to this definition as the *irreflexive semantics*. As an alternative, the *reflexive semantics* is obtained by replacing each $<$ in Fig. 2 with $\leq$. We write $\mathfrak{T}(\leq)$ or $\mathfrak{T}(<)$ to indicate that the semantics is reflexive or, respectively, irreflexive.

Equipped with Allen's relations, we can express, for example, the query asking for the names of the tutorials that are followed by a lunch break and the times of those lunch breaks:

$$\boldsymbol{q}(x, \chi) \quad = \quad \exists \rho \, \big(\mathsf{LunchBreak}(x, \chi) \land \mathsf{TutorialDay}(x, \rho) \land \bar{\mathsf{A}}(\chi, \rho)\big),$$

where χ and ρ are variables ranging over time intervals. Over the ABox with the two statements above, this query would not return any answers because the lunch break is in the middle of the Semantic Web tutorial (D) rather than after it ($\bar{\mathsf{A}}$). In fact, such queries

[2] It is to be noted that there are two slightly different versions of A and $\bar{\mathsf{A}}$ in the literature.

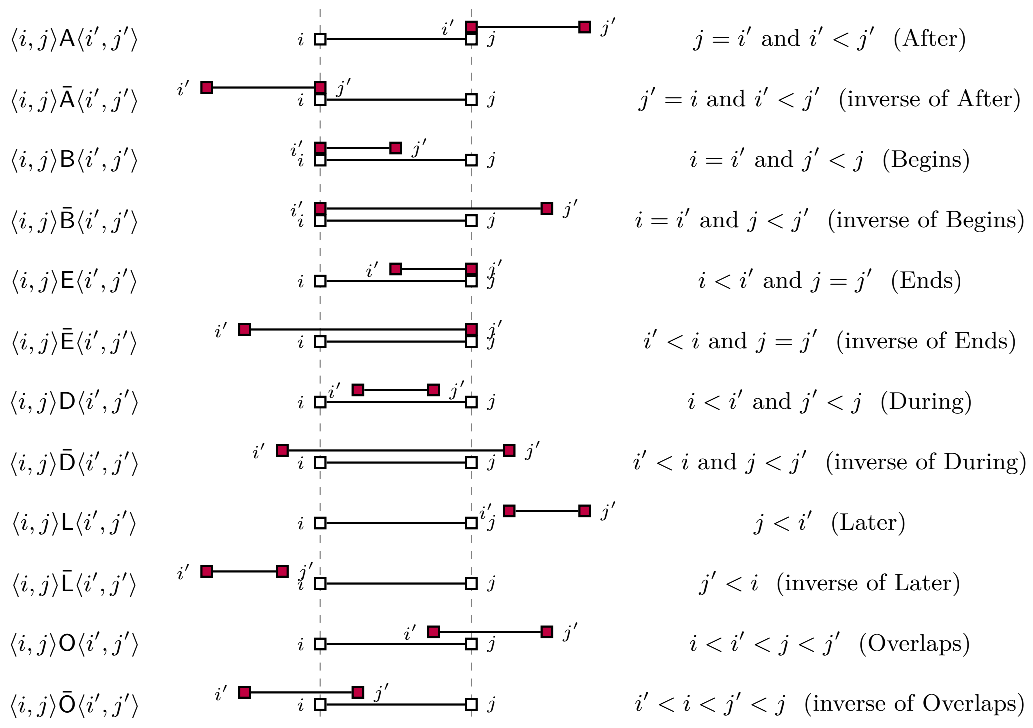

$\langle i,j\rangle \mathsf{A} \langle i',j'\rangle$ $j = i'$ and $i' < j'$ (After)
$\langle i,j\rangle \bar{\mathsf{A}} \langle i',j'\rangle$ $j' = i$ and $i' < j'$ (inverse of After)
$\langle i,j\rangle \mathsf{B} \langle i',j'\rangle$ $i = i'$ and $j' < j$ (Begins)
$\langle i,j\rangle \bar{\mathsf{B}} \langle i',j'\rangle$ $i = i'$ and $j < j'$ (inverse of Begins)
$\langle i,j\rangle \mathsf{E} \langle i',j'\rangle$ $i < i'$ and $j = j'$ (Ends)
$\langle i,j\rangle \bar{\mathsf{E}} \langle i',j'\rangle$ $i' < i$ and $j = j'$ (inverse of Ends)
$\langle i,j\rangle \mathsf{D} \langle i',j'\rangle$ $i < i'$ and $j' < j$ (During)
$\langle i,j\rangle \bar{\mathsf{D}} \langle i',j'\rangle$ $i' < i$ and $j < j'$ (inverse of During)
$\langle i,j\rangle \mathsf{L} \langle i',j'\rangle$ $j < i'$ (Later)
$\langle i,j\rangle \bar{\mathsf{L}} \langle i',j'\rangle$ $j' < i$ (inverse of Later)
$\langle i,j\rangle \mathsf{O} \langle i',j'\rangle$ $i < i' < j < j'$ (Overlaps)
$\langle i,j\rangle \bar{\mathsf{O}} \langle i',j'\rangle$ $i' < i < j' < j$ (inverse of Overlaps)
i
j
i'
j'

Figure 2 Allen's interval relations under the irreflexive semantics.

are supported by the SQL:2011 standard [60], which adopts the strict semantics for some of Allen's relations and the non-strict for others.

The Halpern-Shoham interval temporal logic $\mathcal{HS}$ [53] is a propositional modal logic with diamond operators of the form $\langle \mathsf{R} \rangle$ for Allen's interval relations R. The propositional variables of $\mathcal{HS}$ are interpreted by sets of intervals of a given linear order $\mathfrak{T}$ where they are assumed to hold true, and a formula $\langle \mathsf{R} \rangle \varphi$ is true at an interval $\iota \in \mathsf{int}(\mathfrak{T})$ iff φ is true at some interval ι' such that $\iota \mathrel{\mathsf{R}} \iota'$. This semantics can be extended to first-order or description logic in a natural way. For example, we can give the following definition of 'a morning session' in a DL version of $\mathcal{HS}$:

$$[\mathsf{U}]\big(\langle \bar{\mathsf{B}} \rangle \mathsf{TutorialDay} \sqcap \langle \mathsf{A} \rangle \mathsf{LunchBreak} \sqsubseteq \mathsf{MorningSession}\big), \tag{10}$$

where U is the *universal* relation between intervals, and $[\mathsf{U}]$ means 'at all intervals'. In English, this axiom says that an object d is a $\mathsf{MorningSession}$ in an interval ι – $\mathsf{MorningSession}(d, \iota)$ in symbols – if there is an interval ι' such that $\iota \mathrel{\bar{\mathsf{B}}} \iota'$ and $\mathsf{TutorialDay}(d, \iota')$, and also there is an interval ι'' such that $\iota \mathrel{\mathsf{A}} \iota''$ and $\mathsf{LunchBreak}(d, \iota'')$. The query $\boldsymbol{q}(x, \chi) = \mathsf{MorningSession}(x, \chi)$ mediated by ontology $\{(10)\}$ over the ABox above would return the certain answer

$$(\text{'Semantic Web'}, \langle 07/26/2017\ 08{:}00, 07/26/2017\ 11{:}30\rangle)$$

meaning that Semantic Web in the time slot between 8:00 and 11:30 is a morning session.

The elegance and expressive power of $\mathcal{HS}$ have attracted attention of many areas of computer science and AI. However, promising applications have been hampered by the fact, already discovered by Halpern and Shoham [53], that $\mathcal{HS}$ is highly undecidable (for example, validity over $\mathbb{Z}$ and $\mathbb{R}$ is Π^1_1-hard). For recent studies of the computational complexity of reasoning with various fragments of $\mathcal{HS}$, we refer the reader to [34, 33, 63, 1, 32] and references therein.

A tractable fragment of $\mathcal{HS}$ and its DL-Lite and datalog extensions that can be used for temporal ontology-mediated query answering have recently been suggested [11, 58]. We briefly discuss these two formalisms in the remainder of Section 5.

5.2 Description logic $\mathcal{HS}\text{-}Lite^{\mathcal{H}}_{horn}$

The language of $\mathcal{HS}\text{-}Lite^{\mathcal{H}}_{horn}$ is an extension of $DL\text{-}Lite^{\mathcal{H}}_{horn}$ [6]. It contains *individual names* $a_1, a_2, \dots$, *concept names* $A_1, A_2, \dots$, and *role names* $P_1, P_2, \dots$. *Basic roles* R, *basic concepts* B, *temporal roles* S and *temporal concepts* C are given by the following grammar:

$$B ::= \top \mid A_k \mid \exists R.\top, \qquad C ::= B \mid [\mathsf{R}]C \mid \langle \mathsf{R}\rangle C,$$
$$R ::= P_k, \mid P_k^- \qquad S ::= R \mid [\mathsf{R}]S \mid \langle \mathsf{R}\rangle S,$$

where R is one of Allen's interval relations or the universal relation U and $[\mathsf{R}]$ is the dual of $\langle \mathsf{R}\rangle$, that is, $[\mathsf{R}]\varphi$ holds at an interval ι iff φ holds at all intervals ι' such that $\iota \,\mathsf{R}\, \iota'$. An $\mathcal{HS}\text{-}Lite^{\mathcal{H}}_{horn}$ TBox is a finite set of *concept* and *role inclusions* and *disjointness constraints* of the form

$$C_1 \sqcap \dots \sqcap C_k \sqsubseteq C^+, \qquad C_1 \sqcap \dots \sqcap C_k \sqsubseteq \bot,$$
$$S_1 \sqcap \dots \sqcap S_k \sqsubseteq S^+, \qquad S_1 \sqcap \dots \sqcap S_k \sqsubseteq \bot,$$

where C^+ and S^+ denote temporal concepts and roles *without* occurrences of diamond operators $\langle \mathsf{R}\rangle$; cf. Section 4.2. (The consequences of allowing $\langle \mathsf{R}\rangle$ on the right-hand side of inclusions will be discussed in the sequel). An $\mathcal{HS}\text{-}Lite^{\mathcal{H}}_{horn}$ ABox is a finite set of atoms of the form $A_k(a, \iota)$ and $P_k(a, b, \iota)$, where ι is an interval of the linear order in question.

It was shown [11] that answering atomic OMQs in $\mathcal{HS}\text{-}Lite^{\mathcal{H}}_{horn}$ is PTIME-complete for both combined and data complexity provided that either concept inclusions contain no $\exists R.\top$ on the right-hand side, or role inclusions contain no temporal relations apart from U. Originally, the result was shown for $(\mathbb{Z}, \leq)$ only; however, in the light of later findings [32], it can also be extended to any *dense* linear order $(T, \leq)$ and $(T, <)$. A failure to prove decidability for discrete linear orders under the irreflexive semantics, say, $(\mathbb{Z}, <)$, even for the language without roles, led to a separate systematic investigation of the propositional fragment $\mathcal{HS}^{\Box}_{horn}$ of $\mathcal{HS}$. Formally, this fragment can be defined as pairs of the form $(\mathcal{O}, \{A(a, \iota)\})$, where $\mathcal{O}$ is an $\mathcal{HS}\text{-}Lite^{\mathcal{H}}_{horn}$ TBox without any occurrence of role names. The satisfiability problem for $\mathcal{HS}^{\Box}_{horn}$ was shown [32] to be undecidable for unbounded discrete linear orders such as $(\mathbb{N}, <)$ and $(\mathbb{Z}, <)$ under the *irreflexive* semantics (in contrast to PTIME-completeness for dense orders under any semantics). We illustrate the expressiveness of $\mathcal{HS}^{\Box}_{horn}$ over $(\mathbb{N}, <)$ by the following example.

▶ **Example 17.** Let $\mathcal{A} = \{A(\langle 0, 0\rangle)\}$ and $\mathcal{O}$ be an $\mathcal{HS}^{\Box}_{horn}$ ontology with the following axioms:

$$[\mathsf{E}]A \sqcap \langle \mathsf{E}\rangle \top \sqsubseteq A, \qquad [\mathsf{E}]C \sqcap \langle \mathsf{E}\rangle \top \sqsubseteq C, \qquad \langle \bar{\mathsf{E}}\rangle[\mathsf{B}][\bar{\mathsf{E}}]A \sqsubseteq C, \qquad \langle \bar{\mathsf{E}}\rangle[\mathsf{B}][\bar{\mathsf{E}}]C \sqsubseteq A,$$

Under the irreflexive semantics over $\mathbb{N}$, we have $\mathcal{O}, \mathcal{A} \models A(\iota)$, for any $\iota = \langle n, 2m\rangle$ with $n, m \in \mathbb{N}$; see Fig. 3, where intervals $\langle i, j\rangle$ are represented as the points (i, j) on the Euclidean plane. Under the reflexive semantics, we have $\mathcal{O}, \mathcal{A} \models A(\iota)$ for $\iota = \langle 0, 0\rangle$ only.

Furthermore, by admitting $\langle \mathsf{R}\rangle$-operators on the right-hand side of concept inclusions of $\mathcal{HS}^{\Box}_{horn}$, we make it undecidable under any semantics and any unbounded linear orders. In fact, this extended logic remains undecidable under the irreflexive semantics even when restricted to *binary* concept inclusions (that is, the *core* fragment).

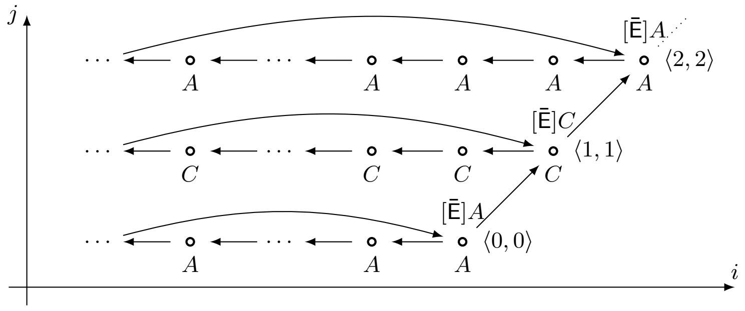


Figure 3 Deriving $A(\iota)$ from $\mathcal{O}$ and $\mathcal{A}$ in Example 17.

5.3 Multidimensional datalog$\mathcal{HS}_n^\Box$

The former of the two tractable fragments of $\mathcal{HS}$-*Lite*$_{horn}^{\mathcal{H}}$ mentioned above can be generalised in two directions [58]. First, we extend *DL-Lite*$_{horn}^{\mathcal{H}}$ TBoxes without $\exists$ on the right-hand side of concept inclusions to arbitrary datalog programs. Second, following [23], we extend the interval logic $\mathcal{HS}$ to a multidimensional hyperrectangle (or block) logic $\mathcal{HS}_n$. Let $\mathfrak{T} = (T, \lhd)$ be either $(\mathbb{Z}, \leq)$ or $(\mathbb{R}, <)$. (In fact, one can take any discrete order under the reflexive semantics and any dense order under the reflexive or irreflexive semantics.) Fix some $n \geq 1$ and a linear order $\mathfrak{T}_\ell = (T_\ell, \lhd_\ell)$ as above, for $1 \leq \ell \leq n$. A *hyperrectangle* in the n-dimensional space $\boldsymbol{\mathfrak{T}} = \prod_{\ell=1}^n \mathfrak{T}_\ell$ is any n-tuple $\boldsymbol{\iota} = (\iota_1, \dots, \iota_n)$ such that $\iota_\ell \in \mathsf{int}(\mathfrak{T}_\ell)$, for $1 \leq \ell \leq n$. The set of hyperrectangles in $\boldsymbol{\mathfrak{T}}$ is denoted by $\mathsf{hyp}(\boldsymbol{\mathfrak{T}})$. Given $\boldsymbol{\iota}, \boldsymbol{\kappa} \in \mathsf{hyp}(\boldsymbol{\mathfrak{T}})$ and an interval relation R, we write $\boldsymbol{\iota}\ \mathsf{R}_\ell\ \boldsymbol{\kappa}$ if $\iota_\ell\ \mathsf{R}\ \kappa_\ell$ and $\iota_i = \kappa_i$, for $i \neq \ell$.

A *data instance* (ABox), $\mathcal{A}$, is now a finite set of *facts* of the form $P(\boldsymbol{c}, \boldsymbol{\iota})$, where P is an m-ary predicate symbol, $\boldsymbol{c}$ an m-tuple of individual constants, for some $m \geq 0$, and $\boldsymbol{\iota} \in \mathsf{hyp}(\boldsymbol{\mathfrak{T}})$. This fact says that $P(\boldsymbol{c})$ is true in the hyperrectangle $\boldsymbol{\iota}$. We denote by $\mathsf{num}_\ell(\mathcal{A})$ the set of $i, j \in T_\ell$ with $\iota_\ell = \langle i, j\rangle$, for some $\boldsymbol{\iota}$ mentioned in $\mathcal{A}$, and by $\mathsf{int}(\mathcal{A})$ the set of $\langle i, j\rangle \in \mathsf{int}(\mathfrak{T}_\ell)$ with $i, j \in \mathsf{num}_\ell(\mathcal{A})$, for $1 \leq \ell \leq n$.

An *individual term*, τ, is an individual variable, x, or a constant, a. A datalog$\mathcal{HS}_n^\Box$ *program*, Π, is a finite set of *rules* of the form

$$A^+ \leftarrow A_1 \wedge \dots \wedge A_k, \qquad\qquad \bot \leftarrow A_1 \wedge \dots \wedge A_k, \tag{11}$$

where $k \geq 1$, each A_i is either an inequality $(\tau \neq \tau')$ with individual terms τ and τ' or defined by the grammar

$$A \quad ::= \quad P(\tau_1, \dots, \tau_m) \quad | \quad [\mathsf{R}]_\ell A \quad | \quad \langle \mathsf{R}\rangle_\ell A, \tag{12}$$

for an m-ary predicate P and individual terms τ_j, and A^+ does not contain any diamond operators $\langle \mathsf{R}\rangle_\ell$. As usual, the atoms $A_1, \dots, A_k$ constitute the *body* of the rule, while A^+ or $\bot$ its *head*. We also impose other standard datalog restrictions on datalog$\mathcal{HS}_n^\Box$ programs. (Clearly, we cannot allow $\langle \mathsf{R}\rangle_\ell$ in the heads as this would make our logic undecidable, as discussed above.)

An *interpretation*, $\mathfrak{M}$, for datalog$\mathcal{HS}_n^\Box$ programs is based on a *domain* $\Delta \neq \emptyset$ (for the individual variables and constants) and the space $\boldsymbol{\mathfrak{T}}$. For any m-ary predicate P, m-tuple $\boldsymbol{c}$ from Δ and $\boldsymbol{\iota} \in \mathsf{hyp}(\boldsymbol{\mathfrak{T}})$, $\mathfrak{M}$ specifies whether P is *true on* $\boldsymbol{c}$ *in* $\boldsymbol{\iota}$, in which case we write $\mathfrak{M}, \boldsymbol{\iota} \models P(\boldsymbol{c})$. Let $\mathfrak{d}$ be an *assignment* of elements of Δ to the individual variables (we adopt the standard name assumption: $\mathfrak{d}(a) = a$, for every individual constant a). We then set

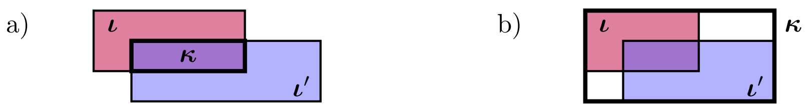


Figure 4 Configurations for Int and Cov in Example 18.

inductively:

$$\begin{array}{rll}
\mathfrak{M}, \boldsymbol{\iota} \models^{\mathfrak{d}} P(\boldsymbol{\tau}) & \text{iff} & \mathfrak{M}, \boldsymbol{\iota} \models P(\mathfrak{d}(\boldsymbol{\tau})), \qquad\qquad \mathfrak{M}, \boldsymbol{\iota} \not\models^{\mathfrak{d}} \bot,\\
\mathfrak{M}, \boldsymbol{\iota} \models^{\mathfrak{d}} \tau \neq \tau' & \text{iff} & \mathfrak{d}(\tau) \neq \mathfrak{d}(\tau'),\\
\mathfrak{M}, \boldsymbol{\iota} \models^{\mathfrak{d}} [\mathsf{R}]_\ell A & \text{iff} & \mathfrak{M}, \boldsymbol{\kappa} \models^{\mathfrak{d}} A \text{ for all } \boldsymbol{\kappa} \text{ with } \boldsymbol{\iota}\, \mathsf{R}_\ell\, \boldsymbol{\kappa},\\
\mathfrak{M}, \boldsymbol{\iota} \models^{\mathfrak{d}} \langle \mathsf{R}\rangle_\ell A & \text{iff} & \mathfrak{M}, \boldsymbol{\kappa} \models^{\mathfrak{d}} A \text{ for some } \boldsymbol{\kappa} \text{ with } \boldsymbol{\iota}\, \mathsf{R}_\ell\, \boldsymbol{\kappa}.
\end{array}$$

We say that $\mathfrak{M}$ *satisfies* Π *under* $\mathfrak{d}$ if

$$\mathfrak{M}, \boldsymbol{\iota} \models^{\mathfrak{d}} A \quad \text{whenever} \quad \mathfrak{M}, \boldsymbol{\iota} \models^{\mathfrak{d}} A_i \text{ for } 1 \leq i \leq k,$$

for all $\boldsymbol{\iota} \in \mathsf{hyp}(\boldsymbol{\mathfrak{T}})$ and all rules $A \leftarrow A_1 \land \dots \land A_k$ in Π. $\mathfrak{M}$ is a *model* of Π and $\mathcal{A}$ if it satisfies Π under every assignment, and $\mathfrak{M}, \boldsymbol{\iota} \models P(\boldsymbol{c})$, for every fact $P(\boldsymbol{c}, \boldsymbol{\iota})$ in $\mathcal{A}$. Π and $\mathcal{A}$ are *consistent* if they have a model.

▶ **Example 18.** Denote by $\langle \mathsf{Int}\rangle$ the binary modal operator such that $A\langle \mathsf{Int}\rangle A'$ holds at a hyperrectangle $\boldsymbol{\kappa}$ iff A holds at some $\boldsymbol{\iota}$, A' at some $\boldsymbol{\iota}'$, and $\boldsymbol{\kappa} = \boldsymbol{\iota} \cap \boldsymbol{\iota}'$. One can show that rules such as $B \leftarrow A\langle \mathsf{Int}\rangle A'$ are expressible as datalog$\mathcal{HS}_n^{\Box}$ programs. For example, for $n = 2$, there are $13^2 = 169$ different relative positions of two rectangles; see, e.g., [65, Fig. 4] for an illustration. Those configurations where the rectangles have non-empty intersection are encoded by datalog$\mathcal{HS}_n^{\Box}$ rules such as $B \leftarrow \langle \bar{\mathsf{E}}\rangle_1 \langle \bar{\mathsf{B}}\rangle_2 A \land \langle \bar{\mathsf{B}}\rangle_1 \langle \bar{\mathsf{E}}\rangle_2 A'$ for the configuration in Fig. 4a.

Similarly, one can express the rule $B \leftarrow A\langle \mathsf{Cov}\rangle A'$ such that $A\langle \mathsf{Cov}\rangle A'$ holds at $\boldsymbol{\kappa}$ iff $\boldsymbol{\kappa}$ is the smallest hyperrectangle containing some $\boldsymbol{\iota}$ with A and $\boldsymbol{\iota}'$ with A'; see Fig. 4b.

An *interval term*, ϑ, is either an interval or an *interval variable*. A *conjunctive query* (CQ) is a formula of the form $\boldsymbol{q}(\boldsymbol{x}, \boldsymbol{\chi}) = \exists \boldsymbol{x}' \exists \boldsymbol{\chi}'\, \Phi(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{\chi}, \boldsymbol{\chi}')$, where Φ is a conjunction of atoms $P(\boldsymbol{\tau}, \boldsymbol{\vartheta})$ for tuples $\boldsymbol{\tau}$ and $\boldsymbol{\vartheta}$ of individual and interval terms, respectively, and $\mathsf{R}(\vartheta, \vartheta')$, for an interval relation R, such that all individual and interval variables in Φ are from $\boldsymbol{x} \cup \boldsymbol{x}'$ and $\boldsymbol{\chi} \cup \boldsymbol{\chi}'$, respectively. A datalog$\mathcal{HS}_n^{\Box}$ program Π and a CQ $\boldsymbol{q}(\boldsymbol{x}, \boldsymbol{\chi})$ constitute an *ontology-mediated query* (OMQ) $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\chi}) = (\Pi, \boldsymbol{q}(\boldsymbol{x}, \boldsymbol{\chi}))$.

▶ **Example 19.** Suppose $\boldsymbol{\mathfrak{T}} = \mathfrak{T}_1 \times \mathfrak{T}_2$, where $\mathfrak{T}_1 = (\mathbb{Z}, \leq)$ represents time and $\mathfrak{T}_2 = (\mathbb{R}, <)$ temperature. Imagine that a turbine monitoring system is receiving from sensors a stream of data of the form $\mathsf{Blade}(\mathsf{ID140}, (\iota_1, \iota_2))$, where $\mathsf{ID140}$ is a blade ID and $\iota_2 \in \mathsf{int}(\mathbb{R}, <)$ is the observed temperature range during the time interval $\iota_1 \in \mathsf{int}(\mathbb{Z}, \leq)$. Then the rule

$$\mathsf{TemperatureRise}(x) \leftarrow \langle \bar{\mathsf{A}}\rangle_1 \langle \bar{\mathsf{O}}\rangle_2 \mathsf{Blade}(x) \land \langle \mathsf{A}\rangle_1 \langle \mathsf{O}\rangle_2 \mathsf{Blade}(x)$$

says that the temperature of blade x is rising over a rectangle (ι_1, ι_2) if $\mathsf{Blade}(x, (\iota_1^-, \iota_2^-))$ and $\mathsf{Blade}(x, (\iota_1^+, \iota_2^+))$ hold at some (ι_1^-, ι_2^-) and (ι_1^+, ι_2^+) located as shown in Fig. 5.

The temperature drop is defined analogously:

$$\mathsf{TemperatureDrop}(x) \leftarrow \langle \bar{\mathsf{A}}\rangle_1 \langle \mathsf{O}\rangle_2 \mathsf{Blade}(x) \land \langle \mathsf{A}\rangle_1 \langle \bar{\mathsf{O}}\rangle_2 \mathsf{Blade}(x).$$

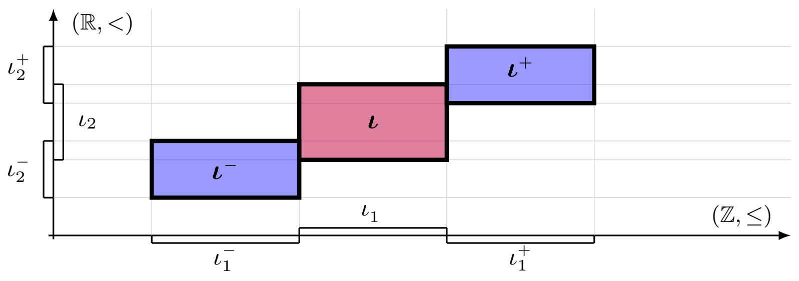


Figure 5 Rule for TemperatureRise in Example 19.

To find the blades x and the time intervals χ such that the temperature of x was rising before χ, reaching 1500° in χ, and dropping after that, we can use the following CQ:

$$\exists\rho\exists\chi^-\exists\rho^-\exists\chi^+\exists\rho^+\big[\mathsf{Blade}(x,(\chi,\rho))\land\mathsf{TemperatureRise}(x,(\chi^-,\rho^-))\land\mathsf{A}(\chi^-,\chi)\land$$
$$\mathsf{TemperatureDrop}(x,(\chi^+,\rho^+))\land\mathsf{A}(\chi,\chi^+)\land\mathsf{O}(\rho,\langle1500,1600\rangle)\big].$$

Let $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\chi})=(\Pi,\boldsymbol{q}(\boldsymbol{x},\boldsymbol{\chi}))$ be an OMQ and $\mathcal{A}$ a data instance. A *certain answer to* $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\chi})$ *over* $\mathcal{A}$ is any pair $(\boldsymbol{a},\boldsymbol{\delta})$ of a tuple $\boldsymbol{a}$ of individual constants in $\mathcal{A}$ and a tuple $\boldsymbol{\delta}$ from $\mathsf{int}(\mathcal{A})$ of the same length as $\boldsymbol{x}$ and $\boldsymbol{\chi}$, respectively, satisfying the following condition: for every model $\mathfrak{M}$ of Π and $\mathcal{A}$, there is a map h of the individual terms in $\boldsymbol{q}$ to Δ and the interval terms to $\bigcup_\ell \mathsf{int}(\mathfrak{T}_\ell)$ preserving constants and dimensions such that $h(\boldsymbol{x})=\boldsymbol{a}$, $h(\boldsymbol{\chi})=\boldsymbol{\delta}$, and

$$\mathfrak{M},h(\boldsymbol{\vartheta})\models P(h(\boldsymbol{\tau})),\text{ for every atom } P(\boldsymbol{\tau},\boldsymbol{\vartheta}) \text{ in } \boldsymbol{q},\qquad\text{and}$$
$$\mathsf{R}(h(\vartheta),h(\vartheta'))\text{ holds in the corresponding }\mathfrak{T}_\ell,\text{ for every atom }\mathsf{R}(\vartheta,\vartheta')\text{ in }\boldsymbol{q}.$$

The problem of checking whether $(\boldsymbol{a},\boldsymbol{\delta})$ is a certain answer to $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\chi})$ over $\mathcal{A}$ is shown to be PTime-complete for data complexity and ExpTime-complete for combined complexity; for propositional datalog$\mathcal{HS}_n^\Box$ programs, the problem is PTime-complete for combined complexity [58]. Any datalog$\mathcal{HS}_n^\Box$ OMQ $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\chi})=(\Pi,\boldsymbol{q}(\boldsymbol{x},\boldsymbol{\chi}))$ can also be rewritten to a standard polynomial-size datalog program $\Pi^\dagger$ with a goal $G(\boldsymbol{x},\boldsymbol{\chi})$ such that, for any data instance $\mathcal{A}$, a tuple $(\boldsymbol{a},\boldsymbol{\delta})$ is a certain answer to $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{\chi})$ over $\mathcal{A}$ iff $\Pi^\dagger,\mathcal{A}\models G(\boldsymbol{a},\boldsymbol{\delta})$. We refer the reader to [58] for some initial experiments on the expressive power and efficiency of ontology-based query answering with datalog$\mathcal{HS}_n^\Box$ using two real-world scenarios.

6 Dense Time and Metric Temporal Logics

The problems with discreteness of time are related to the fact that a minimal unit of time in some cases may be unknown or inconvenient to use. Suppose, for example, that the time unit is set to be 'a minute' for the turbine performance monitoring system with timestamped data of the form $(\mathsf{tb007},1500,11{:}27)$. When a newer model of turbine is installed with measurements taken at the rate of one per second, we shall have to redefine the minimal unit accordingly. This means, in particular, that the timestamps of the old data will also have to be multiplied by 60 together with all the operators used in the ontology and queries (e.g., $\bigwedge_{i=0}^{60}\bigcirc_P^i\mathsf{LowSpeed}\sqsubseteq\mathsf{Alert}$ saying that an alert is to be issued if a turbine maintained low speed for 1 hour). On the other hand, if we assume that time is dense and use rational numbers to refer to time instants, then we can represent timestamps such as 11:27:30 of the new turbine as $i+\frac{1}{2}$ (assuming that 11:27 and 11:28 correspond to integer numbers i and $i+1$, respectively), keeping the old timestamps and the ontology intact. However, for dense time, we cannot use the inherently discrete *LTL* operators in the ontology and queries any longer, and shall have to switch to a different temporal formalism with, say, *metric* interval operators, in which case the axiom above will have to be rewritten as $\boxminus_{[0,60]}\mathsf{LowSpeed} \sqsubseteq \mathsf{Alert}$, where $\boxminus_{[0,60]}\mathsf{LowSpeed}$ is true at a moment i iff $\mathsf{LowSpeed}$ holds at every j such that $i - j \in [0, 60]$.

6.1 *datalogMTL*$^\Box$

In the standard metric temporal logic *MTL* [3], the temporal domain is the real numbers $\mathbb{R}$, while the intervals ϱ in the constrained temporal operators such as $\boxminus_\varrho$ (always in the past within the interval ϱ from now) have natural numbers or ∞ as their endpoints. For various applications, it would be more appropriate to assume that the endpoints of ϱ are non-negative rational numbers or ∞, while the temporal domain is the rational numbers $\mathbb{Q}$ (however, in theory, not much will change if we take $\mathbb{R}$ as the temporal domain). Thus, by an *interval*, ι, we mean in this section any nonempty subset of $\mathbb{Q}$ of the form $[i, j]$, $[i, j)$, $(i, j]$ or (i, j), where $i, j \in \mathbb{Q} \cup \{-\infty, \infty\}$ and $i \leq j$. (We identify $(i, \infty]$ with (i, ∞), $[-\infty, i]$ with $(-\infty, i]$, etc.) The set of all intervals in $\mathbb{Q}$ is denoted by $\mathsf{int}(\mathbb{Q})$. A *range*, ϱ, is an interval with non-negative endpoints.

As in Section 5.3, we take datalog as the domain ontology language and combine it with *MTL*. Thus, a data instance (ABox), $\mathcal{A}$, is a finite set of facts of the form $P(\boldsymbol{a})@\iota$, where P is an m-ary predicate symbol, $\boldsymbol{a}$ an m-tuple of individual constants, for some $m \geq 0$, and $\iota \in \mathsf{int}(\mathbb{Q})$. This fact says that $P(\boldsymbol{a})$ is true at *each point of time* in the interval ι. To reflect this subtle semantical difference from Section 5, we write $P(\boldsymbol{a})@\iota$ rather than $P(\boldsymbol{a}, \iota)$. The following facts are an example of a data instance:

$$\begin{aligned}
&\mathsf{Turbine(tb0)}@(-\infty,\infty), \qquad && \mathsf{ActivePowerAbove1.5(tb0)}@[13{:}00{:}00, 13{:}00{:}10), \\
& && \mathsf{ActivePowerAbove1.5(tb0)}@[13{:}00{:}08, 13{:}00{:}15), \\
& && \mathsf{ActivePowerBelow0.15(tb0)}@[13{:}00{:}17, 13{:}01{:}25).
\end{aligned} \tag{13}$$

Brandt et al. [29] consider *atomic queries* of the form $\boldsymbol{q}(\boldsymbol{x}, \chi) = P(\boldsymbol{\tau})@\chi$, where P is a predicate name, $\boldsymbol{x}$ is a tuple of all individual variables occurring in the terms $\boldsymbol{\tau}$, and χ an *interval variable*. For example, the answers to the query $\boldsymbol{q}(\chi) = \mathsf{ActivePowerAbove1.5(tb0)}@\chi$ over the data instance above contain (among others) the intervals $[13{:}00{:}00, 13{:}00{:}10)$, $(13{:}00{:}05, 13{:}00{:}10)$, and $[13{:}00{:}00, 13{:}00{:}15)$ (the semantics will be defined below), as this information is contained, explicitly or implicitly, in $\mathcal{A}$. In a practical OBDA system, however, the returned result should be limited to the last interval only, $[13{:}00{:}00, 13{:}00{:}15)$, because it *includes* all other answers.

The temporal ontology language *datalogMTL*$^\Box$ uses the rules of the form (11), where the atoms A are defined by the grammar

$$A \quad ::= \quad \top \quad | \quad P(\tau_1, \dots, \tau_m) \quad | \quad \boxplus_\varrho A \quad | \quad \boxminus_\varrho A \quad | \quad A_1 \, \mathcal{S}_\varrho \, A_2 \quad | \quad A_1 \, \mathcal{U}_\varrho \, A_2$$

and A^+ is as above but without any 'non-deterministic' operators $\mathcal{U}_\varrho$ and $\mathcal{S}_\varrho$; cf. (12). We also use standard abbreviations $\diamondminus_\varrho A = \top \, \mathcal{S}_\varrho \, A$ and $\diamondplus_\varrho A = \top \, \mathcal{U}_\varrho \, A$. A *datalogMTL*$^\Box$ program is a finite set of rules.

▶ **Example 20.** For instance, the rule

$$\begin{aligned}
&\mathsf{ActivePowerTrip}(x) \leftarrow \mathsf{Turbine}(x) \land \boxminus_{[0,1m]} \mathsf{ActivePowerBelow0.15}(x) \land \\
&\qquad\qquad\qquad\qquad\qquad\qquad \diamondminus_{[60s,63s]} \boxminus_{[0,10s]} \mathsf{ActivePowerAbove1.5}(x)
\end{aligned} \tag{14}$$

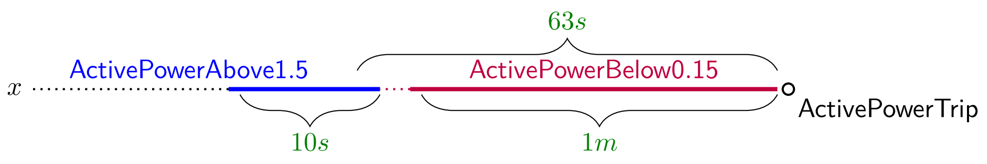


Figure 6 ActivePowerTrip.

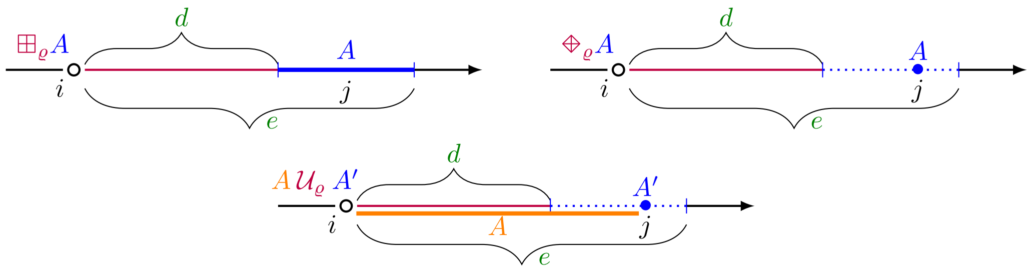


Figure 7 Semantics of metric temporal operators for $\varrho = [d, e]$.

says that an active power trip happens when the active power of a turbine was above 1.5MW for a period of at least 10 seconds, maximum 3 seconds after which there was a period of at least one minute where the active power was below 0.15MW, as shown in Fig. 6.

The semantics of query answering in *datalogMTL*$^\Box$ is essentially point-based. Thus, an *interpretation*, $\mathfrak{M}$, is based on a *domain* $\Delta \neq \emptyset$ for the individual variables and constants. For any m-ary predicate P, m-tuple $\boldsymbol{c}$ from Δ, and any moment of time $i \in \mathbb{Q}$, the interpretation $\mathfrak{M}$ specifies whether P is *true on* $\boldsymbol{c}$ *at* i, in which case we write $\mathfrak{M}, i \models P(\boldsymbol{c})$. As before, $\mathfrak{d}$ is an *assignment* of elements of Δ to the individual variables (we adopt the standard name assumption: $\mathfrak{d}(a) = a$, for every individual constant a). We then set inductively:

$$\mathfrak{M}, i \models^{\mathfrak{d}} \top, \quad \text{and} \quad \mathfrak{M}, i \not\models^{\mathfrak{d}} \bot,$$
$$\mathfrak{M}, i \models^{\mathfrak{d}} P(\boldsymbol{\tau}) \quad \text{iff} \quad \mathfrak{M}, i \models P(\mathfrak{d}(\boldsymbol{\tau})),$$
$$\mathfrak{M}, i \models^{\mathfrak{d}} (\tau \neq \tau') \quad \text{iff} \quad \mathfrak{d}(\tau) \neq \mathfrak{d}(\tau'),$$
$$\mathfrak{M}, i \models^{\mathfrak{d}} \boxplus_\varrho A \quad \text{iff} \quad \mathfrak{M}, j \models^{\mathfrak{d}} A \text{ for all } j \text{ with } j - i \in \varrho,$$
$$\mathfrak{M}, i \models^{\mathfrak{d}} \boxminus_\varrho A \quad \text{iff} \quad \mathfrak{M}, j \models^{\mathfrak{d}} A \text{ for all } j \text{ with } i - j \in \varrho,$$
$$\mathfrak{M}, i \models^{\mathfrak{d}} A_1 \,\mathcal{U}_\varrho\, A_2 \quad \text{iff} \quad \mathfrak{M}, i' \models^{\mathfrak{d}} A_2 \text{ for some } i' \text{ with } i' - i \in \varrho \text{ and}$$
$$\mathfrak{M}, j \models^{\mathfrak{d}} A_1 \text{ for all } j \in (i, i'),$$
$$\mathfrak{M}, i \models^{\mathfrak{d}} A_1 \,\mathcal{S}_\varrho\, A_2 \quad \text{iff} \quad \mathfrak{M}, i' \models^{\mathfrak{d}} A_2 \text{ for some } i' \text{ with } i - i' \in \varrho \text{ and}$$
$$\mathfrak{M}, j \models^{\mathfrak{d}} A_1 \text{ for all } j \in (i', i).$$

Figure 7 illustrates the semantics of the future-time operators for $\varrho = [d, e]$. Note that ranges ϱ in the temporal operators can be punctual $[d, d]$, in which case $\boxplus_{[d,d]} A$ is equivalent to $\diamondplus_{[d,d]} A$, and $\boxminus_{[d,d]} A$ to $\diamondminus_{[d,d]} A$.

We say that $\mathfrak{M}$ *satisfies* a *datalogMTL*$^\Box$ program Π under an assignment $\mathfrak{d}$ if, for *all* $i \in \mathbb{Q}$ and all the rules $A \leftarrow A_1 \wedge \cdots \wedge A_k$ in Π, we have

$$\mathfrak{M}, i \models^{\mathfrak{d}} A \quad \text{whenever} \quad \mathfrak{M}, i \models^{\mathfrak{d}} A_n \quad \text{for } 1 \leq n \leq k.$$

We call $\mathfrak{M}$ a *model* of Π and $\mathcal{A}$ and write $\mathfrak{M} \models (\Pi, \mathcal{A})$ if $\mathfrak{M}$ satisfies Π under every assignment,

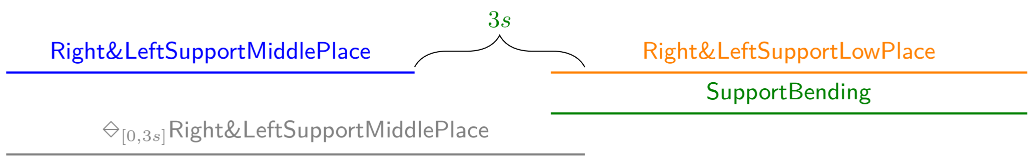

3s
Right&LeftSupportMiddlePlace
Right&LeftSupportLowPlace
SupportBending
$\diamondminus_{[0,3s]}$Right&LeftSupportMiddlePlace

Figure 8 SupportBending in Example 21.

and $\mathfrak{M}, i \models P(\boldsymbol{a})$ for any $P(\boldsymbol{a})@\iota$ in $\mathcal{A}$ and any $i \in \iota$. Π and $\mathcal{A}$ are *consistent* if they have a model.

A *datalogMTL*$^{\square}$ *ontology-mediated query* is of the form $(\Pi, \boldsymbol{q}(\boldsymbol{x}, \chi))$, where Π is a *datalogMTL*$^{\square}$ program and $\boldsymbol{q}(\boldsymbol{x}, \chi)$ is an atomic query $P(\boldsymbol{\tau})@\chi$. A *certain answer* to $(\Pi, \boldsymbol{q}(\boldsymbol{x}, \chi))$ over a data instance $\mathcal{A}$ is a pair $(\boldsymbol{a}, \iota)$ such that $\boldsymbol{a}$ is a tuple of constants from $\mathcal{A}$ of the same length as $\boldsymbol{x}$, ι an interval and, for any $i \in \iota$, any model $\mathfrak{M}$ of Π and $\mathcal{A}$, and any assignment $\mathfrak{d}$ mapping $\boldsymbol{x}$ to $\boldsymbol{a}$, we have $\mathfrak{M}, i \models^{\mathfrak{d}} P(\boldsymbol{\tau})$. In this case, we write $\mathfrak{M}, i \models \boldsymbol{q}(\boldsymbol{a})$. To illustrate, the *datalogMTL*$^{\square}$ query $(\Pi, \mathsf{ActivePowerTrip}(\mathsf{tb0})@\chi)$, where Π consists of rule (14), returns $[13{:}01{:}17, 13{:}01{:}18)$ as a certain answer over the data instance above.

▶ **Example 21.** We illustrate the importance of the operators $\mathcal{S}_{\varrho}$ and $\mathcal{U}_{\varrho}$ using an example inspired by the ballet moves ontology [76]. Suppose we want to say that SupportBending is a move spanning from the beginning to the end of Right&LeftSupportLowPlace provided that it is preceded by Right&LeftSupportMiddlePlace, which ends within $3s$ from the beginning of the Right&LeftSupportLowPlace, as shown in Fig. 8.

We can define the SupportBending move using the following rule:

$$\mathsf{SupportBending} \leftarrow \mathsf{Right\&LeftSupportLowPlace}\ \mathcal{S}_{[0,\infty)} \diamondminus_{[0,3s]} \mathsf{Right\&LeftSupportMiddlePlace}.$$

Note that defining SupportBending in *datalogMTL*$^{\square}$ would be problematic if only the $\square$ and $\diamond$ operators were available.

Atomic OMQ evaluation with *datalogMTL*$^{\square}$ has been studied by Brandt et al. [29]. In particular, it was shown to be decidable and ExpSpace-complete for combined complexity. This result holds even with punctual temporal operators (with range $[d, d]$), in which case the propositional *MTL* is known to be undecidable [4]; on the other hand, the propositional *MTL* is ExpSpace-complete if the punctual operators are not allowed [3]; see also [66, 67]. In fact, the undecidability result in the presence of punctual operators holds even for the propositional (predicates of arity 0 only) fragment of *datalogMTL*$^{\square}$ extended by $\diamondminus_{\varrho}$ and $\diamondplus_{\varrho}$ operators in the head of rules [29] (cf. $\mathcal{HS}^{\square}_{horn}$ in Section 5.2). Furthermore, it was shown that, for *nonrecursive datalogMTL*$^{\square}$ programs, query answering is PSpace-complete for combined complexity and in AC^0 for data complexity.

6.2 Use Cases

The metric temporal ontology language *datalogMTL*$^{\square}$ has been used to construct ontologies and support query answering in three practical use-cases [29, 76], which will be briefly discussed below.

Turbine Monitoring at Siemens At Siemens, service centres store aggregated turbine sensor data instances such as (13). A *datalogMTL*$^{\square}$ ontology has been designed [29] to define events (representing normal or abnormal behaviour) that are of interest to engineers monitoring the performance of turbines. One such event is active power trip defined by (14). As another example, we show a (partial) definition of normal restart:

$$\begin{aligned}
&\mathsf{NormalRestart}(x) \leftarrow \mathsf{NormalStart}(x) \land \diamondminus_{(0,1h]}\mathsf{NormalStop}(x),\\
&\mathsf{NormalStop}(x) \leftarrow \mathsf{CoastDown1500to200}(x) \land \diamondminus_{(0,9m]}\big[\mathsf{CoastDown6600to1500}(x) \land\\
&\qquad\qquad \diamondminus_{(0,2m]}\big(\mathsf{MainFlameOff}(x) \land \diamondminus_{(0,2m]}\mathsf{ActivePowerOff}(x)\big)\big],\\
&\mathsf{MainFlameOff}(x) \leftarrow \boxminus_{[0s,10s]}\mathsf{MainFlameBelow0.1}(x).
\end{aligned}$$

(The complete definition of normal restart contains 12 rules.) The purpose of this ontology is to enable a convenient access to temporal information for an engineer who can pose succinct queries such as $\boldsymbol{q}(x,\chi) = \mathsf{NormalRestart}(x)@\chi$ (find the turbines that had a normal restart) without having to write explicitly (or even to know) the complex definition of this event.

Weather Monitoring The MesoWest[3] project makes publicly available historical records of the weather stations across the US showing such parameters of meteorological conditions as temperature, wind speed and direction, amount of precipitation, etc. From this data, one can extract facts such as

$$\begin{aligned}
&\mathsf{NorthWind(KBVY)}@(15{:}14, 15{:}24], && \mathsf{HurricaneForceWind(KMNI)}@(15{:}21, 15{:}31],\\
&\mathsf{Precipitation(KBVY)}@(15{:}14, 15{:}24], && \mathsf{TempAbove0(KBVY)}@(15{:}14, 15{:}24],\\
&\mathsf{TempAbove0(KMNI)}@(15{:}21, 15{:}31], &&\\
&\mathsf{LocatedInCounty(KBVY, Essex)}@(-\infty,\infty), && \mathsf{LocatedInState(KBVY, MA)}@(-\infty,\infty),
\end{aligned}$$

where KBVY, KMNI are IDs of the stations (according to the standard definition, the hurricane force wind is above 118 km/h). A snippet of a weather ontology giving meteorological definitions (such as 'a hurricane is a hurricane force wind lasting one hour or longer') is shown below:

$$\begin{aligned}
&\boxminus_{[0,1h]}\mathsf{Hurricane}(x) \leftarrow \boxminus_{[0,1h]}\mathsf{HurricaneForceWind}(x),\\
&\mathsf{ShoweryCounty}(x) \leftarrow \mathsf{LocatedInCounty}(u_1,x) \land \mathsf{LocatedInCounty}(u_2,x) \land\\
&\qquad \mathsf{Precipitation}(u_1) \land \mathsf{NoPrecipitation}(u_2) \land \diamondminus_{(0,30m]}\mathsf{Precipitation}(u_2),\\
&\mathsf{HurricaneAffectedState}(x) \leftarrow \mathsf{LocatedInState}(u,x) \land \mathsf{Hurricane}(u),\\
&\boxminus_{[0,24h]}\mathsf{ExcessiveHeat}(x) \leftarrow \boxminus_{[0,24h]}\mathsf{TempAbove24}(x) \land \diamondminus_{[0,24h]}\mathsf{TempAbove41}(x),\\
&\mathsf{HeatAffectedCounty}(x) \leftarrow \mathsf{LocatedInCounty}(u,x) \land \mathsf{ExcessiveHeat}(u),\\
&\mathsf{CyclonePatternState}(x) \leftarrow \mathsf{LocatedInState}(u_1,x) \land \mathsf{LocatedInState}(u_2,x) \land\\
&\qquad \mathsf{LocatedInState}(u_3,x) \land \mathsf{LocatedInState}(u_4,x) \land \mathsf{EastWind}(u_1) \land\\
&\qquad\qquad \mathsf{NorthWind}(u_2) \land \mathsf{WestWind}(u_3) \land \mathsf{SouthWind}(u_4).
\end{aligned}$$

The purpose of using the temporal ontology in the weather use-case is to enable a weather expert to find information about complex meteorological events by using succinct queries.

BalOnSe: Ontology of Dance Movements This use-case is concerned with user annotations of ballet videos such as

$$\mathsf{LeftLegGestureMiddleBack(video1)}@[12s, 13s]$$

[3] `http://mesowest.utah.edu/`

saying that the movement LeftLegGestureMiddleBack is shown in video1 from 00:12:00 to 00:13:00. The ballet ontology [76] reflects the terminology developed by ballet researchers and contains rules such as

$$\begin{aligned}\boxplus_{[0,3s]}\mathsf{PlieReleve}(x) \leftrightarrow\ & \boxplus_{[0,1s]}\ \mathsf{RightSupportMidPlace}(x) \land \boxplus_{[0,1s]}\mathsf{LeftSupportMidPlace}(x) \land \\ & \boxplus_{[1,2s]}\ \mathsf{RightSupportLowPlace}(x) \land \boxplus_{[1,2s]}\mathsf{LeftSupportLowPlace}(x) \land \\ & \boxplus_{[2,3s]}\ \mathsf{RightSupportHighPlace}(x) \land \boxplus_{[2,3s]}\mathsf{LeftSupportHighPlace}(x)\end{aligned}$$

defining the composite movement *plie releve* as a sequence of simpler movements occurring simultaneously or in a sequence. The video annotations together with the ontology are then used to enhance the search capabilities of a video search system for ballet learners and scholars. Thus, searching the term *plie releve* will return the videos (and time spans in them) showing this movement, even if the annotation for this sequence is not explicitly present in the database, but is deducible from the ontology and other annotations.

7 Ontology-Based Data Access and Implementations

In real-world applications, the data instances (ABoxes) are not created from scratch. In fact, they are obtained from existing relational or RDF databases by means of *mappings* (queries in the language of a data source) in order to produce a high-level conceptual view of the data. Such ABoxes can be materialised and stored as, e.g., RDF triples, or remain virtual (as a potential result of applying the mapping to the data), in which case an ontology-mediated query may be evaluated by rewriting it into a set of queries in the language(s) of the data sources. In the this section, we briefly address the problem of converting raw data to an ABox in the context of temporal data. After that we present some prototypical implementations of temporal ontology-based data access and evaluations of their performance.

7.1 From Raw to Conceptual Temporal Data

Suppose turbine sensor measurements are stored in a relational table TB_Sensor:

turbineId	dateTime	activePower	rotorSpeed	mainFlame	...
tb0	2015-04-04 12:20:48	2	1550	0	
tb0	2015-04-04 12:20:49	1.8	1400	null	
tb0	2015-04-04 12:20:52	1.7	1350	1	
		...			

There are three major options for conceptualising this data if, for instance, we are interested in the situations when the rotor speed was below 1500:

- $\mathsf{RotorSpeedBelow1500}(\mathsf{tb0}, i)$, $\mathsf{RotorSpeedBelow1500}(\mathsf{tb0}, i + 1)$, where i is the timestamp in the first row of the table, and so $i + 1$ is the number of the second timestamp. This is the most simplistic approach that ignores the distance between the timestamps, but it is suitable if the timestamps are present in the database at regular intervals (which is not the case above), or only the sequence of events rather than the duration of the gap between them is important.
- $\mathsf{RotorSpeedBelow1500}(\mathsf{tb0}, i)$, $\mathsf{RotorSpeedBelow1500}(\mathsf{tb0}, i + 3)$, where i is the timestamp in the first row of the table. Here, we obviously make an assumption that the time unit in our application domain is 'a second', and so this approach takes into account the duration of the gap between the events.

- RotorSpeedBelow1500(tb0, $\langle i, i+3\rangle$), where $\langle i, i+3\rangle$ is a time interval. Here, we use a real-world assumption that a rotor speed sensor sends its measurements only when the current value of the speed is sufficiently different from the previous measurement, and this value is assumed to hold for all the times until the next one is produced. Note also that some sensors may produce aggregated (e.g., average) value taken over some period.

The choice of how to conceptualise the data depends on the application domain. Below, we follow the third approach and show a mapping (in the syntax similar to the standard R2RML mapping language, where in the body we use standard SQL with window operators) that extracts the data instance related to the situations when active power was above 1.5MW:

```
ActivePowerAbove1.5(tbid)@[ledge, redge) ←
    SELECT tbid, ledge, redge FROM (
        SELECT turbineId AS tbid,  LAG(dateTime, 1) OVER (w) AS ledge,
               LAG(activePower, 1) OVER (w) AS lag_activePower,  dateTime AS redge
        FROM TB_Sensor
        WINDOW w AS (PARTITION BY turbineId ORDER BY dateTime)) tmp
    WHERE lag_activePower > 1.5
```

The mapping above applied to TB_Sensor will produce the following instance:

ActivePowerAbove1.5(tb0)@[12:20:48, 12:20:49),

ActivePowerAbove1.5(tb0)@[12:20:49, 12:20:52).

Note that we use the definition of interval from Section 6 and make an assumption (reflecting our intuition on how sensors produce their measurements) that the intervals involved are all of the form $[i, j)$. Clearly, we can add similar mappings for the concepts RotorSpeedAbove1500 and MainFlameBelow0.1.

7.2 Implementation

We report on the implementation of temporal ontology-based data access and its evaluation [29]. The ontology language supported by this implementation is $\mathit{datalog}_{nr}\mathit{MTL}^{\Box}$ consisting of nonrecursive $\mathit{datalogMTL}^{\Box}$ programs, and the system rewrites $\mathit{datalog}_{nr}\mathit{MTL}^{\Box}$ OMQs to standard SQL queries with views. The performance the rewritings for the Siemens use-case described in Section 6.2 was evaluated on an HP Proliant server with 24 Intel Xeon CPUs (@3.47GHz), 106GB of RAM and five 1TB 15K RPM HD, which used PostgreSQL as a database engine. The maximum physical memory consumption in the experiments was 12.9GB.

Siemens supplied a sample of data for one running turbine, denoted tb0, over 4 days in the form of the table TB_Sensor. This sample was replicated to imitate the data for one turbine over 10 different periods ranging from 32 to 320 months. Four queries ActivePowerTrip(tb0)@χ, NormalStart(tb0)@χ, NormalStop(tb0)@χ, and NormalRestart(tb0)@χ were evaluated with a timeout of 30 minutes. The execution times are given in Fig. 9, which shows their linear growth in the number of months and, consequently, in the size of data.

Note that the normal restart (start) query timeouts on the data for more than 15 (respectively, 20) years, which is more than enough for the monitoring and diagnostics tasks at Siemens, where the two most common application scenarios for sensor data analytics are daily monitoring (that is, analytics of high-frequency data of the previous 24 hours) and fleet-level analytics of key-performance indicators over one year. In both cases, the computation time of the results is far less a crucial cost factor than the lead-time for data preparation.

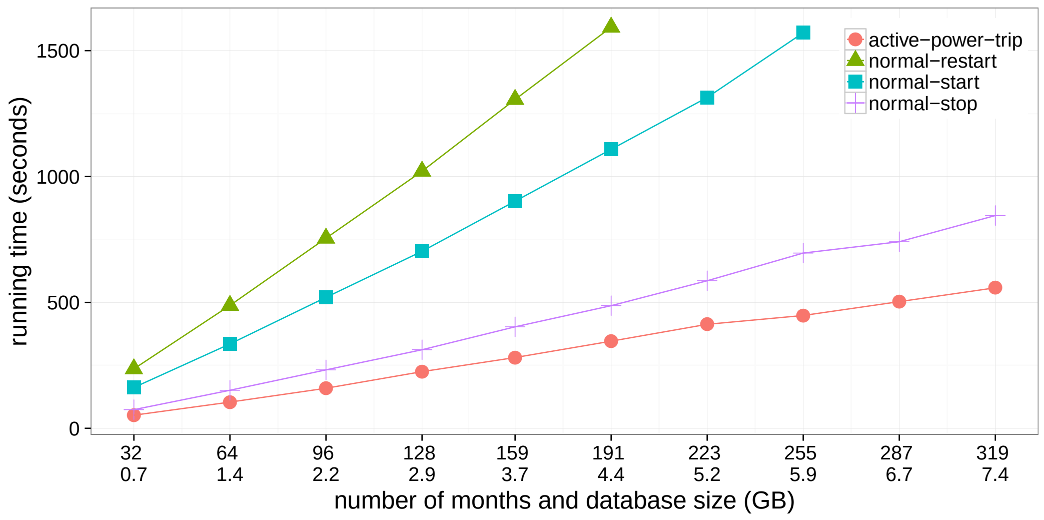


Figure 9 Performance of queries in the Siemens use-case.

The evaluation was performed for the weather OMQs with MesoWest data (see Section 6.2) as well. On the other hand, the system SPARK capable of parallel query processing, in place of PostgreSQL, was evaluated showing large performance improvements in some cases; for details consult [30].

References

1 Luca Aceto, Dario Della Monica, Valentin Goranko, Anna Ingólfsdóttir, Angelo Montanari, and Guido Sciavicco. A complete classification of the expressiveness of interval logics of Allen's relations: the general and the dense cases. *Acta Inf.*, 53(3):207–246, 2016. `doi:10.1007/s00236-015-0231-4`.

2 James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983. `doi:10.1145/182.358434`.

3 Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996. `doi:10.1145/227595.227602`.

4 Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Inf. Comput.*, 104(1):35–77, 1993. `doi:10.1006/inco.1993.1025`.

5 Natalia Antonioli, Francesco Castanò, Spartaco Coletta, Stefano Grossi, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Emanuela Virardi, and Patrizia Castracane. Ontology-based data management for the Italian public debt. In *Proc. of the 8th Int. Conf. on Formal Ontology in Information Systems, FOIS 2014*, pages 372–385. IOS Press, 2014. `doi:10.3233/978-1-61499-438-1-372`.

6 Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The *DL-Lite* family and relations. *J. Artif. Intell. Res. (JAIR)*, 36:1–69, 2009. `doi:10.1613/jair.2820`.

7 Alessandro Artale and Enrico Franconi. Temporal description logics. In *Handbook of Temporal Reasoning in Artificial Intelligence*, volume 1 of *Foundations of Artificial Intelligence*, pages 375–388. Elsevier, 2005. `doi:10.1016/S1574-6526(05)80014-8`.

8 Alessandro Artale, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. First-order rewritability of temporal ontology-mediated queries. In *Proc. of the 24th Int. Joint Conf. on Artificial Intelligence, IJCAI'15*, pages 2706–2712. IJCAI/AAAI, 2015.

9 Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. The complexity of clausal fragments of LTL. In *Proc. of the 19th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR'13*, volume 8312 of *LNCS*, pages 35–52. Springer, 2013. `doi:10.1007/978-3-642-45221-5_3`.

10 Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. A cookbook for temporal conceptual data modelling with description logics. *ACM Trans. Comput. Log.*, 15(3):25:1–25:50, 2014. `doi:10.1145/2629565`.

11 Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. Tractable interval temporal propositional and description logics. In *Proc. of the 29th Conf. on Artificial Intelligence, AAAI'15*, pages 1417–1423. AAAI Press, 2015.

12 Alessandro Artale, Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev. Temporal description logic for ontology-based data access. In *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence, IJCAI'13*, pages 711–717. IJCAI/AAAI, 2013.

13 Franz Baader. Ontology-based monitoring of dynamic systems. In *Proc. of the 14th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR'14*, pages 678–681. AAAI Press, 2014.

14 Franz Baader, Stefan Borgwardt, Patrick Koopmann, Ana Ozaki, and Veronika Thost. Metric temporal description logics with interval-rigid names (extended abstract). In *Proc. of the 30th Int. Workshop on Description Logics, DL'17*. CEUR-WS, 2017.

15 Franz Baader, Stefan Borgwardt, and Marcel Lippmann. Temporalizing ontology-based data access. In *Proc. of the 24th Int. Conf. on Automated Deduction, CADE-24*, volume 7898 of *LNCS*, pages 330–344. Springer, 2013. `doi:10.1007/978-3-642-38574-2_23`.

16 Franz Baader, Stefan Borgwardt, and Marcel Lippmann. Temporal conjunctive queries in expressive description logics with transitive roles. In *Proc. of the 28th Australasian Joint Conf. on Advances in Artificial Intelligence, AI'15*, volume 9457 of *LNCS*, pages 21–33. Springer, 2015. `doi:10.1007/978-3-319-26350-2_3`.

17 Franz Baader, Stefan Borgwardt, and Marcel Lippmann. Temporal query entailment in the description logic SHQ. *J. Web Semantics*, 33:71–93, 2015. `doi:10.1016/j.websem.2014.11.008`.

18 Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, IJCAI-05*, pages 364–369. IJCAI/AAAI, 2005.

19 Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

20 Franz Baader, Silvio Ghilardi, and Carsten Lutz. LTL over description logic axioms. In *Proc. of the 11th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR 2008*, pages 684–694. AAAI Press, 2008.

21 Franz Baader, Ralf Küsters, and Frank Wolter. Extensions to description logics. In *The Description Logic Handbook*, pages 219–261. Cambridge University Press, 2003.

22 Samantha Bail, Sandra Alkiviadous, Bijan Parsia, David Workman, Mark Van Harmelen, Rafael S. Goncalves, and Cristina Garilao. Fishmark: A linked data application benchmark. In *Proc. of SSWS+HPCSW 2012*, pages 1–15. CEUR-WS, 2012.

23 Philippe Balbiani, Jean-François Condotta, and Luis Fariñas del Cerro. Tractability results in the block algebra. *J. Log. Comput.*, 12(5):885–909, 2002. `doi:10.1093/logcom/12.5.885`.

24 Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. on Database Systems*, 39(4):33:1–33:44, 2014. doi:10.1145/2661643.

25 Stefan Borgwardt, Marcel Lippmann, and Veronika Thost. Temporal query answering in the description logic *DL-Lite*. In *Proc. of the 9th Int. Symposium on Frontiers of Combining Systems, FroCoS'13*, volume 8152 of *LNCS*, pages 165–180. Springer, 2013. doi:10.1007/978-3-642-40885-4_11.

26 Stefan Borgwardt, Marcel Lippmann, and Veronika Thost. Temporalizing rewritable query languages over knowledge bases. *J. Web Semantics*, 33:50–70, 2015. doi:10.1016/j.websem.2014.11.007.

27 Stefan Borgwardt and Veronika Thost. Temporal query answering in *DL-Lite* with negation. In *Proc. of the Global Conf. on Artificial Intelligence, GCAI15*, volume 36 of *EPiC Series in Computing*, pages 51–65, 2015.

28 Stefan Borgwardt and Veronika Thost. Temporal query answering in the description logic $\mathcal{EL}$. In *Proc. of the 24h Int. Joint Conf. on Artificial Intelligence, IJCAI'15*, pages 2819–2825. AAAI Press, 2015.

29 Sebastian Brandt, Elem Güzel Kalayci, Roman Kontchakov, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. Ontology-based data access with a horn fragment of metric temporal logic. In *Proc. of the 31st AAAI Conf. on Artificial Intelligence, AAAI'17*, pages 1070–1076. AAAI Press, 2017.

30 Sebastian Brandt, Elem Güzel Kalayci, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. Querying log data with metric temporal logic. *CoRR*, abs/1703.08982, 2017.

31 Torben Braüner and Silvio Ghilardi. First-order modal logic. In *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 549–620. Elsevier, 2007.

32 Davide Bresolin, Agi Kurucz, Emilio Muñoz-Velasco, Vladislav Ryzhikov, Guido Sciavicco, and Michael Zakharyaschev. Horn fragments of the Halpern-Shoham interval temporal logic. *ACM Trans. Comput. Log.*, 18(3), 2017.

33 Davide Bresolin, Dario Della Monica, Valentin Goranko, Angelo Montanari, and Guido Sciavicco. The dark side of interval temporal logic: marking the undecidability border. *Ann. Math. Artif. Intell. (AMAI)*, 71(1-3):41–83, 2014. doi:10.1007/s10472-013-9376-4.

34 Davide Bresolin, Dario Della Monica, Angelo Montanari, and Guido Sciavicco. The light side of interval temporal logic: the Bernays-Schönfinkel fragment of CDT. *Ann. Math. Artif. Intell. (AMAI)*, 71(1-3):11–39, 2014. doi:10.1007/s10472-013-9337-y.

35 Richard J. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1–6):66–92, 1960. doi:10.1002/malq.19600060105.

36 Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The MASTRO system for ontology-based data access. *Semantic Web*, 2(1):43–53, 2011. doi:10.3233/SW-2011-0029.

37 Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007. doi:10.1007/s10817-007-9078-x.

38 Diego Calvanese, Pietro Liuzzo, Alessandro Mosca, José Remesal, Martin Rezk, and Guillem Rull. Ontology-based data integration in EPNet: Production and distribution of food during the Roman Empire. *Eng. Appl. of AI*, 51:212–229, 2016. doi:10.1016/j.engappai.2016.01.005.

39 Jan Chomicki. Polynomial time query processing in temporal deductive databases. In *Proc. of the 9th ACM Symposium on Principles of Database Systems, PODS'90*, pages 379–391. ACM Press, 1990. `doi:10.1145/298514.298589`.

40 Jan Chomicki and Tomasz Imielinski. Temporal deductive databases and infinite objects. In *Proc. of the 7th ACM Symposium on Principles of Database Systems, PODS'88*, pages 61–73. ACM, 1988. `doi:10.1145/308386.308416`.

41 Stéphane Demri, Valentin Goranko, and Martin Lange. *Temporal Logics in Computer Science*. Cambridge University Press, 2016.

42 Michael Fisher, Clare Dixon, and Martin Peim. Clausal temporal resolution. *ACM Trans. Comput. Log.*, 2(1):12–56, 2001. `doi:10.1145/371282.371311`.

43 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. `doi:10.1007/BF01744431`.

44 Dov M. Gabbay, Ian Hodkinson, and Mark Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 1. Oxford University Press, 1994.

45 Dov M. Gabbay, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Elsevier North Holland, 2003.

46 Dov M. Gabbay, Mark A. Reynolds, and Marcelo Finger. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 2. Oxford University Press, 2000.

47 Martin Giese, Ahmet Soylu, Guillermo Vega-Gorgojo, Arild Waaler, Peter Haase, Ernesto Jiménez-Ruiz, Davide Lanti, Martín Rezk, Guohui Xiao, Özgür L. Özçep, and Riccardo Rosati. Optique: Zooming in on big data. *IEEE Computer*, 48(3):60–67, 2015. `doi:10.1109/MC.2015.82`.

48 Claudio Gutierrez, Carlos A. Hurtado, and Alejandro A. Vaisman. Introducing time into RDF. *IEEE Trans. Knowl. Data Eng.*, 19(2):207–218, 2007. `doi:10.1109/TKDE.2007.34`.

49 Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Roman Kontchakov. Temporalized $\mathcal{EL}$ ontologies for accessing temporal data: Complexity of atomic queries. In *Proc. of the 25th Int. Joint Conf. on Artificial Intelligence, IJCAI'16*, pages 1102–1108. IJCAI/AAAI, 2016.

50 Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Ana Ozaki. On metric temporal description logics. In *Proc. of the 22nd European Conf. on Artificial Intelligence, ECAI 2016*, volume 285 of *FAIA*, pages 837–845. IOS Press, 2016. `doi:10.3233/978-1-61499-672-9-837`.

51 Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Thomas Schneider. Lightweight description logics and branching time: A troublesome marriage. In *Proc. of the 14th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR'14*, pages 278–287. AAAI Press, 2014.

52 Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Thomas Schneider. Lightweight temporal description logics with rigid roles and restricted TBoxes. In *Proc. of the 24th Int. Joint Conf. on Artificial Intelligence, IJCAI 2015*, pages 3015–3021. IJCAI/AAAI, 2015.

53 Joseph Y. Halpern and Yoav Shoham. A propositional modal logic of time intervals. *J. ACM*, 38(4):935–962, 1991. `doi:10.1145/115234.115351`.

54 Ian M. Hodkinson, Roman Kontchakov, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. On the computational complexity of decidable fragments of first-order linear temporal logics. In *Proc. of the 10th Int. Symposium on Temporal Representation and Reasoning and the 4th Int. Conf. on Temporal Logic, TIME-ICTL 2003*, pages 91–98. IEEE Computer Society, 2003. `doi:10.1109/TIME.2003.1214884`.

55 Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, IJCAI-05*, pages 466–471. IJCAI/AAAI, 2005.

56 Neil Immerman. *Descriptive complexity.* Springer, 1999. doi:10.1007/978-1-4612-0539-5.

57 Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in *DL-Lite.* In *Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR 2010*, pages 247–257. AAAI Press, 2010.

58 Roman Kontchakov, Laura Pandolfo, Luca Pulina, Vladislav Ryzhikov, and Michael Zakharyaschev. Temporal and spatial OBDA with many-dimensional Halpern-Shoham logic. In *Proc. of the 25th Int. Joint Conf. on Artificial Intelligence, IJCAI'16*, pages 1160–1166. IJCAI/AAAI, 2016.

59 Alisa Kovtunova. *Ontology-Mediated Query Answering with Lightweight Temporal Description Logics.* PhD thesis, KRDB research centre, Faculty of Computer Science, Free University of Bozen-Bolzano, 2017.

60 Krishna G. Kulkarni and Jan-Eike Michels. Temporal features in SQL:2011. *SIGMOD Record*, 41(3):34–43, 2012.

61 Leonid Libkin. *Elements of Finite Model Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004. doi:10.1007/978-3-662-07003-1.

62 Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. Temporal description logics: A survey. In *Proc. of the 15th Int. Symposium on Temporal Representation and Reasoning, TIME'08*, pages 3–14. IEEE Computer Society, 2008. doi:10.1109/TIME.2008.14.

63 Jerzy Marcinkowski and Jakub Michaliszyn. The undecidability of the logic of subintervals. *Fundam. Inform.*, 131(2):217–240, 2014. doi:10.3233/FI-2014-1011.

64 Boris Motik. Representing and querying validity time in RDF and OWL: A logic-based approach. *J. Web Semantics*, 12:3–21, 2012. doi:10.1016/j.websem.2011.11.004.

65 Isabel Navarrete, Antonio Morales, Guido Sciavicco, and M. Antonia Cárdenas Viedma. Spatial reasoning with rectangular cardinal relations – the convex tractable subalgebra. *Ann. Math. Artif. Intell. (AMAI)*, 67(1):31–70, 2013. doi:10.1007/s10472-012-9327-5.

66 Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In *Proc. of the 20th Annual IEEE Symposium on Logic in Computer Science, LICS'05*, pages 188–197. IEEE Computer Society, 2005. doi:10.1109/LICS.2005.33.

67 Joël Ouaknine and James Worrell. Some recent results in metric temporal logic. In *Proc. of the 6th Int. Conf. on Formal Modeling and Analysis of Timed Systems, FORMATS'08*, pages 1–13, 2008. doi:10.1007/978-3-540-85778-5_1.

68 Özgür L. Özçep, Ian Horrocks, Ralf Möller, Thomas Hubauer, Christian Neuenstadt, Mikhail Roshchin, Dmitriy Zheleznyakov, and Evgeny Kharlamov. Deliverable D5.1: A semantics for temporal and stream-based query answering in an OBDA context. Technical report, Deliverable FP7-318338, EU, October 2013.

69 Özgür L. Özçep and Ralf Möller. Ontology based data access on temporal and streaming data. In *Proc. of the 10th Int. Summer School on Reasoning on the Web in the Big Data Era (Reasoning Web'14)*, volume 8714 of *LNCS*, pages 279–312. Springer, 2014. doi:10.1007/978-3-319-10587-1_7.

70 Özgür L. Özçep, Ralf Möller, and Christian Neuenstadt. A stream-temporal query language for ontology based data access. In *Proc. of the 37th Annual German Conf. on AI, KI'14*, pages 183–194. Springer, 2014. doi:10.1007/978-3-319-11206-0_18.

71 Francesco Pagliarecci, Luca Spalazzi, and Gilberto Taccari. Reasoning with temporal aboxes: Combining *DL-Lite*$_{core}$ with CTL. In *Proc. of the 26th Int. Workshop on Description Logics, DL'13*, pages 885–897. CEUR-WS, 2013.

72 Amir Pnueli. The temporal logic of programs. In *Proc. of the 18th Annual Symposium on Foundations of Computer Science, FOCS'77*, pages 46–57. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.32.

73 Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008. `doi:10.1007/978-3-540-77688-8_5`.

74 Andrea Pugliese, Octavian Udrea, and V. S. Subrahmanian. Scaling RDF with time. In *Proc. of the 17th Int. Conf. on World Wide Web, WWW'08*, pages 605–614. ACM, 2008. `doi:10.1145/1367497.1367579`.

75 Alexander Rabinovich. A proof of Kamp's theorem. *Logical Methods in Computer Science*, 10(1), 2014. `doi:10.2168/LMCS-10(1:14)2014`.

76 Katerina El Raheb, Theofilos Mailis, Vladislav Ryzhikov, Nicolas Papapetrou, and Yannis E. Ioannidis. Balonse: Temporal aspects of dance movement and its ontological representation. In *Proc. of the 14th Int. Conf., ESWC 2017, Part II*, pages 49–64, 2017. `doi:10.1007/978-3-319-58451-5_4`.

77 Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. Ontology-based data access: Ontop of databases. In *Proc. of the 12th Int. Semantic Web Conf., ISWC'13, Part I*, volume 8218 of *LNCS*, pages 558–573. Springer, 2013. `doi:10.1007/978-3-642-41335-3_35`.

78 Riccardo Rosati. On conjunctive query answering in $\mathcal{EL}$. In *Proc. of the Int. Workshop on Description Logics, DL'07*, volume 250. CEUR-WS, 2007.

79 Andrea Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. Intelligent Information Systems*, 2(3):265–278, 1993. `doi:doi:10.1007/BF00962071`.

80 Klaus Schild. Combining terminological logics with tense logic. In *Proc. of the 6th Portuguese Conf. on Progress in Artificial Intelligence, EPIA'93*, volume 727 of *Lecture Notes in Computer Science*, pages 105–120. Springer, 1993. `doi:10.1007/3-540-57287-2_41`.

81 Albrecht Schmiedel. Temporal terminological logic. In *Proc. of the 8th National Conf. on Artificial Intelligence, AAAI'90*, pages 640–645. AAAI Press / The MIT Press, 1990.

82 Juan F. Sequeda and Daniel P. Miranker. A pay-as-you-go methodology for ontology-based data access. *IEEE Internet Computing*, 21(2):92–96, 2017. `doi:10.1109/MIC.2017.46`.

83 Ahmet Soylu, Martin Giese, Rudolf Schlatte, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Özgür L. Özçep, Christian Neuenstadt, and Sebastian Brandt. Querying industrial stream-temporal data: An ontology-based visual approach. *J. Ambient Intelligence & Smart Environments*, 9(1):77–95, 2017. `doi:10.3233/AIS-160415`.

84 David Toman. On incompleteness of multi-dimensional first-order temporal logics. In *Proc. of the 10th Int. Symposium on Temporal Representation and Reasoning and the 4th Int. Conf. on Temporal Logic, TIME-ICTL 2003*, pages 99–106. IEEE Computer Society, 2003. `doi:10.1109/TIME.2003.1214885`.

85 Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *Proc. of the 14th Annual ACM Symposium on Theory of Computing, STOC'82*, pages 137–146. ACM, 1982. `doi:10.1145/800070.802186`.

Advances in Quantitative Analysis of Free-Choice Workflow Petri Nets[*]

Javier Esparza

Technische Universität München, Munich, Germany
esparza@in.tum.de

Abstract

We survey recent results on the development of efficient algorithms for the quantitative analysis of business processes modeled as workflow Petri nets. The algorithms can be applied to any workflow net, but have polynomial runtime in the free-choice case.

1998 ACM Subject Classification D.2.4 Software/Program Verification, D.2.2 Design Tools and Techniques, F.1.1 Models of Computation

Keywords and phrases Free-choice Petri Nets, concurrency theory, quantitative verification

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.2

Category Invited Talk

1 Introduction

Workflow Petri nets are a successful formalism for the representation and formal analysis of business processes. They are also much used as a formal back-end for different notations like BPMN (Business Process Modeling Notation), EPC (Event-driven Process Chain), or UML Activity Diagrams [1, 2, 5]. In this note we assume that the reader is familiar with basic Petri net terms: place, transition, token, marking, and the firing rule, that is, the rule that determines whether a transition is enabled at a marking, and how the marking changes when the transition fires.

In a nutshell, a workflow Petri net is just a Petri net with a distinguished initial place and a distinguished final place. These places induce distinguished initial and final markings, which contain a token in the initial/final place and no tokens elsewhere. In a well-designed workflow, every marking reachable from the initial marking enables some firing sequence leading to the final marking, a property known as *soundness* [1]. In particular, sound workflow nets are both deadlock-free and livelock-free.

Workflow Petri nets can be analyzed by constructing their reachability graph (which has the reachable markings as nodes, and the steps allowed by the firing rule as edges), and applying model-checking techniques, see e.g. [16, 23]. However, this approach suffers from the well-known state-explosion problem: Even if reachable markings put at most one token in every place, a workflow net with n places can still have $\Theta(2^n)$ reachable markings. For simple qualitative properties, like soundness, the state-explosion problem can be handled very effectively by tools like LoLA [22, 16]. However, the problem becomes acute for workflows enriched with data, time, and/or probabilities.

Since 2013 my co-authors and I have addressed this question by developing novel analysis algorithms that can be applied to general workflow nets, but provide strong runtime guarantees

[*] Supported by the DFG project "Negotiations: A Model for Tractable Concurrency".

for special classes. The rationale for this approach is that workflow Petri nets modeling real-life business processes tend to have a simpler structure than those found in other application areas, like the analysis of distributed algorithms, concurrent programs, or biological processes. In particular, it has been repeatedly observed that many of these workflow nets are *free-choice* [6]. For example, Workflow Graphs, a simple but effective business process formalism [1, 17, 18, 13], can be translated into free-choice workflow Petri nets, and 1386 of the 1958 workflow nets in the most popular benchmark suite in the literature are free-choice workflow nets [16]. So our goal is to design analysis algorithms that are applicable to arbitrary workflow nets, and provide a better runtime guarantee in the free-choice case. Typically, our algorithms have exponential worst-case complexity in the general case (which is unavoidable due to NP-hardness or PSPACE-hardness results), but polynomial complexity in the free-choice case.

A Petri net is free-choice if every pair of places has either the same set of output transitions or disjoint sets of output transitions. The consequence is that for every reachable marking, if some output transition of a place can fire, then *all* output transitions can fire, that is, the net can *freely choose* which output transition to fire. While free-choice Petri nets have a rich theory, almost all results concerning them are about the basic Petri net model, and do not apply to Petri nets enriched with data, time, or probabilities. In a series of papers, my co-authors and I have developed novel analysis techniques that overcome this problem. In the rest of the note we summarize this work. We first consider our work on reduction algorithms, and then our last paper on decomposition-based algorithms.

▶ Remark. Some of our results are formulated in terms of *deterministic negotiation diagrams*. In a nutshell, negotiation diagrams, introduced in [11], are workflow Petri nets that can be decomposed into communicating sequential Petri nets, a feature that makes them more amenable to theoretical study. A classical theorem of net theory shows that the connection between sound deterministic negotiation diagrams and sound free-choice workflow Petri nets is very tight: There are simple, polynomially computable translations between these two formalisms, which moreover only incur in a linear blow up. More details can be found in [7].

2 Reduction techniques

Reduction algorithms are a very efficient analysis technique for Petri nets and other business modeling formalisms, like EPCs and AND-XOR graphs (see for instance [21, 3, 9, 24]). A reduction algorithm consists of (a) a set of *reduction rules*, whose application allows one to simplify the workflow while preserving important properties, and (b) an algorithmic policy for selecting the next rule to be applied. The algorithm applies the rules exhaustively until it reaches an irreducible workflow net. For certain classes of nets and certain properties, the rules can be *complete*: They can reduce all workflows in the class satisfying the property, and only them, to some unique canonical workflow. Typically, this canonical workflow is the workflow consisting of one single transition with the initial and final places as only input and output place. If the policy followed by the algorithm is guaranteed to reach the canonical workflow, then the reduction algorithm becomes a decision algorithm for the property. Moreover, the algorithm decides the property without having to explore the reachability graph of the net at all.

A set of rules for free-choice Petri nets (not necessarily workflow nets) was presented in [6]. The rules preserve liveness and boundedness, two important properties of Petri nets, and are shown to be complete. In [3] these rules were applied to free-choice workflow Petri nets, and shown to be complete for the soundness property. This leads to a polynomial-time decision procedure for soundness of free-choice workflow nets, in sharp contrast with the PSPACE-hardness of deciding soundness for general workflow nets[1]. However, the rules of [6] have two problems. First, as shown in [13], they do not preserve properties concerning data or timing information. Moreover, as shown in [6], one of the rules is not correct for arbitrary workflow nets. More precisely, applying the rule to a sound, non-free-choice workflow net can make it unsound. In [13] we present a new set of surprisingly simple rules that overcomes these shortcomings. The rules can be applied to Petri nets in which tokens carry data. For example, a token in a certain place can be labeled with a natural number. Transitions collect a tuple of data from their input places, and apply a *transformer* to them, yielding a tuple of data that is sent to the output places.

The rules not only preserve soundness/unsoundness, but also the *input/output relation* of the workflow. This is the relation that assigns to every initial marking the set of final markings reachable from it (observe that initial markings differ only on the value of the token in the initial final place, and final markings on the value of the token in the final place). Therefore, the rules can be applied to decide any property of the input/output relation, and, by suitable reductions, other properties like the worst-case execution time of a given workflow. The rules also solve the second problem: contrary to the original rules, they can be applied to arbitrary workflow nets. Finally, the rules are still complete for free-choice workflow nets, in the sense that they reduce every sound free-choice workflow net to a workflow net with only one transition, but the same input/output relation.

The definitive description of the reduction algorithm, whose correctness proof and complexity analysis are rather complex, is given in [8], an extended and corrected version of [11, 12], which is currently under review. The algorithm completely reduces sound free-choice workflow nets by means of a sequence of rule applications of length at most cubic in the number of places and transitions of the net (in experiments the actual number of rule applications only grows linearly in the size of the net). Further, the sequence of reductions can be found in polynomial time. In [14] we apply this reduction algorithm to the problem of computing the expected cost of a workflow. For this, we define probabilistic workflow nets with costs, and enhance the reduction rules of [13] so that they preserve the expected cost. Using the results of [8] we prove that the expected cost of a free-choice workflow net can be computed in polynomial time.

3 Decomposition-based techniques

Our most recent work is presented in [15]. This paper generalizes the results of [13, 14]. It presents the most versatile analysis algorithm for free-choice workflow nets designed so far, among those that avoid the construction of the reachability graph. In particular, the reduction-based algorithms of [13, 14] can be recast as special cases of this general algorithm. The algorithm can be instantiated to solve problems about the time needed to execute a workflow, and also about its cost[2]. More precisely, the generic algorithm yields polynomial algorithms for the computation of the worst-case and the best-case execution time, and for the worst-case, best-case, and expected cost. It also provides a good algorithm for the computation of the expected time, although in this case the algorithm is not polynomial.

[1] The exact complexity depends on the specifics of the workflow model, for instance whether the workflow Petri net is assumed to be 1-safe or not.

[2] Notice the difference between time and cost: while the cost of executing two concurrent transitions is the sum of the costs, the time is the maximum of the times.

This is however to be expected, since the problem of deciding if the expected time exceeds a given bound is NP-complete, even for acyclic free-choice workflow nets [4].

The paper extends the classical lattice-based formalism for the static analysis of sequential flow-graphs, as presented for example in [20], to workflow Petri nets. A flow-graph consists of a set of nodes, modeling program points, and a set of edges, modeling program instructions, like assignments or guards. In the lattice-based approach one (i) defines a lattice $\mathcal{D}$ of dataflow informations corresponding to the possible results of the analysis to be conducted, (ii) assigns semantic transformers $[\![a]\!] : \mathcal{D} \to \mathcal{D}$ to each action a of the flow-graph, (iii) assigns to a path $a_1 \cdots a_n$ of the flow graph the functional composition $[\![a_n]\!] \circ \cdots \circ [\![a_1]\!]$ of the transformers, and (iv) defines the result of the analysis as the "Merge Over all Paths", i.e., the join of the transformers of all execution paths, usually called the MOP-solution or just the MOP of the dataflow problem. So performing an analysis amounts to computing the MOP of the flow-graph for the given lattice and the given transformers.

Katoen *et al.* have recently shown in [19, 10] that in order to adequately deal with quantitative analyses of concurrent systems one needs a semantics that distinguishes between the inherent nondeterminism of each sequential process, and the nondeterminism introduced by concurrency (the choice of the process that should perform the next step). Following these ideas, we introduce a semantics in which the latter is resolved by an external scheduler, and define the MOP for a given scheduler. The result of a dataflow analysis is then given by the infimum or supremum, depending on the application, of the MOPs for all possible schedulers.

In [15] we define the class of *Mazurkiewicz-invariant frameworks*. Loosely speaking, a framework is Mazurkiewicz-invariant if two executions of the workflow net that differ only in the order of execution of concurrent transitions have the same transformer. We prove a theorem showing a first important property of sound free-choice workflow nets, namely that the MOP is independent of the scheduler for Mazurkiewicz-invariant frameworks. This allows to compute the result of the analysis by fixing a scheduler, and computing the MOP for it. The main contribution of the paper is a method to compute the MOP of a framework for a sound free-choice workflow net. This is achieved by proving a novel and very powerful *decomposition theorem* showing that a sound free-choice workflow net is composed of smaller workflow subnets which are also sound. The algorithm iteratively identifies these subnets, computes their MOP, and replaces the complete subnet by one single transition with the same MOP.

Since the algorithm is generic, its complexity depends on the choice of the lattice and the transformers. However, we obtain a "concurrency-for-free" result: The runtime of the algorithm for computing the MOP for a sound free-choice workflow net is within a polynomial factor of the runtime for computing the MOP of a sequential flow-graph. Notice that this is the case even though the reachability graph of the workflow net – which can be seen as the sequential flow-graph equivalent to it – can be exponentially larger than the net itself.

The generic algorithm is more general than the reduction-based algorithms. More precisely, for every reduction-based algorithm, there is an instance of the generic algorithm with at most the same complexity. Further, using the generic algorithm we can solve in polynomial time quantitative problems for which no reduction algorithm existed so far. An example is the computation of the maximal number of tokens of a reachable marking of a sound free-choice workflow net. As shown in [4], the maximal number of tokens corresponds to the minimal number of resources that guarantee successful completion of the workflow, and is therefore an important parameter. Our generic decomposition-based algorithm can be instantiated to compute the maximal number of tokens in polynomial time[3].

[3] This research, conducted with Philipp Meyer and Hagen Völzer, is still unpublished.

References

1 Wil van der Aalst. The application of Petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

2 Wil van der Aalst and Kees Max van Hee. *Workflow management: models, methods, and systems*. MIT press, 2004.

3 Wil van der Aalst, Alexander Hirnschall, and Eric Verbeek. An alternative way to analyze workflow graphs. In *Advanced Information Systems Engineering*, volume 2348 of *Lecture Notes in Computer Science*, pages 535–552. Springer, 2002.

4 Mirela Botezatu, Hagen Völzer, and Lothar Thiele. The complexity of deadline analysis for workflow graphs with multiple resources. In *BPM*, volume 9850 of *Lecture Notes in Computer Science*, pages 252–268. Springer, 2016.

5 Jörg Desel and Thomas Erwin. Modeling, simulation and analysis of business processes. In *Business Process Management*, volume 1806 of *Lecture Notes in Computer Science*, pages 129–141. Springer, 2000.

6 Jörg Desel and Javier Esparza. *Free choice Petri nets*. Cambridge University Press, 2005.

7 Jörg Desel and Javier Esparza. Negotiations and Petri nets. *Transactions in Petri Nets and Other Models of Concurrency*, 11:203–225, 2016.

8 Jörg Desel, Javier Esparza, and Philipp Hoffmann. Negotiation as concurrency primitive. *CoRR*, abs/1612.07912, 2016.

9 Boudewijn van Dongen, Wil van der Aalst, and Eric Verbeek. Verification of EPCs: Using reduction rules and Petri nets. In *Advanced Information Systems Engineering*, pages 372–386. Springer, 2005.

10 Christian Eisentraut, Holger Hermanns, Joost-Pieter Katoen, and Lijun Zhang. A semantics for every GSPN. In *PETRI NETS*, volume 7927 of *Lecture Notes in Computer Science*, pages 90–109. Springer, 2013.

11 Javier Esparza and Jörg Desel. On negotiation as concurrency primitive. In *CONCUR*, volume 8052 of *Lecture Notes in Computer Science*, pages 440–454. Springer, 2013.

12 Javier Esparza and Jörg Desel. On negotiation as concurrency primitive II: Deterministic cyclic negotiations. In *FoSSaCS*, volume 8412 of *Lecture Notes in Computer Science*, pages 258–273. Springer, 2014.

13 Javier Esparza and Philipp Hoffmann. Reduction rules for colored workflow nets. In *FASE*, volume 9633 of *Lecture Notes in Computer Science*, pages 342–358, 2016.

14 Javier Esparza, Philipp Hoffmann, and Ratul Saha. Polynomial analysis algorithms for free choice probabilistic workflow nets. In *QEST*, volume 9826 of *Lecture Notes in Computer Science*, pages 89–104. Springer, 2016.

15 Javier Esparza, Anca Muscholl, and Igor Walukiewicz. Static analysis of deterministic negotiations. *CoRR*, abs/1704.04190, 2017. To appear in the Proceedings of LICS'17.

16 Dirk Fahland, Cédric Favre, Barbara Jobstmann, Jana Koehler, Niels Lohmann, Hagen Völzer, and Karsten Wolf. Instantaneous soundness checking of industrial business process models. In *BPM*, volume 5701 of *Lecture Notes in Computer Science*, pages 278–293. Springer, 2009.

17 Cédric Favre, Dirk Fahland, and Hagen Völzer. The relationship between workflow graphs and free-choice workflow nets. *Information Systems*, 47:197–219, 2015.

18 Cédric Favre, Hagen Völzer, and Peter Müller. Diagnostic information for control-flow analysis of Workflow Graphs (a.k.a. Free-Choice Workflow nets). In *TACAS*, volume 9636 of *Lecture Notes in Computer Science*, pages 463–479, 2016.

19 Joost-Pieter Katoen. GSPNs revisited: Simple semantics and new analysis algorithms. In *ACSD*, pages 6–11. IEEE Computer Society, 2012.

20 Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of program analysis*. Springer, 1999.

21 Wasim Sadiq and Maria E. Orlowska. Analyzing process models using graph reduction techniques. *Information systems*, 25(2):117–134, 2000.

22 Karsten Schmidt. LoLA: A low level analyser. In *ICATPN*, volume 1825 of *Lecture Notes in Computer Science*, pages 465–474, 2000. Current version available at `http://www.service-technology.org`.

23 Nikola Trcka, Wil M. P. van der Aalst, and Natalia Sidorova. Data-flow anti-patterns: Discovering data-flow errors in workflows. In *CAiSE 2009*, volume 5565 of *Lecture Notes in Computer Science*, pages 425–439, 2009.

24 Eric Verbeek, Moe Thandar Wynn, Wil van der Aalst, and Arthur ter Hofstede. Reduction rules for reset/inhibitor nets. *Journal of Computer and System Sciences*, 76(2):125–143, 2010.

Plan and Program Synthesis: A New Look at Some Old Problems

Sheila A. McIlraith

Department of Computer Science, University of Toronto, Toronto, Canada
sheila@cs.toronto.edu

Abstract

The proliferation of programmable devices, personal assistants, and autonomous systems presents fundamental challenges to the deployment of safe, predictable systems that can work together, interact seamlessly with humans, and that are *taskable* and *instructable* by people who may not know how to program. In this talk, we will revisit the classical problem of program synthesis through the lens of AI automated planning. We will present recent advances in AI automated planning principles and computational methods that support the synthesis of plans with goals and preferences specified in Linear Temporal Logic and Regular Expressions. Moving from automated planning in deterministic domains to planning in nondeterministic domains, we will explore the pathway to synthesizing programs that are taskable and instructable by exploiting state-of-the-art AI planning technology.

1998 ACM Subject Classification I.2.2 Automatic Programming

Keywords and phrases planning, program synthesis, linear temporal logic, regular expressions

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.3

Category Invited Talk

Possible and Certain Answers for Queries over Order-Incomplete Data[*][†]

Antoine Amarilli[1], Mouhamadou Lamine Ba[2], Daniel Deutch[3], and Pierre Senellart[4,5]

1 **LTCI, Télécom ParisTech, Université Paris-Saclay, Paris, France**
2 **University Alioune Diop of Bambey, Bambey, Senegal**
3 **Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel**
4 **DI ENS, ENS, CNRS, PSL Research University, Paris, France**
5 **Inria Paris, Paris, France**

Abstract

To combine and query ordered data from multiple sources, one needs to handle uncertainty about the possible orderings. Examples of such "order-incomplete" data include integrated event sequences such as log entries; lists of properties (e.g., hotels and restaurants) ranked by an unknown function reflecting relevance or customer ratings; and documents edited concurrently with an uncertain order on edits. This paper introduces a query language for order-incomplete data, based on the positive relational algebra with order-aware accumulation. We use partial orders to represent order-incomplete data, and study possible and certain answers for queries in this context. We show that these problems are respectively NP-complete and coNP-complete, but identify many tractable cases depending on the query operators or input partial orders.

1998 ACM Subject Classification H.2.1 [Database Management] Logical Design

Keywords and phrases certain answer, possible answer, partial order, uncertain data

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.4

1 Introduction

Many applications need to combine and transform ordered data (e.g., temporal data, rankings, preferences) from multiple sources. Examples include sequences of readings from multiple sensors, or log entries from different applications or machines, that must be combined to form a complete picture of events; rankings of restaurants and hotels published by different websites, their ranking function being often proprietary and unknown; and concurrent edits of shared documents, where the order of contributions made by different users needs to be merged. Even when the order of items from each individual source is known, the order across sources is often *uncertain*. For instance, even when sensor readings or log entries have timestamps, these may be ill-synchronized across sensors or machines; different websites may follow different rules and rank different hotels, so there are multiple ways to create a unified ranked list; concurrent document editions may be ordered in multiple ways. We say that the resulting information is *order-incomplete*.

This paper studies query evaluation over order-incomplete data in a relational setting [1]. Our running example is that of restaurants and hotels from travel websites, ranked according

[*] An extended version of this article can be found at `https://arxiv.org/abs/1707.07222`.
[†] This research was partially supported by the Israeli Science Foundation (grant 1636/13) and the Blavatnik ICRC.

to proprietary functions. An example query could compute the union of ranked lists of restaurants from distinct websites, or ask for a ranked list of pairs of a restaurant and a hotel in the same district. As we do not know how the proprietary order is defined, the query result may become *uncertain*: there may be multiple reasonable orderings of restaurants in the union result, or multiple orderings of restaurant–hotel pairs. We also study the application of order-aware *accumulation* to the query result, where each possible order may yield a different value: e.g., extracting only the highest ranked pairs, concatenating their names, or assessing the attractiveness of a district based on its best restaurants and hotels.

Our approach is to handle this uncertainty through the classical notions of *possible and certain answers*. First, whenever there is a *certain answer* to the query – i.e., there is only one possible order on query results or one accumulation result – which is obtained no matter the order on the input and in intermediate results, we should present it to the user, who can then browse through the ordered query results (as is typically done in absence of uncertainty, using constructs such as SQL's `ORDER BY`). Certain answers can arise even in non-trivial cases where the combination of input data admits many possible orders: consider user queries that select only a small interesting subset of the data (for which the ordering happens to be certain), or a short summary obtained through accumulation over large data. In many other cases, the different orders on input data or the uncertainty caused by the query may lead to several *possible answers*. In this case, it is still of interest (and non-trivial) to verify whether an answer is possible, e.g., to check whether a given ranking of hotel–restaurant pairs is consistent with a combination of other rankings (the latter done through a query). Thus, we study the problems of deciding whether a given answer is *certain*, and whether it is *possible*.

We note that users may wish to focus on the position of some tuples of interest (e.g., "is it possible/certain that a particular pair of restaurant-hotel is ranked first?", or "is it possible/certain that restaurant A is ranked above restaurant B?). We show these questions may be expressed in our framework through proper choices of accumulation functions.

Main contributions. We introduce a query language with accumulation for order-incomplete data, which generalizes the positive relational algebra [1] with aggregation as the outermost operation. We define a bag semantics for this language, without assuming that a single choice of order can be made (unlike, e.g., rank aggregation [15]): we use *partial orders* to represent all orders that are consistent with the input data. We then undertake the first general study of the *complexity of possible and certain answers for queries over such data*. We show that these problems are respectively NP-complete and coNP-complete, the main difficulties being the existence of duplicate tuple values in the data and the use of order-aware accumulation. Fortunately, we can show many realistic tractable cases: certainty is in PTIME without accumulation, and both problems are tractable under reasonable restrictions on the input and on the query. The rest of this paper is organized as follows. In Section 2, we introduce our data model and our query language. We define and exemplify the problems of possible and certain answers in Section 3. We then study their complexity, first in the general case (Section 4), then in restricted settings that ensure tractability (Sections 5 and 6). We study extensions to the language, namely duplicate elimination and group-by, in Section 7. We compare our model and results with related work in Section 8, and conclude in Section 9. Proof sketches of some important results are given in an appendix, for lack of space.

2 Data Model and Query Language

We fix a countable set of values $\mathcal{D}$ that includes $\mathbb{N}$ and infinitely many values not in $\mathbb{N}$. A *tuple t over $\mathcal{D}$* of *arity* $\mathrm{a}(t)$ is an element of $\mathcal{D}^{\mathrm{a}(t)}$, denoted $\langle v_1, \ldots, v_{\mathrm{a}(t)} \rangle$. The simplest notion

restname	*distr*	
Gagnaire	8	
TourArgent	5	↓

(a) *Rest* table

hotelname	*distr*	
Mercure	5	
Balzac	8	
Mercure	12	↓

(b) *Hotel* table

hotelname	*distr*	
Balzac	8	
Mercure	5	
Mercure	12	↓

(c) *Hotel*$_2$ table

Figure 1 Running example: Paris restaurants and hotels.

$$\begin{tikzcd}
 & \langle \mathrm{TA}, 5, \mathrm{B}, 8 \rangle & \\
\langle \mathrm{G}, 8, \mathrm{B}, 8 \rangle \arrow[ur] & & \langle \mathrm{TA}, 5, \mathrm{M}, 5 \rangle \arrow[ul] \\
 & \langle \mathrm{G}, 8, \mathrm{M}, 5 \rangle \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Figure 2 Example 2.

$$\begin{tikzcd}
\underset{e}{\mathrm{jp}} & \underset{f}{\mathrm{jp}} \\
\underset{c}{\mathrm{fr}} \arrow[u] & \underset{d}{\mathrm{it}} \arrow[u] \arrow[ul] \\
\underset{a}{\mathrm{fr}} \arrow[u] & \underset{b}{\mathrm{it}} \arrow[ul]
\end{tikzcd}$$

Figure 3 Example 11.

of ordered relations are then *list relations* [11, 12]: a list relation of arity $n \in \mathbb{N}$ is an ordered list of tuples over $\mathcal{D}$ of arity n (where the same tuple value may appear multiple times). List relations impose a single order over tuples, but when one combines (e.g., unions) them, there may be multiple plausible ways to order the results.

We thus introduce *partially ordered relations* (*po-relations*). A po-relation $\Gamma = (ID, T, <)$ of arity $n \in \mathbb{N}$ consists of a finite set of *identifiers* ID (chosen from some infinite set closed under product), a *strict partial order* $<$ on ID, and a (generally non injective) mapping T from ID to $\mathcal{D}^n$. The actual identifiers do not matter, but we need them to refer to occurrences of the same tuple value. Hence, we always consider po-relations *up to isomorphism*, where $(ID, T, <)$ and $(ID', T', <')$ are *isomorphic* iff there is a bijection $\varphi : ID \to ID'$ such that $T'(\varphi(id)) = T(id)$ for all $id \in ID$, and $\varphi(id_1) <' \varphi(id_2)$ iff $id_1 < id_2$ for all $id_1, id_2 \in ID$.

A special case of po-relations are *unordered po-relations* (or *bag relations*), where $<$ is empty: we write them (ID, T). The *underlying bag relation* of $\Gamma = (ID, T, <)$ is (ID, T).

The point of po-relations is to represent *sets* of list relations. Formally, a *linear extension* $<'$ of $<$ is a total order on ID such that for each $x < y$ we have $x <' y$. The *possible worlds* $pw(\Gamma)$ of Γ are then defined as follows: for each linear extension $<'$ of $<$, writing ID as $id_1 <' \cdots <' id_{|ID|}$, the list relation $(T(id_1), \ldots, T(id_{|ID|}))$ is in $pw(\Gamma)$. As T is generally not injective, two different linear extensions may yield the same list relation. Po-relations can thus model uncertainty over the *order* of tuples (but not on their *value*: the underlying bag relation is always certain).

Query language. We now define a bag semantics for *positive relational algebra* operators, to manipulate po-relations with queries. The positive relational algebra, written PosRA, is a standard query language for relational data [1]. We will extend PosRA later in this section with *accumulation*, and add further extensions in Section 7. Each PosRA operator applies to po-relations and computes a new po-relation; we present them in turn.

The selection operator restricts the relation to a subset of its tuples, and the order is the restriction of the input order. The *tuple predicates* allowed in selections are Boolean combinations of equalities and inequalities, which can use tuple attributes and values in $\mathcal{D}$.

selection: For any po-relation $\Gamma = (ID, T, <)$ and tuple predicate ψ, we define the selection $\sigma_\psi(\Gamma) := (ID', T_{|ID'}, <_{|ID'})$ where $ID' := \{id \in ID \mid \psi(T(id)) \text{ holds}\}$.

The projection operator changes tuple values in the usual way, but keeps the original tuple ordering in the result, and retains all copies of duplicate tuples (following our *bag semantics*):

projection: For a po-relation $\Gamma = (ID, T, <)$ and attributes $A_1, \ldots, A_n$, we define the projection $\Pi_{A_1,\ldots,A_n}(\Gamma) := (ID, T', <)$ where T' maps each $id \in ID$ to $\Pi_{A_1,\ldots,A_n}(T(id))$.

As for union, we impose the minimal order constraints that are compatible with those of the inputs. We use the *parallel composition* [7] of two partial orders $<$ and $<'$ on disjoint sets ID and ID', i.e., the partial order $<'' := (< \parallel <')$ on $ID \cup ID'$ defined by: every $id \in ID$ is incomparable for $<''$ with every $id' \in ID'$; for each $id_1, id_2 \in ID$, we have $id_1 <'' id_2$ iff $id_1 < id_2$; for each $id'_1, id'_2 \in ID'$, we have $id'_1 <'' id'_2$ iff $id'_1 <' id'_2$.

union: Let $\Gamma = (ID, T, <)$ and $\Gamma' = (ID', T', <')$ be two po-relations of the same arity. We assume that the identifiers of Γ' have been renamed if necessary to ensure that ID and ID' are disjoint. We then define $\Gamma \cup \Gamma' := (ID \cup ID', T'', < \parallel <')$, where T'' maps $id \in ID$ to $T(id)$ and $id' \in ID'$ to $T'(id')$.

The union result $\Gamma \cup \Gamma'$ does not depend on the exact definition of Γ'', i.e., it is unique up to isomorphism. Our definition also implies that $\Gamma \cup \Gamma$ is different from Γ, as per bag semantics. In particular, when Γ and Γ' have only one possible world, $\Gamma \cup \Gamma'$ usually does not.

We next introduce two possible product operators. First, the *direct product* [40] $<_{\texttt{DIR}} := (< \times_{\texttt{DIR}} <')$ of two partial orders $<$ and $<'$ on sets ID and ID' is defined by $(id_1, id'_1) <_{\texttt{DIR}} (id_2, id'_2)$ for each $(id_1, id'_1), (id_2, id'_2) \in ID \times ID'$ iff $id_1 < id_2$ and $id'_1 <' id'_2$. We define the direct product operator over po-relations accordingly: two identifiers in the product are comparable only if *both components* of both identifiers compare in the same way.

direct product: For any po-relations $\Gamma = (ID, T, <)$ and $\Gamma' = (ID', T', <')$, remembering that the sets of possible identifiers is closed under product, we let $\Gamma \times_{\texttt{DIR}} \Gamma' := (ID \times ID', T'', < \times_{\texttt{DIR}} <')$, where T'' maps each $(id, id') \in ID \times ID'$ to the *concatenation* $\langle T(id), T'(id')\rangle$.

Again, the direct product result often has multiple possible worlds even when inputs do not.

The second product operator uses the lexicographic product (or *ordinal product* [40]) $<_{\texttt{LEX}} := (< \times_{\texttt{LEX}} <')$ of two partial orders $<$ and $<'$, defined by $(id_1, id'_1) <_{\texttt{LEX}} (id_2, id'_2)$ for all $(id_1, id'_1), (id_2, id'_2) \in ID \times ID'$ iff either $id_1 < id_2$, or $id_1 = id_2$ and $id'_1 <' id'_2$.

lexicographic product: For any po-relations $\Gamma = (ID, T, <)$ and $\Gamma' = (ID', T', <')$, we define $\Gamma \times_{\texttt{LEX}} \Gamma'$ as $(ID \times ID', T'', < \times_{\texttt{LEX}} <')$ with T'' defined like for direct product.

Last, we define the *constant expressions* that we allow:

const:
- for any tuple t, the singleton po-relation $[t]$ has only one tuple with value t;
- for any $n \in \mathbb{N}$, the po-relation $[\leqslant n]$ has arity 1 and has $pw([\leqslant n]) = \{(1, \ldots, n)\}$.

A natural question is then to determine whether any of our operators is subsumed by the others, but we show that this is not the case:

▶ **Theorem 1.** *No PosRA operator can be expressed through a combination of the others.*

We have now defined a semantics on po-relations for each PosRA operator. We define a *PosRA query* in the expected way, as a query built from these operators and from relation names. Calling *schema* a set $\mathcal{S}$ of relation names and arities, with an attribute name for each position of each relation, we define a *po-database* D as having a po-relation $D[R]$ of the correct arity for each relation name R in $\mathcal{S}$. For a po-database D and a PosRA query Q we denote by $Q(D)$ the po-relation obtained by evaluating Q over D.

▶ **Example 2.** The po-database D in Figure 1 contains information about restaurants and hotels in Paris: each po-relation has a total order (from top to bottom) according to customer ratings from a given travel website, and for brevity we do not represent identifiers.

Let $Q := \mathit{Rest} \times_{\mathtt{DIR}} (\sigma_{\mathit{distr} \neq \text{“12”}}(\mathit{Hotel}))$. Its result $Q(D)$ has two possible worlds: $(\langle\text{G}, 8, \text{M}, 5\rangle, \langle\text{G}, 8, \text{B}, 8\rangle, \langle\text{TA}, 5, \text{M}, 5\rangle, \langle\text{TA}, 5, \text{B}, 8\rangle)$, $(\langle\text{G}, 8, \text{M}, 5\rangle, \langle\text{TA}, 5, \text{M}, 5\rangle, \langle\text{G}, 8, \text{B}, 8\rangle, \langle\text{TA}, 5, \text{B}, 8\rangle)$. In a sense, these *list relations* of hotel–restaurant pairs are *consistent* with the order in D: we do not know how to order two pairs, except when both the hotel and restaurant compare in the same way. The *po-relation* $Q(D)$ is represented in Figure 2 as a Hasse diagram (ordered from bottom to top), again writing tuple values instead of tuple identifiers for brevity.

Consider now $Q' := \Pi(\sigma_{\mathit{Rest.distr}=\mathit{Hotel.distr}}(Q))$, where Π projects out $\mathit{Hotel.distr}$. The possible worlds of $Q'(D)$ are $(\langle\text{G}, \text{B}, 8\rangle, \langle\text{TA}, \text{M}, 5\rangle)$ and $(\langle\text{TA}, \text{M}, 5\rangle, \langle\text{G}, \text{B}, 8\rangle)$, intuitively reflecting two different opinions on the order of restaurant–hotel pairs in the same district. Defining Q'' similarly to Q' but replacing $\times_{\mathtt{DIR}}$ by $\times_{\mathtt{LEX}}$ in Q, we have $\mathit{pw}(Q''(D)) = (\langle\text{G}, \text{B}, 8\rangle, \langle\text{TA}, \text{M}, 5\rangle)$.

We conclude by observing that we can efficiently evaluate PosRA queries on po-relations:

▶ **Proposition 3.** *For any fixed PosRA query Q, given a po-database D, we can construct the po-relation $Q(D)$ in polynomial time in the size of D (the polynomial degree depends on Q).*

Accumulation. We now enrich PosRA with order-aware *accumulation* as the outermost operation, inspired by *right accumulation* and *iteration* in list programming, and *aggregation* in relational databases. We fix a *monoid* $(\mathcal{M}, \oplus, \varepsilon)$ for accumulation and define:

▶ **Definition 4.** For $n \in \mathbb{N}$, let $h : \mathcal{D}^n \times \mathbb{N}^* \to \mathcal{M}$ be a function called an arity-n *accumulation map*. We call $\text{accum}_{h,\oplus}$ an *arity-n accumulation operator*; its result $\text{accum}_{h,\oplus}(L)$ on an arity-n list relation $L = (t_1, \ldots, t_n)$ is $h(t_1, 1) \oplus \cdots \oplus h(t_n, n)$, and it is ε on an empty L. For complexity purposes, we *always* require accumulation operators to be *PTIME-evaluable*, i.e., given any list relation L, we can compute $\text{accum}_{h,\oplus}(L)$ in PTIME.

The accumulation operator maps the tuples with h to $\mathcal{M}$, where accumulation is performed with $\oplus$. The map h may use its second argument to take into account the absolute position of tuples in L. In what follows, we omit the arity of accumulation when clear from context.

The PosRA$^{\text{acc}}$ language. We define the language PosRA$^{\text{acc}}$ that contains all queries of the form $Q = \text{accum}_{h,\oplus}(Q')$, where $\text{accum}_{h,\oplus}$ is an accumulation operator and Q' is a PosRA query. The *possible results* of Q on a po-database D, denoted $Q(D)$, is the set of results obtained by applying accumulation to each possible world of $Q'(D)$, namely:

▶ **Definition 5.** For a po-relation Γ, we define: $\text{accum}_{h,\oplus}(\Gamma) := \{\text{accum}_{h,\oplus}(L) \mid L \in \mathit{pw}(\Gamma)\}$.

Of course, accumulation has exactly one result whenever the operator $\mathrm{accum}_{h,\oplus}$ does not depend on the order of input tuples: this covers, e.g., the standard sum, min, max, etc. Hence, we focus on accumulation operators which *depend on the order of tuples* (e.g., defining $\oplus$ as concatenation), so there may be more than one accumulation result:

▶ **Example 6.** As a first example, let $\mathit{Ratings}(\mathit{user}, \mathit{restaurant}, \mathit{rating})$ be an *unordered* po-relation describing the numerical ratings given by users to restaurants, where each user rated each restaurant at most once. Let $\mathit{Relevance}(\mathit{user})$ be a po-relation giving a partially-known ordering of users to indicate the relevance of their reviews. We wish to compute a *total rating* for each restaurant which is given by the sum of its reviews weighted by a PTIME-computable weight function w. Specifically, $w(i)$ gives a nonnegative weight to the rating of the i-th most relevant user. Consider $Q_1 := \mathrm{accum}_{h_1,+}(\sigma_\psi(\mathit{Relevance} \times_{\mathtt{LEX}} \mathit{Ratings}))$ where we set $h_1(t, n) := t.\mathit{rating} \times w(n)$, and where ψ is the tuple predicate: $\mathit{restaurant} = \text{“Gagnaire”} \wedge \mathit{Ratings.user} = \mathit{Relevance.user}$. The query Q_1 gives the total rating of “Gagnaire”, and each possible world of $\mathit{Relevance}$ may lead to a different accumulation result.

As a second example, consider an unordered po-relation $\mathit{HotelCity}(\mathit{hotel}, \mathit{city})$ indicating in which city each hotel is located, and consider a po-relation $\mathit{City}(\mathit{city})$ which is (partially) ranked by a criterion such as interest level, proximity, etc. Now consider the query $Q_2 := \mathrm{accum}_{h_2,\mathrm{concat}}(\Pi_{\mathit{hotel}}(Q_2'))$, where $Q_2' := \sigma_{\mathit{City.city}=\mathit{HotelCity.city}}(\mathit{City} \times_{\mathtt{LEX}} \mathit{HotelCity})$, where $h_2(t, n) := t$, and where “concat” denotes standard string concatenation. Q_2 concatenates the hotel names according to the preference order on the city where they are located, allowing any possible order between hotels of the same city and between hotels in incomparable cities.

3 Possibility and Certainty

Evaluating a PosRA or PosRA$^{\mathrm{acc}}$ query Q on a po-database D yields a *set of possible results*: for PosRA$^{\mathrm{acc}}$, it yields an explicit set of accumulation results, and for PosRA, it yields a po-relation that represents a set of possible worlds (list relations). The uncertainty among the results may be due to the order of the input relations being partial, due to uncertainty yielded by the query, or both. In some cases, there is only one possible result, i.e., a *certain* answer. In other cases, we may wish to examine multiple *possible* answers. We thus define:

▶ **Definition 7** (Possibility and Certainty). Let Q be a PosRA query, D be a po-database, and L a list relation. The *possibility problem* (`POSS`) asks if $L \in pw(Q(D))$, i.e., if L is a possible result. The *certainty problem* (`CERT`) asks if $pw(Q(D)) = \{L\}$, i.e., if L is the only possible result. Likewise, if Q is a PosRA$^{\mathrm{acc}}$ query with accumulation monoid $\mathcal{M}$, for a result $v \in \mathcal{M}$, the `POSS` problem asks whether $v \in Q(D)$, and `CERT` asks whether $Q(D) = \{v\}$.

Discussion. For PosRA$^{\mathrm{acc}}$, our definition follows the usual notion of possible and certain answers in data integration [28] and incomplete information [30]. For PosRA, we ask for possibility or certainty of an *entire* output list relation, i.e., *instance possibility and certainty* [3]. We now justify that these notions are useful and discuss more “local” alternatives.

First, as we exemplify below, the output of a query may be certain even for complex queries and uncertain input. It is important to identify such cases and present the user with the certain answer in full, like order-by query results in current DBMSs. Our `CERT` problem is useful for this task, because we can use it to decide if a certain output exists, and if yes, we can compute it in PTIME (by choosing any linear extension). However, `CERT` is a challenging problem to solve, because of duplicate values (see “Technical difficulties” below).

▶ **Example 8.** Consider the po-database D of Figure 1 with the po-relations *Rest* and $Hotel_2$. To find recommended pairs of hotels and restaurants in the same district, the user can write $Q := \sigma_{Rest.distr=Hotel_2.distr}(Rest \times_{\mathtt{DIR}} Hotel_2)$. Evaluating $Q(D)$ yields only one possible world, namely, the list relation $(\langle G, 8, B, 8\rangle, \langle TA, 5, M, 5\rangle)$, which is a *certain* result.

This could also happen with larger input relations. Imagine for example that we join hotels and restaurants to find pairs of a hotel and a restaurant located in that hotel. The result can be certain if the relative ranking of the hotels and of their restaurants agree.

If there is no certain answer, deciding possibility of an instance may be considered as "best effort". It can be useful, e.g., to check if a list relation (obtained from another source) is consistent with a query result. For example, we may wish to check if a website's ranking of hotel–restaurant pairs is *consistent* with the preferences expressed in its rankings for hotels and restaurants, to detect when a pair is ranked higher than its components would warrant.

When there is no overall certain answer, or when we want to check the possibility of some aggregate property of the relation, we can use a $\text{PosRA}^{\text{acc}}$ query. In particular, in addition to the applications of Example 6, accumulation allows us to encode alternative notions of `POSS` and `CERT` for *PosRA* queries, and to express them as `POSS` and `CERT` for $\text{PosRA}^{\text{acc}}$. For example, instead of possibility or certainty for a full relation, we can express possibility or certainty of the *location*[1] of particular tuples of interest:

▶ **Example 9.** With accumulation we can model *position-based selection* queries. Consider for instance a *top-k* operator on list relations, which retrieves a list relation of the first k tuples. For a po-relation, the set of results is all possible such list relations. We can implement *top-k* as $\text{accum}_{h_3,\text{concat}}$ with $h_3(t, n)$ being (t) for $n \leqslant k$ and ε otherwise, and with concat being list concatenation. We can similarly compute *select-at-k*, i.e., return the tuple at position k, via $\text{accum}_{h_4,\text{concat}}$ with $h_4(t, n)$ being (t) for $n = k$ and ε otherwise.

Accumulation can also be used for a *tuple-level comparison*. To check whether the first occurrence of a tuple t_1 precedes any occurrence of t_2, we define h_5 for all $n \in \mathbb{N}$ by $h_5(t_1, n) := \top$, $h_5(t_2, n) := \bot$ and $h_5(t, n) := \varepsilon$ for $t \neq t_1, t_2$, and a monoid operator $\oplus$ such that $\top \oplus \top = \top \oplus \bot = \top$, $\bot \oplus \bot = \bot \oplus \top = \bot$: the result is ε if neither t_1 not t_2 is present, $\top$ if the first occurrence of t_1 precedes any occurrence of t_2, $\bot$ otherwise.

We study the complexity of these variants in Section 6. We now give examples of their use:

▶ **Example 10.** Consider $Q = \Pi_{distr}(\sigma_{Rest.distr=Hotel.distr}(Rest \times_{\mathtt{DIR}} Hotel))$, which computes ordered recommendations of districts including both hotels and restaurants. Using accumulation as in Example 9, the user can compute the best district to stay in with $Q' = \text{top-1}(Q)$. If Q' has a certain answer, then there is a dominating hotel–restaurant pair in this district, which answers the user's need. If there is no certain answer, `POSS` allows the user to determine the *possible* top-1 districts.

We can also use `POSS` and `CERT` for $\text{PosRA}^{\text{acc}}$ queries to restrict attention to *tuples* of interest. If the user hesitates between districts 5 and 6, they can apply tuple-level comparison to see whether the best pair of district 5 may be better (or is always better) than that of 6.

Technical difficulties. The main challenge to solve `POSS` and `CERT` for a PosRA query Q on an input po-database D is that the tuple values of the desired result L may occur multiple times in the po-relation $Q(D)$, making it hard to match L and $Q(D)$. In other words, even

[1] Remember that the *existence* of a tuple is not order-dependent and thus vacuous in our setting.

though we may compute the po-relation $Q(D)$ in PTIME (by Proposition 3) and present it to the user, they still cannot easily "read" possible and certain answers out of the po-relation:

▶ **Example 11.** Consider a po-relation $\Gamma = (ID, T, <)$ with $ID = \{id_{\mathrm{a}}, id_{\mathrm{b}}, id_{\mathrm{c}}, id_{\mathrm{d}}, id_{\mathrm{e}}, id_{\mathrm{f}}\}$, with $T(id_{\mathrm{a}}) := \langle \mathrm{Gagnaire}, \mathrm{fr}\rangle$, $T(id_{\mathrm{b}}) := \langle \mathrm{Italia}, \mathrm{it}\rangle$, $T(id_{\mathrm{c}}) := \langle \mathrm{TourArgent}, \mathrm{fr}\rangle$, $T(id_{\mathrm{d}}) := \langle \mathrm{Verdi}, \mathrm{it}\rangle$, $T(id_{\mathrm{e}}) := \langle \mathrm{Tsukizi}, \mathrm{jp}\rangle$, $T(id_{\mathrm{f}}) := \langle \mathrm{Sola}, \mathrm{jp}\rangle$, and with $id_{\mathrm{a}} < id_{\mathrm{c}}$, $id_{\mathrm{b}} < id_{\mathrm{c}}$, $id_{\mathrm{c}} < id_{\mathrm{e}}$, $id_{\mathrm{d}} < id_{\mathrm{e}}$, and $id_{\mathrm{d}} < id_{\mathrm{f}}$. Intuitively, Γ describes a preference relation over restaurants, with their name and the type of their cuisine. Consider the PosRA query $Q := \Pi(\Gamma)$ that projects Γ on type; we illustrate the result (with the original identifiers) in Figure 3. Let L be the list relation $(\mathrm{it}, \mathrm{fr}, \mathrm{jp}, \mathrm{it}, \mathrm{fr}, \mathrm{jp})$, and consider `POSS` for Q, Γ, and L.

We have that $L \in pw(Q(\Gamma))$, as shown by the linear extension $id_{\mathrm{d}} <' id_{\mathrm{a}} <' id_{\mathrm{f}} <' id_{\mathrm{b}} <' id_{\mathrm{c}} <' id_{\mathrm{e}}$ of $<$. However, this is hard to see, because each of it, fr, jp appears more than once in the candidate list as well as in the po-relation; there are thus multiple ways to "map" the elements of the candidate list to those of the po-relation, and only some of these mappings lead to the existence of a corresponding linear extension. It is also challenging to check if L is a certain answer: here, it is not, as there are other possible answers, e.g.: $(\mathrm{it}, \mathrm{fr}, \mathrm{fr}, \mathrm{it}, \mathrm{jp}, \mathrm{jp})$.

For PosRA$^{\mathrm{acc}}$ queries, this technical difficulty is even accrued because of the need to figure out the possible ways in which the desired accumulation result can be obtained.

4 General Complexity Results

We have defined the PosRA and PosRA$^{\mathrm{acc}}$ query languages, and defined and motivated the problems `POSS` and `CERT`. We now start the study of their complexity, which is the main technical contribution of our paper. We will always study their *data complexity*[2], where the query Q is fixed: in particular, for PosRA$^{\mathrm{acc}}$, the accumulation map and monoid, which we assumed to be PTIME-evaluable, is fixed as part of the query, though it is allowed to be infinite. The input to `POSS` and `CERT` for the fixed query Q is the po-database D and the candidate result (a list relation for PosRA, an accumulation result for PosRA$^{\mathrm{acc}}$).

Possibility. We start with `POSS`, which we show to be NP-complete in general.

▶ **Theorem 12.** *The `POSS` problem is in NP for any PosRA or PosRA$^{\mathrm{acc}}$ query. Further, there exists a PosRA query and a PosRA$^{\mathrm{acc}}$ query for which the `POSS` problem is NP-complete.*

In fact, as we will later point out, hardness holds even for quite a restrictive setting, with a more intricate proof: see Theorem 18.

Certainty. We show that `CERT` is coNP-complete for PosRA$^{\mathrm{acc}}$:

▶ **Theorem 13.** *The `CERT` problem is in coNP for any PosRA$^{\mathrm{acc}}$ query, and there is a PosRA$^{\mathrm{acc}}$ query for which it is coNP-complete.*

For PosRA queries, however, we show that `CERT` is in PTIME. As we will see later, this follows from the tractability of `CERT` for PosRA$^{\mathrm{acc}}$ on *cancellative monoids* (Theorem 26).

▶ **Theorem 14.** *`CERT` is in PTIME for any PosRA query.*

We next identify further tractable cases, first for PosRA and then for PosRA$^{\mathrm{acc}}$.

[2] In *combined complexity*, with Q part of the input, `POSS` and `CERT` are easily seen to be respectively NP-hard and coNP-hard, by reducing from the evaluation of Boolean conjunctive queries (which is NP-hard in data complexity [1]) even without order.

5 Tractable Cases for POSS on PosRA Queries

We show that POSS is tractable for PosRA queries if we restrict the allowed operators and if we bound some order-theoretic parameters of the input po-database, such as *poset width*.

We call $\mathrm{PosRA}_{\mathtt{LEX}}$ the fragment of PosRA that disallows the $\times_{\mathtt{DIR}}$ operator, but allows all other operators (including $\times_{\mathtt{LEX}}$). We also define $\mathrm{PosRA}_{\mathtt{DIR}}$ that disallows $\times_{\mathtt{LEX}}$ but not $\times_{\mathtt{DIR}}$.

Totally ordered inputs. We start by the natural case where the individual po-relations are *totally ordered*, i.e., their order relation is a total order (so they actually represent a list relation). This applies to situations where we integrate data from multiple sources that are certain (totally ordered), and where uncertainty only results from the integration query (so that the result may still have exponentially many possible worlds, e.g., the *union* of two total orders has exponentially many possible interleavings). In a sense, the $\times_{\mathtt{DIR}}$ operator is the one introducing the most uncertainty and "complexity" in the result, so we consider the fragment $\mathrm{PosRA}_{\mathtt{LEX}}$ of PosRA queries without $\times_{\mathtt{DIR}}$, and show:

▶ **Theorem 15.** *POSS is in PTIME for $\mathrm{PosRA}_{\mathtt{LEX}}$ queries if input po-relations are totally ordered.*

In fact, we can show tractability for relations of bounded *poset width*:

▶ **Definition 16** ([36])**.** An *antichain* in a po-relation $\Gamma = (ID, T, <)$ is a set $A \subseteq ID$ of pairwise incomparable tuple identifiers. The *width* of Γ is the size of its largest antichain. The *width* of a po-database is the maximal width of its po-relations.

In particular, totally ordered po-relations have width 1, and unordered po-relations have a width equal to their size (number of tuples); the width of a po-relation can be computed in PTIME [18]. Po-relations of low width are a common practical case: they cover, for instance, po-relations that are totally ordered except for a few tied identifiers at each level. We show:

▶ **Theorem 17.** *For any fixed $k \in \mathbb{N}$ and fixed $\mathrm{PosRA}_{\mathtt{LEX}}$ query Q, the POSS problem for Q is in PTIME when all po-relations of the input po-database have width $\leqslant k$.*

We last justify our choice of disallowing the $\times_{\mathtt{DIR}}$ product. Indeed, if we allow $\times_{\mathtt{DIR}}$, then POSS is hard on totally ordered po-relations, even if we disallow $\times_{\mathtt{LEX}}$:

▶ **Theorem 18.** *There is a $\mathrm{PosRA}_{\mathtt{DIR}}$ query for which the POSS problem is NP-complete even when the input po-database is restricted to consist only of totally ordered po-relations.*

Unordered inputs We now show the tractability of POSS for *unordered* input relations, i.e., po-relations that allow all possible orderings over their tuples. This applies, e.g., to contexts where the order on input tuples is irrelevant or unknown; all order information must then be imposed by the (fixed) query, using the ordered constant relations $[\leqslant \bullet]$. We show:

▶ **Theorem 19.** *POSS is in PTIME for any PosRA query if input po-relations are unordered.*

Here again we prove a more general result, capturing the case where the input is "almost unordered". We introduce for this purpose a novel order-theoretic notion, *ia-width*, which decomposes the relation in classes of indistinguishable sets of incomparable elements.

▶ **Definition 20.** Given a poset $(V, <)$, a subset $S \subseteq V$ is an *indistinguishable antichain* if it is both an antichain (there are no $x, y \in S$ such that $x < y$) and an *indistinguishable set* (or *interval* [17]): for all $x, y \in S$ and $z \in V \backslash S$, $x < z$ iff $y < z$, and $z < x$ iff $z < y$.

An *indistinguishable antichain partition* (ia-partition) of a poset is a partition of its domain into indistinguishable antichains. The *cardinality* of such a partition is its number of classes. The *ia-width* of a poset (or po-relation) is the cardinality of its smallest ia-partition. The ia-width of a po-database is the maximal ia-width of its relations.

Hence, any po-relation Γ has ia-width at most $|\Gamma|$, and unordered relations have an ia-width of 1. Po-relations may have low ia-width in practice if order is completely unknown except for a few comparability pairs given by users, or when objects of a constant number of types are ordered based only on some order on the types. We show that ia-width, like width, can be computed in PTIME, and that bounding it ensures tractability (for all PosRA):

▶ **Proposition 21.** *The ia-width of any poset, and a corresponding ia-partition, can be computed in PTIME.*

▶ **Theorem 22.** *For any fixed $k \in \mathbb{N}$ and fixed PosRA query Q, the* `POSS` *problem for Q is in PTIME when all po-relations of the input po-database have ia-width $\leqslant k$.*

Mixing both kinds of relations. We have shown the tractability of `POSS` assuming constant width (only for $\mathrm{PosRA}_{\mathtt{LEX}}$ queries) or assuming constant ia-width. A natural question is then whether we can allow both *totally ordered* and *unordered* po-relations. For instance, we may combine sources whose order is fully unknown or irrelevant, with sources that are completely ordered (or almost totally ordered). More generally, can we allow both bounded-width and bounded-ia-width relations? We show that this is the case if we disallow *both* product operators, i.e., restrict to the language $\mathrm{PosRA}_{\mathrm{no}\times}$ of PosRA queries with no product.

▶ **Theorem 23.** *For any fixed $k \in \mathbb{N}$ and fixed $PosRA_{\mathrm{no}\times}$ query Q, the* `POSS` *problem for Q is in PTIME when all po-relations of the input po-database have either ia-width $\leqslant k$ or width $\leqslant k$.*

Disallowing product is severe, but we can still integrate sources by taking the *union* of their tuples, selecting subsets, and modifying tuple values with projection. In fact, allowing product makes `POSS` intractable when allowing both unordered and totally ordered input:

▶ **Theorem 24.** *There is a $PosRA_{\mathtt{LEX}}$ query and a $PosRA_{\mathtt{DIR}}$ query for which the* `POSS` *problem is NP-complete even when the input po-database is restricted to consist only of one totally ordered and one unordered po-relation.*

6 Tractable Cases for Accumulation Queries

We next study tractable cases for `POSS` and `CERT` in presence of accumulation.

Cancellative monoids. We first consider a natural restriction on the accumulation function:

▶ **Definition 25** ([23])**.** For any monoid $(\mathcal{M}, \oplus, \varepsilon)$, we call $a \in \mathcal{M}$ *cancellable* if, for all $b, c \in \mathcal{M}$, we have that $a \oplus b = a \oplus c$ implies $b = c$, and we also have that $b \oplus a = c \oplus a$ implies $b = c$. We call $\mathcal{M}$ a *cancellative monoid* if all its elements are cancellable.

Many interesting monoids are cancellative; in particular, this is the case of both monoids in Example 6. More generally, all *groups* are cancellative monoids (but some infinite cancellative monoids are not groups, e.g., the monoid of concatenation). For this large class of accumulation functions, we design an efficient algorithm for certainty.

▶ **Theorem 26.** *CERT is in PTIME for any PosRA$^{\text{acc}}$ query that performs accumulation in a cancellative monoid.*

Hence, CERT is tractable for PosRA (Theorem 14), via the concatenation monoid, and CERT is also tractable for top-k (defined in Example 9). The hardness of POSS for PosRA (Theorem 12) then implies that POSS, unlike CERT, is hard even on cancellative monoids.

Other restrictions on accumulation. We next revisit the results of Section 5 for PosRA$^{\text{acc}}$. However, we need to make other assumptions on accumulation (besides PTIME-evaluability). First, in the next results in this section, we assume that the accumulation monoid is *finite*:

▶ **Definition 27.** A PosRA$^{\text{acc}}$ query is said to perform *finite* accumulation if the accumulation monoid $(\mathcal{M}, \oplus, \varepsilon)$ is finite.

For instance, if the domain of the output is assumed to be fixed (e.g., ratings in $\{1, \ldots, 10\}$), then select-at-k and top-k (the latter for fixed k), as defined in Example 9, are finite.

Second, for some of the next results, we require *position-invariant accumulation*, namely, that the accumulation map does not depend on the absolute position of tuples:

▶ **Definition 28.** Recall that the accumulation map h has in general two inputs: a tuple and its position. A PosRA$^{\text{acc}}$ query is said to be *position-invariant* if its accumulation map ignores the second input, so that effectively its only input is the tuple itself.

Note that accumulation in the monoid is still performed in order, so we can still perform, e.g., concatenation. These two restrictions do not suffice to make POSS and CERT tractable, but we will use them to lift the results of Section 5.

Revisiting Section 5. We now extend our previous results to queries with accumulation, for POSS and CERT, under the additional assumptions on accumulation that we presented. We call PosRA$^{\text{acc}}_{\texttt{LEX}}$ and PosRA$^{\text{acc}}_{\text{no}\times}$ the extension of PosRA$_{\texttt{LEX}}$ and PosRA$_{\text{no}\times}$ with accumulation.

We can first generalize Theorem 17 to PosRA$^{\text{acc}}_{\texttt{LEX}}$ queries with *finite* accumulation:

▶ **Theorem 29.** *For any PosRA$^{\text{acc}}_{\texttt{LEX}}$ query performing* finite *accumulation, POSS and CERT are in PTIME on po-databases of bounded width.*

We then extend Theorem 22 to PosRA$^{\text{acc}}$ with *finite* and *position-invariant* accumulation:

▶ **Theorem 30.** *For any PosRA$^{\text{acc}}$ query performing* finite *and* position-invariant *accumulation, POSS and CERT are in PTIME on po-databases of bounded ia-width.*

Last, we can adapt the tractability result for queries without product (Theorem 23):

▶ **Theorem 31.** *For any PosRA$^{\text{acc}}_{\text{no}\times}$ query performing* finite *and* position-invariant *accumulation, POSS and CERT are in PTIME on po-databases whose relations have either bounded width or bounded ia-width.*

The finiteness assumption is important, as the previous result does not hold otherwise. Specifically, there exists a query that performs *position-invariant* but not *finite* accumulation, for which POSS is NP-hard even on unordered po-relations.

Other definitions. Finally, recall that we can use accumulation as in Example 9 to capture *position-based selection* (top-k, select-at-k) and *tuple-level comparison* (whether the first occurrence of a tuple precedes all occurrences of another tuple) for PosRA queries. Using a direct construction for these problems, we can show that they are tractable:

▶ **Proposition 32.** *For any PosRA query Q, the following problems are in PTIME:*

select-at-k: *Given a po-database D, tuple value t, and position $k \in \mathbb{N}$, whether it is possible/certain that $Q(D)$ has value t at position k;*

top-k: *For any* fixed *$k \in \mathbb{N}$, given a po-database D and list relation L of length k, whether it is possible/certain that the top-k values in $Q(D)$ are exactly L;*

tuple-level comparison: *Given a po-database D and two tuple values t_1 and t_2, whether it is possible/certain that the first occurrence of t_1 precedes all occurrences of t_2.*

7 Extensions

We next briefly consider two extensions to our model: group-by and duplicate elimination.

Group-by. First, we extend accumulation with a *group-by* operator, inspired by SQL.

▶ **Definition 33.** Let $(\mathcal{M}, \oplus, \varepsilon)$ be a monoid and $h : \mathcal{D}^k \to \mathcal{M}$ be an accumulation map (cf. Definition 4), and let $\mathbf{A} = A_1, \ldots, A_n$ be a sequence of attributes: we call $\mathrm{accumGroupBy}_{h,\oplus,\mathbf{A}}$ an *accumulation operator with group-by.* Letting L be a list relation with compatible schema, we define $\mathrm{accumGroupBy}_{h,\oplus,\mathbf{A}}(L)$ as an *unordered* relation that has, for each tuple value $t \in \pi_{\mathbf{A}}(L)$, one tuple $\langle t, v_t \rangle$ where v_t is $\mathrm{accum}_{h,\oplus}(\sigma_{A_1=t.A_1,\ldots A_n=t.A_n}(L))$ with π and σ on the list relation L having the expected semantics. The result on a po-relation Γ is the set of unordered relations $\{\mathrm{accumGroupBy}_{h,\oplus,\mathbf{A}}(L) \mid L \in pw(\Gamma)\}$.

In other words, the operator "groups by" the values of $A_1, \ldots, A_n$, and performs accumulation within each group, forgetting the order across groups. As for standard accumulation, we only allow group-by as an outermost operation, calling $\mathrm{PosRA}^{\mathrm{accGBy}}$ the language of PosRA queries followed by one accumulation operator with group-by. Note that the set of possible results is generally not a po-relation, because the underlying bag relation is not certain.

We next study the complexity of POSS and CERT for $\mathrm{PosRA}^{\mathrm{accGBy}}$ queries. Of course, whenever POSS and CERT are hard for some $\mathrm{PosRA}^{\mathrm{acc}}$ query Q on some kind of input po-relations, then there is a corresponding $\mathrm{PosRA}^{\mathrm{accGBy}}$ query for which hardness also holds (with empty $\mathbf{A}$). The main point of this section is to show that the converse is not true: the addition of group-by increases complexity. Specifically, we show that the POSS problem for $\mathrm{PosRA}^{\mathrm{accGBy}}$ is hard even on totally ordered po-relations and without the $\times_{\mathtt{DIR}}$ operator:

▶ **Theorem 34.** *There is a $\mathrm{PosRA}^{\mathrm{accGBy}}$ query Q with finite and position-invariant accumulation, not using $\times_{\mathtt{DIR}}$, such that POSS for Q is NP-hard even on totally ordered po-relations.*

This result contrasts with the tractability of POSS for $\mathrm{PosRA}_{\mathtt{LEX}}$ queries (Theorem 15) and for $\mathrm{PosRA}^{\mathrm{acc}}_{\mathtt{LEX}}$ queries with finite accumulation (Theorem 29) on totally ordered po-relations.

By contrast, it is not hard to see that the CERT problem for $\mathrm{PosRA}^{\mathrm{accGBy}}$ reduces to CERT for the same query without group-by, so it is no harder than the latter problem. Specifically:

▶ **Theorem 35.** *All CERT tractability results from Section 6 extend to $\mathrm{PosRA}^{\mathrm{accGBy}}$ when imposing the same restrictions on query operators, accumulation, and input po-relations.*

Duplicate elimination. We last study the problem of consolidating tuples with *duplicate values*. To this end, we define a new operator, dupElim, and introduce a semantics for it. The main problem is that tuples with the same values may be ordered differently relative to other tuples. To mitigate this, we introduce the notion of *id-sets*:

▶ **Definition 36.** Given a totally ordered po-relation $(ID, T, <)$, a subset ID' of ID is an *indistinguishable duplicate set* (or *id-set*) if for every $id_1, id_2 \in ID'$, we have $T(id_1) = T(id_2)$, and for every $id \in ID \backslash ID'$, we have $id < id_1$ iff $id < id_2$, and $id_1 < id$ iff $id_2 < id$.

▶ **Example 37.** Consider the totally ordered relation $\Gamma_1 := \Pi_{hotelname}(Hotel)$, with $Hotel$ as in Figure 1. The two "Mercure" tuples are not an id-set: they disagree on their ordering with "Balzac". Consider now a totally ordered relation $\Gamma_2 = (ID_2, T_2, <_2)$ whose only possible world is a list relation (A, B, B, C) for some tuples A, B, and C over $\mathcal{D}$. The set $\{id \in ID_2 \mid T_2(id) = B\}$ is an id-set in Γ_2. Note that a singleton is always an id-set.

We define a semantics for dupElim on a totally ordered po-relation $\Gamma = (ID, T, <)$ via id-sets. First, check that for every tuple value t in the image of T, the set $\{id \in ID \mid T(id) = t\}$ is an id-set in Γ. If this holds, we call Γ *safe*, and set dupElim(Γ) to be the singleton $\{L\}$ of the only possible world of the restriction of Γ obtained by picking one representative element per id-set (clearly L does not depend on the chosen representatives). Otherwise, we call Γ *unsafe* and say that duplicate consolidation has *failed*; we then set dupElim(Γ) to be an empty set of possible worlds. Intuitively, duplicate consolidation tries to reconcile (or "synchronize") order constraints for tuples with the same values, and fails when it cannot be done.

▶ **Example 38.** In Example 37, we have dupElim$(\Gamma_1) = \emptyset$ but dupElim$(\Gamma_2) = (A, B, C)$.

We then extend dupElim to po-relations by considering all possible results of duplicate elimination on the possible worlds, ignoring the unsafe possible worlds. If no possible worlds are safe, then we *completely fail*:

▶ **Definition 39.** For each list relation L, we let Γ_L be a po-relation such that $pw(\Gamma_L) = \{L\}$. Letting Γ be a po-relation, we set dupElim$(\Gamma) := \bigcup_{L \in pw(\Gamma)}$ dupElim(Γ_L). We say that dupElim(Γ) *completely fails* if dupElim$(\Gamma) = \emptyset$, i.e., dupElim$(\Gamma_L) = \emptyset$ for every $L \in pw(\Gamma)$.

▶ **Example 40.** Consider the totally ordered po-relation $Rest$ from Figure 1, and a totally ordered po-relation $Rest_2$ whose only possible world is (Tsukizi, Gagnaire). Consider $Q :=$ dupElim$(\Pi_{restname}(Rest) \cup Rest_2)$. Intuitively, Q combines restaurant rankings, using duplicate consolidation to collapse two occurrences of the same name to a single tuple. The only possible world of Q is (Tsukizi, Gagnaire, TourArgent), since duplicate elimination fails in the other possible worlds: indeed, this is the only possible way to combine the rankings.

We next show that the result of dupElim can still be represented as a po-relation, up to complete failure (which may be efficiently identified).

▶ **Theorem 41.** *For any po-relation Γ, we can test in PTIME if* dupElim(Γ) *completely fails; if it does not, we can compute in PTIME a po-relation Γ' such that $pw(\Gamma') =$* dupElim(Γ).

We note that dupElim is not redundant with any of the other PosRA operators, generalizing Theorem 1:

▶ **Theorem 42.** *No operator among those of PosRA and* dupElim *can be expressed through a combination of the others.*

Last, we observe that dupElim can indeed be used to undo some of the effects of bag semantics. For instance, we can show the following:

▶ **Proposition 43.** *For any po-relation Γ, we have* $\mathrm{dupElim}(\Gamma \cup \Gamma) = \mathrm{dupElim}(\Gamma)$*: in particular, one completely fails iff the other does.*

We can also show that most of our previous tractability results still apply when the duplicate elimination operator is added:

▶ **Theorem 44.** *All `POSS` and `CERT` tractability results of Sections 4–6, except Theorem 23 and Theorem 31, extend to PosRA and PosRA*$^{\mathrm{acc}}$ *where we allow* dupElim *(but impose the same restrictions on query operators, accumulation, and input po-relations).*

Furthermore, if in a set-semantics spirit we *require* that the query output has no duplicates, `POSS` and `CERT` are always tractable (as this avoids the technical difficulty of Example 11):

▶ **Theorem 45.** *For any PosRA query Q, `POSS` and `CERT` for* $\mathrm{dupElim}(Q)$ *are in PTIME.*

Discussion. The introduced group-by and duplicate elimination operators have some shortcomings: the result of group-by is in general not representable by po-relations, and duplicate elimination may fail. These are both consequences of our design choices, where we capture only uncertainty on order (but not on tuple values) and design each operator so that its result corresponds to the result of applying it to each individual world of the input (see further discussion in Section 8). Avoiding these shortcomings is left for future work.

8 Comparison With Other Formalisms

We next compare our formalism to previously proposed formalisms: query languages over bags (with no order); a query language for partially ordered multisets; and other related work. To our knowledge, however, none of these works studied the possibility or certainty problems for partially ordered data, so that our technical results do not follow from them.

Standard bag semantics. We first compare to related work on the *bag semantics* for relational algebra. Indeed, a natural desideratum for our semantics on (partially) ordered relations is that it should be a faithful extension of bag semantics. We first consider the BALG^1 language on bags [21] (the "flat fragment" of their language BALG on nested relations). We denote by BALG^1_+ the fragment of BALG^1 that includes the standard extension of positive relational algebra operations to bags: additive union, cross product, selection, projection. We observe that, indeed, our semantics faithfully extends BALG^1_+: *query evaluation commutes with "forgetting" the order.* Formally, for a po-relation Γ, we denote by $\mathrm{bag}(\Gamma)$ its underlying bag relation, and define likewise $\mathrm{bag}(D)$ for a po-database D as the database of underlying bag relations. For the following comparison, we identify $\times_{\mathtt{DIR}}$ and $\times_{\mathtt{LEX}}$ with the $\times$ of [21] and our union with the additive union of [21]; the following holds:

▶ **Proposition 46.** *For any PosRA query Q and a po-relation D,* $\mathrm{bag}(Q(D)) = Q(\mathrm{bag}(D))$ *where $Q(D)$ is defined according to our semantics and $Q(\mathrm{bag}(D))$ is defined by $BALG^1_+$.*

The full BALG^1 language includes additional operators, such as bag intersection and subtraction, which are non-monotone and as such may not be expressed in our language: it is also unclear how they could be extended to our setting (see further discussion in "Algebra on pomsets" below). On the other hand, BALG^1 does not include aggregation, and so $\mathrm{PosRA}^{\mathrm{acc}}$ and BALG^1 are incomparable in terms of expressive power.

A better yardstick to compare against for accumulation could be [33]: they show that their basic language $\mathcal{BQL}$ is equivalent to BALG, and then further extend the language with aggregate operators, to define a language called $\mathcal{NRL}^{\text{aggr}}$ on nested relations. On flat relations, $\mathcal{NRL}^{\text{aggr}}$ captures functions that cannot be captured in our language: in particular the average function AVG is non-associative and thus cannot be captured by our accumulation function (which anyway focuses on order-dependent functions, as POSS/CERT are trivial otherwise). On the other hand, $\mathcal{NRL}^{\text{aggr}}$ cannot test parity (Corollary 5.7 in [33]) whereas this is easily captured by our accumulation operator. We conclude that $\mathcal{NRL}^{\text{aggr}}$ and PosRA$^{\text{acc}}$ are incomparable in terms of captured transformations on bags, even when restricted to flat relations.

Algebra on pomsets. We now compare our work to algebras defined on *pomsets* [20, 22], which also attempt to bridge partial order theory and data management (although, again, they do not study possibility and certainty). *Pomsets* are labeled posets quotiented by isomorphism (i.e., renaming of identifiers), like po-relations. A major conceptual difference between our formalism and that of [20, 22] is that their language focuses on processing *connected components* of the partial order graph, and their operators are tailored for that semantics. As a consequence, their semantics is *not* a faithful extension of bag semantics, i.e., their language would not satisfy the counterpart of Proposition 46 (see for instance the semantics of union in [20]). By contrast, we manipulate po-relations that stand for sets of possible list relations, and our operators are designed accordingly, unlike those of [20] where transformations take into account the structure (connected components) of the entire poset graph. Because of this choice, [20] introduces non-monotone operators that we cannot express, and can design a duplicate elimination operator that cannot fail. Indeed, the possible failure of our duplicate elimination operator is a direct consequence of its semantics of operating on each possible world, possibly leading to contradictions.

If we consequently disallow duplicate elimination in both languages for the sake of comparison, we note that the resulting fragment $\mathcal{P}\text{om-}\mathcal{A}\text{lg}_{\varepsilon_n}$ of the language of [20] can yield only series-parallel output (Proposition 4.1 of [20]), unlike PosRA queries whose output order may be arbitrary. Hence, $\mathcal{P}\text{om-}\mathcal{A}\text{lg}_{\varepsilon_n}$ does not subsume PosRA.

Incompleteness in databases. Our work is inspired by the field of incomplete information management, studied for various models [5, 30], in particular relational databases [24]. This field inspires our design of po-relations and study of possibility and certainty [3, 34]. However, uncertainty in these settings typically focuses on *whether* tuples exist or on their *values* (e.g., with nulls [10], including the novel approach of [31, 32]; with c-tables [24], probabilistic databases [42] or fuzzy numerical values as in [38]). To our knowledge, though, our work is the first to study possible and certain answers in the context of *order*-incomplete data. Combining order incompleteness with standard tuple-level uncertainty is left as a challenge for future work. Note that some works [8, 29, 32] use partial orders on *relations* to compare the informativeness of representations. This is unrelated to our partial orders on *tuples*.

Ordered domains. Another line of work has studied relational data management where the *domain elements* are (partially) ordered [25, 35, 43]. However, the perspective is different: we see order on tuples as part of the relations, and as being constructed by applying our operators; these works see order as being given *outside* of the query, hence do not study the propagation of uncertainty through queries. Also, queries in such works can often directly access the order relation [43, 6]. Some works also study uncertainty on totally ordered *numerical* domains [38, 39], while we look at general order relations.

Temporal databases. *Temporal databases* [9, 37] consider order on facts, but it is usually induced by timestamps, hence total. A notable exception is [16] which considers that some facts may be *more current* than others, with constraints leading to a partial order. In particular, they study the complexity of retrieving query answers that are certainly current, for a rich query class. In contrast, we can *manipulate* the order via queries, and we can also ask about aspects beyond currency, as shown throughout the paper (e.g., via accumulation).

Using preference information. Order theory has been also used to handle *preference information* in database systems [26, 4, 27, 2, 41], with some operators being the same as ours, and for *rank aggregation* [15, 26, 14], i.e. retrieving top-k query answers given multiple rankings. However, such works typically try to *resolve* uncertainty by reconciling many conflicting representations (e.g. via knowledge on the individual scores given by different sources and a function to aggregate them [15], or a preference function [2]). In contrast, we focus on maintaining a faithful model of *all* possible worlds without reconciling them, studying possible and certain answers in this respect.

9 Conclusion

This paper introduced an algebra for order-incomplete data. We have studied the complexity of possible and certain answers for this algebra, have shown the problems to be generally intractable, and identified multiple tractable cases. In future work we plan to study the incorporation of additional operators (in particular non-monotone ones), investigate how to combine order-uncertainty with uncertainty on values, and study additional semantics for dupElim. Last, it would be interesting to establish a dichotomy result for the complexity of `POSS`, and a complete syntactic characterization of cases where `POSS` is tractable.

Acknowledgements We are grateful to Marzio De Biasi, Pálvölgyi Dömötör, and Mikhail Rudoy, from `http://cstheory.stackexchange.com`, for helpful suggestions.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*. Addison-Wesley, 1995.

2 Bogdan Alexe, Mary Roth, and Wang-Chiew Tan. Preference-aware integration of temporal data. *PVLDB*, 8(4), 2014. `doi:10.14778/2735496.2735500`.

3 Lyublena Antova, Christoph Koch, and Dan Olteanu. World-set decompositions: Expressiveness and efficient algorithms. In *ICDT*, volume 4353 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2007. URL: `https://arxiv.org/abs/0705.4442`.

4 Anastasios Arvanitis and Georgia Koutrika. PrefDB: Supporting preferences as first-class citizens in relational databases. *IEEE TKDE*, 26(6), 2014.

5 Pablo Barceló, Leonid Libkin, Antonella Poggi, and Cristina Sirangelo. XML with incomplete information. *J. ACM*, 58(1), 2010. `doi:10.1145/1870103.1870107`.

6 Michael Benedikt and Luc Segoufin. Towards a characterization of order-invariant queries over tame graphs. *Journal of Symbolic Logic*, 74, 2009.

7 Andeas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. Posets. In *Graph Classes. A Survey*, chapter 6. SIAM, 1987.

8 Peter Buneman, Achim Jung, and Atsushi Ohori. Using powerdomains to generalize relational databases. *TCS*, 91(1), 1991.

9 Jan Chomicki and David Toman. Time in database systems. In *Handbook of Temporal Reasoning in Artificial Intelligence*. Elsevier, 2005.

10 Edgar F. Codd. Extending the database relational model to capture more meaning. *TODS*, 4(4), 1979.

11 Latha S. Colby, Edward L. Robertson, Lawrence V. Saxton, and Dirk Van Gucht. A query language for list-based complex objects. In *PODS*, 1994.

12 Latha S. Colby, Lawrence V. Saxton, and Dirk Van Gucht. Concepts for modeling and querying list-structured data. *Information Processing & Management*, 30(5), 1994.

13 Robert P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 1950.

14 Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the Web. In *WWW*, 2001.

15 Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

16 Wenfei Fan, Floris Geerts, and Jef Wijsen. Determining the currency of data. *TODS*, 37(4), 2012.

17 Roland Fraïssé. L'intervalle en théorie des relations; ses genéralisations, filtre intervallaire et clôture d'une relation. *North-Holland Math. Stud.*, 99, 1984.

18 D. R. Fulkerson. Note on Dilworth's decomposition theorem for partially ordered sets. In *Proc. Amer. Math. Soc*, 1955.

19 Michael R. Garey and David S. Johnson. *Computers And Intractability. A Guide to the Theory of NP-completeness.* W. H. Freeman, 1979.

20 Stéphane Grumbach and Tova Milo. An algebra for pomsets. In *ICDT*, 1995.

21 Stéphane Grumbach and Tova Milo. Towards tractable algebras for bags. *JCSS*, 52(3), 1996. `doi:10.1006/jcss.1996.0042`.

22 Stéphane Grumbach and Tova Milo. An algebra for pomsets. *Inf. Comput.*, 150(2), 1999. `doi:10.1006/inco.1998.2777`.

23 John M. Howie. *Fundamentals of semigroup theory.* Oxford: Clarendon Press, 1995.

24 Tomasz Imieliński and Witold Lipski. Incomplete information in relational databases. *J. ACM*, 31(4), 1984.

25 Neil Immerman. Relational queries computable in polynomial time. *Inf. Control*, 68(1-3), 1986.

26 Marie Jacob, Benny Kimelfeld, and Julia Stoyanovich. A system for management and analysis of preference data. *VLDB Endow.*, 7(12), 2014.

27 Werner Kiessling. Foundations of preferences in database systems. In *VLDB*, 2002.

28 Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, 2002. `doi:10.1145/543613.543644`.

29 Leonid Libkin. A semantics-based approach to design of query languages for partial information. In *Semantics in Databases*, 1998.

30 Leonid Libkin. Data exchange and incomplete information. In *PODS*, 2006. `doi:10.1145/1142351.1142360`.

31 Leonid Libkin. Incomplete data: What went wrong, and how to fix it. In *PODS*, 2014.

32 Leonid Libkin. SQL's three-valued logic and certain answers. In *ICDT*, 2015. `doi:10.4230/LIPIcs.ICDT.2015.94`.

33 Leonid Libkin and Limsoon Wong. Query languages for bags and aggregate functions. *J. Comput. Syst. Sci.*, 55(2), 1997. `doi:10.1006/jcss.1997.1523`.

34 Witold Lipski, Jr. On semantic issues connected with incomplete information databases. *TODS*, 4(3), 1979.

35 Wilfred Ng. An extension of the relational data model to incorporate ordered domains. *TODS*, 26(3), 2001.

36 Bernd Schröder. *Ordered Sets: An Introduction.* Birkhäuser, 2003.
37 Richard T. Snodgrass, Jim Gray, and Jim Melton. *Developing time-oriented database applications in SQL.* Morgan Kaufmann, 2000.
38 Mohamed A. Soliman and Ihab F. Ilyas. Ranking with uncertain scores. In *ICDE*, 2009. doi:10.1109/ICDE.2009.102.
39 Mohamed A. Soliman, Ihab F. Ilyas, and Shalev Ben-David. Supporting ranking queries on uncertain and incomplete data. *VLDBJ*, 19(4), 2010.
40 Richard P. Stanley. *Enumerative Combinatorics.* Cambridge University Press, 1986.
41 Kostas Stefanidis, Georgia Koutrika, and Evaggelia Pitoura. A survey on representation, composition and application of preferences in database systems. *TODS*, 36(3), 2011.
42 Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.
43 Ron van der Meyden. The complexity of querying indefinite data about linearly ordered domains. *JCSS*, 54(1), 1997.
44 Manfred K Warmuth and David Haussler. On the complexity of iterated shuffle. *JCSS*, 28(3), 1984.

A Proof Sketches for Section 4 (General Complexity Results)

▶ **Theorem 12.** *The* POSS *problem is in NP for any PosRA or PosRA*$^{\mathrm{acc}}$ *query. Further, there exists a PosRA query and a PosRA*$^{\mathrm{acc}}$ *query for which the* POSS *problem is NP-complete.*

Proof Sketch. The membership for PosRA in NP is clear: guess a linear extension and check that it realizes the candidate possible result. For hardness, as in previous work [44], we reduce from the UNARY-3-PARTITION problem [19]: given a number B and $3m$ numbers written in unary, decide if they can be partitioned in triples that all sum to B. We reduce this to POSS for the identity PosRA query, on an arity-1 input po-relation where each input number n is represented as a chain of $n+2$ elements. The first and last elements of each chain are respectively called start and end markers, and elements of distinct chains are pairwise incomparable. The candidate possible world L consists of m repetitions of the following pattern: three start markers, B elements, three end markers. A linear extension achieves L iff the triples matched by $<$ to each copy of the pattern are a solution to UNARY-3-PARTITION, hence POSS for Q is NP-hard. This implies hardness for PosRA$^{\mathrm{acc}}$, when accumulating with the identity map and concatenation (so that any list relation is mapped to itself). ◀

▶ **Theorem 13.** *The* CERT *problem is in coNP for any PosRA*$^{\mathrm{acc}}$ *query, and there is a PosRA*$^{\mathrm{acc}}$ *query for which it is coNP-complete.*

Proof Sketch. Again, membership is immediate. We show hardness of CERT by studying a PosRA$^{\mathrm{acc}}$ query Q_{a} that checks if two input po-relations Γ and Γ' have some common possible world: Q_{a} does so so by testing if one can alternate between elements of Γ and Γ' with the same label, using accumulation in the transition monoid of a deterministic finite automaton. We show hardness of POSS for Q_{a} (as in the previous result), and further ensure that Q_{a} always has at most two possible accumulation results, no matter the input. Hence, POSS for Q_{a} reduces to the negation of CERT for Q_{a}, so that CERT is also hard. ◀

B Proof Sketches for Section 5 (Tractable Cases for POSS on PosRA Queries)

▶ **Theorem 17.** *The* POSS *problem is in NP for any PosRA or PosRA*$^{\mathrm{acc}}$ *query. Further, there exists a PosRA query and a PosRA*$^{\mathrm{acc}}$ *query for which the* POSS *problem is NP-complete.*

Proof Sketch. As $\times_{\mathtt{DIR}}$ is disallowed, we can show that the po-relation $\Gamma := Q(D)$ has width k' depending only on k and the query Q (but not on D). We can then compute in PTIME a *chain partition* of Γ [13, 18], namely, a decomposition of Γ in totally ordered chains, with additional order constraints between them. This allows us to apply a dynamic algorithm to decide POSS: the state of the algorithm is the position on the chains. The number of states is polynomial with degree k', which is a constant when Q and k are fixed. ◀

▶ **Theorem 22.** *For any fixed $k \in \mathbb{N}$ and fixed PosRA query Q, the POSS problem for Q is in PTIME when all po-relations of the input po-database have ia-width $\leqslant k$.*

Proof Sketch. As in the proof of Theorem 17, we first show that the query result Γ also has ia-width depending only on k and the query. We then consider the order relation on indistinguishable antichains of Γ. For each linear extension τ of this order, we apply a greedy algorithm to decide POSS, for which we show correctness. The algorithm reads the candidate possible world in order and maps each tuple to an identifier of Γ with the correct value that was not mapped yet: we pick it in the first possible class according to the order τ. ◀

C Proof sketches for Section 6 (Tractable Cases for Accumulation Queries)

▶ **Theorem 26.** *CERT is in PTIME for any PosRA$^{\mathrm{acc}}$ query that performs accumulation in a cancellative monoid.*

Proof Sketch. We show that the accumulation result in cancellative monoids is certain iff the po-relation on which we apply accumulation respects the following *safe swaps* criterion: for all tuples t_1 and t_2 and consecutive positions p and $p+1$ where they may appear, we have $h(t_1, p) \oplus h(t_2, p+1) = h(t_2, p) \oplus h(t_1, p+1)$. We can check this in PTIME. ◀

Constraint Identification Using Modified Hoare Logic on Hybrid Models of Gene Networks[*]

Jonathan Behaegel[1], Jean-Paul Comet[2], and Maxime Folschette[3]

1 Université Côte d'Azur, CNRS, I3S, Sophia Antipolis, France
2 Université Côte d'Azur, CNRS, I3S, Sophia Antipolis, France
3 Université de Nantes, LS2N, Nantes, France

Abstract

We present a new hybrid Hoare logic dedicated for a class of linear hybrid automata well suited to model gene regulatory networks. These automata rely on Thomas' discrete framework in which qualitative parameters have been replaced by continuous parameters called celerities. The identification of these parameters remains one of the keypoints of the modelling process, and is difficult especially because the modelling framework is based on a continuous time. We introduce Hoare triples which handle biological traces and pre/post-conditions. Observed chronometrical biological traces play the role of an imperative program for classical Hoare logic and our hybrid Hoare logic, defined by inference rules, is proved to be sound. Furthermore, we present a weakest precondition calculus (a la Dijkstra) which leads to constraints on dynamical parameters. Finally, we illustrate our "constraints generator" with a simplified circadian clock model describing the rhythmicity of cells in mammals on a 24-hour period.

1998 ACM Subject Classification I.6.4 Model Validation and Analysis

Keywords and phrases Hoare logic, weakest precondition, linear hybrid automata, constraint synthesis, gene networks

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.5

1 Introduction

Formal methods from computer science have been largely applied to model and analyse biological systems [5, 17]. In particular, verification tools like model-checking have been used to verify dynamical properties of discrete models [3, 7] in which the temporal aspects are only present through the succession of events: the delay between two successive events is not taken into account. Unfortunately, continuous time turns out to be important in most biological systems, in particular for gene regulatory networks.

Gene regulatory networks are models representing influences between genes leading to the modification of the synthesis of associated proteins. Because proteins can regulate their target genes, positive or negative feedback loops emerge making possible a large variety of behaviours. These gene regulatory networks are designed to apprehend and predict effects of a component on the system but such models are also useful to confront hypotheses with the up-to-date collected knowledge on the gene interactions.

Several modelling framework have been devoted to gene networks. These frameworks differ by the aspects they highlight. Stochastic models emphasize non-determinism, differential models represent a system with a lot of details (transcription, traduction, transports . . .) [14]

[*] This work is partially supported by the French National Agency for Research (ANR-14-CE09-0011).

and give precise trajectories in terms of concentrations; qualitative models [16, 15] focus on the major features that explain the observations (only main causalities are taken into account); and hybrid models link qualitative aspects with continuous variables such as time.

Whatever the modelling framework, the main difficulty of building such networks is the identification of parameters governing the dynamics of the system. The determination of these parameters is crucial to observe a behaviour consistent with biological knowledge. We already showed [3] that formal methods can help in this parameter identification step in the René Thomas' discrete modelling framework [16]. Unfortunately, this framework abstracts temporal information, often necessary for tuning the models. This discrete approach is based on the splitting of homogeneous concentration areas which have the same effects on other components. In order to refine this complete discrete framework, we associated with each such area a celerity describing the evolution speed of each component. These parameters lead to a class of linear hybrid automata. We also showed how the known experimental traces can be used to establish constraints on dynamical parameters (celerities) and to restrain the set of admissible parameters [1].

Numerous works already focus on the study of hybrid automata [9]. Especially, several tools already exist to tackle the model checking of linear hybrid automata, either with classical exhaustive approaches [10] or using abstract interpretation [8]. Communicating sequential processes (CSP), a process algebra for describing patterns of interaction in concurrent systems, has also been extended to hybrid systems and hybrid Hoare logic has been proved to be useful in such context [18]. These methods among others propose parameter synthesis in some ways. Nevertheless, these tools are tailored for a general purpose and will not take into account the specificities coming from biological contexts.

We divert Hoare logic (whose aim is to rigorously reason about the correctness of imperative programs) in order to determine constraints on celerities in such models. This approach was already developed, for the discrete framework [2, 4] and we extend it in the present paper for hybrid automata. Hoare logic has already been extended to real time systems [11] in which continuous evolutions are not taken into account whereas they are important in our biological context. Hoare logic relies on triples of the form $\{Pre\}\ p\ \{Post\}$ where *Pre* and *Post* are conditions on states of the system and p is a path of the system. A Hoare triple is considered true if, whenever the system is reset at a state satisfying the *Pre* condition, the path p is possible and leads the system into a state which satisfies the condition *Post*. Following E. Dijkstra [6], the aim of the game is then to determine, for each path p and postcondition *Post*, the weakest precondition *Pre*, thus covering the largest set of states, making the Hoare triple true. In our temporal approach, the time spent in each state becomes crucial to determine the constraints on parameters.

We illustrate our formalism with a biological process named circadian clock which synchronises all cells in mammals at a 24-hour rhythmicity. We design a hybrid automaton modelling this biological cycle and, from the observed trace of this process, we build constraints for each celerity of this hybrid automaton using the aforementioned hybrid Hoare logic. We finally show that simulations, run parameters values satisfying the constraints, exhibit curves which are similar to experimental data.

The paper is organised as follows. We first define in Section 2 the formalism of the hybrid modelling framework. Then Section 3 focuses on the syntax and the semantics of the modified Hoare logic, and the weakest precondition calculus, whereas Section 4 is devoted to the soundness of our hybrid Hoare logic. We illustrate in Section 5 the use of this Hoare logic for identifying the constraints on parameters of the simplified circadian clock model. Finally in Section 6, we discuss the limits of this approach and we expose some possible extensions.

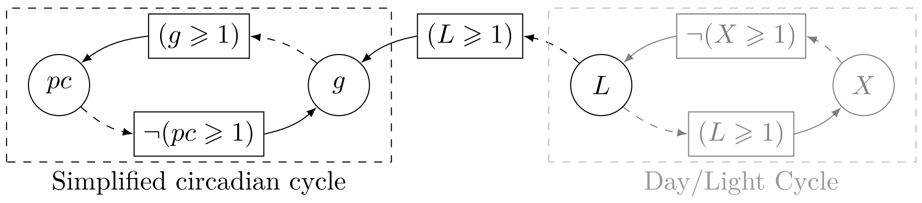


Figure 1 Simplified model of the circadian clock in mammals. L is a zeitgeber (see Section 5).

2 Hybrid Modelling Framework of Gene Network

A gene network is visualised as a labelled directed graph (interaction graph) in which vertices are either variables (within circles) or multiplexes (within rectangles), see Fig. 1. Variables abstract genes and their products, and multiplexes contain formulas that encode situations in which a variable or a group of variables (inputs of multiplexes, dashed arrows) influences the evolution of some other variables (output of multiplexes, plain arrows). A multiplex can encode the formation of molecular complexes, phosphorylation by a protein, competition of entities for the activation of a promoter, etc. Definition 1 gives the formal details of a gene network.

▶ **Definition 1** (Hybrid Gene Regulatory Network)**.** A hybrid gene regulatory network (GRN for short) is a tuple $R = (V, M, E, \mathcal{C})$ where:

- V is a finite set whose elements are called *variables* of the network. With each variable $v \in V$ is associated a boundary $b_v \in \mathbb{N}^*$.
- M is a finite set whose elements are called *multiplexes*. With each multiplex $m \in M$ is associated a formula φ_m in the *multiplex language* formed of the atoms "$(v \geqslant n)$", where $n \in [\![1, b_v]\!]$[1], and the usual logical connectives $\neg$, $\wedge$, $\vee$ and $\Rightarrow$.
- E is a set of edges of the form $(m \to v) \in M \times V$.
- $\mathcal{C} = \{C_{v,\omega,n}\}$ is a family of real numbers indexed by a tuple (v, ω, n) where $v \in V$, ω is a subset of $R^-(v)$ where $R^-(v) = \{m \mid (m \to v) \in E\}$, that is, ω is a set of predecessors of v, and $n \in [\![0, b_v]\!]$. $C_{v,\omega,n}$ is called the *celerity* of v for ω at the level n and these celerities have to satisfy the following constraints:

$$\forall v \in V,\ \forall \omega \subset R^-(v),\ \forall n \in [\![0, b_v]\!], \quad C_{v,\omega,n} = 0 \quad \Rightarrow \quad \begin{cases} \forall i \in [\![n+1, b_v]\!] & C_{v,\omega,i} < 0 \\ \forall i \in [\![0, n-1]\!] & C_{v,\omega,i} > 0 \end{cases}$$

$$\forall v \in V,\ \forall \omega \subset R^-(v),\ \forall k \in [\![0, b_v - 1]\!], \quad C_{v,\omega,k} \times C_{v,\omega,k+1} \geq 0.$$

Let us remark that the dashed arrows of Fig. 1 are not present in the previous definition. When representing a gene network, it is convenient to visualise the variables contributing to a particular multiplex, but from a formal point of view, this information is redundant with the formula of the considered multiplex.

Celerities (noted $C_{v,\omega,n}$) give the evolution of each variable v when it is under the active regulation of the set ω of its predecessors and when it is in the qualitative state n. They code for the dynamics of the system and we aim at the identification of these celerities. The constraints on celerities given in the previous definition link the signs of celerities to the underlying dynamics and may need some explanations. The first one deals with the case where a celerity value is null for a given set of active predecessors ω of a variable v. This models an equilibrium state, thus the related constraint states that celerities around this equilibrium, for the same set of active predecessors ω, must force v to reach this equilibrium.

[1] $[\![a, b]\!] = \{n \in \mathbb{N} \mid a \leqslant n \leqslant b\}$.

As a consequence, there is a single null celerity at most for a given couple of v and ω. If, on the other hand, no celerity is null for these v and ω, a consequence of the second constraint is that they are all of the same sign. This models that v either always decreases until full degradation or always increases up to saturation.

In the remainder of this section, we focus on the dynamics of a gene network. Definition 2 introduces the hybrid states whereas Def. 3 explains the crucial notion of resources of a variable in a particular state.

▶ **Definition 2** (State of a GRN). Let $R = (V, M, E, \mathcal{C})$ be a GRN. A *hybrid state* of R is a tuple $h = (\eta, \pi)$ where

- η is a function from V to $\mathbb{N}$ such that for all $v \in V$, $0 \leqslant \eta(v) \leqslant b_v$;
- π is a function from V to the interval $[0, 1]$ of real numbers.

η is called the *discrete state* or *qualitative state* of h whereas π is called its *fractional part.* For simplicity, we note in the sequel $\eta_v = \eta(v)$ and $\pi_v = \pi(v)$. We denote S the set of hybrid states of R. When there is no ambiguity, we often use η and π without explicit mention of h.

Figure 2-Centre illustrates an example of hybrid state. The tuple of all fractional parts represents coordinates inside the current qualitative state.

▶ **Definition 3** (Resources). Let $R = (V, M, E, \mathcal{C})$ be a GRN and let $v \in V$. The satisfaction relation $h \vDash \varphi$, where $h = (\eta, \pi)$ is a hybrid state of R and φ is a formula of the multiplex language, is inductively defined as follows:

- If φ is the atom $(v \geqslant n)$ with $n \in [\![1, b_v]\!]$, then $h \vDash \varphi$ iff $\eta_v \geqslant n$;
- The usual meaning of the logical connectives is used.

The set of *resources* of a variable v at a state h is defined by: $\rho(h, v) = \{m \in R^-(v) \mid h \vDash \varphi_m\}$, that is, the multiplexes that are predecessors of v and whose formula is satisfied.

We note that the set $\rho(h, v)$ only depends on the discrete state of h: all hybrid states having the same discrete part thus have the same resources. Indeed, inside a discrete state η, the dynamics of v is controlled in the same manner, thus the celerity is the same for all states $h = (\eta, \pi)$, that is: $C_{v,\rho(h,v),\eta_v}$. By abuse of language, we also use the notation $\rho(\eta, v)$. From this celerity, and given a particular hybrid state, one can compute the touch delay (Def. 4) of each variable, which measures the time necessary for the variable to reach a border of the discrete state.

▶ **Definition 4** (Touch Delay). Let $R = (V, M, E, \mathcal{C})$ be a GRN, v be a variable of V and $h = (\eta, \pi)$ be a hybrid state. We note $\delta_h(v)$ the *touch delay* of v in h for reaching the border of the discrete state. More precisely, δ_h is the function from V to $\mathbb{R}^+ \cup \{+\infty\}$ defined by:

- If $C_{v,\rho(h,v),\eta_v} = 0$ then $\delta_h(v) = +\infty$;
- If $C_{v,\rho(h,v),\eta_v} > 0$ (*resp.* < 0) then $\delta_h(v) = \frac{1-\pi_v}{C_{v,\rho(h,v),\eta_v}}$ (*resp.* $\frac{-\pi_v}{C_{v,\rho(h,v),\eta_v}}$).

Nevertheless, reaching the border of a discrete state is not sufficient to go beyond the frontier: there may be no other qualitative level beyond the border (we call such a border an external wall) or the celerity in the neighbour state may be of the opposite sign (internal wall), as given in Def. 5.

▶ **Definition 5** (External and Internal Walls). Let $R = (V, M, E, C)$ be a GRN, let $v \in V$ be a variable and $h = (\eta, \pi)$ a hybrid state.

1. v is said to *face an external wall* at state h if:
$$\big((C_{v,\rho(h,v),\eta_v} < 0) \wedge (\eta_v = 0)\big) \vee \big((C_{v,\rho(h,v),\eta_v} > 0) \wedge (\eta_v = b_v)\big) \ .$$

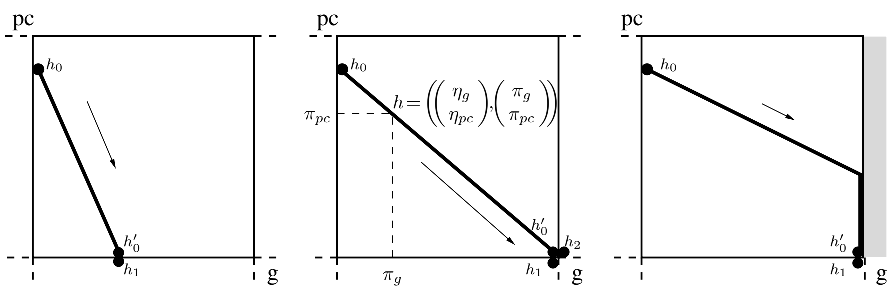

pc
h_0
h_0'
h_1
g
pc
h_0
$h = \left(\begin{pmatrix}\eta_g\\ \eta_{pc}\end{pmatrix},\begin{pmatrix}\pi_g\\ \pi_{pc}\end{pmatrix}\right)$
π_{pc}
h_0'
h_2
h_1
π_g
g
pc
h_0
h_0'
h_1
g

Figure 2 Continuous transitions. Inside each state, a continuous transition ($h_0 \to h_0'$) goes from the initial point h_0 to the unique point h_0' from which a discrete transition takes place ($h_0' \to h_1$). **Left:** The celerity vector allows, without sliding mode, the trajectory to directly reach a border which is crossed. **Center:** From h_0' two possible discrete transitions can occur: $h_0' \to h_1$ or $h_0' \to h_2$. Moreover (π_g, π_{pc}) corresponds to the fractionnal coordinates of a hybrid state h along the path. **Right:** The grey zone depicts an external or internal wall. The only possible discrete transition is $h_0' \to h_1$.

2. Let $h' = (\eta', \pi')$ be another hybrid state s.t. $\eta'_v = \eta_v + \mathrm{sgn}(C_{v,\rho(h,v),\eta_v})$ and $\eta'_u = \eta_u$ for all $u \neq v$. Variable v is said to *face an internal wall* at state h if $\mathrm{sgn}(C_{v,\rho(h,v),\eta_v}) \times \mathrm{sgn}(C_{v,\rho(h',v),\eta'_v}) = -1$, where sgn is the classical sign function.

We note $\mathsf{sv}(h)$ the set of *sliding variables*, that is, variables that face an internal or external wall in the qualitative state of h.

We note that a sliding variable $v \in \mathsf{sv}(h)$ may not be actually sliding if it has not reached its border yet. However, if in addition $\delta_h(v) = 0$, then v is located on an internal wall or external wall. In this case, its fractional part cannot evolve anymore in the current qualitative state (see Fig. 2-Right where variable g reaches an external wall). We introduce the notion of first changing variables in Def. 6 which are the first variables able to change their discrete levels.

▶ **Definition 6** (First Changing Variables). Let $R = (V, M, E, C)$ be a GRN and $h = (\eta, \pi)$ be a hybrid state. The set of *first changing variables* is defined by:

$$\mathsf{first}(h) = \{v \in V \setminus \mathsf{sv}(h) \mid \delta_h(v) \neq +\infty \wedge \forall u \in V \setminus \mathsf{sv}(h), \delta_h(u) \geqslant \delta_h(v)\} \ .$$

Moreover, $\delta_h^{\mathsf{first}}$ denotes the time spent in the qualitative state of h when starting from h: for any $v \in \mathsf{first}(h)$, $\delta_h^{\mathsf{first}} = \delta_h(v)$, or $\delta_h^{\mathsf{first}} = +\infty$ if $\mathsf{first}(h) = \emptyset$.

The set $\mathsf{first}(h)$ represents the set of variables whose qualitative coordinates can change first. If a variable is on an external or internal wall, it cannot evolve as long as other variables do not change, thus: $\mathsf{first}(h) \cap \mathsf{sv}(h) = \emptyset$. Similarly, if the celerity of v in the current state is null, its qualitative value cannot change because of an infinite touch delay.

Figure 2 illustrates several evolutions of a gene network. From a particular hybrid state h_0, the dynamics alternates continuous transitions (within the discrete state) and discrete transitions (when changing the discrete state). When the trajectory reaches an external or internal wall (see Fig. 2-Right), the variable slides along the wall only if the celerity of some other variable can drive the trajectory in such a direction. This description leads to the following definition.

▶ **Definition 7** (Hybrid State Space). Let $R = (V, M, E, C)$ be a GRN. We note $\mathcal{R} = (S, cT, dT)$ the *hybrid state space* of R where S is the set of hybrid states, and cT (resp. dT) is the set of *continuous* (resp. *discrete*) *transitions*:

1. There exists a continuous transition in cT from state $h = (\eta, \pi)$ to state $h' = (\eta', \pi')$ iff:
   - **a.** Either $\mathsf{first}(h) \neq \emptyset$ and there exists a variable $v \in \mathsf{first}(h)$ such that:
     - **i.** $\delta_h(v) \neq 0$, where $\delta_h(v)$ is called the *duration* of the (continuous) transition,
     - **ii.** $\eta' = \eta \quad \text{and} \quad \pi'_u = \begin{cases} 0 \text{ if } C_{u,\rho(h,u),\eta_u} < 0 \\ 1 \text{ if } C_{u,\rho(h,u),\eta_u} > 0 \end{cases} \quad \text{for all } u \in (\mathsf{first}(h) \cup \sigma)$
       where $\sigma = \{x \in \mathsf{sv}(h) \mid \delta_h(x) \leqslant \delta_h^{\mathsf{first}}\}$,
     - **iii.** $\forall z \in V \setminus (\mathsf{first}(h) \cup \sigma)$, then $\pi'_z = \pi_z + \delta_h(v) \times C_{z,\rho(h,z),\eta_z}$.
   - **b.** Or $\mathsf{first}(h) = \emptyset$ (meaning that each variable v either reaches an equilibrium state: $C_{v,\rho(h,v),\eta_v} = 0$; or faces a wall: $v \in \mathsf{sv}(h)$) and:
     - **i.** $\forall v \in \mathsf{sv}(h)$, $\pi'_v = \begin{cases} 0 \text{ if } C_{v,\rho(h,v),\eta_v} < 0 \\ 1 \text{ if } C_{v,\rho(h,v),\eta_v} > 0 \end{cases}$
     - **ii.** $\forall u \notin \mathsf{sv}(h)$, $\pi'_u = \pi_u$ (since in this case $C_{u,\rho(h,u),\eta_u} = 0$).
2. There exists a discrete transition in dT from state $h' = (\eta', \pi')$ to state $h'' = (\eta'', \pi'')$ iff there exists a variable $v \in \mathsf{first}(h')$ such that:
   - **a.** $\delta_{h'}(v) = 0$, where $\delta_{h'}(v)$ is called the *duration* of the (discrete) transition,
   - **b.** $\eta''_v = \eta'_v + \mathrm{sgn}(C_{v,\rho(h',v),\eta'_v}) \quad \text{and} \quad \pi''_v = \begin{cases} 0 \text{ if } C_{v,\rho(h',v),\eta'_v} > 0 \\ 1 \text{ if } C_{v,\rho(h',v),\eta'_v} < 0 \end{cases}$,
   - **c.** $\forall u \in V \setminus \{v\}$, $\eta''_u = \eta'_u$ and $\pi''_u = \pi'_u$.

The states from which there do not exist any transitions (discrete or continuous) are called **steady states**.

The continuous transitions lead to the last hybrid state inside the current discrete state, at which point a qualitative change can happen. The *instantaneous* discrete transitions make the system evolve, as soon as the system can (that is, when $\delta_{h'}(v) = 0$), into the next qualitative state by going through a border. These two different kinds of transitions can be observed on Fig. 3 where the discrete transitions are in dotted lines and the continuous transitions are in plain lines. Let us remark that there is a unique continuous transition starting at a given hybrid state. Indeed, assuming that there exist two continuous transitions $h \to h_1$ and $h \to h_2$ from the same hybrid state h, the item 1 of the previous definition leads to the equality $h_1 = h_2$ regarding the ends of the continuous transitions (whatever the value of the set $\mathsf{first}(h)$).

Let us notice that the defined linear hybrid automata leads to an undeterministic behaviour: when the celerity vector allows the trajectory to reach more than one border at the same time, several discrete transitions can be considered (see Fig. 2-Center). Some of these discrete transitions can be forbidden in case of internal wall.

3 Hybrid Hoare Logic

This section is dedicated to the presentation of the Hoare logic adapted to our hybrid formalism. Hoare logic is based on Hoare triples noted $\{Pre\}\ p\ \{Post\}$ meaning that if a program p is executed from a state satisfying a precondition Pre, then after execution, the postcondition $Post$ is true. In our case, the program p is replaced by a biological trace characterising biological knowledge on the chronometrical qualitative behaviour of the system.

For this, we define the property language used for pre- and postconditions in Subsec. 3.1 and the path language used to describe observed traces in Subsec. 3.2. Then, Hoare logic is defined using these languages and we give in Subsec. 3.3 an adaptation of the weakest precondition calculus, that is, the computation of the weakest (the most general) precondition that makes the trace possible and such that the postcondition $Post$ is satisfied afterwards.

In the rest of this section, we denote by $\square$ any of the usual comparison symbols on integers or real numbers: $<$, $\leq$, $>$, $\geq$, $=$, $\neq$.

3.1 Property Language

We first define the property language describing pre- and postconditions.

▶ **Definition 8** (Property Language $\mathcal{L}_C$). The *terms* of the property language $\mathcal{L}_C$ are inductively defined as follows:

- A *discrete term* is a variable η_u with $u \in V$, or a constant of $\mathbb{N}$;
- A *continuous term* is a variable π_u or π'_u with $u \in V$, or $C_{u,\omega,n}$ with $u \in V$, $\omega \subset R^-(u)$ and $n \in [\![0, b_u]\!]$, or a constant of $\mathbb{R}$;
- The connectives $+$, $-$, $\times$ and $/$ create new terms by composition, the latter being only valid for continuous terms. We use their usual semantics.

Discrete atoms are of the form $n \square n'$ where n and n' are discrete terms and *continuous atoms* are of the form $f \square f'$ where f and f' are continuous terms.

The *discrete conditions* are defined by: $D \quad ::== \quad a_d \mid \neg D \mid D \wedge D \mid D \vee D$ where a_d is a discrete atom.

The *hybrid conditions* are defined by: $H \quad ::== \quad a_d \mid a_c \mid \neg H \mid H \wedge H \mid H \vee H$ where a_d and a_c are respectively a discrete atom and a continuous one.

A *property* is a couple (D, H) formed by a discrete and a hybrid condition. All such couples (D, H) form the *property language* $\mathcal{L}_C$.

A hybrid state h satisfies a property $\varphi = (D, H) \in \mathcal{L}_C$ iff both D and H hold in h, by using the usual meaning of the connectives; in this case, we note $h \vDash \varphi$.

3.2 Path Language

The path language given in Def. 12 takes the role of an imperative program in a Hoare triple by describing a biological behaviour. Such a path consists in explicit discrete transitions as given in Def. 9, but also in continuous transitions described by duration and some information, see Def. 10. The characterisation of continuous transitions is based on two kinds of atoms: $C_v \square c$ constrains the value of the current celerity of v, and $\mathsf{slide}(v)$ constrains v to slide.

▶ **Definition 9** (Discrete Path Atom). The *(discrete) path atoms* are defined by:
$dpa \quad ::== \quad v+ \mid v-$ where $v \in V$ is a variable name.

For any states $h = (\eta, \pi)$ and $h' = (\eta', \pi')$, the transition $h \xrightarrow{v+} h'$ (resp. $h \xrightarrow{v-} h'$) is satisfied iff there exists a discrete transition from h to h' so that $\eta'_v = \eta_v + 1$ (resp. $\eta'_v = \eta_v - 1$).

In the following, if $v \in V$ is a variable, $v\pm$ refers indistinctly to $v+$ or $v-$.

▶ **Definition 10** (Assertion Language $\mathcal{L}_A$). The *assertion language* $\mathcal{L}_A$ is defined by the following grammar:

$$a \quad ::== \quad \top \mid C_v \square c \mid \mathsf{slide}(v) \mid \mathsf{slide}^+(v) \mid \mathsf{slide}^-(v) \mid \neg a \mid a \wedge a \mid a \vee a$$

where $v \in V$ is a variable name and $c \in \mathbb{R}$ is a real number.

A couple $(\Delta t, a) \in \mathbb{R}^+ \times \mathcal{L}_A$ of a non-negative real number and an element of the assertion language is called an *assertion couple*.

The following definition gives the semantics of such assertion couples. From an informal point of view, for any states $h = (\eta, \pi)$ and $h' = (\eta, \pi')$ in the same qualitative state η, the continuous transition $h \to h'$ *satisfies* the assertion couple $(\Delta t, a)$ if the continuous transition exists and if it lasts Δt units of time and it respects the assertion a: $\top$ is always true; $C_v \square c$

is satisfied iff $C_{v,\rho(h,v),\eta_v} \,\square\, c$ where $C_{v,\rho(h,v),\eta_v}$ is the celerity of v in the current qualitative state; $\mathsf{slide}^+(v)$ (resp. $\mathsf{slide}^-(v)$) is satisfied iff v faces and reaches a wall at the top of the domain (resp. at the bottom); $\mathsf{slide}(v)$ is a shorthand for $\mathsf{slide}^+(v) \vee \mathsf{slide}^-(v)$; and logical connectives have their usual meanings. We note indifferently $(h,h') \vDash (\Delta t, a)$ or $h \xrightarrow{(\Delta t,a)} h'$.

Regarding Def. 10, the special case where Δt equals 0 characterises a situation where the system enters a qualitative state in a "corner" and no continuous transition is required between two successive discrete transitions.

▶ **Definition 11** (Semantics of the Assertion Couple $(\Delta t, a)$)**.** Let us consider a hybrid state $h = (\eta, \pi)$ and the unique continuous transition starting from h and ending in $h' = (\eta, \pi')$. The satisfaction relation between the continuous transition $h \longrightarrow h'$ and an assertion couple $(\Delta t, a) \in \mathbb{R}^+ \times \mathcal{L}_A$ is noted $(h,h') \vDash (\Delta t, a)$, by overloading of notation, and is defined as follows:

- If $a \equiv \top$, $(h,h') \vDash (\Delta t, a)$ iff $\delta_h^{\mathsf{first}} = \Delta t$.
- If a is of the form $(C_u \,\square\, c)$, $(h,h') \vDash (\Delta t, a)$ iff $\delta_h^{\mathsf{first}} = \Delta t$ and $(C_{u,\rho(h,u),\eta_u} \,\square\, c)$.
- If a is of the form $\mathsf{slide}(v)$, $(h,h') \vDash (\Delta t, a)$ iff $\delta_h^{\mathsf{first}} = \Delta t$ and $\delta_h(v) < \delta_h^{\mathsf{first}}$.
- If a is of the form $\mathsf{slide}^+(v)$ (resp. $\mathsf{slide}^-(v)$), $(h,h') \vDash (\Delta t, a)$ iff $\delta_h^{\mathsf{first}} = \Delta t$ and $\delta_h(v) < \delta_h^{\mathsf{first}}$ and $C_{v,\rho(h,v),\eta_v} > 0$ (resp. $C_{v,\rho(h,v),\eta_v} < 0$).
- If a is of the form $\neg a'$, $(h,h') \vDash (\Delta t, a)$ iff $\delta_h^{\mathsf{first}} = \Delta t$ and $(h,h') \nvDash (\Delta t, a')$.
- If a is of the form $a' \wedge a''$ (resp. $a' \vee a''$), $(h,h') \vDash (\Delta t, a)$ iff $(h,h') \vDash (\Delta t, a')$ and (resp. or) $(h,h') \vDash (\Delta t, a'')$.

▶ **Definition 12** (Path Language $\mathcal{L}_P$)**.** The *(discrete) paths* are defined by:

$$p ::== \ \varepsilon \mid (\Delta t, a, v\pm) \mid p \ ; \ p$$

where $(\Delta t, a)$ is an assertion couple and $v\pm$ is a discrete path atom. The semantics of a path p is given by the binary relation $\xrightarrow{p}$ between states defined by:

- If $p = \varepsilon$, then $h_1 \xrightarrow{p} h_2$ iff $h_1 = h_2$;
- If $p = (\Delta t, a, v\pm)$, then $h_1 \xrightarrow{p} h_2$ iff there exists a state h'_1 s.t. $h_1 \xrightarrow{(\Delta t,a)} h'_1$ and $h'_1 \xrightarrow{v\pm} h_2$;
- If $p \equiv p_1; p_2$, then $h_1 \xrightarrow{p} h_2$ iff there exists a state h_3 s.t. $h_1 \xrightarrow{p_1} h_3$ and $h_3 \xrightarrow{p_2} h_2$.

A path containing only $(\Delta t, a, v\pm)$ is called an *elementary path.*

The path language allows the modeller to express experimental biological traces as sequences of elementary paths. The next section shows how such information can be formally taken into account in order to help the identification of celerities compatible with such paths.

3.3 Hoare Triples and Weakest Precondition

We are now able to give the definition (Def. 13) of a Hoare triple in the scope of our hybrid formalism which is a natural extension of the classical definition. Figure 3 gives an example of a valid Hoare triple.

▶ **Definition 13** (Hybrid Hoare Triples)**.** A Hoare triple for a given GRN is an expression of the form $\{\mathit{Pre}\}\ p\ \{\mathit{Post}\}$ where *Pre* and *Post*, called *precondition* and *postcondition* respectively, are properties of $\mathcal{L}_C$, and p is a path from $\mathcal{L}_P$. A Hoare triple $\{\mathit{Pre}\}\ p\ \{\mathit{Post}\}$ is *satisfied* iff for all state $h_1 \vDash \mathit{Pre}$, there exists another state h_2 so that $h_1 \xrightarrow{p} h_2$ and $h_2 \vDash \mathit{Post}$.

Now that the semantics of this new Hoare logic is defined, we aim at adapting the weakest precondition calculus as proposed by Dijkstra [6] to our hybrid framework (Def. 14).

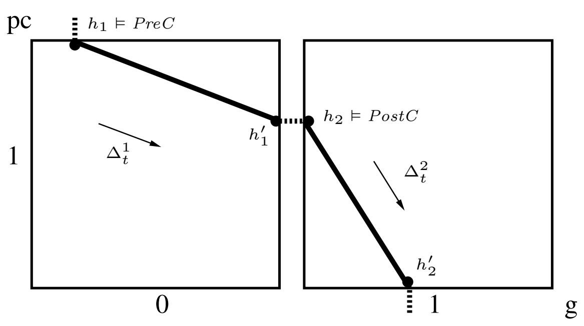


Figure 3 Hoare triple example: $\{PreC\}$ $(\Delta_t^1, \top, g+)$ $\{PostC\}$**.** Starting from the hybrid state $h_1 \vDash PreC$, and considering the path in bold line, it is possible to chain a continuous transition $(h_1 \to h'_1)$ of duration Δ_t^1 and a discrete transition $(h'_1 \to h_2)$ leading to a $h2 \vDash PostC$: this Hoare triple is therefore satisfied.

Edsger Dijkstra introduced a *predicate transformer semantics*: the semantics of an imperative programming language is defined by assigning to each instruction in this language a corresponding predicate transformer. For each elementary instruction *EI* of the imperative programming language, the weakest precondition of *EI* is a function mapping any postcondition *Post* to a precondition *Pre*. Actually, this function returns the weakest precondition on the initial state ensuring that the execution of *EI* terminates in a final state satisfying *Post*. For each sequential imperative program "$P; EI$" whose last instruction is *EI*, and for each postcondition *Post*, the predicate transformer of *EI* allows us to first determine the weakest precondition just before the last instruction and by iterating the same process, it becomes possible to determine the weakest precondition of the whole imperative program (loops are treated in a particular way with the help of *invariants*).

In our setting, the same approach leads to build the minimal constraints on the celerities insuring that starting from a state satisfying the precondition *Pre*, the model exhibits the known path p (corresponding to a biological trace) leading to a state satisfying the postcondition *Post*. Each constraint depends on each elementary path which is defined by the time Δt spent in the current qualitative state, the assertion a and the discrete path atom $v\pm$. Each elementary path takes the role of an elementary instruction.

▶ **Definition 14** (Weakest Precondition)**.** Let p be a path program and $Post = (D, H_f)$ be a post-condition parameterized by a final state index f. The *weakest precondition* attributed to p and *Post* is a property: $\mathsf{WP}^i_f(p, Post) \equiv (D', H'_{i,f})$, parameterized by a fresh initial state index i and the same final state f, and whose value is recursively defined by:

- If $p = \varepsilon$ is the empty sequence program, then $D' \equiv D$ and $H'_{i,f} \equiv H_f$;
- If $p = (\Delta t, a, v+)$ is an atom, with $v \in V$:
  - $D' \equiv D[\eta_v \backslash \eta_v + 1]$,
  - $H'_{i,f} \equiv H_f[\eta_v \backslash \eta_v + 1] \wedge \Phi_v^+(\Delta t) \wedge \mathcal{F}(\Delta t) \wedge \neg \mathcal{W}_v^+ \wedge \mathcal{A}(\Delta t, a) \wedge \mathcal{J}_v$;
- If $p = (\Delta t, a, v-)$ is an atom, with $v \in V$:
  - $D' \equiv D[\eta_v \backslash \eta_v - 1]$,
  - $H'_{i,f} \equiv H_f[\eta_v \backslash \eta_v - 1] \wedge \Phi_v^-(\Delta t) \wedge \mathcal{F}(\Delta t) \wedge \neg \mathcal{W}_v^- \wedge \mathcal{A}(\Delta t, a) \wedge \mathcal{J}_v$;
- If $p = p_1; p_2$ is a concatenation of programs:

$$\mathsf{WP}^i_f(p_1; p_2, Post) \equiv \mathsf{WP}^i_m(p_1, \mathsf{WP}^m_f(p_2, Post))$$

  which is parameterized by a fresh intermediate state index m;

where $\Phi_v^+(\Delta t)$, $\Phi_v^-(\Delta t)$, $\mathcal{W}_v^+$, $\mathcal{W}_v^-$, $\mathcal{F}(\Delta t)$, $\mathcal{A}(\Delta t, a)$ and $\mathcal{J}_v$ are sub-properties detailed in Appendix A.

We note that in the cases corresponding to atoms $p = (\Delta t, a, v\pm)$, the formula $H'_{i,f}$ contains H_f with substitutions, in conjunction with $\Phi_v^\pm(\Delta t)$, $\neg\mathcal{W}_v^\pm$, $\mathcal{F}(\Delta t)$, $\mathcal{A}(\Delta t, a)$ and $\mathcal{J}_v$, which makes the weakest precondition of a sequence of instructions very big and difficult to compute or analyse by hand. Nevertheless, each of the previous subformulas corresponds to a condition which has to be met to allow the execution of an atomic instruction $(\Delta t, a, v\pm)$:

- The sign of the celerity of v in the current state is given by $v\pm$;
- Traversing the qualitative state lasts Δt units of time ($\Phi_v^\pm(\Delta t)$);
- There is no internal or external wall preventing v to increase or decrease its qualitative state ($\neg\mathcal{W}_v^\pm$);
- All components other than first changing variables must either reach their border after v, or face an internal or external wall ($\mathcal{F}(\Delta t)$).
- The assertion a is verified along the continuous transition ($\mathcal{A}(\Delta t, a)$);
- The continuous transition inside a discrete state links the fractional parts of v, its celerity and time spent in the current discrete state. Similarly the discrete transition indicates that the fractional parts of states before and after a discrete transition are the same except for the variable v changing its discrete level ($\mathcal{J}_v$).

Finally, the computation of the weakest precondition for a given Hoare triple $\{Pre\}\ p\ \{Post\}$ is automated using the classical backward proof strategy:

- If p is of the form $(\Delta t, a, v\pm)$ or ε, then we compute the precondition.
- If $p = p_1; p_2$ with $p_2 = (\Delta t, a, v\pm)$, we compute the precondition before p_2 and we iterate for path p_1 (we never consider p_2 as ε).

These two items are mutually exclusive which means that the proof strategy generates a unique proof tree.

An implementation of this weakest precondition calculus has been realised[2]. Section 5 details its result on a model of the circadian clock and before that, next section gives the theorem of its soundness.

4 Soundness of the Hybrid Hoare Logic

4.1 Inference Rules and Axioms

The considered Hoare logic for hybrid gene regulatory networks is defined by the following inference rules:

Incrementation rule:
$$\frac{}{\begin{Bmatrix} D[\eta_v \backslash \eta_v + 1] \\ H'_{i,f} \end{Bmatrix} \begin{pmatrix} \Delta t \\ a \\ v+ \end{pmatrix} \begin{Bmatrix} D \\ H_f \end{Bmatrix}}$$

Decrementation rule:
$$\frac{}{\begin{Bmatrix} D[\eta_v \backslash \eta_v - 1] \\ H'_{i,f} \end{Bmatrix} \begin{pmatrix} \Delta t \\ a \\ v- \end{pmatrix} \begin{Bmatrix} D \\ H_f \end{Bmatrix}}$$

where v is a variable, η_v its expression level, D (resp. H) the discrete (resp. hybrid) condition, $H'_{i,f} \equiv H_f[\eta_v \backslash \eta_v + 1] \wedge \Phi_v^+(\Delta t) \wedge \mathcal{F}(\Delta t) \wedge \neg\mathcal{W}_v^+ \wedge \mathcal{A}(\Delta t, a) \wedge \mathcal{J}_v$ (Incrementation rule), or $H'_{i,f} \equiv H_f[\eta_v \backslash \eta_v - 1] \wedge \Phi_v^-(\Delta t) \wedge \mathcal{F}(\Delta t) \wedge \neg\mathcal{W}_v^- \wedge \mathcal{A}(\Delta t, a) \wedge \mathcal{J}_v$ (Decrementation rule),

[2] Available at: `http://www.i3s.unice.fr/~comet/DOCUMENTS/hybridisation.tar.gz`.

both detailed in Appendix A, Δt the time spent inside the current discrete state and a an assertion. The last inference rule is the sequential composition rule:

$$\textbf{Sequential composition rule:}\ \frac{\{Q_1\}\ p_1\ \{Q_3\} \qquad \{Q_3\}\ p_2\ \{Q_2\}}{\{Q_1\}\ p_1; p_2\ \{Q_2\}}$$

where Q_1, Q_2, Q_3 are properties of the form (D, H) having the role of pre- and postconditions, and p_1 and p_2 are particular paths deduced from biological experiments.

The two following axioms, based on the semantics of the hybrid model, complement the inference rules:

- $\eta_v \geq 0 \wedge \eta_v \leq b_v$ (the expression level has to be in its definition domain),
- $C_{v,\omega,\eta_v} \times C_{v,\omega,\eta_v+1} \geq 0$ (for two neighbour qualitative states, if the variable v is controlled by the same resources, then the celerities of v cannot be of opposite signs).

4.2 Soundness of the Hoare logic

The following lemmas are useful for the proof of soundness. Lemma 15 states that the time spent in the current discrete state is equal to the time mandatory, for the variable which changes first, to reach its border. Lemma 16 expresses the fact that the truth value of a formula remains the same after a continuous transition.

▶ **Lemma 15** (Time Spent in a Discrete State). *Let h be a hybrid state. If $h \vDash (D_1, H_1)$ and $H_1 \Rightarrow \Phi_v^+(\Delta t) \wedge \mathcal{F}(\Delta t)$ (resp. $H_1 \Rightarrow \Phi_v^-(\Delta t) \wedge \mathcal{F}(\Delta t)$), then: $\delta_h^{\mathsf{first}} = \delta_h(v) = \Delta t$.*

Proof. Let us consider $H_1 \Rightarrow \Phi_v^+(\Delta t) \wedge \mathcal{F}(\Delta t)$ (resp. $H_1 \Rightarrow \Phi_v^-(\Delta t) \wedge \mathcal{F}(\Delta t)$). Let $h = (\eta, \pi)$ be a hybrid state such that $h \vDash (D_1, H_1)$.
From the definition of the sub-property $\Phi_v^+(\Delta t)$, see Appendix A.2, variable v reaches its upper border ($\pi_v^{i'} = 1$) and its celerity is positive ($C_{v,\omega,n} > 0$). Let $h' = (\eta, \pi')$ be the hybrid state where v first touches this border. The time spent in the current qualitative state η corresponds to the time necessary to reach the border where v changes its qualitative level. Indeed, from $\Phi_v^+(\Delta t)$ we deduce:

$$\pi_v^i = \pi_v^{i'} - C_{v,\omega,n} \cdot \Delta t = 1 - C_{v,\omega,n} \cdot \Delta t, \qquad \text{that is,} \qquad \Delta t = \frac{1 - \pi_v^i}{C_{v,\omega,n}}.$$

From Def. 4, we have $\Delta t = \delta_h(v)$.

Finally, the sub-property $\mathcal{F}(\Delta t)$, see Appendix A.4, expresses the fact that if a variable u different from v reaches its border before v, u faces an internal or external wall (see Appendix A.3). Thus, since $\delta_h(v) = \Delta t$ and according to Def. 6, we deduce $\delta_h^{\mathsf{first}} = \delta_h(v) = \Delta t$. ◀

▶ **Lemma 16** (Preservation of Formulas Evaluation Along a Continuous Transition). *Let us consider a Hoare triple $\{(D', H'_{i,f})\}\ p\ \{(D, H_f)\}$ obtained by the weakest precondition calculus, a hybrid state $h = (\eta, \pi^i)$, and finally the unique continuous transition $h \to h'$ starting from h. If $h \vDash (D', H'_{i,f})$, then $h' \vDash (D', H'_{i,f})$.*

Proof.

- Since h and h' belong to the same discrete state, the expression levels of all variables are the same. The evaluation of D' in h' is then the evaluation of D' in h.
- The atoms of $H'_{i,f}$ (see subformulas in Appendix A) concern either celerities or discrete or continuous coordinates of different points of the trajectory (entrance and arrival points in different discrete states). These points are either outside the current discrete state, or are the points h or h'.

Moreover, the celerities are constants, the points of the trajectory which do not belong to the current discrete state as well as the points h and h' do not change.

The interpretation of $(D', H'_{i,f})$ is therefore the same in h and h'. ◀

The soundness of the modified Hoare logic, adapted for the hybrid modelling framework, means that if a Hoare triple is built in agreement with the inference rules (Def. 14) then this Hoare triple is satisfied according to the semantics of Hoare triples (Def. 13).

▶ **Theorem 17.** *The hybrid Hoare logic is sound.*

The proof is detailed in Appendix B.

5 Example: Simplified Circadian Cycle

The circadian rhythm is a biological process regulating cells of an organism with a 24-hour period and controlling the electrical and metabolic processes.

5.1 Presentation of the Circadian Cycle

In mammals, the main circadian cycle is located in the suprachiasmatic nucleus and regulates the peripheral clocks. It is affected by light, acting like a synchronizer called *Zeitgeber*, which means "giver of time".

The circadian rhythm is mainly controlled by two protein complexes which are PER/CRY and BMAL1/CLOCK. When light appears, the BMAL1/CLOCK complex activates the *per* and *cry* genes by binding the *E-box* response element in the promoter upstream these genes [12]. The PER and CRY proteins are synthesized and dimerised in the cytoplasm. During night, this complex is found inside the nucleus and inhibits BMAL1/CLOCK implying a negative feedback of PER/CRY on its genes. Finally, PER/CRY is degraded by proteasome. The circadian cycle completes and a new one begins with the transcription of genes *bmal1* and *clock*.

We decided to use an interaction graph which focuses on the *per* and *cry* components, as represented in Fig. 1. The Light/Day cycle (whose duration is 12h/12h) is represented by the node named L. (Let us notice that the node labelled X is a modelling artefact to get an oscillating feature for light.) This node enhances the *per* and *cry* genes (modelled with node g) when the light is activated, that is, when the qualitative value of L is at level 1. These genes synthesize their proteins which complex and spread inside the nucleus. When the complex is activated (which is modelled by an expression level of 1 for pc), those genes are inhibited, blocking the synthesis during night. Because genes are disabled, the protein complex will be degraded by proteasome and a new cycle begins with the reactivation of the genes. All four nodes of this model have two qualitative levels of expression named 0 (not active) and 1 (active).

5.2 Hoare Triple and Results

The steps of the circadian clock explained in Subsec. 5.1 are represented in the Hoare Triple below. The time spent in each qualitative state comes from biological information obtained during a light/day cycle of 12h/12h. The assertions $\mathsf{slide}^{+}(g)$ and $\mathsf{slide}^{+}(pc)$ (resp. $\mathsf{slide}^{-}(L)$) characterize a saturation (resp. complete degradation) of g and pc (resp. of L, corresponding

Table 1 Constraints obtained by computation of the weakest precondition. Left: Constraints on celerities of g and pc. Right: Constraints on celerities of L and X.

Constraints on celerities of g and pc		**Constraints on celerities of L and X**	
$C_{g,\emptyset,0} < 0$	$0 < C_{g,\{pc,L\}0}$	$-\frac{1}{0.6} < C_{L,\emptyset,0} < 0$	$C_{X,\emptyset,0} < 0$
$C_{g,\emptyset,1} < 0$	$0 < C_{g,\{pc,L\},1} < \frac{1}{5.53}$	$C_{L,\emptyset,1} < 0$	$-\frac{1}{6} \leqslant C_{X,\emptyset,1} < 0$
$C_{g,\{L\},0} < 0$	$C_{pc,\emptyset,0} < 0$	$0 < C_{L,\{X\},0}$	$0 < C_{X,\{L\},0} < \frac{1}{5.1}$
$C_{g,\{L\},1} < 0$	$C_{pc,\emptyset,1} = -\frac{0.12}{0.9}$	$0 < C_{L,\{X\},1}$	$0 < C_{X,\{L\},1}$
$C_{g,\{pc\},0} > 0$	$0 < C_{pc,\{g\},0} < \frac{1}{6.13}$		
$C_{g,\{pc\},1} > 0$	$0 < C_{pc,\{g\},1} < \frac{1}{5.4}$		

to the beginning of the night). This information is summed up in the following Hoare triple:

$$\begin{Bmatrix} D_8 \\ H_8 \end{Bmatrix} \begin{pmatrix} 0.9 \\ \top \\ pc- \end{pmatrix}; \begin{pmatrix} 4.5 \\ \top \\ g+ \end{pmatrix}; \begin{pmatrix} 0.6 \\ \top \\ X+ \end{pmatrix}; \begin{pmatrix} 5.53 \\ \mathsf{slide}^+(g) \\ pc+ \end{pmatrix}; \begin{pmatrix} 0.47 \\ \top \\ L- \end{pmatrix}; \begin{pmatrix} 5.4 \\ \mathsf{slide}^+(pc) \\ g- \end{pmatrix}; \begin{pmatrix} 0.6 \\ \mathsf{slide}^-(L) \\ X- \end{pmatrix}; \begin{pmatrix} 6 \\ \top \\ L+ \end{pmatrix} \begin{Bmatrix} D_0 \\ H_0 \end{Bmatrix}$$

$$\text{where} \quad \begin{cases} D_0 \equiv (\eta_g = 0) \wedge (\eta_{pc} = 1) \wedge (\eta_L = 1) \wedge (\eta_X = 0) \\ H_0 \equiv (\pi_g = 0.12) \wedge (\pi_{pc} = 0.12) \wedge (\pi_L = 0) \wedge (\pi_X = 0) \end{cases}$$

Using this Hoare triple, we compute *via* the backward strategy the weakest precondition iteratively by crossing the intermediate states of the path. In addition, because the behaviour is cyclic, we know that the starting hybrid state of this path is equal to the finishing one. The provided implementation allows us to identify and simplify the constraints of the celerities obtained through our weakest precondition calculus. After some automatic and manual simplifications, we finally obtain the constraints summed up in Table 1 which make the known cyclic behaviour possible.

In order to illustrate the validity of this process, we used the constraint solver Ibex[3] to extract values satisfying the previous constraints. This constraint solver takes as input only conjunctions. Thus the obtained constraints are transformed in a disjunctive normal form (we obtained 3 terms in disjunction) and all terms of this disjunction are successively given to Ibex to extract possible values for variables. Amongst the values returned by Ibex, we arbitrarily chose one set of possible values to be injected in the model for simulation. Interestingly, the obtained constraints fully characterize one of the hybrid states along the limit cycle, which gives us an initial state for the simulation.

The simulated traces have then to be compared to biological experimental data. Because such data for the PER/CRY protein complex inside the nucleus are not published, the simulation (Fig. 4) is compared to experimental data of genes *per1*, *per2*, *cry1* and *cry2* as well as their respective proteins taken separately [13]. We noticed that the maximal activity of *per* genes of our simulation and experimental data occurs at the end of a day, and the curves of proteins are maximal during night at the same time slot for both curves. Thus our simplified circadian cycle model is consistent with biological experimental data although we arbitrarily used parameter values satisfying constraints. This simulation reinforces reliability of our formalism for determining constraints.

[3] See `http://www.ibex-lib.org/`.

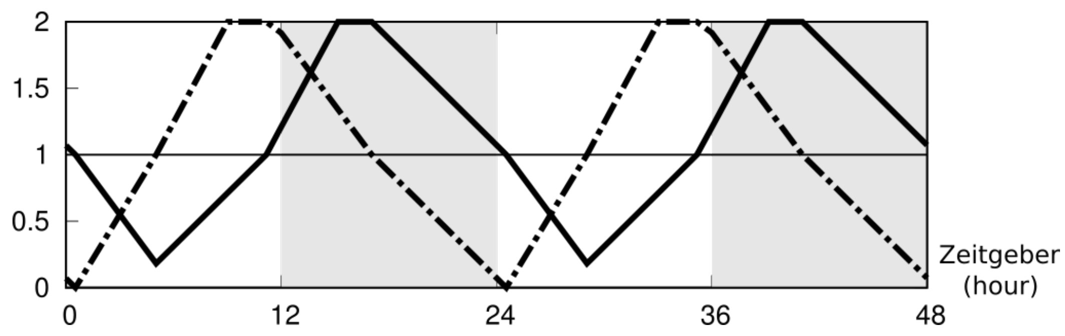


Figure 4 Model simulation based on arbitrarily chosen celerities satisfying the deduced constraints. The plain (resp. dashed) line represents the PER/CRY complex (resp. gene) activity.

6 Conclusion

In this paper, we have developed a suitable approach to determine constraints on the parameters of a linear hybrid automaton. Our hybrid Hoare logic combined with experimental biological traces including precise chronometrical information leads to constraints on celerities which have to be satisfied to allow the model to represent the observed behaviour (HCSP [18] is a formalism similar to ours but does not include chronometrical information along the path). The obtained constraints *via* the weakest precondition calculus are analysed using the solver Ibex which extracts all admissible intervals of celerities. Choosing celerity values satisfying these constraints leads to a model which exhibits simulation traces similar to the aforementioned experimental data, this approach has been tested on the simplified circadian clock model.

The soundness of our hybrid Hoare logic is proved which means that simulations obtained with parameters satisfying the computed precondition leads to simulated traces which are in concordance with the path representing the experimental data.

This work opens many outlooks. Generally, it is useful to prove the completeness of the weakest precondition calculus. Because of the continuous terms (real numbers), our hybrid Hoare logic cannot be complete regarding all possible formulas. Nevertheless, we think that our hybrid Hoare logic should be complete regarding closed propositional formulas constructed from polynomial (in)equations and logical connectors. The decidability of the theory of real closed fields implies that the precondition constraints can be analysed; it does not mean that for each semantically correct Hoare triple, there exists a proof tree built on our inference rules. Completness of our framework would mean that if a Hoare triple is semantically correct and if pre- and postconditions are expressions in the first order language of real closed fields, then there exists a proof tree for this Hoare triple. Finally, because of the combinatorial explosion of the size of the weakest precondition formula and despite some on-the-fly simplifications, it could be interesting to investigate other ways to simplify the result in some particular cases.

Acknowledgements. We are grateful to F. Delaunay for having shared his expertise on the circadian clock and to E. Cornillon and G. Bernot for fruitful discussions about the hybrid formalism.

References

1 J. Behaegel, J.-P. Comet, G. Bernot, E. Cornillon, and F. Delaunay. A hybrid model of cell cycle in mammals. *JBCB*, 14(1):1640001 [17 pp.], 2016.

2 G. Bernot, J.-P. Comet, Z. Khalis, A. Richard, and O. Roux. A genetically modified Hoare logic. *ArXiv: 1506.05887*, June 2015.

3 G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. Application of formal methods to biological regulatory networks: extending thomas' asynchronous logical approach with temporal logic. *Journal of theoretical biology*, 229(3):339–347, 2004.

4 G. Bernot, J.-P. Comet, and O. Roux. A genetically modified Hoare logic that identifies the parameters of a gene networks. In O. Roux and J. Bourdon, editors, *CMSB'15*, volume 9308 of *LNBI*, pages 8–12, 2015.

5 F. Corblin, E. Fanchon, and L. Trilling. Applications of a formal approach to decipher discrete genetic networks. *BMC Bioinformatics*, 11:385, 2010. doi:10.1186/1471-2105-11-385.

6 E. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM*, 18:453–457, August 1975. doi:10.1145/360933.360975.

7 F. Fages. Temporal logic constraints in the biochemical abstract machine BIOCHAM. In *Logic Based Program Synthesis and Transformation, 15th International Symposium, LOPSTR 2005, London, UK, September 7-9, 2005, Revised Selected Papers*, pages 1–5, 2005. doi:10.1007/11680093_1.

8 N. Halbwachs, Y.-É. Proy, and P. Raymond. Verification of linear hybrid systems by means of convex approximations. In Baudouin Le Charlier, editor, *Proceedings of the First International Static Analysis Symposium (SAS'94)*, pages 223–237. Springer Berlin Heidelberg, 1994.

9 T. Henzinger. The theory of hybrid automata. In M. Kemal Inan and Robert P. Kurshan, editors, *Verification of Digital and Hybrid Systems*, pages 265–292. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. doi:10.1007/978-3-642-59615-5_13.

10 T. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. In Orna Grumberg, editor, *Proceedings of the 9th International Conference on Computer Aided Verification (CAV'97)*, pages 460–463. Springer Berlin Heidelberg, 1997.

11 J. Hooman. Extending Hoare logic to real-time. *Formal Aspects of Computing*, 6:801–825, 1994.

12 T. Hunt and P. Sassone-Corsi. Riding tandem: circadian clocks and the cell cycle. *Cell*, 129(3):461–464, 2007.

13 C. Lee, J.-P. Etchegaray, F. Cagampang, A. Loudon, and S. Reppert. Posttranslational mechanisms regulate the mammalian circadian clock. *Cell*, 107(7):855–867, 2001.

14 T. Liu, X. Zhang, and X. Gao. Stability analysis for continuous-time and discrete-time genetic regulatory networks with delays. *Applied mathematics and computation*, 274:628–643, 2016.

15 R. Thomas. Boolean formalization of genetic control circuits. *Journal of theoretical biology*, 42(3):563–585, 1973.

16 R. Thomas and M. Kaufman. Multistationarity, the basis of cell differentiation and memory. II. logical analysis of regulatory networks in terms of feedback circuits. *Chaos*, 11:180–195, 2001.

17 P. Traynard, A. Fauré, F. Fages, and D. Thieffry. Logical model specification aided by model-checking techniques: application to the mammalian cell cycle regulation. *Bioinformatics*, 32(17):772–780, 2016. doi:10.1093/bioinformatics/btw457.

18 N. Zhan, S. Wang, and H. Zhao. Formal modelling, analysis and verification of hybrid systems. In *Unifying Theories of Programming and Formal Engineering Methods*, pages 207–281. Springer, 2013.

A Appendix: Sub-properties of the Weakest Precondition Calculus

In this appendix, we detail each subformula of the weakest precondition in Def. 14.

A.1 Weakest Precondition

In order to fully compute the weakest precondition, it is required to label the fractional parts of the states mentioned in the properties. For this, we use labels called below f (final), i (initial) and m (intermediate). Moreover, by convention, we use π' (resp. π) to specify the fractional part of the exit from the current discrete state (resp. entrance into the current discrete state).

Let us notice that all the following properties depend on the indices i and f used in Def. 14, although for readability issues we did not mention them on the names of each sub-property. Furthermore, for a given index i, we call by convention π_u^i (resp. $\pi_u^{i\,\prime}$) the fractional part of the entering (resp. exiting) state inside the discrete state i.

Finally, for all variable $u \in V$ and all $\omega \subset R^-(v)$ subset of predecessors of u, we define:

$$\Phi_v^\omega \equiv \Big(\bigwedge_{m \in \omega} \varphi_m \Big) \wedge \Big(\bigwedge_{m \in R^{-1}(v) \setminus \omega} \neg \varphi_m \Big) \ .$$

In other words, Φ_v^ω is true in a state h if and only if the resources of u are exactly ω, that is, $\rho(h, v) = \omega$.

A.2 Discrete Transition to the Next Discrete State

For all component $v \in V$, $\Phi_v^+(\Delta t)$ (resp. $\Phi_v^-(\Delta t)$) describes the conditions in which v increases (resp. decreases) its discrete expression level after Δt units of time: its celerity in the current state must be positive (resp. negative) and its fractional part only depends on Δt in the way given at the very end of Section 2.

$$\Phi_v^+(\Delta t) \equiv (\pi_v^{i\,\prime} = 1) \wedge \bigwedge_{\substack{\omega \subset R^-(v) \\ n \in [\![0, b_v]\!]}} \Bigg(\begin{pmatrix} \Phi_v^\omega \ \wedge \\ (\eta_v = n) \end{pmatrix} \Rightarrow (C_{v,\omega,n} > 0) \wedge (\pi_v^i = \pi_v^{i\,\prime} - C_{v,\omega,n} \cdot \Delta t) \Bigg),$$

$$\Phi_v^-(\Delta t) \equiv (\pi_v^{i\,\prime} = 0) \wedge \bigwedge_{\substack{\omega \subset R^-(v) \\ n \in [\![0, b_v]\!]}} \Bigg(\begin{pmatrix} \Phi_v^\omega \ \wedge \\ (\eta_v = n) \end{pmatrix} \Rightarrow (C_{v,\omega,n} < 0) \wedge (\pi_v^i = \pi_v^{i\,\prime} - C_{v,\omega,n} \cdot \Delta t) \Bigg).$$

A.3 Internal and External Walls

For all component $u \in V$, $\mathcal{W}_u^+$ (resp. $\mathcal{W}_u^-$) states that there is a wall preventing u to increase (resp. decrease) its qualitative state. This wall can either be an external wall EW_u^+ (resp. EW_u^-) or an internal wall IW_u^+ (resp. IW_u^-). Furthermore, $\Phi_{u+}^{\omega'}$ (resp. $\Phi_{u-}^{\omega'}$), which is required in these subformulas, is true if and only if the set of resources of u is exactly ω' in the state where u is increased (resp. decreased) by 1.

$$\mathcal{W}_u^+ \equiv \mathsf{IW}_u^+ \vee \mathsf{EW}_u^+ \qquad \text{and} \qquad \mathcal{W}_u^- \equiv \mathsf{IW}_u^- \vee \mathsf{EW}_u^-$$

where:

$$\mathsf{EW}_u^+ \equiv (\eta_u = b_u) \wedge \bigwedge_{\omega \subset R^-(u)} (\Phi_u^\omega \Rightarrow C_{u,\omega,b_u} > 0) \ \ ,$$

$$\mathsf{EW}_u^- \equiv (\eta_u = 0) \wedge \bigwedge_{\omega \subset R^-(u)} (\Phi_u^\omega \Rightarrow C_{u,\omega,0} < 0) \ \ ,$$

$$\mathsf{IW}_u^+ \equiv (\eta_u < b_u) \wedge \bigwedge_{\substack{\omega,\omega' \subset R^-(u)\\ n\in[\![0,b_u]\!]}} \Big(\begin{pmatrix} (\eta_u = n) \wedge \\ (m = n+1) \wedge \\ \Phi_u^{\omega} \wedge \Phi_{u+}^{\omega'} \end{pmatrix} \Rightarrow C_{u,\omega,n} > 0 \wedge C_{u,\omega',m} < 0 \Big) \ ,$$

$$\mathsf{IW}_u^- \equiv (\eta_u > 0) \wedge \bigwedge_{\substack{\omega,\omega' \subset R^-(u)\\ n\in[\![0,b_u]\!]}} \Big(\begin{pmatrix} (\eta_u = n) \wedge \\ (m = n-1) \wedge \\ \Phi_u^{\omega} \wedge \Phi_{u-}^{\omega'} \end{pmatrix} \Rightarrow C_{u,\omega,n} < 0 \wedge C_{u,\omega',m} > 0 \Big) \ ,$$

$$\Phi_{u+}^{\omega'} \equiv (\eta_u < b_u) \wedge \bigwedge_{n\in[\![0,b_u]\!]} \big((\eta_u = n) \Rightarrow \Phi_u^{\omega'}[\eta_u \backslash \eta_u + 1] \big) \ ,$$

$$\Phi_{u-}^{\omega'} \equiv (\eta_u > 0) \wedge \bigwedge_{n\in[\![0,b_u]\!]} \big((\eta_u = n) \Rightarrow \Phi_u^{\omega'}[\eta_u \backslash \eta_u - 1] \big) \ .$$

A.4 First Changing Variables

$\mathcal{F}(\Delta t)$ states that all components that are not first changing variables must either reach their border after the first changing variables, or face an internal or external wall.

$$\mathcal{F}(\Delta t) \equiv \bigwedge_{u \in V \backslash \mathsf{first}(h_i)} \left[\begin{array}{c} \left(\bigwedge_{\substack{\omega \subset R^-(u)\\ n\in[\![0,b_u]\!]}} \begin{pmatrix} (\eta_u = n) \wedge \Phi_u^{\omega} \wedge \\ C_{u,\omega,n} > 0 \wedge \\ \pi_u^i > \pi_u^{i'} - C_{u,\omega,n} \cdot \Delta t \end{pmatrix} \Rightarrow \mathcal{W}_u^+ \right) \wedge \\ \left(\bigwedge_{\substack{\omega \subset R^-(u)\\ n\in[\![0,b_u]\!]}} \begin{pmatrix} (\eta_u = n) \wedge \Phi_u^{\omega} \wedge \\ C_{u,\omega,n} < 0 \wedge \\ \pi_u^i < \pi_u^{i'} - C_{u,\omega,n} \cdot \Delta t \end{pmatrix} \Rightarrow \mathcal{W}_u^- \right) \end{array} \right] .$$

A.5 Hybrid Assertions

The sub-property $\mathcal{A}(\Delta t, a)$ allows one to translate all assertion symbols given about the continuous transition related to the instruction (celerities and slides) into a property:

$$\mathcal{A}(\Delta t, a) \equiv \bigwedge_{\substack{k \in [\![1,n]\!]\\ \omega_k \in R^-(v_k)\\ n_k \in [\![0,b_{v_k}]\!]}} \left(\bigwedge_{l \in [\![1,n]\!]} \Big((\eta_{v_l} = n_l) \wedge \Phi_{v_l}^{\omega_l} \Big) \Rightarrow a \begin{bmatrix} C_{v_l} \backslash C_{v_l,\omega_l,n_l} \\ \mathsf{slide}(v_l) \backslash \mathcal{S}_{v_l,\omega_l,n_l}(\Delta t) \\ \mathsf{slide}^+(v_l) \backslash \mathcal{S}^+_{v_l,\omega_l,n_l}(\Delta t) \\ \mathsf{slide}^-(v_l) \backslash \mathcal{S}^-_{v_l,\omega_l,n_l}(\Delta t) \end{bmatrix} \right)$$

where a is the assert part of the instruction $P = (\Delta t, a, v\pm)$, and, for all variable $u \in V$:

$$\mathcal{S}^+_{u,\omega,n}(\Delta t) \equiv (\pi_u^{i\,\prime} = 1) \wedge (\pi_u^i > \pi_u^{i\,\prime} - C_{u,\omega,n} \cdot \Delta t) \ ,$$

$$\mathcal{S}^-_{u,\omega,n}(\Delta t) \equiv (\pi_u^{i\,\prime} = 0) \wedge (\pi_u^i < \pi_u^{i\,\prime} - C_{u,\omega,n} \cdot \Delta t) \ ,$$

$$\mathcal{S}_{u,\omega,n}(\Delta t) \equiv \mathcal{S}^+_{u,\omega,n}(\Delta t) \vee \mathcal{S}^-_{u,\omega,n}(\Delta t) \ .$$

These sub-properties indicate that the exit position of the corresponding variable u is located on a threshold. In addition, the constraints $\pi_u^i > \pi_u^{i\,\prime} - C_{u,\omega,n} \cdot \Delta t$ and $\pi_u^i < \pi_u^{i\,\prime} - C_{u,\omega,n} \cdot \Delta t$ mean that the duration before reaching the border is lower that the one spent inside the current state (Δt). The sign of the celerity of the sliding variable u is constrained by the sub-property $\mathcal{F}$ and the constraint $\pi_u^i > \pi_u^{i\,\prime} - C_{u,\omega,n} \cdot \Delta t$ (resp. $\pi_u^i < \pi_u^{i\,\prime} - C_{u,\omega,n} \cdot \Delta t$) of the sub-property $\mathcal{S}^+_{u,\omega,n}(\Delta t)$ (resp. $\mathcal{S}^-_{u,\omega,n}(\Delta t)$) or $\mathcal{S}_{u,\omega,n}(\Delta t)$.

A.6 Junctions

A.6.1 Continuous Junctions Inside Discrete States

For all component $v \in V$, and for a continuous transition between two hybrid states $h = (\eta, \pi)$ and $h' = (\eta, \pi')$, $\mathcal{CJ}_v$ establishes a relationship between the fractional parts and the celerity of the variable v. If the exit fractional part of v is 0 or 1, the sign of the celerity can be deduced and the time mandatory to v to reach the border is lower than the time spent in the current discrete state. If v does not reach its border, the exact position of the entrance fractional part of v can be deduced from the exit position, the time spent in the current discrete state and the celerity.

$$\mathcal{CJ}_v \equiv \begin{cases} & (\pi'_v = 0) & \Rightarrow & C_{v,\rho(h,v),\eta_v} < 0 \wedge (\pi_v \leq \pi'_v - C_{v,\rho(h,v),\eta_v} \times \delta_h^{\text{first}}) \\ \wedge & (\pi'_v = 1) & \Rightarrow & C_{v,\rho(h,v),\eta_v} > 0 \wedge (\pi_v \geq \pi'_v - C_{v,\rho(h,v),\eta_v} \times \delta_h^{\text{first}}) \\ \wedge & (0 < \pi'_v < 1) & \Rightarrow & (\pi_v = \pi'_v - C_{v,\rho(h,v),\eta_v} \times \delta_h^{\text{first}}) \ . \end{cases}$$

A.6.2 Discrete Junctions Between Discrete States

For all component $v \in V$, and for a discrete transition happening on component v between an initial and a final state corresponding to the indices i and f, $\mathcal{DJ}_v$ establishes a junction between the fractional parts of these states. This formula states that the fractional part of v switches from 1 to 0 for an increase, or from 0 to 1 for a decrease, whereas all other fractional parts are unchanged:

$$\mathcal{DJ}_v \equiv (\pi_v^f = 1 - \pi_v^{i\,\prime}) \wedge \bigwedge_{u \in V \setminus \{v\}} (\pi_u^f = \pi_u^{i\,\prime}) \ .$$

Finally, we define:

$$\mathcal{J}_v \equiv \mathcal{DJ}_v \wedge \bigwedge_{u \in V} \mathcal{CJ}_u$$

These relationships can be easily observed on Fig. 3 on the discrete transition in the centre: all fractional parts are left the same, except for the variable performing the transition.

B Soundness Proof

The soundness proof is made for each inference rule which depends on its corresponding assertion (Def. 11). Each of them is treated according to the assertion type. We focus here on the proof of the soundness of the incrementation rule since the proof of the soundness of the decrementation rule is similar, and that the one for the sequential composition rule is classical. In this subsection, we consider the Hoare triple associated with the incrementation rule, described in Subsection 4.1, and a hybrid state $h = (\eta, \pi)$ satisfying the precondition.

B.1 First Case: $a = \top$

(α) Let us first prove the existence of the continuous transition. According to the sub-property $\Phi_v^+(\Delta t)$, $\pi'_v = 1$ (arrival at the top border of v), $C_{v,\omega,n} > 0$ and $\pi'_v = \pi_v + C_{v,\omega,n} \times \Delta t$ (the time spent in the current state is Δt) if Φ_v^ω and $\eta_v = n$ are satisfied. Let us also consider the unique hybrid state $h' = (\eta, \pi')$ such that the continuous transition $h \to h'$ exists. Thus, according to Lemma 15 and Definition 11, $(h, h') \vDash (\Delta t, \top)$.

(β) Let us now prove the existence of the discrete transition. Let us simplify the subformula $\neg\mathcal{W}_v^+ \equiv \neg\mathsf{EW}_v^+ \wedge \neg\mathsf{IW}_v^+$ satisfied at h'. We have:

$$\neg\mathsf{EW}_v^+ \equiv \neg\Big[\underbrace{(\eta_v = b_v)}_{\perp} \wedge \bigwedge_{\omega \subset R^-(v)} (\Phi_v^\omega \Rightarrow C_{v,\omega,b_v} > 0)\Big] \equiv \top$$

which is evaluated to true because v increases its level ($v+$ is the discrete path atom) and thus is not already at its maximal discrete value. Thus, $\neg\mathcal{W}_v^+ \equiv \neg\mathsf{IW}_v^+$:

$$\neg\mathcal{W}_v^+ \equiv \neg\left[(\eta_v < b_v) \wedge \bigwedge_{\substack{n\in[\![0,b_v]\!]\\ \omega,\omega'\subset R^-(v)}} \left(\begin{pmatrix} (\eta_v = n)\wedge \\ (m = n+1)\wedge \\ \Phi_v^\omega \wedge \Phi_{v+}^{\omega'} \end{pmatrix} \Rightarrow \begin{pmatrix} C_{v,\omega,n} > 0 \\ \wedge \\ C_{v,\omega',m} < 0 \end{pmatrix}\right)\right]$$

$$\equiv \underbrace{\underbrace{\neg(\eta_v < b_v)}_{\perp}}_{\text{since } \eta_v < b_v} \vee \bigvee_{\substack{n\in[\![0,b_v]\!]\\ \omega,\omega'\subset R^-(v)}} \neg\left(\begin{pmatrix} (\eta_v = n)\wedge \\ (m = n+1)\wedge \\ \Phi_v^\omega \wedge \Phi_{v+}^{\omega'} \end{pmatrix} \Rightarrow \begin{pmatrix} C_{v,\omega,n} > 0 \\ \wedge \\ C_{v,\omega',m} < 0 \end{pmatrix}\right)$$

Amongst all premises of the remaining disjunctions, only one is satisfied because the current qualitative state and the next state have a unique qualitative level ($\eta_v = n$ and $m = n+1$) and a unique set of resources (Φ_v^ω and $\Phi_{v+}^{\omega'}$). Replacing ω and ω' by the right resources of the corresponding states $\rho(\eta, v)$ and $\rho(\eta'', v)$ and naming η'' the next state, we deduce:

$$\neg\mathcal{W}_v^+ \equiv \neg\left(\underbrace{\neg\begin{bmatrix} (\eta_v = n) \;\; \wedge \\ (m = n+1) \;\; \wedge \\ \Phi_v^{\rho(\eta,v)} \wedge \Phi_{v+}^{\rho(\eta'',v)} \end{bmatrix}}_{\perp} \vee \begin{pmatrix} C_{v,\rho(\eta,v),\eta_v} > 0 \\ \wedge \\ C_{v,\rho(\eta'',v),\eta''_v} < 0 \end{pmatrix}\right) \equiv \begin{pmatrix} C_{v,\rho(\eta,v),\eta_v} \leq 0 \\ \vee \\ C_{v,\rho(\eta'',v),\eta''_v} \geq 0 \end{pmatrix}$$

However, since $\Phi_v^+(\Delta t)$ is true at h', we have $C_{v,\rho(\eta,v),\eta_v} > 0$. Thus $\neg\mathcal{W}_v^+$ is equivalent to $C_{v,\rho(\eta'',v),\eta''_v} \geq 0$ and the previous inequation is true since $\neg\mathcal{W}_v^+$ is satisfied at h'. Thus the variable v reaches its threshold in Δt time ($\Phi_v^+(\Delta t)$) and crosses it ($\neg\mathcal{W}_v^+$) allowing a discrete transition $h' \to h''$ which increases v because the signs of the celerities of v in h' and in h'' are the sames.

(γ) Let us finally prove that the postcondition is satisfied after the elementary path. We previously proved that there exists a unique continuous transition $h \to h'$ and a discrete one $h' \to h''$. Since $h \vDash (D[\eta_v \backslash \eta_v + 1], H'_{i,f})$, we deduce with Lemma 16: $h' \vDash (D[\eta_v \backslash \eta_v + 1], H'_{i,f})$. The discrete transition increases the variable v ($\eta''_v = \eta_v + 1$), we deduce that:

$$h'' \vDash (D[\eta_v \backslash \eta_v + 1][\eta_v \backslash \eta_v - 1], H'_{i,f}[\eta_v \backslash \eta_v - 1]), \quad \text{that is,} \quad h'' \vDash (D, H'_{i,f}[\eta_v \backslash \eta_v - 1])$$

The hybrid condition $H'' \equiv H'_{i,f}[\eta_v \backslash \eta_v - 1]$ is satisfied in h'':

$$H'' \equiv \Big(H_f[\eta_v \backslash \eta_v + 1] \wedge \Phi_v^+(\Delta t) \wedge \mathcal{F}(\Delta t) \wedge \neg\mathcal{W}_v^+ \wedge \mathcal{A}(\Delta t, a) \wedge \mathcal{J}_v\Big)[\eta_v \backslash \eta_v - 1]$$
$$\equiv H_f \wedge \Big(\Phi_v^+(\Delta t) \wedge \mathcal{F}(\Delta t) \wedge \neg\mathcal{W}_v^+ \wedge \mathcal{A}(\Delta t, a) \wedge \mathcal{J}_v\Big)[\eta_v \backslash \eta_v - 1]$$

So, the discrete and hybrid conditions D and H_f are satisfied at h'' and the postcondition is verified.

B.2 Second Case: $a = \mathsf{slide}^+(u)$

(α) Similarly to the first case, we consider the unique hybrid state $h' = (\eta, \pi')$ such that the continuous transition $h \to h'$ exists. The time spent in the current qualitative state is also Δt (sub-property $\Phi_v^+(\Delta t)$). Since $a \neq \top$, the sub-property $\mathcal{A}$ plays a crucial rule:

$$\mathcal{A}(\Delta t, a) \equiv \bigwedge_{\substack{k \in [\![1,n]\!] \\ \omega_k \in R^-(v_k) \\ n_k \in [\![0, b_{v_k}]\!]}} \left(\bigwedge_{l \in [\![1,n]\!]} \left((\eta_{v_l} = n_l) \wedge \Phi_{v_l}^{\omega_l} \right) \Rightarrow a \begin{bmatrix} C_{v_l} \backslash C_{v_l,\omega_l,n_l} \\ \mathsf{slide}(v_l) \backslash \mathcal{S}_{v_l,\omega_l,n_l}(\Delta t) \\ \mathsf{slide}^+(v_l) \backslash \mathcal{S}^+_{v_l,\omega_l,n_l}(\Delta t) \\ \mathsf{slide}^-(v_l) \backslash \mathcal{S}^-_{v_l,\omega_l,n_l}(\Delta t) \end{bmatrix} \right)$$

Amongst all premises of these conjunctions, only one is satisfied because the current qualitative state has a unique qualitative level for each variable v_l ($\eta_{v_l} = n_l$) and a unique set of resources for each v_l ($\Phi_{v_l}^{\omega_l}$). We can then replace $\mathsf{slide}^+(u)$ by the sub-property $\mathcal{S}^+$:

$$\mathcal{S}^+_{u,\omega,n}(\Delta t) \equiv ({\pi_u^i}' = 1) \wedge (\pi_u^i > {\pi_u^i}' - C_{u,\omega,n} \cdot \Delta t)$$

where ω is the resources of u and n its current qualitative level. This formula means that the exit position of the current qualitative state is on the top border (${\pi_u^i}' = 1$). We then deduce:

$$\begin{aligned} C_{u,\omega,n} \cdot \Delta t &> 1 - \pi_u^i \\ \Delta t &> \frac{1 - \pi_u^i}{C_{u,\omega,n}} = \delta_{h'}(u) \qquad \text{because } 1 - \pi_u^i \geq 0,\ \Delta t \geq 0 \text{ and } C_{u,\omega,n} > 0 \end{aligned}$$

According to Lemma 15, we have $\delta_{h'}(v) = \delta_{h'}^{\mathsf{first}} = \Delta t$ and so $\delta_{h'}(v) > \delta_{h'}(u)$. In other words, u reaches its top border before v reaches its one. Thus, the continuous transition $h \to h'$ is such that $(h, h') \vDash (\Delta t, a)$, see Definition 11.

(β and γ) The proof of the discrete transition existence from h' is similar to the first case. This transition leads to h'' which satisfies the postcondition $h'' \vDash (D, H)$ (see the stages β and γ of the first case).

B.3 Third Case: $a = C_u \mathbin{\square} c$ with $c \in \mathbb{R}$

(α) The sub-property $\mathcal{A}(\Delta t, a)$ allows one to replace the celerity C_u in the assertion a by the celerity indexed by the relevant set of resources of the current qualitative state. So, we deduce $C_{u,\rho(\eta,u),\eta_u} \mathbin{\square} c$. From Definition 11, the unique continuous transition $h \to h'$ where $h' = (\eta, \pi')$ is such that $(h, h') \vDash (\Delta t, a)$.

(β and γ) The proof of the discrete transition existence from h' is similar to the first case. This transition leads to h'' which satisfies the postcondition $h'' \vDash (D, H)$ (see the stages β and γ of the first case).

B.4 Fourth Case: $a = a_1 \wedge a_2$

(α) From the previous cases, it is possible to construct two hybrid states h_1 and h_2 such that $(h, h_1) \vDash (\Delta t, a_1)$ and $(h, h_2) \vDash (\Delta t, a_2)$. Because the continuous transition starting at h is unique, $h_1 = h_2$. Thus, $(h, h_1) \vDash (\Delta t, a_1 \wedge a_2)$.

($\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$) The proof of the discrete transition existence from $\boldsymbol{h'}$ is similar to the first case. This transition leads to $\boldsymbol{h''}$ which satisfies the postcondition $\boldsymbol{h''} \vDash \boldsymbol{(D, H)}$ (see the stages $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ of the first case).

This proof is generalisable for all logical connectives and recursively to all formulas.

Hierarchical Cost-Parity Games

Laura Bozzelli[1], Aniello Murano[*2], Giuseppe Perelli[†3], and Loredana Sorrentino[4]

1 **Università degli Studi di Napoli Federico II, Naples, Italy**
2 **Università degli Studi di Napoli Federico II, Naples, Italy**
3 **University of Oxford, Oxford, UK**
4 **Università degli Studi di Napoli Federico II, Naples, Italy**

Abstract

Cost-parity games are a fundamental tool in system design for the analysis of reactive and distributed systems that recently have received a lot of attention from the formal methods research community. They allow to reason about the time delay on the requests granted by systems, with a bounded consumption of resources, in their executions.

In this paper, we contribute to research on Cost-parity games by combining them with *hierarchical systems*, a successful method for the succinct representation of models. We show that determining the winner of a *Hierarchical Cost-parity Game* is PSPACE-COMPLETE, thus matching the complexity of the proper special case of *Hierarchical Parity Games*. This shows that reasoning about temporal delay can be addressed at a free cost in terms of complexity.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Parity Games, Cost-Parity Games, Hierarchical Systems, System Verification

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.6

1 Introduction

In formal system design and verification [11, 12, 20, 26], *Parity Games* represent a fundamental machinery for the automatic synthesis and verification of concurrent and reactive systems [5, 6, 7, 21, 22]. The determinacy and the memorylessness of parity games is crucial in various theoretical areas useful in formal verification, among which we mention automata theory, temporal and modal logics, and monadic second-order logics. For instance, the emptiness problem of alternating tree automata [14] as well as model checking and satisfiability in modal μ-calculus [18] can be reduced to deciding the winner of a parity game. In particular, model checking μ-calculus is equivalent via linear time reduction to this problem [13].

As pointed out in [15, 23, 24], the parity winning condition corresponds to a qualitative *request-response condition* [17]: Player 0 wins a play of infinite duration if all but finitely many odd colors (which we think of as requests) are followed by larger even colors (which we think of as responses). In this setting, there is no bound on the wait time, *i.e.*, the number of steps that elapse between a request and its first response in the play. On the other hand, in many applications, it is important to bound the wait time. In the last decade, many papers have focused on quantitative aspects, in particular boundedness requirements, of

[*] The author is partially supported by the INdAM research project 2017 "Logica e Automi per il Model Checking".
[†] The author acknowledges with gratitude the the financial support of the ERC Advanced Investigator grant 291528 ("RACE") at Oxford.

formal verification [1, 19, 10], including parity games [10, 15, 23, 24]. In [19], the authors introduce Prompt LTL, an extension of standard LTL [25] with the prompt-eventually operator $\mathtt{F}_p$: a finite system satisfies a Prompt LTL formula φ iff there is a bound on the wait time for all the prompt-eventually subformulas of φ in all the computations of the system. The automata-theoretic counterpart of the $\mathtt{F}_p$ operator has been investigated in [1]. Parity games extended with promptness requirements, the so-called finitary parity games, have been studied in [10]. The finitary parity condition [10] extends the parity condition by additionally requiring the existence (along the given play) of a bound k such that almost every odd color is answered within at most k steps. Surprisingly, finitary parity games are solvable in polynomial time, and thus simpler than parity games (according to the state-of-the-art). A meaningful generalization of finitary games is represented by the class of parity games with costs [15] (in the following, referred as cost-parity games). In such games, transitions are labeled by non-negative integers (costs). The cost of traversing a transition can be used to model resource consumption. The goal of Player 0 consists then in ensuring the underlying parity condition by using bounded resources: a play is winning for Player 0 if there is a bound k such that almost every odd color is followed by a larger even color that is reached with cost at most k. On the other hand, Player 1's goal is to exhaust the resources by making the cost unbounded. Note that Player 1's objective is not an ω-regular property, and in general, Player 1 needs infinite memory to win such games. However, cost-parity games enjoy some nice properties: Player 0 has memoryless winning strategies and determining the winner lies in $\mathrm{NP} \cap \mathrm{coNP}$. This upper bound has been recently improved to $\mathrm{UP} \cap \mathrm{coUP}$ in [24], proving thus that the increased expressiveness with respect to parity conditions comes at a free cost in terms of complexity.

In the recent years, many other quantitative extensions of parity games have been introduced. Among them we would like to mention Mean-Payoff Parity Games [9], whose winning condition is a combination of a parity and a mean-payoff objective, and Energy Parity Games [8]. These last ones are played over weighted arenas, and the winning condition extends the parity condition by additionally requiring that the sum of the weights along a play (interpreted as level of energy, or resource usage) remains always positive.

A well-known issue in formal verification is that the translation of a high-level description of a system into a formal model, typically given by a finite-state machine (FSM), often involves an exponential blow-up in the size of the FSM, thus affecting the efficiency of the analysis procedures both in theory and practice. Several sources of this blow-up have been identified in the literature. A well-studied one is the ability of components in the system to work in parallel and communicating with each other, possibly using variables. The impact of the concurrent setting on analysis problems is well-known: it costs an exponential, leading to the so called state-explosion problem. Another source of the blow-up in the translation of systems into FSMs is that in high-level sequential programming, one can specify components only once and then can reuse them in different contexts, leading to modularity and succinct system representation. A smart way to represent such modularity is by means of *hierarchical FSM*, where some of the states of the FSM are boxes (superstates) which correspond to nested FSMs (the reused components). The naive approach to model checking such systems is to 'flatten' them by repeatedly substituting references to sub-structures with copies of them. This results in a flat FSM whose size is exponential in the nesting depth of the hierarchical system. However, differently from the concurrent setting, a wiser approach avoiding flattening, for the case of model checking against temporal logics like LTL, CTL and the more expressive modal μ-calculus, is beneficial in terms of complexity [3, 4, 5, 16]. Parity games have also been investigated under the hierarchical setting. In [5], Aminof et al. prove that deciding the winner in a *Hierarchical Parity Game* (*HPG*) is a PSPACE-complete problem. The technique used in [5] is based on the observation that even though a sub-arena may appear in different contexts, it is possible to extract information about the sub-arena that is independent of the context in which it appears.

In this paper, we further investigate the power of hierarchical representation by introducing and studying *Cost-parity Games over Hierarchical Systems* (*HCPG*). As main result, we establish that the problem of solving *HCPG* is PSPACE-complete, which matches the complexity of the proper special case of hierarchical parity games (*HPG*). The proposed approach for solving the considered problem generalizes in a non-trivial and sophisticated manner the one exploited in [5] for solving *HPG*, and is based on the notion of summary function for a memoryless strategy σ of Player 0 in a given sub-arena. Such a function records in a finite and efficient way the overall behavior of all the finite plays of σ leading to exit states of the sub-arena with respect to requests and responses, by finitely abstracting the set of associated costs and delays. The algorithm for solving *HCPG* then solves a sequence of flat cost-parity games obtained by replacing sub-arenas by simple gadgets (depending only on the set of colors and exit states of the sub-arena) that implement the summary functions.

The sequel of the paper is structured as follows. In Section 2, we first recall the framework of cost-parity games. Then, we introduce hierarchical cost-parity games and describe our solution approach in Section 3. Finally, we give few conclusions and future work directions in Section 4. Due to space constraints, some proofs are omitted.

2 Preliminaries

Let $\mathbb{N}$ be the set of natural numbers. For all $i, j \in \mathbb{N}$, with $i \leq j$, $[i, j]$ denotes the set of natural numbers h such that $i \leq h \leq j$. We fix a non-empty finite set C of natural numbers of the form $[0, j]$ for some $j \in \mathbb{N}$, which represents the set of colors for the given cost-parity winning condition. We denote by C_e and C_o the sets of even and odd colors in C, respectively. We assume that the maximal color j in C, denoted by C_o^{max}, is odd.

For an alphabet Σ, and a non-empty finite or infinite word w over Σ, we denote by $|w|$ the length of w (we set $|w| = \infty$ if w is infinite). Moreover, for all $i, j \geq 1$, with $i \leq j$, $w(i)$ is the i-th letter of w, while $w[i, j]$ denotes the finite subword of w given by $w(i) \cdots w(j)$, and w^i the prefix of w from position i, i.e., the word $w(i)w(i+1)\ldots$.

2.1 Cost-Parity Games

We recall the framework of Cost-parity games [15] which are two-player turn-based games played on finite graphs equipped with a Cost-parity winning condition. In such a setting, Player 0 wins a play of infinite duration if there is a bound $\ell \in \mathbb{N}$ such that almost all odd colors (which we think of as requests) are followed by larger even colors (which we think of as responses) that are reached with cost at most ℓ.

A state-transition graph or FSM is a tuple $\langle \mathrm{S}, \mathrm{R}, in\rangle$ consisting of a finite set S of states, a transition relation $\mathrm{R} \subseteq \mathrm{S} \times \mathrm{S}$, and an initial state $in \in \mathrm{S}$. For a state $s \in \mathrm{S}$, we write $\mathrm{R}(s) = \{s' \in \mathrm{S} \mid (s, s') \in \mathrm{R}\}$ for the set of successors of s. A path in the FSM is a non-empty finite or infinite word π over S such that $\pi(i+1) \in \mathrm{R}(\pi(i))$ for all $i \in [1, |\pi| - 1]$.

An *arena* is a tuple $\mathcal{A} = \langle \mathrm{S}, \mathrm{S}_0, \mathrm{S}_1, \mathrm{R}, in\rangle$ consisting of an FSM $\langle \mathrm{S}, \mathrm{R}, in\rangle$ and a partition $\{\mathrm{S}_0, \mathrm{S}_1\}$ of S into the states of Player 0 (drawn as circles) and the states of Player 1 (drawn as rectangles). A play of a game over $\mathcal{A}$ proceeds by moving a token on the states of $\mathcal{A}$, starting at some state. If the token is placed on a state $s \in \mathrm{S}_0$ (resp., $s \in \mathrm{S}_1$), then the play ends if s has no successors (we call such a state a *terminal state*); otherwise, Player 0 (resp., Player 1) chooses a successor s' of s and moves the token to s'. Formally, a *play* of $\mathcal{A}$ is a *maximal* path of $\mathcal{A}$, i.e., a path π in the underlying FSM such that either π is infinite, or π is finite and ends at a terminal state.

Let $p \in \{0, 1\}$ and S_p^N be the set of non-terminal states of Player p. A *strategy* for Player p is a mapping $\sigma : \mathrm{S}^* \cdot \mathrm{S}_p^N \mapsto \mathrm{S}$ assigning to each non-empty sequence of states $w \cdot s \in \mathrm{S}^* \cdot \mathrm{S}_p^N$ leading to a non-terminal state s of Player p, a successor of s. A play π is consistent with the strategy σ if for all $k \in [1, |\pi| - 1]$ such that $\pi(k) \in \mathrm{S}_p^N$, it holds that $\pi(k + 1) = \sigma(\pi[1, k])$. The strategy σ is *memoryless* if its output does not depend on the whole prefix of the play, but only on the last position, i.e, if for all $w \cdot s \in \mathrm{S}^* \cdot \mathrm{S}_p^N$, $\sigma(w \cdot s) = \sigma(s)$. We can thus represent a memoryless strategy as a mapping $\sigma : \mathrm{S}_p^N \to \mathrm{S}$.

A (zero-sum) game is a pair $\langle \mathcal{A}, \mathrm{Win} \rangle$ consisting of an arena $\mathcal{A} = \langle \mathrm{S}, \mathrm{S}_0, \mathrm{S}_1, \mathrm{R}, in \rangle$ and a subset Win of infinite plays which are winning for Player 0. An infinite play π is *winning for Player* 1 if it is not winning for Player 0. A *finite* play π is winning for Player p if π ends at a state of the opponent Player $1 - p$. A strategy σ for Player p is *winning from a state* s if all the plays π starting from s which are consistent with the strategy σ are winning for Player p. In such a case, we say that state s is winning for Player p. A game is determined if for each state s, s is winning for one of the players. Note that since for all strategies σ^0 and σ^1 of Player 0 and Player 1, respectively, there is a unique play starting from s which is consistent with both σ^0 and σ^1, in (zero-sum) games, a state s cannot be winning for both the players. Solving a game consists in checking whether the initial state is winning for Player 0.

Cost-parity winning conditions

We, now, recall the class of Cost-parity winning conditions. A *Cost-parity arena* $\mathcal{G} = \langle \mathcal{A}, \mathrm{Cost}, \Omega \rangle$ over the set C of colors consists of an arena $\mathcal{A} = \langle \mathrm{S}, \mathrm{S}_0, \mathrm{S}_1, \mathrm{R}, in \rangle$, a transition-labeling $\mathrm{Cost} : \mathrm{R} \mapsto \{0, 1\}$ (cost function), and a coloring mapping $\Omega : \mathrm{S} \mapsto C$ assigning to each state a color in C. Note that according to [15], the definition of transition-labelling only allows cost 0 or 1 on a transition. Having arbitrary costs in $\mathbb{N}$ would not change our results, as we are interested in boundedness questions only. We extend the transition-labeling to a cost function Cost over paths π obtained by counting the number of increment transitions (i.e., 1-labeled transitions) traversed along the path, i.e., $\mathrm{Cost}(\pi) = \sum_{i=2}^{i=|\pi|} \mathrm{Cost}(\pi(i - 1), \pi(i))$. Note that $\mathrm{Cost}(\pi) \in \mathbb{N} \cup \{\infty\}$.

The pair (Cost, Ω) induces a winning condition for Player 0, where the occurrence of an odd color along a play π is interpreted as a *request*, for which there has to be a *response* later on the play by a higher even color. Formally, let π be a finite or infinite path of $\mathcal{A}$. A *request in* π is a position k along π such that $\pi(k)$ has *odd* color. For an odd color c, a c-request in π is a request k in π such that $\Omega(\pi(k)) = c$. Moreover, we define $\mathrm{Ans}(c) = \{c' \in C_e \mid c' \geq c\}$, i.e., the set of even colors that answers a request of color c. For a request k in π, let r_k be the smallest position $k' \geq k$ that answers to request k, i.e., such that $\Omega(\pi(k')) \in \mathrm{Ans}(\Omega(\pi(k)))$, if such positions k' exist, and let $r_k = |\pi|$ otherwise. In the first (resp., second) case, we say that the request k is *answered* (resp., *unanswered*) in π. The *delay* of the request k in π, denoted by $\mathrm{dl}(\pi, k)$, then is defined as the cost of the infix of π from the request k to position r_k, i.e., $\mathrm{Cost}(\pi[k, r_k])$ if $r_k \neq \infty$, and $\mathrm{Cost}(\pi^k)$ otherwise. The *cost-parity* winning condition induced by (Cost, Ω), written $\mathrm{CostParity}(\mathrm{Cost}, \Omega)$, is then the set of infinite plays π such that there is $n \geq 1$ and a bound $\ell \in \mathbb{N}$ so that for all requests k in π with $k \geq n$, $\mathrm{dl}(\pi, k) \leq \ell$ and the request k is answered in π. Thus, an infinite play $\pi \in \mathrm{CostParity}(\mathrm{Cost}, \Omega)$ iff there is bound ℓ such that all but finitely many requests are answered with cost less than ℓ. Note that $\mathrm{CostParity}(\mathrm{Cost}, \Omega)$ is *prefix-independent*, i.e., for all infinite plays π and $k \geq 1$, $\pi \in \mathrm{CostParity}(\mathrm{Cost}, \Omega)$ iff $\pi^k \in \mathrm{CostParity}(\mathrm{Cost}, \Omega)$. We recall the following known result.

▶ **Theorem 1** ([15]). *Cost-parity games are determined and Player* 0 *has memoryless winning strategies from the winning Player* 0 *states. Moreover, solving a cost-parity game* $\mathcal{G} = \langle \mathcal{A}, Cost, \Omega \rangle$ *with* k *colors can be done in time* $|\mathcal{A}|^{O(k \cdot \log k)}$ *and in polynomial space.*

For technical convenience, we also consider a generalization of cost-parity arenas, called *partial cost-parity arenas*, where one considers as additional input a subset *Exit* of the set of terminal states, called *exit states*. Finite plays ending at states in *Exit* are assumed to be *non-winning* for either player and have an undefined value. In this setting, a *non-loosing strategy for Player p from state s* is a strategy σ for Player p such that each play starting from s which is consistent with σ and does not lead to an exit state is winning for Player p. A *non-loosing strategy* is a non-loosing strategy for Player 0 from the initial state *in*. For a strategy σ for Player 0, an *exit play* of σ is a finite play starting from *in* and ending at an exit state which is consistent with σ. For $s \in Exit$, an s-exit play of σ is an exit play of σ leading to s. Two partial cost-parity arenas $\mathcal{G} = \langle \mathcal{A}, \mathrm{Cost}, \Omega, Exit \rangle$ and $\mathcal{G}' = \langle \mathcal{A}', \mathrm{Cost}', \Omega', Exit' \rangle$ *have the same interface* if $Exit = Exit'$, $\mathcal{G}$ and $\mathcal{G}'$ have the same initial state *in*, and for each $s \in \{in\} \cup Exit$, the colors and the players of state s in $\mathcal{G}$ and $\mathcal{G}'$ coincide.

2.2 Hierarchical Cost-Parity Games

A *Hierarchical Cost-Parity Game* is a cost-parity game played over a (flat) arena induced by a *hierarchical arena*. The latter is a standard hierarchical FSM [4] in which the set of nodes of each of the underlying FSMs is partitioned into nodes belonging to Player 0 and nodes belonging to Player 1. We refer to the underlying FSMs as modular sub-arenas. Formally, a hierarchical arena is a tuple $\mathcal{V} = \langle \mathcal{V}_1, \dots, \mathcal{V}_n \rangle$ of modular sub-arenas, where each $\mathcal{V}_i$ is in turn a tuple of the form $\langle \mathrm{N}_i, \mathrm{N}_i^0, \mathrm{N}_i^1, \mathrm{B}_i, in_i, Exit_i, \mathrm{Y}_i, \mathrm{E}_i \rangle$ consisting of the following components:

- A finite set N_i of nodes which is partitioned into a set N_i^0 of nodes of Player 0 and a set N_i^1 of nodes of Player 1, and a finite set B_i of *boxes*. We assume that $\mathrm{N}_1, \dots, \mathrm{N}_n, \mathrm{B}_1, \dots, \mathrm{B}_n$ are pairwise disjoint.
- An initial node or entry $in_i \in \mathrm{N}_i$, [1] and a subset $Exit_i$ of N_i called *exit-nodes*. We assume that $Exit_1 = \emptyset$, i.e., the top-level sub-arena $\mathcal{V}_1$ has no exits.
- An indexing function $\mathrm{Y}_i : \mathrm{B}_i \to \{i+1, \dots, n\}$ that maps each box b of $\mathcal{V}_i$ to an index $\mathrm{Y}_i(b) > i$. The box b represents a reference to the definition of the sub-arena $\mathcal{V}_{\mathrm{Y}_i(b)}$.
- An edge relation E_i. Each edge in E_i is a pair (u, v) such that: *(i)* the source u is either a node of $\mathcal{V}_i$ or a pair (b, e), where b is a box of $\mathcal{V}_i$ and e is an exit-node of the sub-arena that b refers to, and *(ii)* the target v is either a node or a box of $\mathcal{V}_i$.

Define $\mathrm{N} = \bigcup_{i=1}^{n} \mathrm{N}_i$ (the set of $\mathcal{V}$-nodes), $\mathrm{E} = \bigcup_{i=1}^{n} \mathrm{E}_i$ (the set of $\mathcal{V}$-edges), and $Exit = \bigcup_{i=1}^{n} Exit_i$ (the set of $\mathcal{V}$-exit-nodes). In a modular sub-arena, the edges connect nodes and boxes with one another. Edges entering a box implicitly lead to the unique entry-node of the sub-arena that the box refers to. On the other hand, an edge exiting a box needs to explicitly specify the identity of the exit-node among the possible exit-nodes of the sub-arena associated with that box. The *size* $|\mathcal{V}_i|$ of a modular sub-arena $\mathcal{V}_i$ is $|\mathrm{N}_i| + |B_i| + |\mathrm{E}_i|$. The size $|\mathcal{V}|$ of $\mathcal{V}$ is $\sum_{i=1}^{i=n} |\mathcal{V}_i|$. The *nesting depth* of $\mathcal{V}$ is the length of the longest chain $i_1, i_2, \dots, i_j$ of indices in $[1, n]$ such that a box of $\mathcal{V}_{i_l}$ is mapped to i_{l+1} for all $l \in [1, j-1]$. Note that the fact that boxes of a sub-arena can only refer to sub-arenas with a greater index implies that the nesting depth of $\mathcal{V}$ is finite. Such a restriction does not exist in the *recursive* setting [2].

[1] We assume a single entry for each sub-arena. Multiple entries can be handled by duplicating sub-arenas.

A *Hierarchical Cost-Parity Arena* (*HCPA*, for short) over C is a tuple $\mathcal{H} = \langle \mathcal{V}, \mathrm{Cost}, \Omega \rangle$ consisting of a hierarchical arena $\mathcal{V} = \langle \mathcal{V}_1, \ldots, \mathcal{V}_n \rangle$ equipped with a cost function $\mathrm{Cost} : \mathrm{E} \mapsto \{0, 1\}$ for the set of $\mathcal{V}$-edges, and a coloring mapping $\Omega : \mathrm{N} \mapsto C$ for the set of $\mathcal{V}$-nodes. We can associate to $\mathcal{H}$ an ordinary cost-parity arena (called its *flat expansion*) by recursively substituting each box by a copy of the modular sub-arena it refers to. Since different boxes can refer to the same sub-arena, nodes may appear in different contexts. In general, a state of the flat expansion is a vector whose last component is a node, and the remaining components are boxes that specify the context. Formally, for each modular sub-arena $\mathcal{V}_i$, we inductively define its flat expansion as the *partial* Cost-parity arena $\mathcal{H}_i^F = \langle \mathcal{A}_i, \mathrm{Cost}_i, \Omega_i, \mathit{Exit}_i \rangle$, with $\mathcal{A}_i = \langle \mathrm{S}_i, \mathrm{S}_i^0, \mathrm{S}_i^1, \mathrm{R}_i, \mathit{in}_i \rangle$, defined as follows:

- The set of states S_i is inductively defined as follows: *(i)* if u is a node in $\mathcal{V}_i$, then $u \in \mathrm{S}_i$, and *(ii)* if b is a box of $\mathcal{V}_i$ and $s \in \mathrm{S}_{\mathrm{Y}_i(b)}$, then $(b, s) \in \mathrm{S}_i$.
- S_i^0 (resp., S_i^1) is the set of states in S_i whose node-component belongs to Player 0 (resp., Player 1), and the coloring function Ω_i assigns to each state s of $\mathcal{A}_i$, the color $\Omega(u)$ of the node-component u of s.
- The transition relation R_i and the cost function Cost_i are inductively defined as follows.
  - If $(u, v) \in \mathrm{E}_i$ and the target v is a node, then $(u, v) \in \mathrm{R}_i$ and $\mathrm{Cost}_i(u, v) = \mathrm{Cost}(u, v)$. If $(u, b) \in \mathrm{E}_i$ and the target b is a box, then $(u, (b, \mathit{in}_{\mathrm{Y}_i(b)})) \in \mathrm{R}_i$ and $\mathrm{Cost}_i(u, (b, \mathit{in}_{\mathrm{Y}_i(b)})) = \mathrm{Cost}(u, b)$.
  - If b is a box of $\mathcal{V}_i$ and $(s, s') \in \mathrm{R}_{\mathrm{Y}_i(b)}$, then $((b, s), (b, s')) \in \mathrm{R}_i$ and $\mathrm{Cost}_i((b, s), (b, s')) = \mathrm{Cost}_{\mathrm{Y}_i(b)}(s, s')$.

Note that since $\mathit{Exit}_1 = \emptyset$, $\mathcal{H}_1^F$ is an ordinary Cost-parity arena (i.e., it is not partial), called the flat expansion of $\mathcal{H}$. Moreover, observe that each state of $\mathcal{H}_1^F$ is a vector of length at most the nesting depth d of $\mathcal{V}$, and that the number of states in $\mathcal{H}_1^F$ can be exponential in d. Solving the game on the *HCPA* $\mathcal{H}$ consists in checking whether the initial state in_1 of the cost-parity arena $\mathcal{H}_1^F$ is winning for Player 0.

3 Solving Hierarchical Cost-Parity Games

The naive approach for solving games on *HCPA* $\mathcal{H}$ consisting in applying Theorem 1 on the flat expansion of $\mathcal{H}$ would lead to an exponential space procedure. In this section, we show that solving hierarchical cost-parity games is PSPACE-complete. Our approach is based on the notion of *summary function* for a strategy σ of Player 0 in a partial cost-parity arena, which records in a finite and efficient way the overall behavior of all the exit plays of σ with respect to requests and responses. The proposed algorithm for solving the game on the given *HCPA* $\mathcal{H}$ then solves a sequence of partial cost-parity games, obtained by replacing each box b referring to a sub-arena $\mathcal{V}_i$ with simple partial-cost parity arenas (*summary-gadget arenas*) having the same interface as the flat expansion $\mathcal{H}_i^F$ of $\mathcal{V}_i$ and depending only on the set of colors and exit states. These gadgets represent the behavior of Player 0 as a choice among the possible summary functions associated with the non-loosing memoryless strategies in $\mathcal{H}_i^F$, and also take into account the possibility that the game will stay forever in the sub-arena $\mathcal{V}_i$ for the given context b. The rest of this section is organized as follows: in Subsection 3.1, we introduce the notions of summary and summary-gadget arena, and in Subsection 3.2 we show how to check that a summary is associated with non-loosing memoryless strategies. Finally, in Subsection 3.3, we illustrate the proposed algorithm for solving *HCPA* games.

3.1 Summaries in partial cost-parity games

In this section, for a given partial cost-parity arena $\mathcal{G}$, we show how to define a finite abstraction of the set of non-loosing strategies (of Player 0). Such an abstraction is based on the notion of summary for a strategy σ of Player 0, which is a mapping assigning to each exit state s a value ranging over a finite set (depending only on the set of colors). Such a value summarizes the overall behavior of all the s-exit plays of σ with respect to requests and responses by finitely abstracting the set of associated costs and delays. Then, we associate to each summary $\mathcal{S}$ of $\mathcal{G}$ a simple partial-cost parity arena $\mathit{Gad}(\mathcal{G}, \mathcal{S})$ – exposing the same interface as $\mathcal{G}$ (the initial state and the set of exit states) – which depends only on the set of colors and exit states, and is independent of the set of 'internal' states in $\mathcal{G}$. The set of summary-gadget arenas $\mathit{Gad}(\mathcal{G}, \mathcal{S})$ such that $\mathcal{S}$ is achieved by some non-loosing memoryless strategy is 'context-equivalent' to $\mathcal{G}$, i.e., for each memoryless strategy σ achieving some summary $\mathcal{S}$, $\mathcal{G}$ can be equivalently replaced with $\mathit{Gad}(\mathcal{G}, \mathcal{S})$ in any hierarchical context where $\mathcal{G}$ is exploited as a sub-arena and Player 0 chooses strategy σ when entering $\mathcal{G}$. [2]

Fix a partial cost-parity arena $\mathcal{G} = \langle \mathcal{A}, \mathrm{Cost}, \Omega, \mathit{Exit} \rangle$ over the set C of colors, where $\mathcal{A} = \langle \mathrm{S}, \mathrm{S}_0, \mathrm{S}_1, \mathrm{R}, \mathit{in} \rangle$ and Exit is the designated set of exit states.

In order to describe the relative merit of colors, we define an ordering $\succeq_0$ over the given set C of colors by letting $c \succeq_0 c'$ when c is better for Player 0 than c'. Formally, $c \succeq_0 c'$ if: either *(i)* c and c' are even and $c \geq c'$, or *(ii)* c and c' are odd and $c' \geq c$, or *(iii)* c' is odd and c is even. Moreover, in order to summarize in a finite way cost measures, we exploit three special symbols, namely, bnd_0, bnd_1, and unb to denote bounded behavior with zero-cost, bounded behavior with non-zero cost, and unbounded behavior (cost ∞), respectively. Additionally, we denote by $\succeq_b$ the ordering on $\{\mathit{bnd}_0, \mathit{bnd}_1, \mathit{unb}\}$ defined as: $\mathit{bnd}_0 \succeq_b \mathit{bnd}_1$ and $\mathit{bnd}_1 \succeq_b \mathit{unb}$. Intuitively, $\mathit{bnd}_0 \succeq_b \mathit{bnd}_1$ and $\mathit{bnd}_1 \succeq_b \mathit{unb}$ express that bounded zero-cost is better for Player 0 then non-zero bounded cost, the latter being in turn better than unbounded cost. Define $\tilde{C} = C \setminus \{C_o^{max}\}$ and $\tilde{C}_o = (C_o \setminus \{C_o^{max}\}) \cup \{0\}$.

In order to formalize the notion of summary for a strategy σ of Player 0, we consider various cost measures with respect to the requests and the responses along the exit plays of σ. For this, we extend the cost function Cost to (possibly infinite) sets Π of *finite* paths. Formally, $\mathrm{Cost}(\Pi)$ is the least upper bound over the costs of the paths in Π, i.e., $\mathrm{Cost}(\Pi) = \sup\{\mathrm{Cost}(\nu) \mid \nu \in \Pi\}$ where $\sup \emptyset = 0$. Note that $\mathrm{Cost}(\Pi) \in \mathbb{N} \cup \{\infty\}$. For a *finite* path ν of $\mathcal{G}$ and an even color $c_e \in C_e$, a *c_e-response in* ν is a position k of ν such that $\nu(k)$ ha color c_e. For such a response k, the cost of response k in ν is the cost of the prefix of ν leading to position k, i.e., $\mathrm{Cost}(\nu[1, k])$. The *c_e-response cost* of ν, denoted by $\mathrm{ResCost}(\nu, c_e)$, is the cost $\mathrm{Cost}(\nu[1, k])$ of the prefix of ν up to the minimal c'_e-response k in ν for some even color $c'_e \geq c_e$ if such c'_e-responses exist, and it is 0 otherwise. The *maximal even color* of the path ν is the maximal even color visited by ν if ν visits some even color, and it is 0 otherwise (note that a 0-response cannot answer to any request). We exploit the following cost measures for the (possibly infinite) set of exit plays of a given strategy σ of Player 0 leading to a designated exit state.

▶ **Definition 2** (Cost measures of Player 0 strategies). Let $s \in \mathit{Exit}$, σ a strategy of Player 0, Π_s the (possibly empty) set of exit plays of σ leading to s, and $c_e \in C_e$ an even color.

- *Cost of σ w.r.t. s*, denoted $\mathrm{Cost}(\sigma, s)$: it is $\mathrm{Cost}(\Pi_s)$.
- *Even c_e-cost of σ w.r.t. s*, denoted $\mathrm{Cost}_e(\sigma, s, c_e)$: it is $\mathrm{Cost}(\Pi_{c_e})$, where Π_{c_e} is the (possibly empty) set of exit plays in Π_s whose maximal even color is at most c_e.

[2] The formal proof of such a context-equivalence is postponed to Section 3.3.

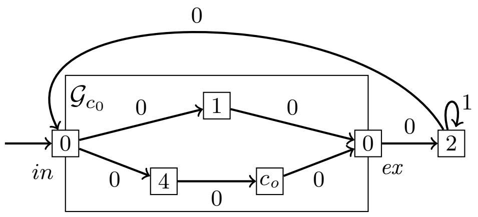
- *c_e-response cost of σ w.r.t. s*, denoted $\text{ResCost}(\sigma, s, c_e)$: it is the least upper bound over the c_e-response costs of the exit plays in Π_s, i.e., $\sup\{\text{ResCost}(\nu, c_e) \mid \nu \in \Pi_s\}$.
- *Request-cost of σ w.r.t. s*, denoted $\text{ReqCost}(\sigma, s)$: it is the least upper bound over the delays associated with the requests along the exit plays in Π_s, i.e., $\sup\{\text{dl}(\nu, k) \mid \nu \in \Pi_s$ and k is a request in $\nu\}$.

Note that $\text{Cost}_e(\sigma, s, -)$ is monotonic in the third argument, i.e., $\text{Cost}_e(\sigma, s, c'_e) \geq \text{Cost}_e(\sigma, s, c_e)$ for all $c_e, c'_e \in C_e$ such that $c'_e \geq c_e$. We, now, introduce the notion of summary for a strategy σ of Player 0 which records for each exit state s, a value, called *exit value*, ranging over a finite set depending only on the set of colors. This value summarizes the overall behavior of the exit plays of σ leading to s. We distinguish three situations (recall that $C_o^{max} = \max(C)$ and C_o^{max} is odd):
- The best scenario for Player 0 is when there is no exit play of σ leading to s. We represent this situation by exploiting the special symbol $\vdash$.
- The worst scenario is when the request-cost of σ w.r.t. s is infinite, or there is an s-exit play of σ having a C_o^{max}-request. We use the color C_o^{max} to describe this scenario.
- If none of the two previous conditions is fulfilled, then the exit value is a sextuple of elements: *(i)* the first element summarizes the cost of σ w.r.t. s, *(ii)* the second element keeps track of the minimal color w.r.t. $\preceq_0$ over the maximal colors along the s-exit plays of σ, *(iii)* the third element represents the maximal odd color associated with an unanswered request, and *(iv)* the last three elements in the tuple summarize the overall response behavior of the s-exit plays of σ.

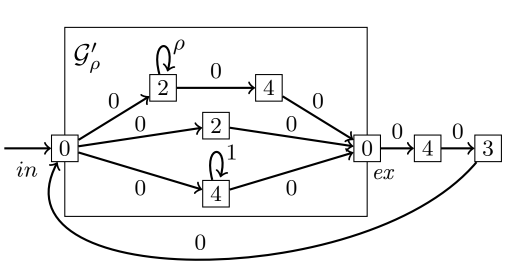
The formal definition of exit values for a strategy of Player 0 follows.

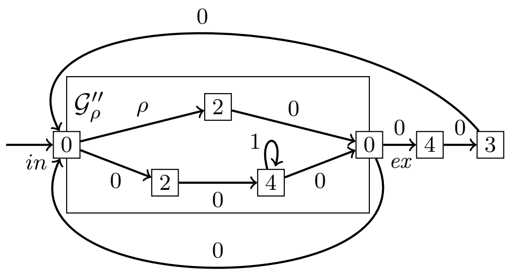
▶ **Definition 3** (Exit values of Player 0 strategies). Let $s \in \mathit{Exit}$, σ a strategy of Player 0, and Π_s the set of exit plays of σ leading to s. The *exit value $\mathit{value}(\sigma, s)$ of strategy σ w.r.t. s* is defined as follows. If $\Pi_s = \emptyset$, then $\mathit{value}(\sigma, s) = \vdash$. If instead either $\text{ReqCost}(\sigma, s) = \infty$ or there is $\nu \in \Pi_s$ having a C_o^{max}-request, then $\mathit{value}(\sigma, s) = C_o^{max}$. Otherwise, $\mathit{value}(\sigma, s) = (\mathit{value}_{\text{Cost}}(\sigma, s), \mathit{value}_{pr}(\sigma, s), \mathit{value}_o(\sigma, s), \mathit{value}_e^L(\sigma, s), \mathit{value}_e^M(\sigma, s), \mathit{value}_e^R(\sigma, s)) \in \{\mathit{bnd}_0, \mathit{bnd}_1, \mathit{unb}\} \times \tilde{C} \times \tilde{C_o} \times C_e \times (C_e \cup \{\bot\}) \times C_e$, and the following holds:
- *Cost value $\mathit{value}_{Cost}(\sigma, s)$*: *(i)* $\mathit{value}_{\text{Cost}}(\sigma, s) = \mathit{unb}$ if $\text{Cost}(\Pi_s) = \infty$, *(ii)* $\mathit{value}_{\text{Cost}}(\sigma, s) = 0$ if $\text{Cost}(\Pi_s) = 0$, and *(iii)* $\mathit{value}_{\text{Cost}}(\sigma, s) = \mathit{bnd}_1$ otherwise.
- *Parity value $\mathit{value}_{pr}(\sigma, s)$*: it is $\min_{\preceq_0}\{c \in C \mid c$ is the maximal color of some $\nu \in \Pi_s\}$.
- *Odd value $\mathit{value}_o(\sigma, s)$*: it is the *greatest* odd color $c_o \in C_o$ such that for some $\nu \in \Pi_s$, ν has an unanswered c_o-request if such an odd color c_o exists; otherwise, it is 0.
- *Even-left value $\mathit{value}_e^L(\sigma, s)$*: it is the *greatest* even color $c_e \in C_e$ such that $\text{ResCost}(\sigma, s, c_e) \neq \infty$ and for each $\nu \in \Pi_s$, the maximal even color in ν is at least c_e, if such an even color c_e exists; otherwise, it is 0.
- *Even-middle value $\mathit{value}_e^M(\sigma, s)$*: it is the *smallest* even color c_e such that $\text{Cost}_e(\sigma, s, c_e) \in \mathbb{N} \setminus \{0\}$ if such a color c_e exists, and $\mathit{value}_e^M(\sigma, s) = \bot$ otherwise ($\bot$ is for 'undefined').
- *Even-right value $\mathit{value}_e^R(\sigma, s)$*: it is the *greatest* even color $c_e \in C_e$ such that $\text{ResCost}(\sigma, s, c_e) \neq \infty$ and for each $c'_e \in C_e$ with $c'_e < c_e$, $\text{Cost}_e(\sigma, s, c'_e) \neq \infty$, if such an even color c_e exists; otherwise, it is 0.

Note that for parity winning conditions, the parity value $\mathit{value}_{pr}(\sigma, s)$ suffices for summarizing the s-exit behavior of strategy σ [5]. For cost-parity winning conditions, we also need to keep track of the maximal odd color $\mathit{value}_o(\sigma, s)$ associated with unanswered requests. Note that $\mathit{value}_o(\sigma, s) \preceq_0 \mathit{value}_{pr}(\sigma, s)$, and $\mathit{value}_o(\sigma, s) \prec_0 \mathit{value}_{pr}(\sigma, s)$ whenever the maximal unanswered request is associated with s-exit plays whose maximal color is even. As an example, let un consider the sub-arena $\mathcal{G}_{c_o}$ – parametric in the color c_o – in the figure below:



Note that all the states are controlled by Player 1. The instances $\mathcal{G}_1$ and $\mathcal{G}_3$ of $\mathcal{G}_{c_o}$ have parity value 1, and odd value 1 and 3, respectively. While by using $\mathcal{G}_1$, all the plays starting from state *in* are winning for Player 0, the same does not hold by using $\mathcal{G}_3$ since in this case, there are plays where the request 3 is answered in an unbounded way.





For what concerns the even values, the even-left value $value^L_e(\sigma, s)$ represents, intuitively, the maximal even color that the s-exit plays of σ offer for answering – in a bounded way – to previous requests in an arbitrary context. The even-right value $value^R_e(\sigma, s)$, where $value^R_e(\sigma, s) \geq value^L_e(\sigma, s)$, instead represents the maximal even color which may answer to a request preceding an s-exit play ν of σ in a bounded way: if the maximal even color in ν is smaller than $value^R_e(\sigma, s)$, then the overall cost of all s-exit plays of σ whose maximal even color is smaller than $value^R_e(\sigma, s)$ is finite. As an example, let us consider the sub-arena $\mathcal{G}'_\rho$ – parametric in the cost ρ of the self-loop on the state with color 2 – in the left part of the figure above, where all the states are controlled by Player 1. The instances $\mathcal{G}'_0$ and $\mathcal{G}'_1$ of $\mathcal{G}'_\rho$ have even-left value 2, and even-right value 4 and 2, respectively. While for $\mathcal{G}'_0$, all the plays starting from state *in* are winning for Player 0, the same does not hold for $\mathcal{G}'_1$, since in this case, there are plays where the external request 3 is answered in an unbounded way. Finally, in order to illustrate the importance of the even-middle value, let us consider the sub-arena $\mathcal{G}''_\rho$ in the right part of the figure above, where again all the states are controlled by Player 1. The instances $\mathcal{G}''_0$ and $\mathcal{G}''_1$ of $\mathcal{G}''_\rho$ have even-left value 2, right-even value 4, and even-middle value $\perp$ and 2, respectively. While for $\mathcal{G}''_0$, all the plays starting from *in* are winning for Player 0, for $\mathcal{G}''_1$, there are plays where the external request 3 is answered in an unbounded way. We make the following observations which easily follow from Definition 3.

▶ **Proposition 4.** *Let σ be a strategy of Player 0 in $\mathcal{G}$ and $s \in Exit$ such that $value(\sigma, s) = (f, c_{pr}, c_o, c^L_e, c^M_e, c^R_e)$. Then:*

- $c_o \preceq_o c_{pr}$*,* $c^L_e \leq c_{pr}$*,* $c^L_e \leq c^R_e$*, and* $c^M_e \in [c^L_e, c^R_e]$ *if* $c^M_e \neq \perp$*.*
- $c^M_e = \perp$ *if* $f = bnd_0$*, and* $c^M_e \neq \perp$ *if* $f = bnd_1$*.*
- $c^R_e = \max(C_e)$ *if* $f \neq unb$*, and* $c^R_e < \max(C_e)$ *if* $f = unb$ *and* $c^M_e = c^R_e$*.*
- $c^L_e = c_{pr}$ *if* $c_{pr} \in C_e$ *and either* $f \neq unb$*, or* $c^L_e < c^R_e$*, or* $c^M_e = c^R_e$*.*

▶ **Definition 5** (Summaries of Player 0 strategies)**.** The set $\mathcal{E}_C$ of *exit values* for the set C of colors is the finite set $\{\vdash, C^{max}_o\} \cup \mathcal{E}'_C$, where $\mathcal{E}'_C$ is the set of tuples $(f, c_{pr}, c_o, c^L_e, c^M_e, c^R_e) \in \{bnd_0, bnd_1, unb\} \times \tilde{C} \times \tilde{C}_o \times C_e \times (C_e \cup \{\perp\}) \times C_e$ satisfying Conditions (1)–(4) in Proposition 4.

A *summary* of $\mathcal{G}$ is a mapping $\mathcal{S} : Exit \mapsto \mathcal{E}_C$ such that for all $s \in Exit$ with $\mathcal{S}(s) = (f, c_{pr}, c_o, c^L_e, c^M_e, c^R_e)$, it holds that $c^L_e \succeq_0 \Omega(in)$, $c_o \preceq_0 \Omega(s)$, and $\Omega(in) \leq c^R_e$. The summary

$\mathcal{S}(\sigma)$ of a strategy σ of Player 0 in $\mathcal{G}$ is the summary of $\mathcal{G}$ associating to each $s \in \mathit{Exit}$, the exit value $\mathit{value}(\sigma, s)$.

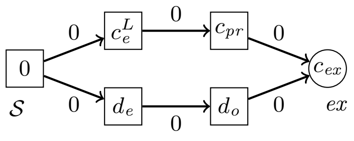
For each summary $\mathcal{S}$, we now define a partial-cost parity game $\mathit{Gad}(\mathcal{G}, \mathcal{S})$, exposing the same interface as $\mathcal{G}$ and independent of the set of 'internal' states in $\mathcal{G}$, such that there is a unique strategy $\sigma_{\mathcal{S}}$ of Player 0 in $\mathit{Gad}(\mathcal{G}, \mathcal{S})$. Moreover, $\sigma_{\mathcal{S}}$ is non-loosing and the exit values of $\sigma_{\mathcal{S}}$ correspond to the exit values of any strategy of Player 0 in $\mathcal{G}$ having $\mathcal{S}$ as summary.

▶ **Definition 6** (Summary-Gadget Arena). Let $\mathcal{S}$ be a summary of $\mathcal{G}$. Given $ex \in \mathit{Exit}$, we first define the sub-gadget $\mathit{Gad}(\mathcal{G}, \mathcal{S}, ex)$ of $\mathcal{G}$ for summary $\mathcal{S}$ and ex, which is the partial cost-parity game with set of states $\mathrm{S}_{ex} \cup \{\mathcal{S}, ex\}$ and set of edges R_{ex}, where:

- All the states in $\mathrm{S}_{ex} \cup \{\mathcal{S}\}$ are controlled by Player 1, $\mathcal{S}$ has color 0 and is the initial state, ex is the unique exit state, and the color and the player of state ex is as in $\mathcal{G}$.

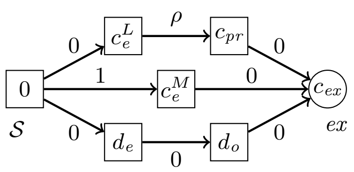
Moreover, if $\mathcal{S}(ex) = \dashv$, then $\mathrm{S}_{ex} = \emptyset$, and $\mathrm{R}_{ex} = \emptyset$. On the opposite side, if $\mathcal{S}(ex) = C_o^{max}$, then S_{ex} consists of a unique state s having color C_o^{max}, and R_{ex} consists of two edges, one from state $\mathcal{S}$ to state s with cost 0, and the other one from s to ex with cost 0 as well. Otherwise, let $\mathcal{S}(ex) = (f, c_{pr}, c_o, c_e^L, c_e^M, c_e^R)$. Then, we distinguish six cases, where *(i)* c_{ex} is the color of ex, and *(ii)* $d_o = c_{pr}$ and $d_e = c_e^L$ if $c_o \in \{0, c_{pr}\}$, and $d_o = c_o$ and $d_e = \max(\{c_e, c_o + 1\})$ otherwise, where $c_e = c_e^L$ if $c_{pr} \in C_o$, and $c_e = c_{pr}$ otherwise. In the figures illustrating the construction, we assume that ex is controlled by Player 0.

Case $f = bnd_0$



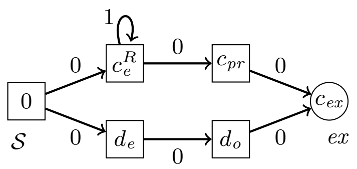
In this case, we have $c_e^M = \bot$ and $c_e^R = \max(C_e)$. The sub-gadget $\mathit{Gad}(\mathcal{G}, \mathcal{S}, ex)$ for this case is a DAG and is illustrated on the left. Note that the cost of any path from state $\mathcal{S}$ to the exit state ex is 0.

Case $f = bnd_1$



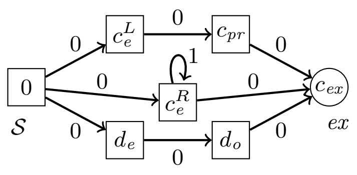
In this case, we have that $c_e^M \in C_e$, $c_e^M \in [c_e^L, c_e^R]$, and $C_e^R = \max(C_e)$. The associated sub-gadget is a DAG and it is illustrated on the right, where $\rho = 0$ if $c_e^L < c_e^M$, and $\rho = 1$ otherwise. Note that the overall cost of all paths from state $\mathcal{S}$ to the exit state ex is 1. Moreover, according to the definition of even-middle value, c_e^M represents the smallest even color c_e such that the cost of all exit plays leading to ex and having maximal even color c_e is finite and non-null. Additionally, if $c_e^L < c_e^M$, according to the definition of even-left value, there are exit plays leading to ex whose maximal even color is c_e^L, and the overall cost of such exit plays is 0.

Case $f = \textit{unb}$, $c_e^M = \bot$, and $c_e^L = c_e^R$



The sub-gadget $\textit{Gad}(\mathcal{G}, \mathcal{S}, \textit{ex})$ for this case is illustrated on the left. When $f = \textit{unb}$, the overall cost of all exit plays leading to $\textit{ex}$ is infinite. This is implemented by a self-loop with cost 1 on the state having color c_e^R. Note that for a strategy σ of Player 0 with $\textit{value}_{\text{Cost}}(\sigma, \textit{ex}) = \textit{unb}$ and $\textit{value}_e^R(\sigma, \textit{ex}) = c_e^R$, the overall cost of all $\textit{ex}$-exit plays having maximal even color at most c_e^R maybe finite. However, in this case, $c_e^R < \max(C_e)$ and $\text{ResCost}(\sigma, \textit{ex}, c_e^R + 2) = \infty$. Thus, the self-loop with cost 1 in the sub-gadget above takes into account also these possible scenarios.

Case $f = \textit{unb}$, $c_e^M = \bot$, and $c_e^L < c_e^R$



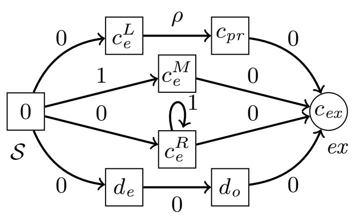
This case is similar to the previous one. The unique difference is that now $c_e^L < c_e^R$. Thus, the associated sub-gadget – illustrated on the right – summarizes strategies σ of Player 0 for which, in particular, $\textit{value}_e^L(\sigma, \textit{ex}) = c_e^L$ and there are exit plays leading to $\textit{ex}$ whose maximal even color is c_e^L, and the overall cost of such exit plays is 0.

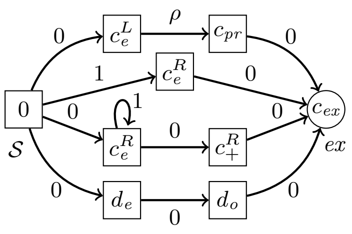
Case $f = \textit{unb}$, $c_e^M \in C_e$, and $c_e^M < c_e^R$

This case is similar to the previous one, but now $c_e^M \in C_e$, hence, $c_e^M \in [c_e^L, c_e^R]$. The associated sub-gadget is illustrated in the left part of the figure below, where $\rho = 0$ if $c_e^L < c_e^M$, and $\rho = 1$ otherwise.

Case $f = \textit{unb}$, $c_e^M \in C_e$, and $c_e^M = c_e^R$

In this case, we have that $c_e^M \in [c_e^L, c_e^R]$ and $c_e^R < \max(C_e)$. The associated sub-gadget is illustrated on the right of the figure below, where $c_+^R = c_e^R + 2$, $\rho = 0$ if $c_e^L < c_e^M$, and $\rho = 1$ otherwise. In this case there is an even color, namely c_+^R, whose response-cost with respect to $\textit{ex}$ is infinite. This is consistent with the fact that for all strategies σ of Player 0 such that $\textit{value}_{\text{Cost}}(\sigma, \textit{ex}) = \textit{unb}$, $\textit{value}_e^R(\sigma, \textit{ex}) = c_e^R$, and $\text{Cost}_e(\sigma, \textit{ex}, c_e^R) \neq \infty$, we have that $\text{ResCost}_e(\sigma, \textit{ex}, c_e^R + 2) = \infty$.





We now define the gadget arena $\textit{Gad}(\mathcal{G}, \mathcal{S})$ for the given summary $\mathcal{S}$, which is intuitively obtained by merging the sub-gadgets $\textit{Gad}(\mathcal{G}, \mathcal{S}, \textit{ex})$ for the various exit states $\textit{ex} \in \textit{Exit}$ and by adding the state in. Formally, assuming that $S_{ex} \cap S_{ex'} = \emptyset$ (i.e., sub-gadgets associated with distinct exit states share only state $\mathcal{S}$), $Gad(\mathcal{G}, \mathcal{S})$ has the same interface as $\mathcal{G}$ and satisfies the following: the set of states of $Gad(\mathcal{G}, \mathcal{S})$ is $\{in, \mathcal{S}\} \cup Exit \cup \bigcup_{ex \in Exit} \mathrm{S}_{ex}$ and the set of transitions is $\{(in, \mathcal{S})\} \cup \bigcup_{ex \in Exit} \mathrm{R}_{ex}$, where transition $(in, \mathcal{S})$ has cost 0.

▶ Remark. Note that in a summary-gadget arena $Gad(\mathcal{G}, \mathcal{S})$, every state which is not in $\{in\} \cup Exit$ is controlled by Player 1. In particular, there is exactly one strategy of Player 0, and such a strategy is non-loosing.

By construction, we easily obtain the following result.

▶ **Proposition 7.** *Let $\mathcal{G} = \langle \mathcal{A}, Cost, \Omega, Exit \rangle$ be a partial-cost parity arena, σ a strategy of Player 0, σ_S the unique strategy of Player 0 in $Gad(\mathcal{G}, \mathcal{S}(\sigma))$, and $s \in Exit$. Then, $value(\sigma, s) = value(\sigma_S, s)$. Moreover, if $value(\sigma, s) \neq C_o^{max}$, the following holds:*

- *Let ν be an s-exit play of σ_S with maximal even color c_e. Then, either* (i) *$Cost_e(\sigma, s, c_e) \geq Cost_e(\sigma_S, s, c_e)$, and there is an s-exit play ν' of σ whose maximal even color is at most c_e, or* (ii) *$Cost_e(\sigma_S, s, c_e) = \infty$, $c_e < \max(C_e)$, and $ResCost(\sigma, s, c_e + 2) = \infty$.*
- *For each $c_e \in C_e$, $ResCost(\sigma_S, s, c_e) = \infty$ entails that $ResCost(\sigma, s, c_e) = \infty$.*

Not all the summaries of $\mathcal{G}$ are associated with non-loosing strategies (of Player 0). On the other hand, checking whether a summary is associated with a non-loosing strategy is not an easy task since we have to check the fulfillment of unboundedness conditions. However, we can get around the problem by exploiting monotonicity properties of the cost-parity winning conditions. We define a reflexive and transitive relation $\sqsupseteq$ over the set of summaries. Intuitively, $\mathcal{S} \sqsupseteq \mathcal{S}'$ when $\mathcal{S}$ is not worse than $\mathcal{S}'$ for Player 0. A summary $\mathcal{S}$ is then *relevant* if $\mathcal{S}(\sigma) \sqsupseteq \mathcal{S}$ for some non-loosing *memoryless* strategy σ. As we will see in Section 3.2, checking whether a summary is relevant can be done in polynomial space.

▶ **Definition 8** (Relevant summaries)**.** Let $\sqsupseteq$ be a binary relation over $\mathcal{E}_C$ defined as follows:

- $\vdash \sqsupseteq ev$ for all $ev \in \mathcal{E}_C$;
- $ev \sqsupseteq C_o^{max}$ for all $ev \in \mathcal{E}_C$;
- $(f, c_{pr}, c_o, c_e^L, c_e^M, c_e^R) \sqsupseteq (\tilde{f}, \tilde{c_{pr}}, \tilde{c_o}, \tilde{c_e^L}, \tilde{c_e^M}, \tilde{c_e^R})$ if $f \succeq_b \tilde{f}$, $c_{pr} \succeq_0 \tilde{c_{pr}}$, $c_o \succeq_0 \tilde{c_o}$, $c_e^L \succeq_0 \tilde{c_e^L}$, $c_e^R \succeq_0 \tilde{c_e^R}$, and the following holds:
  - if $c_e^M \neq \bot$, then either $\tilde{c_e^M} \neq \bot$ and $c_e^M \geq \tilde{c_e^M}$, or $\tilde{c_e^M} = \bot$ and $c_e^M \geq \tilde{c_e^R}$.

Given two summaries $\mathcal{S}$ and $\mathcal{S}'$ of $\mathcal{G}$, we say that $\mathcal{S}$ *is not worse than $\mathcal{S}'$ for Player* 0, written $\mathcal{S} \sqsupseteq \mathcal{S}'$, if $\mathcal{S}(s) \sqsupseteq \mathcal{S}'(s)$ for all $s \in Exit$. A summary $\mathcal{S}$ of $\mathcal{G}$ is *relevant* iff there is a memoryless non-loosing strategy σ in $\mathcal{G}$ such that $\mathcal{S}(\sigma) \sqsupseteq \mathcal{S}$.

▶ Remark. The binary relation $\sqsupseteq$ over the set of summaries is reflexive and transitive.

Note that if $\mathcal{G}$ has no exits, then the unique summary is the empty set, and such a summary is relevant iff there is a memoryless winning strategy of Player 0 from in. By construction, we easily obtain the following result, which represents the converse of Proposition 7.

▶ **Proposition 9.** *Let $\mathcal{G} = \langle \mathcal{A}, Cost, \Omega, Exit \rangle$ be a partial-cost parity arena, $\mathcal{S}$ a summary of $\mathcal{G}$, σ a strategy of Player 0 such that $\mathcal{S}(\sigma) \sqsupseteq \mathcal{S}$, σ_S the unique strategy of Player 0 in $Gad(\mathcal{G}, \mathcal{S})$, and $s \in Exit$. Then $value(\sigma, s) \sqsupseteq value(\sigma_S, s)$. Moreover, if $\mathcal{S}(s) \neq C_o^{max}$, the following holds:*

- *Let ν be an s-exit play of σ, c_e the maximal even color of ν, and $Cost_e(\sigma, s, c_e) = m \in \mathbb{N} \cup \{\infty\}$. Then, either* (i) *$Cost_e(\sigma_S, s, c_e) = m'$ where $m' > 0$ if $m > 0$, and $m' = \infty$ if $m = \infty$, and there is a s-exit play ν' of σ_S whose maximal even color is at most c_e, or* (ii) *$Cost_e(\sigma, s, c_e) = \infty$, $c_e < \max(C_e)$, and $ResCost(\sigma_S, s, c_e + 2) = \infty$.*

- *For each $c_e \in C_e$, if $ResCost(\sigma, s, c_e) = \infty$, one of the following holds:*
  - *either $ResCost(\sigma_S, s, c'_e) = \infty$ for some even color $c'_e \leq c_e$,*
  - *or there is an even color $c'_e \leq value^R_e(\sigma, s) < c_e$ such that $Cost_e(\sigma_S, s, c'_e) = \infty$.*

Note that the set of relevant summaries in $\mathcal{G}$ is empty iff there does not exist a memoryless non-loosing strategy in $\mathcal{G}$. By Theorem 1, checking this condition can be done in polynomial space. In this case, we associate with $\mathcal{G}$ a simple partial cost-parity arena (bad gadget), where Player 0 always loses.

▶ **Definition 10** (Bad-Gadget Arena). The *bad-gadget arena $BadGad(\mathcal{G})$* of $\mathcal{G}$ is the partial cost-parity game having the same interface as $\mathcal{G}$ and defined as follows: $BadGad(\mathcal{G})$ has a unique 'internal' state $s \notin \{in\} \cup Exit$, which has color 0 and is controlled by Player 0, and a unique transition, namely (in, s), which has cost 0.

3.2 Checking relevance of summaries

We reduce the problem of checking summary relevance in partial cost-parity arenas to verifying the existence of *memoryless* strategies in cost-parity arenas under a simple imperfect-information setting. Formally, an *observation-based cost-parity arena* (*OCPA*) is a cost-parity arena $\mathcal{G} = \langle \mathcal{A}, \mathrm{Cost}, \Omega, Obs\rangle$ equipped with an observability equivalence relation $Obs \subseteq \mathrm{S} \times \mathrm{S}$ over the set of states. An *observation-based memoryless* strategy of Player 0 is a memoryless strategy σ of Player 0 such that, for all non-terminal states s and s' controlled by Player 0, $(s, s') \in Obs \Rightarrow (\sigma(s), \sigma(s')) \in Obs$. The following easily follows.

▶ **Theorem 11.** *Let $\mathcal{G} = \langle \mathcal{A}, Cost, \Omega, Obs\rangle$ be an OCPA. Checking the existence of a winning observation-based memoryless strategy of Player 0 from the initial state can be done in polynomial space.*

▶ **Theorem 12** (Checking summary relevance). *Let $\mathcal{G} = \langle \mathcal{A}, Cost, \Omega, Exit\rangle$ be a partial cost-parity arena over C with $\mathcal{A} = \langle \mathrm{S}, \mathrm{S}_0, \mathrm{S}_1, \mathrm{R}, in\rangle$ and $\mathcal{S}$ a summary of $\mathcal{G}$. Then, one can check in polynomial space whether $\mathcal{S}$ is relevant.*

Proof. We build in polynomial time an *OCPA* $\mathcal{G}_\mathcal{S}$ such that, there is a winning observation-based memoryless strategy of Player 0 in $\mathcal{G}_\mathcal{S}$ from the initial state iff $\mathcal{S}$ is relevant in $\mathcal{G}$. We first construct a partial *OCPA* $\mathcal{G}'$ obtained from $\mathcal{G}$ by extending every state of $\mathcal{G}$ with additional information which keeps tracks of the maximal even color and the maximal *unanswered* odd color visited in the current play-prefix from in and a flag indicating whether such a prefix has cost zero. Formally, $\mathcal{G}' = \langle \mathcal{A}', \mathrm{Cost}', \Omega', Exit', Obs\rangle$ where $\mathcal{A} = \langle \mathrm{S}', \mathrm{S}'_0, \mathrm{S}'_1, \mathrm{R}', in'\rangle$ and:

- $\mathrm{S}' = \mathrm{S}\times C_e \times \tilde{C}_o \times \{0,1\}$, $Exit' = Exit \times C_e \times \tilde{C}_o \times \{0,1\}$, $in' = (in, 0, 0, 0)$, $\Omega'((s, c_e, c_o, d)) = \Omega(s)$, and $((s, c_e, c_o, d), (s', c'_e, c'_o, d')) \in Obs$ *iff* $s = s'$. Moreover, the player of each state (s, c_e, c_o, d) is the player of s in $\mathcal{G}$ if $s \notin Exit$, Player 0 if $s \in Exit$ and $\mathcal{S}(s) = \vdash$, and Player 1 otherwise.
- $((s, c_e, c_o, d), (s', c'_e, c'_o, d')) \in \mathrm{E}'$ iff *(i)* $(s, s') \in \mathrm{E}$, *(ii)* $c'_e = \max_{\succeq_0}(\{c_e, \Omega(s)\})$, *(iii)* $c'_o = 0$ if $\Omega(s') \in C_e$ and $\Omega(s') \geq c_o$, and $c_o = \min_{\succeq_0}(\{c_o, \Omega(s)\})$ otherwise, and *(iv)* $d' = 0$ if $d = 0$ and $\mathrm{Cost}(s, s') = 0$, and $d' = 1$ otherwise;
- $\mathrm{Cost}'((s, c_e, c_o, d), (s', c'_e, c'_o, d')) = \mathrm{Cost}(s, s')$.

Note that by construction, there is a bijection, denoted by Obs, between the memoryless strategies σ of Player 0 in $\mathcal{G}$, and the observation-based memoryless strategies of Player 0 in $\mathcal{G}'$. Formally, for each non-terminal state (s, c_e, c_o, d) of $\mathcal{G}'$ controlled by Player 0, $Obs(\sigma)((s, c_e, c_o, d))$ is the unique successor of (s, c_e, c_o, d) having as S-component $\sigma(s)$.

For each $ex \in Exit$, let $Exit'_{ex}$ be the set of exit states of $\mathcal{G}'$ having ex as S-component. The game $\mathcal{G}_\mathcal{S}$ is obtained from $\mathcal{G}'$ by adding for each exit state $ex \in Exit$ such that $\mathcal{S}(ex) \notin \{\vdash, C_o^{max}\}$, a gadget (subgraph) consisting of states controlled by Player 1 that connects the exit states of $\mathcal{G}'$ in $Exit'_{ex}$ with the initial state $in' = (in, 0, 0, 0)$, and an additional terminal state $\Box$ which is controlled by Player 0. If $\mathcal{S}(ex) = \vdash$, then for every strategy σ of Player 0 in $\mathcal{G}$, $\mathcal{S}(\sigma)(ex) = \vdash$ if $\mathcal{S}(\sigma) \sqsupseteq \mathcal{S}$, and our choice (states in $Exit'_{ex}$ are controlled by Player 0) allows to capture only the non-loosing strategies σ of $\mathcal{G}$ for which there is no exit play leading to ex. On the other hand, if $\mathcal{S}(ex) = C_o^{max}$, then for each strategy σ of Player 0 in $\mathcal{G}$, $\mathcal{S}(\sigma)(ex) \sqsupseteq C_o^{max}$, and accordingly, states in $Exit'_{ex}$ are controlled by Player 1.

Now, we describe the construction of the gadget for ex when $\mathcal{S}(ex) \notin \{\vdash, C_o^{max}\}$, i.e., $\mathcal{S}(ex)$ is of the form $(f, c_{pr}, c_o, c_e^L, c_e^M, c_e^R) \in \{bnd_0, bnd_1, unb\} \times \tilde{C} \times \tilde{C}_o \times C_e \times (C_e \cup \{\perp\}) \times C_e$. Due to space limitations, here we focus only on the case where $f = unb$ and $c_o > c_e^R$. Note that for each strategy σ of Player 0, it holds that $\mathrm{Cost}(\sigma, ex) \succeq_b unb$. The gadget for this case is obtained by adding 2 new states *controlled by Player* 1, namely ex_e and ex_o^R, a new terminal state $\Box$ controlled by Player 0, and new transitions. State $\Box$ has color 0, state ex_e has the even color $c_o + 1$, and state ex_o^R has color 0 if $c_e^R = 0$, and the odd color $c_e^R - 1$ otherwise. The new transitions have cost 0 and are as follows:

- for each $s = (ex, c'_e, c'_o, d) \in Exit'_{ex}$ such that one of the following bad conditions is satisfied, we add the transition $(s, \Box)$.
  - *Bad conditions:* either *(i)* $\max(\{c'_o, c'_e\}) \prec_o c_{pr}$, or *(ii)* $c'_e < c_e^L$, or *(iii)* $d = 1$, $c_e^M = \perp$, and $c'_e < c_e^R$, or *(iv)* $d = 1$, $c_e^M \neq \perp$, and $c'_e < c_e^M$.
- the transitions (ex_e, ex_o^R) and (ex_o^R, in'), and for each $s \in Exit'_{ex}$, the transition (s, ex_e).

The transitions having as target state $\Box$ are exploited to capture the strategies σ of Player 0 in $\mathcal{G}$ satisfying the following: *(i)* $value_{pr}(\sigma, ex) \succeq_o c_{pr}$, *(ii)* in each exit play ν of σ leading to ex, the maximal even color of ν is at least c_e^L and *(iii)* if $value_e^M(\sigma, ex) \neq \perp$, then either $c_e^M = \perp$ and $value_e^M(\sigma, ex) \geq c_e^R$, or $c_e^M \neq \perp$ and $value_e^M(\sigma, ex) \geq c_e^M$.

Moreover, given a memoryless strategy σ of Player 0 in $\mathcal{G}$, the chains of transitions (s, ex_e), (ex_e, ex_o^R) and (ex_o^R, in') entering the initial state in', where $s \in Exit'_{ex}$, are responsible of cycles consistent with $Obs(\sigma)$ of the form $\nu \cdot ex_e \cdot ex_o^R \cdot in'$, where ν is an arbitrary exit play of $Obs(\sigma)$ leading to some exit state $s \in Exit'_{ex}$. By concatenating these cycles, one obtains infinite plays consistent with $Obs(\sigma)$ which are winning for Player 0 iff $value(\sigma, ex) \neq C_o^{max}$ (the request cost of σ w.r.t. 0 is finite), $value_o(\sigma, ex) \succeq_0 c_o$, and $value_e^R(\sigma, ex) \geq c_e^R$.

By construction, for each memoryless strategy σ of Player 0 in $\mathcal{G}$, σ is non-loosing and $\mathcal{S}(\sigma) \sqsupseteq \mathcal{S}$ *iff* $Obs(\sigma)$ is winning for Player 0 from state in'. Thus, since Obs is a bijection between the memoryless strategies of Player 0 in $\mathcal{G}$ and the observation-based memoryless strategies of Player 0 in $\mathcal{G}_\mathcal{S}$, by Theorem 11, Theorem 12 follows. ◀

3.3 Algorithm for solving games on *HCPA*

In this section, by exploiting the summary-gadget arena construction of Section 3.1, we derive a polynomial space algorithm for solving hierarchical cost-parity games. In particular, we describe an NPSPACE procedure which solves the considered problem (recall that by Savitch's theorem, PSPACE = NPSPACE). The outline of the nondeterministic procedure, called Algorithm 1, is given in Fig. 1.

Given an *HCPA* $\mathcal{H} = \langle \mathcal{V}, \mathrm{Cost}, \Omega \rangle$ with $\mathcal{V} = \langle \mathcal{V}_1, \ldots, \mathcal{V}_n \rangle$, Algorithm 1 proceeds in phases corresponding to the iterations of the repeat loop. In each phase, the modular sub-arenas of $\mathcal{H}$ are processed in increasing order w.r.t. the hierarchical level, starting with the lowest level sub-arena $\mathcal{V}_n$ which has no boxes, and, therefore, corresponds to its flat expansion $\mathcal{H}_n^F$. At

```
Algorithm 1
Input: $\mathcal{H} = \langle\langle\mathcal{V}_1, \ldots, \mathcal{V}_n\rangle, \mathrm{Cost}, \Omega\rangle$
repeat
  for $i = n$ downto 1 do
    guess $\mathcal{G}_{app,i} \in \mathit{AppSimplify}(\mathcal{V}_i, g_{app,i})$;
    $g_{app,i}(b) = \mathcal{G}_{app,\Upsilon_i(b)}$ for all $b \in B_i$;
  if Player 0 wins in $\mathcal{G}_{app,1}$ then accepts
```

Figure 1 NPSPACE procedure.

step $n \geq i \geq 1$ of the current phase, the algorithm nondeterministically chooses a partial cost-parity arena $\mathcal{G}_{app,i}$ from a finite set of partial cost-parity arenas obtained by applying the operation *AppSimplify* (approximated simplification) to the modular sub-arena $\mathcal{V}_i$ and the *substitution* mapping $g_{app,i}$. A substitution for a modular sub-arena $\mathcal{V}_i$ is a mapping associating to each box b of $\mathcal{V}_i$ with $\Upsilon_i(b) = k$, a partial cost-parity arena having the same interface as the flat expansion $\mathcal{H}^F_k$ of $\mathcal{V}_k$ (hence, $g(b)$ has initial state in_k and set of exit states Exit_k). In our case, the substitution $g_{app,i}$ considered by the algorithm at iteration $n \geq i \geq 1$ maps each box b of $\mathcal{V}_i$ with the the guessed approximation $\mathcal{G}_{app,\Upsilon_i(b)}$ in the previous iteration $\Upsilon_i(b)$ of the current phase (recall that $\Upsilon_i(b) > i$). The essence of the operation *AppSimplify* is to replace each box b of $\mathcal{V}_i$ with a copy of the summary-gadget arena of $\mathcal{G}_{app,\Upsilon_i(b)}$ associated with some relevant summary of $\mathcal{G}_{app,\Upsilon_i(b)}$. Note that at iteration n, $g_{app,n}$ is empty and $\mathcal{G}_{app,n}$ coincides with $\mathcal{H}^F_n$. If Player 0 wins in $\mathcal{G}_{app,1}$ (recall that the top-level arena $\mathcal{V}_1$ has no exit, hence, $\mathcal{G}_{app,1}$ has no exit as well), then the algorithm accepts the input $\mathcal{H}$. Otherwise, a new phase is started. We now formally define the approximated simplification operation.

▶ **Definition 13** (Approximated simplification)**.** Let $\mathcal{H} = \langle\mathcal{V}, \mathrm{Cost}, \Omega\rangle$ be an *HCPA* with $\mathcal{V} = \langle\mathcal{V}_1, \ldots, \mathcal{V}_n\rangle$, $i \in [1, n]$, and g be a substitution for $\mathcal{V}_i$. For a box b of $\mathcal{V}_i$ and a relevant summary $\mathcal{S}$ of $g(b)$, we denote by $\mathit{Gad}_b(g(b), \mathcal{S})$ the copy of the summary-gadget arena $\mathit{Gad}(g(b), \mathcal{S})$ associated with $g(b)$ and $\mathcal{S}$ obtained by replacing each state s in $\mathit{Gad}(g(b), \mathcal{S})$ with the copy (b, s). The b-copy $\mathit{BadGad}_b(g(b))$ of the bad-gadget arena $\mathit{BadGad}(g(b))$ for $g(b)$ is defined in a similar way. Note that the copies of the states in $\{in_k\} \cup \mathit{Exit}_k$, where $k = \Upsilon_i(b)$, are states in the flat expansion $\mathcal{H}^F_i$ of $\mathcal{V}_i$.

The *simplification* $\mathit{Simplify}(\mathcal{V}_i, g, b)$ *of* $\mathcal{V}_i$ *w.r.t. the substitution* g *and the box* b (resp., the *simplification* $\mathit{Simplify}(\mathcal{V}_i, g, b, \mathcal{S})$ *of* $\mathcal{V}_i$ *w.r.t. the substitution* g*, the box* b*, and a relevant summary* $\mathcal{S}$ *of* $g(b)$) is the partial cost-parity arena obtained from $\mathcal{H}^F_i$ as follows:

- all the states in $\mathcal{H}^F_i$ of the form (b, s) which are not in $\mathit{BadGad}_b(g(b))$ (resp, $\mathit{Gad}_b(g(b), \mathcal{S})$) are removed together with the associated transitions, and all the states in $\mathit{BadGad}_b(g(b))$ (resp, $\mathit{Gad}_b(g(b), \mathcal{S})$) are added together with the associated transitions.

An *approximated simplification of* $\mathcal{V}_i$ *w.r.t.* g is a partial cost-parity arena obtained by applying for each box b of $\mathcal{V}_i$, the simplification operation w.r.t. g and b if the set of relevant summaries of $g(b)$ is empty, and the simplification operation w.r.t. g, b, and some relevant summary $\mathcal{S}_b$ of $g(b)$ otherwise. We denote by $\mathit{AppSimplify}(\mathcal{V}_i, g)$ the set of approximated simplifications of $\mathcal{V}_i$ w.r.t. g.

Note that the arenas in $\mathit{AppSimplify}(\mathcal{V}_i, g)$ can be constructed directly from $\mathcal{V}_i$ without constructing the flat expansion $\mathcal{H}^F_i$. By Propositions 7 and 9, we deduce that the *AppSimplify* operation preserves the set of relevant summaries. In particular, the following holds, where for a partial cost-parity arena $\mathcal{G}$, $\mathit{RS}(\mathcal{G})$ is the set of relevant summaries in $\mathcal{G}$ (we extend the notation RS to sets of partial cost-parity arenas).

▶ **Lemma 14.** *Let $\mathcal{H} = \langle \mathcal{V}, Cost, \Omega \rangle$ with $\mathcal{V} = \langle \mathcal{V}_1, \ldots, \mathcal{V}_n \rangle$ be an HCPA, $i \in [1, n]$, and g a substitution for $\mathcal{V}_i$. Assume that for each box b of $\mathcal{V}_i$, $RS(\mathcal{H}^F_{\Upsilon_i(b)}) = RS(g(b))$. Then, $RS(\mathcal{H}^F_i) = RS(AppSimplify(\mathcal{V}_i, g))$.*

By Lemma 14, we deduce the main result of this paper.

▶ **Theorem 15.** *Solving hierarchical cost-parity games is* PSPACE*-complete.*

4 Conclusion

Cost-parity games represent a powerful machinery for the verification of temporal requirements that are bounded in time. As in many settings, the representation of systems by means of cost-parity games is affected by an exponential blow-up in the size of the resulting game. To overcome this, many techniques exploiting system regularities have been successfully applied. Among them, hierarchical systems deserve a special mention. In this paper, we have introduced and investigated the problem of solving cost-parity games over hierarchical FSMs, showing that the problem is PSPACE-complete, thus not harder than solving parity games over hierarchical models. As future work, we aim to adapt the proposed approach to all the other winning bounded conditions introduced in [24]. Moreover, it would be interesting to investigate cost-parity conditions over concurrent game structures, the last one being a suitable formalism for modelling strategic environments where there is simultaneous interaction between multiple players. Other relevant research directions include the study of cost-parity games in the imperfect information setting as well as for infinite-state systems.

References

1 S. Almagor, Y. Hirshfeld, and O. Kupferman. Promptness in omega-Regular Automata. In *ATVA'10*, LNCS 7388, pages 22–36, 2010.

2 R. Alur, M. Benedikt, K. Etessami, P. Godefroid, T. W. Reps, and M. Yannakakis. Analysis of recursive state machines. *TOPLAS*, 27(4):786–818, 2005.

3 R. Alur, S. Kannan, and M. Yannakakis. Communicating hierarchical state machines. In *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, pages 169–178, 1999.

4 R. Alur and M. Yannakakis. Model checking of hierarchical state machines. *ACM Trans. Program. Lang. Syst.*, 23(3):273–303, 2001.

5 B. Aminof, O. Kupferman, and A. Murano. Improved Model Checking of Hierarchical Systems. *Inf. Comput.*, 210:68–86, 2012. doi:10.1016/j.ic.2011.10.008.

6 B. Aminof, F. Mogavero, and A. Murano. Synthesis of Hierarchical Systems. In *FACS'11*, LNCS 7253, pages 42–60. Springer, 2011.

7 B. Aminof, F. Mogavero, and A. Murano. Synthesis of Hierarchical Systems. *SCP*, 83:56–79, 2014.

8 K. Chatterjee and L. Doyen. Energy Parity Games. *Theor. Comput. Sci.*, 458:49–60, 2012.

9 K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Mean-Payoff Parity Games. In *LICS'05*, pages 178–187, 2005.

10 K. Chatterjee, T. A. Henzinger, and F. Horn. Finitary winning in ω-regular games. *ACM Trans. Comput. Logic*, 11(1), 2009.

11 E. M. Clarke and E. A. Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *LP'81*, LNCS 131, pages 52–71. Springer, 1981.

12 E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press, 2002.

13 E. A. Emerson and C. Jutla. Tree Automata, μ-Calculus and Determinacy. In *FOCS*, pages 368–377, 1991.

14 E. A. Emerson and C. S. Jutla. The Complexity of Tree Automata and Logics of Programs. *SJM*, 29(1):132–158, 1999.

15 N. Fijalkow and M. Zimmermann. Parity and Streett Games with Costs. *Logical Methods in Computer Science*, 10(2), 2014.

16 S. Göller and M. Lohrey. Fixpoint logics on hierarchical structures. In *FSTTCS'05*, LNCS 3821, pages 483–494. Springer, 2005.

17 M. Jurdzinski, M. Paterson, and U. Zwick. A Deterministic Subexponential Algorithm for Solving Parity Games. *SIAM J. Comput.*, 38(4):1519–1532, 2008.

18 D. Kozen. Results on the Propositional muCalculus. *TCS*, 27(3):333–354, 1983.

19 O. Kupferman, N. Piterman, and M. Y. Vardi. From Liveness to Promptness. *Formal Methods in System Design*, 34(2):83–103, 2009.

20 O. Kupferman, M. Y. Vardi, and P. Wolper. An Automata Theoretic Approach to Branching-Time Model Checking. *JACM*, 47(2):312–360, 2000.

21 O. Kupferman, M. Y. Vardi, and P. Wolper. Module Checking. *IC*, 164(2):322–344, 2001.

22 V. Malvone, A. Murano, and L. Sorrentino. Concurrent Multi-Player Parity Games. In *AAMAS'16*, pages 689–697, 2016.

23 F. Mogavero, A. Murano, and L. Sorrentino. On Promptness in Parity Games. In *LPAR'13*, LNCS 8312, pages 601–618. Springer, 2013.

24 F. Mogavero, A. Murano, and L. Sorrentino. On Promptness in Parity Games. *Fundam. Inform.*, 139(3):277–305, 2015. `doi:10.3233/FI-2015-1235`.

25 A. Pnueli. The Temporal Logic of Programs. In *FOCS'77*, pages 46–57. IEEE Computer Society, 1977.

26 J. P. Queille and J. Sifakis. Specification and Verification of Concurrent Programs in Cesar. In *SP'81*, LNCS 137, pages 337–351. Springer, 1981.

Timed-Automata-Based Verification of MITL over Signals*

Thomas Brihaye[1], Gilles Geeraerts[2], Hsi-Ming Ho[3], and Benjamin Monmege[4]

1 Université de Mons, Mons, Belgium
thomas.brihaye@umons.ac.be
2 Université libre de Bruxelles, Brussels, Belgium
gigeerae@ulb.ac.be
3 Université de Mons, Mons, Belgium
hsi-ming.ho@umons.ac.be
4 Aix Marseille Univ, CNRS, LIF, Marseille, France
benjamin.monmege@univ-amu.fr

Abstract

It has been argued that the most suitable semantic model for real-time formalisms is the non-negative real line (signals), i.e. the continuous semantics, which naturally captures the continuous evolution of system states. Existing tools like UPPAAL are, however, based on ω-sequences with timestamps (timed words), i.e. the pointwise semantics. Furthermore, the support for logic formalisms is very limited in these tools. In this article, we amend these issues by a compositional translation from Metric Temporal Interval Logic (MITL) to signal automata. Combined with an emptiness-preserving encoding of signal automata into timed automata, we obtain a practical automata-based approach to MITL model-checking over signals. We implement the translation in our tool MIGHTYL and report on case studies using LTSMIN as the back-end.

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

Keywords and phrases real-time temporal logic, timed automata, real-time systems

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.7

1 Introduction

Many computer programs nowadays control critical applications, and need to enforce complex requirements in order to guarantee safe, dependable and efficient operation of the whole system. Among these requirements, real-time specifications (such as 'every request is eventually followed by an acknowledgement *within* 3 time units') are common. In this framework, computers interact with an environment that is intrinsically continuous, and ensuring complex real-time constraints is known to be a very difficult task.

Different kinds of formalisms have been proposed over the past 30 years to *specify* those real-time models (often by means of automata) and requirements (usually by means of some logic language). On the automata side, the model of timed automata [2] is arguably widely accepted today, a success which is due in part to the tool support provided by UPPAAL [35] and other verification tools such as KRONOS [11], TIAMO [10]... As far as logics are concerned, several proposals have been made in the literature during the past 30 years (such

* This work has been supported by the FRS/F.N.R.S. PDR grant SyVeRLo, and (partially) funded by the DeLTA project (ANR-16-CE40-0007) and the SensAS project (INS2I JCJC'17).

as MTL [33], TPTL [6], TCTL [1]. . .) but the recent research seems to focus mainly on MTL, for theoretical reasons (we think here of the works of Ouaknine and Worrell on the decidability of MTL [38]); and on MITL [3] for more practical motivations [36, 12, 31, 13, 15, 9].

Indeed, since its introduction in 1996, MITL has been advocated as a good '*trade-off between realistic modelling of time and feasible verification of timing properties*' [3]. MITL is at the same time a real-time extension of LTL, the most widely accepted logic in the non-real-time case; and a restriction of MTL, whose expressive power makes it undecidable in most practical cases [5, 38]. Unfortunately, tool support for MITL is still lacking today, albeit MITL's clear practical interest (and indeed, the need for such tool support is repeatedly emphasised in several papers [3, 36, 9]). Uppaal, the most prominent real-time model checker, supports only a restricted subset of TCTL; and the alternatives are either not publicly available, or too restricted, or too experimental (see the *related work* hereinafter for a more comprehensive picture). We believe this is due to the relative lack of maturity of automata-based support for MITL, at least when compared with LTL.

Another point of debate in the community is the choice of the semantics for real-time models. The two different options are known as the *pointwise* and *continuous* semantics. In the pointwise semantics, executions of the system are timed words, i.e. sequences of pairs (timestamp, system state). That is, the system's states can only be observed at selected timestamps (which are non-negative real values). In the continuous semantics, executions are *signals*, i.e. sequences of contiguous intervals during which the state of the system does not change and can be continuously observed. While the pointwise semantics is the most common today (probably due to the success of timed automata which have initially been defined in this framework), it has been argued [7, 29] that the continuous semantics models time more faithfully, and it is indeed adopted in many works about control of hybrid systems [41], synthetic biology [8], etc. Apart from these practical considerations, the difference between these two semantics matters as it changes the expressive power of the logic.[1] For example, the following formula (which requires p to hold exactly in $[0, a]$ for some $a \geq 0$) is satisfiable in the continuous semantics only: $p \wedge \mathbf{F}(\neg p) \wedge \mathbf{G}\big(\neg p \Rightarrow \mathbf{G}(\neg p)\big) \wedge \neg\big(p \,\mathbf{U}\, (\neg p)\big)$.

Contribution. In order to remedy the lack of comprehensive tool support for MITL in the *pointwise* semantics, we have recently introduced MightyL [14], an efficient tool that turns MITL formulae into a network of timed automata (expressed in the Uppaal language) accepting the same language. These timed automata can then be used to perform satisfiabilty or model-checking, using off-the-shelf model checkers such as Uppaal or LTSMin. The central point of the efficiency of our construction is its *compositional* feature: we output a network of timed automata (one per subformula) instead of a single, monolithic, one. In the present work, we extend this line of research to the realm of *continuous semantics* by revisiting the compositional translation of MITL into *signal automata* (i.e. automata akin to timed automata, but that accept signals instead of timed words).

More precisely, we introduce, in Section 3, a compositional translation that turns an MITL formula φ into a network of signal automata $\mathcal{C}_{init} \times \prod_{\chi} \mathcal{C}_{\chi}$, one for each subformula χ in φ, plus an extra signal automaton $\mathcal{C}_{init}$ (extending the ideas of our previous work [14] to the continuous setting). However, as is, this translation would not allow us to rely on the currently available tools for timed systems since most of them (and in particular, Uppaal) rely on the pointwise semantics. So, in Section 4, we present an emptiness-preserving and compositional transformation from signal automata to timed automata (see Theorem 11).

[1] As for MTL, for instance, which becomes decidable on finite words in the pointwise semantics [38].

Concretely, given a signal automaton $\mathcal{A}$ modelling a system, and a property φ to be checked on $\mathcal{A}$, we can perform model-checking by: (i)$\pm$ building the network of signal automata $\mathcal{C}_{init} \times \prod_\chi \mathcal{C}_\chi$ from $\neg\varphi$ using the procedure of Section 3; (ii) translating, using the techniques of Section 4, $\mathcal{A}$, $\mathcal{C}_{init}$ and all $\mathcal{C}_\chi$ into corresponding *timed* automata $\mathcal{B}^{\mathcal{A}}$, $\mathcal{B}_{init}$ and $\mathcal{B}_\chi$ (for all subformulae χ) respectively; and (iii) checking (using a model-checker for timed automata) whether the language $\mathcal{B}^{\mathcal{A}} \times \mathcal{B}_{init} \times \prod_\chi \mathcal{B}_\chi$ is empty. If this is the case, then our translation ensures that the language of $\mathcal{A} \times \mathcal{C}_{init} \times \prod_\chi \mathcal{C}_\chi$ is empty, which in turn holds if and only if $\mathcal{A} \models \varphi$, by construction. We have implemented this approach as an extension of MightyL and report on experiments in Section 5. The preliminary results are very encouraging, as our approach compares well or outperforms previous approaches from the literature.

Related work. The most similar work to ours is [32] where the authors propose a compositional translation from MITL with past operators [4] to signal automata. The translation works by rewriting the input formula into one with only past operators using projections [21]. Each past subformula can then be handled by a simple component, and the resulting automaton is obtained by synchronising the components via newly introduced propositions. An advantage of this approach is that it directly supports past operators. Unfortunately, the rewriting step does not work for unbounded future operators; this severely limits the applicability of the translation (for example, the liveness property $\mathbf{GF}p$ cannot be expressed in the bounded-future fragment). Also, as far as we know, it has never been implemented. By contrast, while our translation deals only with future MITL, one may use projections to remove past operators from the input formula.

Compositional translations that support unbounded future operators also exist in the literature [36, 37, 20]. One difference of these with our translation is that they are formulated in terms of non-standard models such as *timed signal transducers* or *hierarchical timed automata*. This deviation from the more common models, we believe, has contributed to the lack of implementation of these translations.[2] Another difference is that the components constructed by these approaches are *testers* whereas those constructed by ours are *positive testers* [40, 16, 23]; that is, suppose we introduce a new proposition p_χ for the subformula $\chi = \varphi_1 \mathbf{U} \varphi_2$, a tester enforces $p_\chi \Leftrightarrow \varphi_1 \mathbf{U} \varphi_2$ to hold at all times while a positive tester only enforces the weaker formula $p_\chi \Rightarrow \varphi_1 \mathbf{U} \varphi_2$ to hold at all times. This may affect the performance of verification algorithms [43]. Moreover, the weaker condition allows us to impose some minimality criteria on transitions for further performance gains (see Section 5).

The original translation from MITL to signal automata in [3] is a monolithic tableau-based procedure which follows roughly the same lines as the tableau-based translation from LTL to Büchi automata [27]: the locations of the resulting automaton are labelled by sets of subformulae, and the transitions between them are obtained by 'expanding' the labels. Like our translation, it also enforces minimality when generating transitions. However, the procedure is much more involved than the LTL counterpart and seems difficult to realise in practice. A simplified tableau-based translation is given in [26, 25] where an implementation – the only implementation of an MITL to signal automata translation we are aware of – is also reported. Nevertheless, the translation only works for the upper-bound fragment of MITL, and the tool is not publicly available.

Besides automata-based approaches, there are also proposals to apply SMT (Satisfiability Modulo Theories) solvers [18] to satisfiability/model-checking for MITL over signals [31, 9]. The SMT approach is straightforward to implement and there are publicly available tools.

[2] These models are, however, not more expressive than signal automata.

However, it is essentially a ‘bounded model-checking’ approach and therefore is inherently incomplete, unless very large (impractical) bounds are used.

2 Model-checking signal automata against MITL

This section introduces the main objects we study – the logic MITL over signals and signal automata – as well as the model-checking problem we tackle.

Signals. An *interval* I is a non-empty convex subset of $\mathbb{R}_{\geq 0}$. If I is bounded ($\sup(I)$ exists), we write $|I|$ for $\sup(I) - \inf(I)$. Let AP be a finite set of atomic propositions. A *state* σ over AP is a subset of AP, i.e. $\sigma \in 2^{\mathsf{AP}}$. A *signal* γ over 2^{AP} is a function that maps each $t \in \mathbb{R}_{\geq 0}$ to a state over AP. Throughout this work, we restrict ourselves to signals that are *finitely variable*, i.e. the number of discontinuities is finite in each bounded interval. We rely on *timed state sequences* to represent signals. Intuitively, a timed state sequence partitions the reals into a sequence of contiguous time intervals during which the state remains constant. A *state sequence* $\overline{\sigma} = \sigma_0\sigma_1\sigma_2\cdots$ over 2^{AP} is an infinite sequence of states $\sigma_i \in 2^{\mathsf{AP}}$. An *interval sequence* $\overline{I} = I_0I_1I_2\cdots$ is an infinite sequence of intervals such that: (1) for all $i \geq 0$, I_i and I_{i+1} are adjacent, i.e. $\sup(I_i) = \inf(I_{i+1})$ and $I_i \cap I_{i+1} = \emptyset$; (2) for each $t \in \mathbb{R}_{\geq 0}$, we have $t \in I_i$ for some $i \geq 0$. An interval sequence is said to be *bipartite* if it alternates between singular and open intervals, i.e. I_i is singular for all even $i \geq 0$. Then, a *timed state sequence* over 2^{AP} is a pair $\kappa = (\overline{\sigma}, \overline{I})$ where $\overline{\sigma}$ is a state sequence over 2^{AP} and $\overline{I}$ is an interval sequence. We let $\kappa(t) = \sigma_i$ if $t \in I_i$ for some $i \geq 0$. We write $[\![\gamma]\!]$ (respectively, $[\![\gamma]\!]^{bp}$) for the set of all timed state sequences (respectively, timed state sequences with bipartite interval sequences) κ such that $\kappa(t) = \gamma(t)$ for all $t \in \mathbb{R}_{\geq 0}$.

Metric Interval Temporal Logic (MITL). We consider the satisfiability and model-checking problems for *Metric Interval Temporal Logic* (MITL), a real-time extension of Linear Temporal Logic (LTL), allowing temporal operators to be labelled with *non-singular* intervals. Formally, MITL formulae over AP are generated by the grammar:

$$\varphi := \top \mid p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \,\mathbf{U}_I\, \varphi,$$

where $p \in \mathsf{AP}$ and I is a non-singular interval with endpoints in $\mathbb{N}_{\geq 0} \cup \{\infty\}$ (I is assumed to be $(0, \infty)$ when omitted).

In this work, we focus on the *continuous semantics* for MITL, in which formulae are interpreted over signals. Given a signal γ over 2^{AP}, $t \in \mathbb{R}_{\geq 0}$, and an MITL formula φ, the satisfaction relation $\gamma, t \models \varphi$ is defined as follows (following [3], we adopt the *strict-future* semantics for the temporal operators):

- $\gamma, t \models \top$; $\quad \gamma, t \models p$ if $p \in \gamma(t)$;
- $\gamma, t \models \varphi_1 \wedge \varphi_2$ if $\gamma, t \models \varphi_1$ and $\gamma, t \models \varphi_2$; $\quad \gamma, t \models \neg\varphi$ if $\gamma, t \not\models \varphi$;
- $\gamma, t \models \varphi_1 \,\mathbf{U}_I\, \varphi_2$ if there exists $t' > t$ such that $t' - t \in I$, $\gamma, t' \models \varphi_2$ and $\gamma, t'' \models \varphi_1$ for all $t'' \in (t, t')$.

We write $\mathcal{S}(\varphi)$ for the set of all signals γ such that $\gamma \models \varphi$.

We will use standard syntactic sugar, e.g. $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\bot \equiv \neg\top$, $\varphi_1 \Rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$, the ‘eventually’ operator $\mathbf{F}_I\varphi \equiv \top \,\mathbf{U}_I\, \varphi$, the ‘globally’ operator $\mathbf{G}_I\varphi \equiv \neg\mathbf{F}_I\neg\varphi$, and the ‘release’ operator $\varphi_1 \,\mathbf{R}_I\, \varphi_2 \equiv \neg((\neg\varphi_1) \,\mathbf{U}_I\, (\neg\varphi_2))$. Hence, the semantics of the release operator can be defined as follows:

- $\gamma, t \models \varphi_1 \,\mathbf{R}_I\, \varphi_2$ if for all $t' > t$ such that $t' - t \in I$, $\gamma, t' \models \varphi_2$ or there exists $t'' \in (t, t')$ such that $\gamma, t'' \models \varphi_1$.

In particular, we can make use of these operators to transform every formula φ into its *negative normal form* where the negations are pushed inwards so that they range on atomic propositions only.

Signal automata. Our tool support for MITL will be based on automata. We first give a formal definition of signal automata, and we will also present classical timed automata afterwards. Like [3], we equip these automata with generalised Büchi acceptance conditions. From now on, a *propositional constraint* ϕ over AP is a set of states over AP; that we denote by means of a Boolean formula over AP. For example, assuming $\mathsf{AP} = \{p, q, r\}$, the propositional constraint $p \wedge \neg q$ denotes $\{\{p, r\}, \{p\}\}$. Let X be a finite set of clocks. The set $\mathcal{G}(X)$ of *clock constraints* g over X is generated by the grammar $g := \top \mid \bot \mid g \wedge g \mid x \bowtie c$ where $\bowtie \in \{\leq, <, \geq, >\}$, $x \in X$ and $c \in \mathbb{N}$. A *valuation* v of X is a mapping $v\colon X \to \mathbb{R}_{\geq 0}$. We denote by $\mathbf{0}$ the valuation that maps every clock to 0. The satisfaction of a constraint g by a valuation v is defined in the usual way and denoted $v \models g$. For $t \in \mathbb{R}_{\geq 0}$, let $v + t$ be the valuation defined by $(v + t)(x) = v(x) + t$ for all $x \in X$. For $\lambda \subseteq X$, let $v[\lambda \leftarrow 0]$ be the valuation defined by $(v[\lambda \leftarrow 0])(x) = 0$ if $x \in \lambda$, and $(v[\lambda \leftarrow 0])(x) = v(x)$ otherwise.

▶ **Definition 1.** A *signal automaton* (SA) over 2^{AP} is a tuple $\mathcal{A} = (L, L_0, \alpha, X, \beta, \Delta, \mathcal{F})$ where

- L is a finite set of locations;
- $L_0 \subseteq L$ is the set of initial locations;
- α is the location labelling function that assigns to each location $\ell \in L$ a propositional constraint $\alpha(\ell) \subseteq 2^{\mathsf{AP}}$;
- X is a finite set of clocks;
- β is the location labelling function that assigns to each location $\ell \in L$ a clock constraint $\beta(\ell) \in \mathcal{G}(X)$;
- $\Delta \subseteq L \times 2^X \times L$ is the set of transitions where each transition consists of the source location, the clocks to be reset with this transition, and the target location;
- $\mathcal{F} \subseteq 2^L$ is the family of sets of accepting locations.

A *run* π of $\mathcal{A}$ on a signal γ over 2^{AP} is an infinite sequence of the following form:

$$\xrightarrow[v_0]{} (\ell_0, I_0) \xrightarrow[v_1]{\lambda_1} (\ell_1, I_1) \xrightarrow[v_2]{\lambda_2} (\ell_2, I_2) \xrightarrow[v_3]{\lambda_3} \ldots$$

where: (1) for all $i \geq 0$, ℓ_i is a location of $\mathcal{A}$; (2) the sequence $I_0 I_1 I_2 \cdots$ is an interval sequence; (3) for all $i \geq 0$: $\lambda_i \subseteq X$; (4) for all $i \geq 0$: v_i is a valuation of X; and that satisfies the following:

- **Initialisation:** $\ell_0 \in L_0$ and $v_0 = \mathbf{0}$; and
- **Consecution:** For all $i \geq 0$: $(\ell_i, \lambda_{i+1}, \ell_{i+1}) \in \Delta$ and $v_{i+1} = (v_i + |I_i|)[\lambda_{i+1} \leftarrow 0]$; and
- **Timing:** $v_\pi(t) \models \beta(\ell_\pi(t))$ for all $t \geq 0$, assuming $v_\pi(t) = v_i + (t - \inf(I_i))$ and $\ell_\pi(t) = \ell_i$ if $t \in I_i$ for some $i \geq 0$; and
- **Adequation:** $\gamma(t) \in \alpha(\ell_\pi(t))$ for all $t \geq 0$.

We say that π is *bipartite* if $I_0 I_1 I_2 \cdots$ is bipartite. We say that π is *accepting* if for all $F \in \mathcal{F}$: $\{i \mid \ell_i \in F\}$ is infinite. A signal γ is *accepted* by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on γ. We write $\mathcal{S}(\mathcal{A})$ for the set of signals accepted by $\mathcal{A}$. For two SAs $\mathcal{A}_1$ and $\mathcal{A}_2$, we denote by $\mathcal{A}_1 \times \mathcal{A}_2$ their (asynchronous) product, defined in a manner similar to [3]: intuitively, in each location of this product, we can either fire only a transition of $\mathcal{A}_1$ (provided that the guard in the current location of $\mathcal{A}_2$ holds after the transition), or only a transition of $\mathcal{A}_2$, or one in $\mathcal{A}_1$ and one in $\mathcal{A}_2$, provided that the guards on their (respective) target locations are satisfied afterwards. In particular, we have $\mathcal{S}(\mathcal{A}_1 \times \mathcal{A}_2) = \mathcal{S}(\mathcal{A}_1) \cap \mathcal{S}(\mathcal{A}_2)$.

We focus on the class of bipartite SA whose runs are bipartite by construction. An SA $\mathcal{A} = (L, L_0, \alpha, X, \beta, \Delta, \mathcal{F})$ is *bipartite* if there exists a partition of L into L^s, L^o respecting the conditions given hereinafter. Intuitively, on reading a signal γ, $\mathcal{A}$ is in a location of L^s (L^o) when it sees a singular (respectively open) interval of $\kappa \in [\![\gamma]\!]^{bp}$:

- $L_0 \subseteq L^s$;
- if $(\ell_1, \lambda, \ell_2) \in \Delta$ then $\ell_1 \in L^s$ if and only if $\ell_2 \in L^o$;
- for each $\ell \in L_0$, $\beta(\ell)$ has $x = 0$ as a conjunct for some clock $x \in X$;
- if $(\ell_1, \lambda, \ell_2) \in \Delta$ with $\ell_1 \in L^o$ (and thus $\ell_2 \in L^s$), then there is a clock $x \in X$ such that $x \in \lambda$ and $\beta(\ell_2)$ has $x = 0$ as a conjunct.

In the rest of the paper, we will assume that all SAs are bipartite.[3] There is no loss of generality, thanks to the following proposition from [3] (see Appendix A for a proof):

▶ **Proposition 2.** *Any* SA $\mathcal{A}$ *can be turned into a bipartite* SA $\mathcal{A}^{bp}$ *such that* $\mathcal{S}(\mathcal{A}) = \mathcal{S}(\mathcal{A}^{bp})$.

From now on, when depicting bipartite SA, we use rectangle and rounded rectangles for the locations from L^s and L^o respectively. Figure 1 shows an example of bipartite SA.

Satisfiability and model-checking problems. In this work, we consider two classical problems: satisfiability and model-checking of MITL. The *satisfiability problem* asks, given an MITL formula φ, whether $\mathcal{S}(\varphi) \neq \emptyset$ (if it is the case, we say that φ is satisfiable). The *model-checking problem* asks, given an SA $\mathcal{A}$ and an MITL formula φ whether $\mathcal{S}(\mathcal{A}) \subseteq \mathcal{S}(\varphi)$. If it is the case, we write $\mathcal{A} \models \varphi$.

3 From MITL to signal automata

Our approach to MITL model-checking over signals is based upon a compositional translation from MITL to signal automata. The core idea is similar to the translation for the pointwise semantics reported in our previous work [14]: we keep track of the satisfiability of each temporal subformula (i.e. a subformula whose outermost operator is temporal) χ with an SA $\mathcal{C}_\chi$. From now on, we fix a set AP of atomic propositions and a *negative normal form* MITL formula φ over AP. To simplify the exposition, we restrict ourselves to a fragment of MITL in which only untimed and upper-bound operators are allowed, i.e. each bounding interval I is either $(0, \infty)$ or $(0, a)$, or $(0, a]$ for some positive integer a. This fragment, however, is already expressively complete for the full MITL [28, 37]. Moreover, we regard all temporal subformulae of φ as distinct formulae.

Triggers. Let Φ be the set of temporal subformulae of φ. We introduce a new atomic proposition p_χ for each $\chi \in \Phi$ and we let $\mathsf{AP}_\Phi = \{p_\chi \mid \chi \in \Phi\}$. Each p_χ is called a *trigger* (for χ). Intuitively, *pulling* the trigger p_χ (i.e. setting p_χ to true) at some point means that χ is required to hold at that point. On the other hand, p_χ being false at some point does not mean that χ must not hold at that point – its satisfaction is simply not required there. The point of the triggers is to enable communication between the different component automata: when χ is a subformula of ψ, the component SA $\mathcal{C}_\psi$ will pull the trigger p_χ whenever the satisfaction of χ is needed to check the value of ψ. A key point of our construction is to avoid unnecessary pulling of triggers, in order to reduce the number of behaviours of the product automaton and mitigate the state explosion problem during the model checking phase. This

[3] Note that a product of bipartite SAs is a bipartite SA.

is the point of the formulae $\overline{\psi}$, $*\psi$, $\sim\psi$ and $\widehat{\psi}$ that we introduce hereinafter. Concretely, the outcome of our construction for an MITL formula φ is a network of SA that accepts an AP_Φ-decorated version of $\mathcal{S}(\varphi)$. In other words, the signals accepted by our construction are over $\mathsf{AP} \cup \mathsf{AP}_\Phi$ and their projections on AP yields $\mathcal{S}(\varphi)$, as stated in Theorem 8 at the end of the section.

For each (not necessarily temporal) subformula ψ of φ, we denote by $\mathcal{P}_\psi$ the set of atomic propositions $p_\chi \in \mathsf{AP}_\Phi$ such that χ is a top-level temporal subformula of ψ, i.e. the outermost operator of χ is $\mathbf{U}_I$ or $\mathbf{R}_I$, yet χ does not occur under the scope of another $\mathbf{U}_I$ or $\mathbf{R}_I$ in ψ. For instance, $\mathcal{P}_{p\mathbf{U}_I q \vee r\mathbf{U}_I(s\mathbf{R}t)} = \{p_{p\mathbf{U}_I q}, p_{r\mathbf{U}_I(s\mathbf{R}t)}\}$. For a signal γ' over $2^{\mathsf{P}'}$ (where P' is a set of atomic propositions) and $\mathsf{P} \subseteq \mathsf{P}'$, we denote by $\mathsf{proj}_{\mathsf{P}}(\gamma')$ the *projection* of γ' onto P, i.e. the signal obtained from γ' by hiding all the atomic propositions $p \notin \mathsf{P}$. For a set of signals $\mathcal{S}$ over $2^{\mathsf{P}'}$ and $\mathsf{P} \subseteq \mathsf{P}'$, we write $\mathsf{proj}_{\mathsf{P}}(\mathcal{S}) = \{\mathsf{proj}_{\mathsf{P}}(\gamma') \mid \gamma' \in \mathcal{S}\}$. Conversely, we say a signal γ' over $2^{\mathsf{P}'}$ *extends* a signal γ over 2^{P} ($\mathsf{P} \subseteq \mathsf{P}'$) if $\mathsf{proj}_{\mathsf{P}}(\gamma') = \gamma$.

Formulae over $\mathsf{AP} \cup \mathsf{AP}_\Phi$. We define some syntactic operations on Boolean combinations over $\mathsf{AP} \cup \mathsf{AP}_\Phi$ that will be used in the components described later. Specifically, for a subformula ψ of φ, we define formulae $\overline{\psi}$ (introducing the trigger variables), $*\psi$ (ensuring that we do not pull any trigger of ψ), $\sim\psi$ (checking that ψ does not hold, while none of its triggers are pulled), and $\widehat{\psi}$ (checking ψ while triggering a minimal set of triggers).

The formula $\overline{\psi}$ is obtained from ψ by replacing all top-level temporal subformulae with their corresponding triggers. Formally, $\overline{\psi}$ is defined inductively as follows (where $p \in \mathsf{AP}$):

$$\overline{\psi_1 \wedge \psi_2} = \overline{\psi_1} \wedge \overline{\psi_2} \qquad \overline{\psi} = \psi \text{ when } \psi \text{ is } \top \text{ or } \bot \text{ or } p \text{ or } \neg p$$
$$\overline{\psi_1 \vee \psi_2} = \overline{\psi_1} \vee \overline{\psi_2} \qquad \overline{\psi} = p_\psi \text{ when } \psi \text{ is } \psi_1 \mathbf{U}_I \psi_2 \text{ or } \psi_1 \mathbf{R}_I \psi_2\,.$$

The formula $*\psi$, read as "do not pull the triggers of ψ", is used to ensure that our components only follow the 'minimal models' of ψ. It is defined as the conjunction of the negations of all $p_\chi \in \mathcal{P}_\psi$ (if $\mathcal{P}_\psi = \emptyset$ then $*\psi = \top$). As a concrete example,

$$*((\neg p \vee \psi_1 \mathbf{U} \psi_2) \wedge (q \vee \psi_3 \mathbf{R} (\psi_4 \mathbf{U} \psi_5))) = \neg p_{\psi_1 \mathbf{U} \psi_2} \wedge \neg p_{\psi_3 \mathbf{R} (\psi_4 \mathbf{U} \psi_5)}.$$

The formula $\sim\psi$ asserts that $\overline{\psi}$ is false and none of its triggers are pulled: $\sim\psi = \neg\overline{\psi} \wedge *\psi$. Finally, the formula $\widehat{\psi}$ is defined as $\mathsf{mm}(\overline{\psi})$ where $\mathsf{mm}(\phi)$ is defined inductively as follows:[4]

$$\mathsf{mm}(\top) = \top \qquad \mathsf{mm}(\bot) = \bot \qquad \mathsf{mm}(p) = p \qquad \mathsf{mm}(\neg p) = \neg p$$
$$\mathsf{mm}(\phi_1 \vee \phi_2) = \big(\mathsf{mm}(\phi_1) \wedge \sim\phi_2\big) \vee \big(\mathsf{mm}(\phi_2) \wedge \sim\phi_1\big) \vee \big((\phi_1 \wedge \phi_2) \wedge *\phi_1 \wedge *\phi_2\big)$$
$$\mathsf{mm}(\phi_1 \wedge \phi_2) = \mathsf{mm}(\phi_1) \wedge \mathsf{mm}(\phi_2)\,.$$

First of all, we notice that formulae $\overline{\psi}$ and $\widehat{\psi}$ are equivalent, once we have projected away the propositions that are not in AP, in the following sense:

▶ **Proposition 3.** *For a subformula ψ of φ, if $\sigma \models \overline{\psi}$ for some state σ over $\mathsf{AP} \cup \mathcal{P}_\psi$, there is a state σ' over $\mathsf{AP} \cup \mathcal{P}_\psi$ such that $\sigma' \models \widehat{\psi}$ and $\mathsf{proj}_{\mathsf{AP}}(\sigma) = \mathsf{proj}_{\mathsf{AP}}(\sigma')$ (and vice versa).*

Proof. By induction on the structure of $\overline{\psi}$. For the direct implication, if $\overline{\psi} = \overline{\psi_1} \vee \overline{\psi_2}$ then one of the following must hold:

- $\sigma \models \overline{\psi_1}$ and $\sigma \not\models \overline{\psi_2}$: apply the induction hypothesis on $\sigma \setminus \mathcal{P}_{\psi_2}$ and $\overline{\psi_1}$ (note that $\mathcal{P}_{\psi_1} \cap \mathcal{P}_{\psi_2} = \emptyset$, and ψ_2 is in negative normal form).

[4] Note that the size of $\widehat{\psi}$ is at most quadratic in the size of ψ.

- $\sigma \not\models \overline{\psi_1}$ and $\sigma \models \overline{\psi_2}$: apply the induction hypothesis on $\sigma \setminus \mathcal{P}_{\psi_1}$ and $\overline{\psi_2}$.
- $\sigma \models \overline{\psi_1}$ and $\sigma \models \overline{\psi_2}$: If $\sigma \setminus \mathcal{P}_{\psi_2} \not\models \overline{\psi_2}$, apply the induction hypothesis on $\sigma \setminus \mathcal{P}_{\psi_2}$ and $\overline{\psi_1}$. Otherwise if $\sigma \setminus \mathcal{P}_{\psi_1} \not\models \overline{\psi_1}$, apply the induction hypothesis on $\sigma \setminus \mathcal{P}_{\psi_1}$ and $\overline{\psi_2}$. Otherwise let $\sigma' = \sigma \setminus (\mathcal{P}_{\psi_1} \cup \mathcal{P}_{\psi_2})$.

The other cases of $\overline{\psi}$ are immediate. The other implication of the proof is simpler. ◀

Minimality of triggers. The real impact of $\widehat{\psi}$ with respect to ψ is to ensure the minimality of triggers pulled during an execution. Indeed, we now show that if φ is satisfied by a signal γ (over 2^{AP}), then there must be a way to extend γ into a signal γ' over $2^{\mathsf{AP} \cup \mathsf{AP}_\Phi}$ such that the triggers AP_Φ are only pulled when necessary in γ', and vice versa. This will be crucial to make our approach efficient in practice, as it reduces the behaviours of the product SA that accepts the whole formula φ. This observation is formalised in the following two propositions.

▶ **Proposition 4.** *For a signal γ over 2^{AP}, we have $\gamma, 0 \models \varphi$ if and only if there exists a signal γ' over $2^{\mathsf{AP} \cup \mathcal{P}_\varphi}$ extending γ such that $\gamma', 0 \models \widehat{\varphi}$, and for all $p_\chi \in \mathcal{P}_\varphi$ and $t \in \mathbb{R}_{\geq 0}$, $\gamma', t \models (p_\chi \Rightarrow \chi)$.*

Proof. For the direct implication, let ζ be a signal over $2^{\mathsf{AP} \cup \mathcal{P}_\varphi}$ extending γ such that $p_\chi \in \zeta(t)$ if and only if $\gamma, t \models \chi$ for each $p_\chi \in \mathcal{P}_\varphi$ and $t \in \mathbb{R}_{\geq 0}$ (note that ζ is necessarily finitely-variable as γ is finitely-variable [3]). If $\zeta, 0 \models \widehat{\varphi}$, simply let $\gamma' = \zeta$ and we are done. If $\zeta, 0 \not\models \widehat{\varphi}$, apply Proposition 3 to $\zeta(0)$ and $\overline{\varphi}$ to obtain a state σ such that $\sigma \models \widehat{\varphi}$. Finally, let $\gamma'(0) = \sigma$ and $\gamma'(t) = \zeta(t) \setminus \mathcal{P}_\varphi$ for all $t \in \mathbb{R}_{>0}$. The other implication is immediate. ◀

▶ **Proposition 5.** *For a signal γ over $2^{\mathsf{AP} \cup \{p_\chi\}}$ where $\chi \in \Phi$ and either $\chi = \psi_1 \, \mathbf{U}_I \, \psi_2$ or $\chi = \psi_1 \, \mathbf{R}_I \, \psi_2$, we have $\gamma, t \models (p_\chi \Rightarrow \chi)$ for all $t \in \mathbb{R}_{\geq 0}$ if and only if there exists a signal γ' over $2^{\mathsf{AP} \cup \{p_\chi\} \cup \mathcal{P}_{\psi_1} \cup \mathcal{P}_{\psi_2}}$ extending γ such that*

- *if $\chi = \psi_1 \, \mathbf{U}_I \, \psi_2$ then, for each $t \in \mathbb{R}_{\geq 0}$, $\gamma', t \models p_\chi \Rightarrow \mathsf{Expand}_\chi$ with*

$$\begin{aligned}\mathsf{Expand}_\chi = \quad & \Big[(\widehat{\psi_1} \wedge {\sim}\psi_2) \, \mathbf{U}_I \, ({*}\,\psi_1 \wedge \widehat{\psi_2})\Big] \vee \Big[(\widehat{\psi_1} \wedge \widehat{\psi_2}) \, \mathbf{U} \top\Big] \\ & \vee \Big[\mathbf{F}_I \widehat{\psi_2} \wedge (\widehat{\psi_1} \wedge {\sim}\psi_2) \, \mathbf{U} \, \big(\widehat{\psi_1} \wedge {\sim}\psi_2 \wedge (\widehat{\psi_1} \wedge \widehat{\psi_2}) \, \mathbf{U} \top\big)\Big]\end{aligned}$$

- *if $\chi = \psi_1 \, \mathbf{R}_I \, \psi_2$ then, for each $t \in \mathbb{R}_{\geq 0}$, $\gamma', t \models p_\chi \Rightarrow \mathsf{Expand}_\chi$ with*

$$\begin{aligned}\mathsf{Expand}_\chi = \quad & \Big[({\sim}\psi_1 \wedge \widehat{\psi_2}) \, \mathbf{U}_I \, (\widehat{\psi_1} \wedge \widehat{\psi_2})\Big] \vee \Big[(\widehat{\psi_1} \wedge {*}\,\psi_2) \, \mathbf{U} \top\Big] \vee \Big[\mathbf{G}_I({\sim}\psi_1 \wedge \widehat{\psi_2})\Big] \\ & \vee \Big[\mathbf{F}_I \widehat{\psi_1} \wedge \big({\sim}\psi_1 \wedge \widehat{\psi_2}\big) \, \mathbf{U} \, \big({\sim}\psi_1 \wedge \widehat{\psi_2} \wedge (\widehat{\psi_1} \wedge {*}\,\psi_2) \, \mathbf{U} \top\big)\Big]\end{aligned}$$

- *for each $p_\theta \in \mathcal{P}_{\psi_1} \cup \mathcal{P}_{\psi_2}$, we have $\mathsf{proj}_{\mathsf{AP} \cup \{p_\theta\}}(\gamma'), t \models (p_\theta \Rightarrow \theta)$ for all $t \in \mathbb{R}_{\geq 0}$.*

Proof. Assume that $\chi = \psi_1 \, \mathbf{U}_I \, \psi_2$ and let ζ be a signal over $2^{\mathsf{AP} \cup \{p_\chi\} \cup \mathcal{P}_{\psi_1} \cup \mathcal{P}_{\psi_2}}$ extending γ such that $p_\theta \in \zeta(t)$ if and only if $\gamma, t \models \theta$ for each $p_\theta \in \mathcal{P}_{\psi_1} \cup \mathcal{P}_{\psi_2}$ and $t \in \mathbb{R}_{\geq 0}$. For each $t \in \mathbb{R}_{\geq 0}$ such that $\gamma, t \models p_\chi$, since $\gamma, t \models \chi$ also holds, exactly one of the following must be true (note that $\inf(I) = 0$):

- there is $t' > t$, $t' - t \in I$ such that $\gamma, t' \models \psi_2$ and $\gamma, t'' \models \psi_1 \wedge \neg\psi_2$ for all $t'' \in (t, t')$;
- there is $t' > t$ such that $\gamma, t'' \models \psi_1 \wedge \psi_2$ for all $t'' \in (t, t')$;
- there are $t' > t$ and $t'' > t'$ such that in γ, $\psi_1 \wedge \neg\psi_2$ always holds in $(0, t']$ and $\psi_1 \wedge \psi_2$ always holds in (t', t'').

It follows that we can obtain a 'minimal labelling' from ζ via Proposition 3. More precisely, we apply Proposition 3 to constant segments of ζ and $\overline{\psi_1}$, $\overline{\psi_2}$, or both $\overline{\psi_1}$ and $\overline{\psi_2}$, as required by the interpretation of p_χ in γ. For example, in the first case above, $\gamma'(t'')$ for each $t'' \in (t, t')$

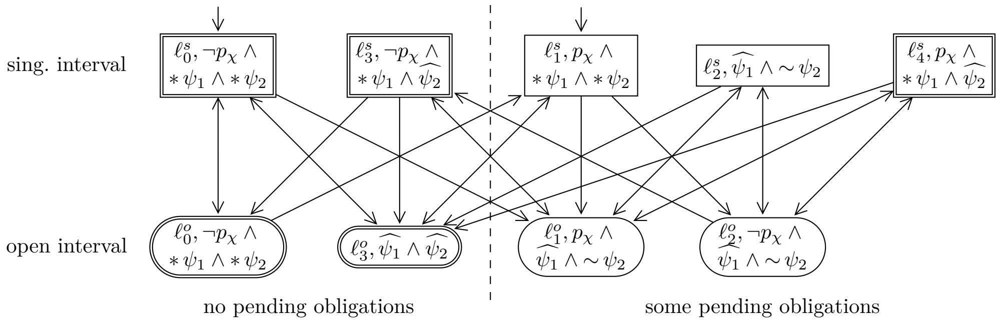

sing. interval
open interval
$\ell_0^s, \neg p_\chi \wedge * \psi_1 \wedge * \psi_2$
$\ell_3^s, \neg p_\chi \wedge * \psi_1 \wedge \widehat{\psi_2}$
$\ell_1^s, p_\chi \wedge * \psi_1 \wedge * \psi_2$
$\ell_2^s, \widehat{\psi_1} \wedge \sim \psi_2$
$\ell_4^s, p_\chi \wedge * \psi_1 \wedge \widehat{\psi_2}$
$\ell_0^o, \neg p_\chi \wedge * \psi_1 \wedge * \psi_2$
$\ell_3^o, \widehat{\psi_1} \wedge \widehat{\psi_2}$
$\ell_1^o, p_\chi \wedge \widehat{\psi_1} \wedge \sim \psi_2$
$\ell_2^o, \neg p_\chi \wedge \widehat{\psi_1} \wedge \sim \psi_2$
no pending obligations
some pending obligations

Figure 1 The component SA $\mathcal{C}_\chi$ for $\chi = \psi_1 \mathbf{U} \psi_2$.

is obtained by applying Proposition 3 to $\zeta(t'') \setminus \mathcal{P}_{\psi_2}$ and $\overline{\psi_1}$; $\gamma'(t')$ is obtained by applying Proposition 3 to $\zeta(t') \setminus \mathcal{P}_{\psi_1}$ and $\overline{\psi_2}$. Similar arguments can be made for $\chi = \psi_1 \mathbf{R} \psi_2$. The other implication is simpler. ◀

▶ **Corollary 6.** *For a signal γ over 2^{AP}, we have $\gamma, 0 \models \varphi$ if and only if there exists a signal γ' over $2^{\mathsf{AP} \cup \mathsf{AP}_\Phi}$ extending γ such that $\gamma', 0 \models \widehat{\varphi}$, and for all $\chi \in \Phi$ and $t \in \mathbb{R}_{\geq 0}$, $\gamma', t \models p_\chi \Rightarrow \mathsf{Expand}_\chi$ (where Expand_χ is one of the formulae in Proposition 5).*

The components. We are now ready to present the components $\mathcal{C}_\chi$ for $\chi \in \Phi$.

The component $\mathcal{C}_\chi$ for $\chi = \psi_1 \mathbf{U} \psi_2$ is given in Figure 1. We now explain how it has been produced. Thanks to Proposition 2, we provide a bipartite SA (in particular, we will read timed sequences with bipartite interval sequences only), where 'singular' locations are on top, and 'open' locations at the bottom. First, we focus on locations ℓ_0^s and ℓ_0^o, that are used as long as trigger p_χ is not pulled: then, there is no need to pull any trigger of ψ_1 nor ψ_2, which is ensured via the use of formula $* \psi_1 \wedge * \psi_2$. Consider then the first time when trigger p_χ is pulled (by another component automaton): it is either in a singular interval in which case we jump into location ℓ_1^s (this creates a pending obligation, since such an 'until' with our strict semantics cannot be fulfilled right away in a singular interval: this means, in particular, that we do not need to pull any trigger for ψ_1 or ψ_2, thus checking $* \psi_1 \wedge * \psi_2$), or in an open interval in which case we jump either into location ℓ_1^o if ψ_2 does not hold (i.e. if $\sim \psi_2$ holds), or into location ℓ_3^o if ψ_2 holds (i.e. if $\widehat{\psi_2}$ is in the guard) which fulfils right away the new obligation (notice that, in the figure, we did not put p_χ in the guard of this location, for simplification: we will discuss this point more in detail afterwards).

When p_χ is first pulled in an open interval (which means we jump into location ℓ_1^o or ℓ_3^o), by the semantics of the 'until' operator, ψ_1 must also hold in that interval. When in ℓ_3^o, the successors are the same as in ℓ_0^o. When in ℓ_1^o with a pending obligation, there are two cases for the next jump:

- either ψ_2 holds in the next singular interval, and then no trigger of ψ_1 needs to be pulled (i.e. guard $* \psi_1 \wedge \widehat{\psi_2}$): if there are no new pulled trigger p_χ, we jump into location ℓ_3^s; otherwise, we jump into location ℓ_4^s where we still have a new pending obligation, but the location is still made accepting to record the fact that the previous obligation has been fulfilled.
- or ψ_2 does not hold, in which case ψ_1 should hold (i.e. guard $\widehat{\psi_1} \wedge \sim \psi_2$): we then jump into location ℓ_2^s whether or not a new trigger p_χ is pulled.

When p_χ is first pulled in a singular interval (which means we jump into location ℓ_1^s), there is no need to pull any trigger of ψ_1 nor ψ_2. Then, while in one of the 'singular' locations ℓ_1^s, ℓ_2^s or ℓ_4^s, with a pending obligation, in the next jump, there are two cases:

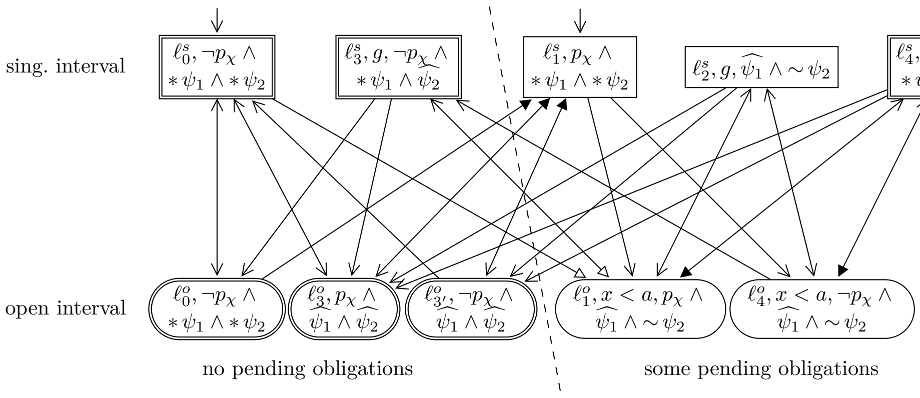

sing. interval
$\ell_0^s, \neg p_\chi \wedge * \psi_1 \wedge * \psi_2$
$\ell_3^s, g, \neg p_\chi \wedge * \psi_1 \wedge \widehat{\psi_2}$
$\ell_1^s, p_\chi \wedge * \psi_1 \wedge * \psi_2$
$\ell_2^s, g, \widehat{\psi_1} \wedge \sim \psi_2$
$\ell_4^s, g, p_\chi \wedge * \psi_1 \wedge \widehat{\psi_2}$
open interval
$\ell_0^o, \neg p_\chi \wedge * \psi_1 \wedge * \psi_2$
$\ell_3^o, p_\chi \wedge \widehat{\psi_1} \wedge \widehat{\psi_2}$
$\ell_{3'}^o, \neg p_\chi \wedge \widehat{\psi_1} \wedge \widehat{\psi_2}$
$\ell_1^o, x < a, p_\chi \wedge \widehat{\psi_1} \wedge \sim \psi_2$
$\ell_4^o, x < a, \neg p_\chi \wedge \widehat{\psi_1} \wedge \sim \psi_2$
no pending obligations
some pending obligations

Figure 2 The component SA $\mathcal{C}_\chi$ for $\psi_1 \, \mathbf{U}_{(0,a)} \, \psi_2$. We use a Boolean variable `si` to signify whether the oldest pending obligation has been pulled in a singular interval or not. The transitions with ▶ or ▷ reset x; the ones with ▶ (resp. ▷) set `si` to true (resp. false). The clock constraint g is defined as $(\mathtt{si} \wedge x < a) \vee (\neg\mathtt{si} \wedge x \leq a)$.

- either ψ_2 holds in the next open interval, in which case ψ_1 should still hold (because of the semantics of the 'until' operator): we can jump into the previously introduced location ℓ_3^o.
- or ψ_2 does not hold (then, ψ_1 should hold anyway) and we jump either in location ℓ_1^o if a new trigger p_χ is pulled, or in ℓ_2^o is no new trigger p_χ is pulled. Location ℓ_2^o has the same successors as ℓ_1^o but we still need to distinguish them since ℓ_1^o must check that a new pending obligation is pulled.

Initially, we do not want to pull any trigger of ψ_1 or ψ_2, therefore, ℓ_0^s and ℓ_1^s are the two initial locations, depending on whether trigger p_χ is initially pulled or not. Accepting locations are the one where either there are no more pending obligations, or a pending obligation has been fulfilled while a new trigger is being pulled (location ℓ_4^s).

Notice that, thanks to the use of $* \psi_i$ and $\widehat{\psi_i}$ formulae, only the necessary triggers in $\mathcal{P}_{\psi_1} \cup \mathcal{P}_{\psi_2}$ are pulled during an execution of this component. Indeed, this is not true for location ℓ_3^o: when going from locations ℓ_0^s or ℓ_3^s, to pull only minimal sets of triggers, we must make sure to go in ℓ_3^o only when a new trigger p_χ is pulled. This requires to split this location into two (one where p_χ holds, the other where it does not). For simplicity, we did not do it in the figure, but we apply this splitting in the next component we present.

This next component $\mathcal{C}_\chi$ is the one for $\chi = \psi_1 \, \mathbf{U}_{(0,a)} \, \psi_2$ (Figure 2), that is obtained by adding a clock x and suitable clock constraints. Intuitively, it suffices to use only one clock because for $I = (0, a)$, all new obligations are implied by the oldest pending obligation. This means that the clock should be reset when entering in a location where a trigger is pulled while all the previous obligations have been fulfilled: this is a priori the case when entering in locations ℓ_1^s, ℓ_1^o, and ℓ_4^s from locations $\{\ell_0^s, \ell_3^s, \ell_0^o, \ell_3^o, \ell_{3'}^o\}$. Now, the valuation of x would fix a deadline for the satisfaction of ψ_2. Indeed, as long as ψ_2 does not hold, we must check that $x < a$. When ψ_2 is next fulfilled, we also check that $x < a$. However, this is not correct for two reasons.

First, when checking the requirements $x < a$, this is not correct if the oldest pending obligations appeared in an open interval: indeed, it is still correct to fulfil ψ_2 in a singular interval where $x = a$. This requires that we register, when resetting clock x, if the trigger is pulled in a singular interval or not. To ease the presentation, we use a Boolean variable si to record that the trigger has been pulled in a singular interval. Pictorially, we use transitions with ▶ heads to reset the clock x and setting si to true, while transitions with ▷ heads reset clock x and set si to false. Then, the clock constraint that must be checked in singular interval (whether or not ψ_2 is currently fulfilled) is not $x < a$ but g defined by $(\texttt{si} \wedge x < a) \vee (\neg\texttt{si} \wedge x \leq a)$: in particular, the guard g in location ℓ_2^s models the fact that if the oldest obligation has been triggered in an open interval (si is false), it is not a contradiction to not yet fulfil ψ_2 at time $x = a$, but then, the only fireable transitions are the one towards ℓ_3^o and $\ell_{3'}^o$ where ψ_2 then holds. This also explains why guard g does not need to be checked when entering in ℓ_3^o and $\ell_{3'}^o$.

Second, this cannot be done as such when entering location ℓ_4^s since the guard g must be checked *before* resetting clock x that records the deadline of the next pending obligation. Indeed, we simply delay the reset and modification of variable si to the next transition towards ℓ_1^o or ℓ_4^o.

The component for $\psi_1 \mathbf{U}_{(0,a]} \psi_2$ is similar and hence omitted. The components for 'release' operators follow the same pattern as the ones for 'until'. Due to lack of space, we present them in Appendix B. Then:

▶ **Proposition 7.** *For each $\chi \in \Phi$, the component $\mathcal{C}_\chi$ accepts exactly all signals γ over $2^{\mathsf{AP} \cup \mathsf{AP}_\Phi}$ such that $\gamma, t \models p_\chi \Rightarrow \mathsf{Expand}_\chi$ for all $t \in \mathbb{R}_{\geq 0}$ (where Expand_χ is one of the formulae in Proposition 5).*

Finally, we need a simple initial component $\mathcal{C}_{init}$ which enforces $\widehat{\varphi}$ at $t = 0$ and $*\varphi$ at all $t > 0$, as suggested by Proposition 5. We can now state the main theorem of this section.

▶ **Theorem 8.** $\mathsf{proj}_{\mathsf{AP}}\left(\mathcal{S}(\mathcal{C}_{init} \times \prod_{\chi \in \Phi} \mathcal{C}_\chi)\right) = \mathcal{S}(\varphi)$.

4 From signal automata to timed automata

In this section, we provide a new approach to check the emptiness of *signal* automata that can be implemented by relying on existing tools for *timed* automata. To this end, we explain how to encode an SA $\mathcal{A}$ into a timed automaton $\mathcal{B}^{\mathcal{A}}$ that accepts exactly the 'timed words' counterparts of the signals accepted by $\mathcal{A}$. Moreover, the construction can be used in a compositional manner: if $\mathcal{A}$ is the product of a number of component SAs, $\mathcal{B}^{\mathcal{A}}$ can be obtained as the product of the TAs that result from applying the construction to the components of $\mathcal{A}$. As the construction is *emptiness-preserving*, it can serve as a bridge between the MITL-to-SA translation in the previous section and existing TA-based tools. We start by recalling the formal definition of timed automata.

Timed words and timed automata. A *time sequence* is an infinite sequence $\overline{\tau} = \tau_0\tau_1\tau_2\ldots$ of non-negative reals (called *timestamps*) such that (1) $\tau_0 = 0$; (2) for all $i \geq 0$, $\tau_i \leq \tau_{i+1}$; (3) for all $t \in \mathbb{R}_{\geq 0}$, there is some $i \geq 0$ such that $\tau_i > t$. A *timed word* $\rho = (\overline{\sigma}, \overline{\tau})$ over 2^{AP} is a pair of a state sequence $\overline{\sigma}$ over 2^{AP} and a time sequence $\overline{\tau}$. Alternatively, we may see ρ as an infinite sequence $(\sigma_0, \tau_0)(\sigma_1, \tau_1)(\sigma_2, \tau_2)\cdots$ of *events* (σ_i, τ_i). We now define timed automata, with generalised acceptance conditions as before (used by [27] in the untimed setting).

▶ **Definition 9.** A *timed automaton* (TA) over 2^{AP} is a tuple $\mathcal{A} = (L, L_0, X, \Delta, \mathcal{F})$ where

- L is a finite set of locations;
- $L_0 \subseteq L$ is the set of initial locations;
- X is a finite set of clocks;

- $\Delta \subseteq L \times 2^{2^{\mathsf{AP}}} \times \mathcal{G}(X) \times 2^X \times L$ is the set of transitions;
- $\mathcal{F} \subseteq 2^L$ is the family of sets of accepting locations.

A *run* π of $\mathcal{A}$ on a timed word $\rho = (\sigma_0, \tau_0)(\sigma_1, \tau_1)(\sigma_2, \tau_2) \cdots$ over 2^{AP} is an infinite sequence

$$(\ell_0, v_0) \xrightarrow[\sigma_0, d_0]{\lambda_1} (\ell_1, v_1) \xrightarrow[\sigma_1, d_1]{\lambda_2} (\ell_2, v_2) \xrightarrow[\sigma_2, d_2]{\lambda_3} \cdots$$

where, for all $i \geq 0$: (1) ℓ_i is a location of $\mathcal{A}$; (2) v_i is a valuation of X; (3) $d_i = \tau_i - \tau_{i-1}$ (assuming $\tau_{-1} = 0$) (4) $\lambda_i \subseteq X$; and that satisfies the following:

- **Initiality:** $\ell_0 \in L_0$; and
- **Consecution:** for all $i \geq 0$: $(\ell_i, \phi, g, \lambda_{i+1}, \ell_{i+1}) \in \Delta$ with $\sigma_i \in \phi$ and $v_i + d_i \models g$; and
- **Timing:** for all $i \geq 0$, $v_{i+1} = (v_i + d_i)[\lambda_{i+1} \leftarrow 0]$.

We say that π is *accepting* if for all accepting sets $F \in \mathcal{F}$, the set $\{i \mid \ell_i \in F\}$ is infinite. A timed word ρ is *accepted* by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on ρ. We write $\mathcal{L}(\mathcal{A})$ for the set of timed words accepted by $\mathcal{A}$. For two TAs $\mathcal{A}_1$ and $\mathcal{A}_2$, we denote by $\mathcal{A}_1 \times \mathcal{A}_2$ their (synchronous) product [2]. In particular, we have $\mathcal{L}(\mathcal{A}_1 \times \mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

Translation from SA to TA. We first explain how we map signals to timed words. To do so, we select a bipartite state sequence κ corresponding to γ, and we express the state changes along κ in a timed word. Formally, for a signal γ and a timed state sequence $\kappa = (\sigma_0, I_0)(\sigma_1, I_1) \cdots$ s.t. $\kappa \in [\![\gamma]\!]^{bp}$ (i.e., I_i is singular for all even $i \geq 0$), we define:

$$[\kappa]_{tw} = (\sigma_0, \sup(I_0))(\sigma_1, \inf(I_1))(\sigma_2, \sup(I_2))(\sigma_3, \inf(I_3)) \cdots .$$

Note that we represent a state change at time t by two events with timestamp t (note that $\sup(I_i) = \inf(I_{i+1})$ for each even $i \geq 0$). Abusing notations, we write $[\gamma]_{tw} = \{[\kappa]_{tw} \mid \kappa \in [\![\gamma]\!]^{bp}\}$ and $[\mathcal{S}]_{tw} = \bigcup_{\gamma \in \mathcal{S}} [\gamma]_{tw}$ for a set $\mathcal{S}$ of signals.

▶ **Proposition 10.** *Given a (bipartite)* SA *$\mathcal{A}$, we can construct a* TA *$\mathcal{B}^{\mathcal{A}}$ such that $\mathcal{L}(\mathcal{B}^{\mathcal{A}}) = [\mathcal{S}(\mathcal{A})]_{tw}$. In particular, if $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ then $\mathcal{L}(\mathcal{B}^{\mathcal{A}_1} \times \cdots \times \mathcal{B}^{\mathcal{A}_n}) = [\mathcal{S}(\mathcal{A})]_{tw}$.*

Proof (Sketch). For a clock constraint $g \in \mathcal{G}(X)$, let $g^{\leftarrow}$ be the clock constraint obtained from g by replacing all clauses of the form '$x \leq c$' with '$x < c$' and all '$x > c$' with '$x \geq c$'. Likewise, let $g^{\rightarrow}$ be the clock constraint obtained from g by replacing all '$x < c$' with '$x \leq c$' and all '$x \geq c$' with '$x > c$'. The following statements hold (for a valuation v of X):

- $v \models g^{\leftarrow}$ if and only if for some $\delta \in \mathbb{R}_{>0}$, we have $v + t \models g$ for all $t \in (0, \delta]$.
- $v \models g^{\rightarrow}$ if and only if for some $\delta \in \mathbb{R}_{>0}$, we have $v' \models g$ for all valuations v' of X such that $v' + t = v$ for some $t \in (0, \delta]$.

In what follows, we write $g[\lambda \leftarrow 0]$ for the clock constraint obtained from g by replacing all occurrences of clocks $x \in \lambda$ with 0. For $\mathcal{A} = (L, L_0, \alpha, X, \beta, \Delta, \mathcal{F})$ (which by assumption is bipartite and $L = L^s \uplus L^o$), define $\mathcal{B} = (L^{\mathcal{A}}, L_0^{\mathcal{A}}, X^{\mathcal{A}}, \Delta^{\mathcal{A}}, \mathcal{F}^{\mathcal{A}})$ where

- $L^{\mathcal{A}} = \{\ell^s \mid \ell \in L^s\} \cup \{\dot{\ell}^s, \ell^o, \dot{\ell}^o \mid \ell \in L^o\} \cup \{\ell^{init}\}$;
- $L_0^{\mathcal{A}} = \{\ell^{init}\}$;
- $X^{\mathcal{A}} = X \cup \{y\}$ where y is a fresh clock;
- $$\begin{aligned}\Delta^{\mathcal{A}} = \quad & \{(\ell^{init}, \alpha(\ell), \beta(\ell) \wedge y = 0, \emptyset, \ell^s) \mid \ell \in L_0\} \\ & \cup \{(\ell_1^s, \alpha(\ell_2), \beta(\ell_2)^{\leftarrow}[\lambda \leftarrow 0] \wedge y = 0, \lambda, \ell_2^o) \mid (\ell_1, \lambda, \ell_2) \in \Delta\} \\ & \cup \{(\ell_1^o, \alpha(\ell_2), \beta(\ell_1)^{\rightarrow} \wedge \beta(\ell_2)[\lambda \leftarrow 0] \wedge y > 0, \lambda \cup \{y\}, \ell_2^s) \mid (\ell_1, \lambda, \ell_2) \in \Delta\} \\ & \cup \{(\ell^o, \alpha(\ell), \beta(\ell)^{\rightarrow} \wedge \beta(\ell)[\lambda \leftarrow 0] \wedge y > 0, \lambda \cup \{y\}, \dot{\ell}^s) \mid \ell \in L^o\} \\ & \cup \{(\dot{\ell}^s, \alpha(\ell), \beta(\ell)^{\leftarrow}[\lambda \leftarrow 0] \wedge y = 0, \lambda, \dot{\ell}^o) \mid \ell \in L^o\} \\ & \cup \{(\dot{\ell}^o, \alpha(\ell), \beta(\ell)^{\rightarrow} \wedge \beta(\ell)[\lambda \leftarrow 0] \wedge y > 0, \lambda \cup \{y\}, \dot{\ell}^s) \mid \ell \in L^o\} \\ & \cup \{(\dot{\ell}_1^o, \alpha(\ell_2), \beta(\ell_1)^{\rightarrow} \wedge \beta(\ell_2)[\lambda \leftarrow 0] \wedge y > 0, \lambda \cup \{y\}, \ell_2^s) \mid (\ell_1, \lambda, \ell_2) \in \Delta\}\end{aligned}$$
- $\mathcal{F}^{\mathcal{A}} = \{\{\ell^s \mid \ell \in L^s \cap F\} \cup \{\ell^o \mid \ell \in L^o \cap F\} \mid F \in \mathcal{F}\}$.

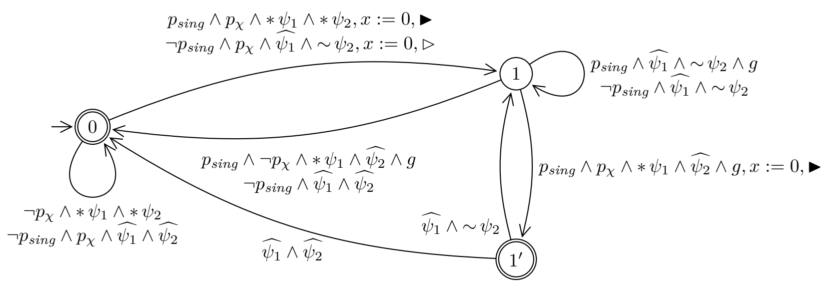

$p_{sing} \land p_\chi \land *\psi_1 \land *\psi_2, x := 0, \blacktriangleright$
$\neg p_{sing} \land p_\chi \land \widehat{\psi_1} \land \sim\psi_2, x := 0, \triangleright$
$p_{sing} \land \widehat{\psi_1} \land \sim\psi_2 \land g$
$\neg p_{sing} \land \widehat{\psi_1} \land \sim\psi_2$
$p_{sing} \land \neg p_\chi \land *\psi_1 \land \widehat{\psi_2} \land g$
$\neg p_{sing} \land \widehat{\psi_1} \land \widehat{\psi_2}$
$p_{sing} \land p_\chi \land *\psi_1 \land \widehat{\psi_2} \land g, x := 0, \blacktriangleright$
$\neg p_\chi \land *\psi_1 \land *\psi_2$
$\neg p_{sing} \land p_\chi \land \widehat{\psi_1} \land \widehat{\psi_2}$
$\widehat{\psi_1} \land \sim\psi_2$
$\widehat{\psi_1} \land \widehat{\psi_2}$
0
1
1′

Figure 3 The component TA $\mathcal{B}_\chi$ for $\chi = \varphi_1 \mathbf{U}_{(0,a)} \varphi_2$. We use a Boolean variable $\mathtt{si}$ to signify whether the current p_χ-interval is left-closed. The transitions with ▶ (respectively, ▷) set $\mathtt{si}$ to true (respectively, false). The clock constraint g is defined as $(\mathtt{si} \land x < a) \lor (\neg\mathtt{si} \land x \leq a)$.

Intuitively, the 'dotted' locations $\dot{\ell}^s$, $\dot{\ell}^o$ are used to allow interleaving and stuttering as $\mathcal{A}$ stays in $\ell \in L^o$: this is crucial to make the asynchronous product $\mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ and the synchronous product $\mathcal{B}^{\mathcal{A}_1} \times \cdots \times \mathcal{B}^{\mathcal{A}_n}$ match. Finally, for pragmatic reasons, we make suitable modifications to $\mathcal{B}$ to obtain a *strongly non-Zeno* TA $\mathcal{B}^{\mathcal{A}}$ (i.e. a TA in which time progresses), as in [22]. ◀

The proposition above works for any (bipartite) SA. For $\mathcal{C}_{init}$ or each component $\mathcal{C}_\chi$ ($\chi \in \Phi$) in the previous section, however, we can suppress all the 'dotted' locations $\dot{\ell}^s$, $\dot{\ell}^o$ and build a much simpler TA (which we denote by $\mathcal{B}_{init}$ or $\mathcal{B}_\chi$, respectively).[5] Our main result can then be stated as the following theorem, where the projection operator proj is defined in a similar way as in the setting of signals.

▶ **Theorem 11.** $\mathsf{proj}_{\mathsf{AP}}\left(\mathcal{L}(\mathcal{B}^{\mathcal{A}} \times \mathcal{B}_{init} \times \prod_{\chi\in\Phi} \mathcal{B}_\chi)\right) = \mathsf{proj}_{\mathsf{AP}}\left(\left[\mathcal{S}(\mathcal{A} \times \mathcal{C}_{init} \times \prod_{\chi\in\Phi} \mathcal{C}_\chi)\right]_{tw}\right)$ *for any given* SA $\mathcal{A}$ *over* $2^{\mathsf{AP}\cup\mathsf{AP}_\Phi}$ *whose propositional constraints can be written as Boolean combinations over* AP *(i.e. do not involve atomic propositions in* AP_Φ*).*

As an example, the component TA $\mathcal{B}_\chi$ for $\chi = \psi_1 \mathbf{U}_{(0,a)} \psi_2$ (in which we use a new atomic proposition p_{sing} that holds on 'singular' transitions) is depicted in Figure 3.

5 Implementation and experiments

We have implemented the translation as an extension of our tool MightyL [14]. Given a formula φ over AP in MITL,[6] the tool generates the model TA $\mathcal{B}^{\mathcal{A}}$ where $\mathcal{A}$ is a universal SA over $2^{\mathsf{AP}\cup\mathsf{AP}_\Phi}$, the initial component TA $\mathcal{B}_{init}$, and the corresponding component TAs $\mathcal{B}_\chi$ for each temporal subformula χ of φ in the Uppaal xml format. The user can, of course, replace $\mathcal{B}^{\mathcal{A}}$ with the model TA $\mathcal{M}$ of their choice and perform model-checking with existing TA-based tools.[7] Our implementation is publicly available and can be executed directly on the webpage: `http://www.ulb.ac.be/di/verif/mightyl`. In the following experiments,

[5] Alternatively, one may translate $\neg\varphi$ into a 'pointwise' formula and invoke the approach in [14] directly; the number of temporal subformulae (and hence the number of clocks) would be doubled, however.

[6] More precisely, our tool accepts all temporal operators that are labelled with intervals of the form $(0,\infty)$, $[0,\infty)$, $(0,a)$, $[0,a)$, $(0,a]$ or $[0,a]$. If 0 is included in the interval, the temporal operator is given a *weak-future* interpretation [39], e.g., $\psi_1 \mathbf{U}^w_{[0,a)} \psi_2 \iff \psi_2 \lor (\psi_1 \land \psi_1 \mathbf{U}_{(0,a)} \psi_2)$. Remember that general MITL formulae can be rewritten into formulae of this fragment, e.g., $\mathbf{F}_{(a,\infty)}\psi \iff \mathbf{G}_{(0,a]}\mathbf{F}\psi$.

[7] We require $\mathcal{M}$ to be strongly non-Zeno and $\mathcal{L}(\mathcal{M}) = [\mathcal{S}(\mathcal{A})]_{tw}$ where $\mathcal{A}$ is an SA over $2^{\mathsf{AP}\cup\mathsf{AP}_\Phi}$ that satisfies the conditions in Theorem 11.

Table 1 Execution times for the 'parametric formulae' benchmark set. The columns 'Pointwise' correspond to the approach of [14] and the columns 'Continuous' correspond to the approach of this article (where OOM stands for out-of-memory). The three numbers of each entry correspond to the time taken by OPAAL to translate UPPAAL XML into C++, the time taken by the **g++** compiler, and the actual model-checking time taken by LTSMIN, respectively.

Formula	Continuous	Pointwise	Formula	Continuous	Pointwise
$F(5,[0,\infty))$	0.41s/1.03s/0.16s	0.36s/1.07s/0.16s	$G(5,[0,\infty))$	0.45s/1.04s/0.32s	0.35s/1.02s/0.28s
$F(10,[0,\infty))$	0.74s/1.31s/0.35s	0.62s/1.23s/0.32s	$G(10,[0,\infty))$	0.77s/1.29s/43.46s	0.63s/1.20s/39.31s
$F(5,[0,5])$	0.61s/1.16s/0.18s	0.40s/1.04s/0.13s	$G(5,[0,5])$	1.11s/1.43s/0.45s	0.52s/1.10s/0.29s
$F(10,[0,5])$	1.18s/1.53s/0.43s	0.72s/1.29s/8.26s	$G(10,[0,5])$	2.16s/2.06s/98.76s	0.91s/1.44s/17.71s
$F(2,(5,\infty))$	0.66s/1.17s/0.18s	0.22s/0.89s/0.04s	$G(2,(5,\infty))$	0.44s/1.07s/0.12s	0.20s/0.88s/0.07s
$F(5,(5,\infty))$	1.48s/1.73s/3.02s	0.36s/1.01s/0.15s	$G(5,(5,\infty))$	0.93s/1.41s/2.77s	0.34s/1.03s/0.27s
$F(10,(5,\infty))$	OOM	0.63s/1.21s/0.31s	$G(10,(5,\infty))$	OOM	0.60s/1.20s/13.30s
$U(5,[0,\infty))$	0.36s/0.98s/0.19s	0.32s/1.01s/0.06s	$R(5,[0,\infty))$	0.38s/1.00s/0.26s	0.30s/0.97s/0.23s
$U(10,[0,\infty))$	0.67s/1.23s/0.25s	0.57s/1.19s/0.26s	$R(10,[0,\infty))$	0.70s/1.26s/8.12s	0.56s/1.18s/9.74s
$U(5,[0,5])$	0.54s/1.08s/0.08s	0.35s/0.99s/0.08s	$R(5,[0,5])$	0.93s/1.32s/0.30s	0.41s/1.01s/0.24s
$U(10,[0,5])$	1.08s/1.46s/0.40s	0.65s/1.24s/2.96s	$R(10,[0,5])$	1.96s/1.93s/3.59s	0.81s/1.32s/19.21s
$U(2,(5,\infty))$	0.41s/1.02s/0.13s	0.18s/0.85s/0.03s	$R(2,(5,\infty))$	0.31s/0.92s/0.05s	0.17s/0.86s/0.03s
$U(5,(5,\infty))$	1.40s/1.63s/12.17s	0.33s/0.97s/0.10s	$R(5,(5,\infty))$	0.93s/1.36s/0.37s	0.32s/0.97s/0.23s
$U(10,(5,\infty))$	OOM	0.59s/1.17s/0.29s	$R(10,(5,\infty))$	OOM	0.53s/1.17s/18.99s

we use LTSMIN [30] (with OPAAL [34], which enables support for UPPAAL XML files) as the back-end model checker and report its execution times (using only a single core) on a Pentium B970 (2.3GHz) machine with 6GB RAM running Ubuntu 17.04. We omit the execution times for MIGHTYL as it is less than 0.1s on all our benchmarks.

Satisfiability of parametric formulae. We consider the satisfiability of a set of parametric MITL formulae modified from [24, 13]. The goal of this benchmark set is to give a rough comparison between the performance of our approach in the *pointwise* semantics (the original aim of MIGHTYL; we refer the reader to [39, 14] for more details) with that in the continuous semantics (this article). For $k \geq 2$ and an interval I, let:

$$
\begin{aligned}
&F(k, I) = \textstyle\bigwedge_{i=1}^{k} \mathbf{F}_I^w p_i,\\
&G(k, I) = \textstyle\bigwedge_{i=1}^{k} \mathbf{G}_I^w p_i,\\
&U(k, I) = (\ldots (p_1 \mathbin{\mathbf{U}_I^w} p_2) \mathbin{\mathbf{U}_I^w} \ldots) \mathbin{\mathbf{U}_I^w} p_k,\\
&R(k, I) = (\ldots (p_1 \mathbin{\mathbf{R}_I^w} p_2) \mathbin{\mathbf{R}_I^w} \ldots) \mathbin{\mathbf{R}_I^w} p_k,
\end{aligned}
$$

where $\mathbf{F}_I^w$, $\mathbf{G}_I^w$, etc., are *weak-future* temporal operators [39]. The formulae in the benchmark set are given in Table 1. For the pointwise case, these are the actual formulae that we pass to MIGHTYL; for the continuous case, standard rewriting rules are applied to handle the lower-bound temporal operators (e.g. $\mathbf{F}_{(5,\infty)}p \iff \mathbf{G}_{(0,5]}\mathbf{F}p$).[8] From the execution times in Table 1, it is evident that OPAAL and **g++** are not performance bottlenecks. For smaller formulae, the times taken by LTSMIN are very short. For larger formulae, however, as LTSMIN uses depth-first search for OPAAL-generated models, it sometimes goes very deep into the state space and results in out-of-memory.

[8] Of course, the resulting formulae are interpreted over signals, in contrast to their pointwise counterparts; but we expect the computational efforts needed to check their satisfiability to be similar.

Table 2 Execution times for the satisfiability, validity and redundancy checks in [19].

Formula	Our approach	Our approach w/o minimality	[19]
$\phi_1 = \mathbf{F}_{[0,30]} p_1 \wedge \mathbf{F}_{[0,20]} p_1$	6.06s	7.50s	14s
$\phi_2 = \mathbf{F}_{[0,30]} (p_1 \Rightarrow \mathbf{G}_{[0,20]} p_1)$	3.08s	4.36s	7s
$\phi_4 = \mathbf{G}_{[0,40]} p_1 \wedge \mathbf{G}_{[0,40]} \mathbf{F}_{[0,10]} p_1$	7.10s	38.15s	29s
$\phi_5 = \mathbf{F}_{[0,40]} (p_1 \vee p_3) \wedge \mathbf{F}_{[0,40]} p_2 \wedge \mathbf{F}_{[0,40]} \mathbf{G}_{[0,30]} p_1$	12.04s	>1200s	126s

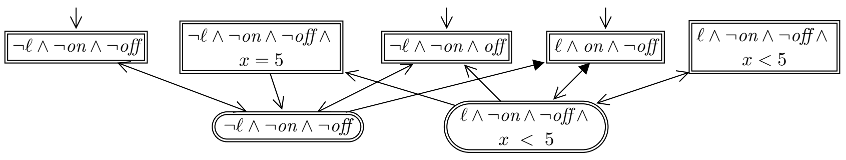


Figure 4 The SA $\mathcal{A}_{lamp}$. The transitions with solid tips reset clock x.

Validity and redundancy of specifications. We say that an MITL formula φ is *valid* if $\neg\varphi$ is not satisfiable. If φ is of the form $\bigwedge_{1 \leq i \leq k} \varphi_i$, we say that the conjunct φ_i is *redundant* in φ if the formula $(\bigwedge_{\substack{1 \leq j \leq k \\ j \neq i}} \varphi_j) \Rightarrow \varphi_i$ is valid. In [19], MITL specifications created by non-expert users are checked for satisfiability, validity and redundancy. We report the execution times of our approach on some of their checks in Table 2. To see the effect of forcing minimal triggers, we also give the execution times when this is not imposed. We also reproduce the execution times reported in [19] in the table; since we do not impose *a priori* bounds on state changes (as opposed to [19]) and we use a much less powerful CPU, these numbers are not meant for direct comparison but rather for reference.

Model-checking a timed lamp. We consider a case study of a timed lamp from [9]. The lamp is controlled by two buttons 'on' and 'off', which can only be pressed instantaneously but not simultaneously. The buttons turn the lamp on and off as expected, and the lamp turns off automatically 5 time units after the last time 'on' was pressed. In [9], the system is given as an MITL formula (with past temporal operators) over atomic propositions $\{\ell, on, off\}$. While we can make use of projections to remove the past temporal operators [44, 28], it turned out that the resulting formula is too large. For this reason, we model the system directly as an SA $\mathcal{A}_{lamp}$ (Figure 4). Then, via Proposition 10 and Theorem 11, we perform the same verification tasks as [9]: (1) checking the emptiness of $\mathcal{A}_{lamp}$; (2) model-checking $\mathcal{A}_{lamp}$ against $\varphi_1 = \mathbf{G}_{[0,\infty)} \big(\mathbf{F}_{[0,5]}(\neg\ell)\big)$, i.e. the lamp never stays lit for more than 5 time units; (3) model-checking $\mathcal{A}_{lamp}$ against $\varphi_2 = \mathbf{F}_{[0,\infty)} \big(\mathbf{G}_{[0,5]}\ell\big) \Rightarrow \mathbf{F}_{[0,\infty)} \big(on \wedge \mathbf{F}_{(0,5]}\, on\big)$, i.e. if at some point the light stays on for more than 5 time units, then there is an instant when 'on' is pressed, and then it is pressed again before 5 time units. The execution times (with and without minimality criteria) are given in Table 3, where we also reproduce the execution times reported in [9]. Again, these numbers are not meant to be compared directly.

6 Conclusion and future work

We proposed a translation from MITL to signal automata based on the same principles as our previous work in the pointwise setting [14]. The main advantages of this translation over the existing ones are that it is compositional and integrates easily with existing tools. To the best of our knowledge, this is the first practical automata-based approach to MITL model-checking over signals. We plan to add to MightyL support for general MITL operators (via rewriting

Table 3 Execution times for the verification tasks in [9].

Task	Our approach	Our approach w/o minimality	[9]
$\mathcal{S}(\mathcal{A}_{lamp}) = \emptyset$?	1.17s	–	4.24s
$\mathcal{S}(\mathcal{A}_{lamp} \times \mathcal{A}_{\neg\varphi_1}) = \emptyset$?	1.73s	1.77s	17.2s
$\mathcal{S}(\mathcal{A}_{lamp} \times \mathcal{A}_{\neg\varphi_2}) = \emptyset$?	2.36s	13.18s	257.1s

or by components) and other temporal operators (such as those from ECL [28]). A possible future theoretical direction is to investigate whether the translation can be generalised (possibly with the techniques in [17] or [42]) to deal with signals that are not finitely-variable.

Acknowledgements. We thank the reviewers of this article for their helpful comments.

References

1 Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

2 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

3 Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

4 Rajeev Alur and Thomas A. Henzinger. Back to the future: Towards a theory of timed regular languages. In *33rd Annual Symposium on Foundations of Computer Science*, pages 177–186. IEEE, 1992.

5 Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993. doi:10.1006/inco.1993.1025.

6 Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–203, January 1994. doi:10.1145/174644.174651.

7 Eugene Asarin, Paul Caspi, and Oded Maler. A Kleene theorem for timed automata. In *LICS'97*, pages 160–171. IEEE Computer Society Press, 1997.

8 Ezio Bartocci, Luca Bortolussi, and Laura Nenzi. A temporal logic approach to modular design of synthetic biological circuits. In *CMSB'13*, volume 8130 of *LNCS*, pages 164–177. Springer, 2013. doi:10.1007/978-3-642-40708-6.

9 Marcello M. Bersani, Matteo Rossi, and Pierluigi San Pietro. A tool for deciding the satisfiability of continuous-time metric temporal logic. *Acta Informatica*, 53(2):171–206, 2016. doi:10.1007/s00236-015-0229-y.

10 Patricia Bouyer, Maximilien Colange, and Nicolas Markey. Symbolic optimal reachability in weighted timed automata. In *CAV'16*, volume 9779 of *LNCS*, pages 513–530. Springer, 2016. doi:10.1007/978-3-319-41528-4.

11 Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. KRONOS: A model-checking tool for real-time systems (tool-presentation). In *FTRTFT'98*, volume 1486 of *LNCS*, pages 298–302. Springer, 1998. doi:10.1007/BFb0055357.

12 Thomas Brihaye, Morgane Estiévenart, and Gilles Geeraerts. On MITL and alternating timed automata. In *FORMATS'13*, volume 8053 of *LNCS*, pages 47–61. Springer, 2013.

13 Thomas Brihaye, Morgane Estiévenart, and Gilles Geeraerts. On MITL and alternating timed automata of infinite words. In *FORMATS'14*, volume 8711 of *LNCS*. Springer, 2014.

14 Thomas Brihaye, Gilles Geeraerts, Hsi-Ming Ho, and Benjamin Monmege. MightyL: A compositional translation from MITL to timed automata. In *CAV'17*, LNCS. Springer, 2017. URL: https://hal.archives-ouvertes.fr/hal-01525524.

15 Peter E. Bulychev, Alexandre David, Kim G. Larsen, and Guangyuan Li. Efficient controller synthesis for a fragment of $\mathrm{MTL}_{0,\infty}$. *Acta Informatica*, 51(3-4):165–192, 2014. `doi:10.1007/s00236-013-0189-z`.

16 Koen Claessen, Niklas Een, and Baruch Sterin. A circuit approach to LTL model checking. In *FMCAD'13*. IEEE, 2013.

17 Julien Cristau. Automata and temporal logic over arbitrary linear time. In *FSTTCS'09*, volume 4 of *LIPIcs*, pages 133–144. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2009. `doi:10.4230/LIPIcs.FSTTCS.2009.2313`.

18 Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: introduction and applications. *Communications of the ACM*, 54(9):69–77, September 2011. `doi:10.1145/1995376.1995394`.

19 Adel Dokhanchi, Bardh Hoxha, and Georgios Fainekos. Formal requirement debugging for testing and verification of cyber-physical systems. Research Report 1607.02549, arXiv, 2016.

20 Deepak D'Souza and RM Matteplackel. A clock-optimal hierarchical monitoring automaton construction for mitl. Research Report 2013-1, IIS, 2013. URL: `http://www.csa.iisc.ernet.in/TR/2013/1/lics2013-tr.pdf`.

21 Deepak D'Souza, M Raj Mohan, and Pavithra Prabhakar. Eliminating past operators in metric temporal logic. *Perspectives in Concurrency*, pages 86–106, 2008.

22 A. Pnueli E. Asarin, O. Maler and J. Sifakis. Controller synthesis for timed automata. In *SSC'98*, pages 469–474. Elsevier, 1998.

23 Javier Esparza and Jan Kretínský. From LTL to deterministic automata: A safraless compositional approach. In *CAV'14*, volume 8559 of *LNCS*, pages 192–208. Springer, 2014.

24 Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In *CAV'01*, volume 2102 of *LNCS*, pages 53–65. Springer, 2001.

25 Marc Geilen. An improved on-the-fly tableau construction for a real-time temporal logic. In *CAV'03*, volume 2725 of *LNCS*, pages 394–406. Springer, 2003. `doi:10.1007/978-3-540-45069-6_37`.

26 Marc Geilen and Dennis Dams. An on-the-fly tableau construction for a real-time temporal logic. In *FTRTFT*, volume 1926 of *LNCS*, pages 276–290. Springer, 2000. `doi:10.1007/3-540-45352-0_23`.

27 Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *PSTV'95*, pages 3–18. Chapman & Hall, 1995.

28 Thomas A. Henzinger, Jean-François Raskin, and Pierre-Yves Schobbens. The regular real-time languages. In *ICALP'98*, volume 1443 of *LNCS*, pages 580–591. Springer, 1998. `doi:10.1007/BFb0055086`.

29 Yoram Hirshfeld and Alexander Moshe Rabinovich. Logics for real time: Decidability and complexity. *Fundamenta Informaticae*, 62(1):1–28, 2004.

30 Gijs Kant, Alfons Laarman, Jeroen Meijer, Jaco van de Pol, Stefan Blom, and Tom van Dijk. LTSmin: High-performance language-independent model checking. In *TACAS'15*, volume 9035 of *LNCS*, pages 692–707. Springer, 2015.

31 Roland Kindermann, Tommi A. Junttila, and Ilkka Niemelä. Bounded model checking of an MITL fragment for timed automata. In *ACSD'13*, pages 216–225. IEEE Computer Society, 2013.

32 Dileep Raghunath Kini, Shankara Narayanan Krishna, and Paritosh K. Pandya. On construction of safety signal automata for $MITL[\mathcal{U},\mathcal{S}]$ using temporal projections. In *FORMATS*, volume 6919 of *LNCS*, pages 225–239. Springer, 2011. `doi:10.1007/978-3-642-24310-3_16`.

33 Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

34 Alfons Laarman, Mads Chr. Olesen, Andreas Engelbredt Dalsgaard, Kim Guldstrand Larsen, and Jaco van de Pol. Multi-core emptiness checking of timed Büchi automata using inclusion abstraction. In *CAV'13*, volume 8044 of *LNCS*, pages 968–983. Springer, 2013.

35 Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.

36 Oded Maler, Dejan Nickovic, and Amir Pnueli. From MITL to timed automata. In *FORMATS'06*, volume 4202 of *LNCS*, pages 274–289. Springer, 2006.

37 Dejan Nickovic. *Checking Timed and Hybrid Properties: Theory and Applications. (Vérification de propriétés temporisées et hybrides: théorie et applications)*. PhD thesis, Joseph Fourier University, Grenoble, France, 2008.

38 Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In *LICS'05*, pages 188–197. IEEE Computer Society Press, 2005.

39 Joël Ouaknine and James Worrell. On the decidability and complexity of metric temporal logic over finite words. *Logical Methods in Computer Science*, 3(1), 2007.

40 Amir Pnueli and Aleksandr Zaks. On the merits of temporal testers. In *25 Years of Model Checking*, volume 5000 of *LNCS*, pages 172–195. Springer, 2008. `doi:10.1007/978-3-540-69850-0_11`.

41 Vasumathi Raman, Alexandre Donzé, Dorsa Sadigh, Richard M. Murray, and Sanjit A. Seshia. Reactive synthesis from signal temporal logic specifications. In *HSCC'15*, pages 239–248. ACM, 2015.

42 Mark Reynolds. The complexity of the temporal logic with "until" over general linear time. *Journal of Computer and System Sciences*, 66(2):393–426, 2003.

43 Kristin Y. Rozier and Moshe Y. Vardi. A multi-encoding approach for LTL symbolic satisfiability checking. In *FM'11*, volume 6664 of *LNCS*, pages 417–431. Springer, 2011. `doi:10.1007/978-3-642-21437-0`.

44 Thomas Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *FTRTFT'94*, volume 863 of *LNCS*, pages 694–715. Springer, 1994.

A Making signal automata bipartite

Proof of Proposition 2. For $\mathcal{A} = (L, L_0, \alpha, X, \beta, \Delta, \mathcal{F})$, we define a corresponding SA $\mathcal{A}^{bp} = (L^{bp}, L_0^{bp}, \alpha^{bp}, X^{bp}, \beta^{bp}, \Delta^{bp}, \mathcal{F}^{bp})$ where

- $L^{bp} = \{\ell^s, \dot{\ell}^s, \ell^o \mid \ell \in L\}$;
- $L_0^{bp} = \{\ell^s \mid \ell \in L_0\}$;
- $\alpha^{bp}(\ell^s) = \alpha^{bp}(\dot{\ell}^s) = \alpha^{bp}(\ell^o) = \alpha(\ell)$ for every $\ell \in L$;
- $X^{bp} = X \cup \{y\}$ where y is a fresh clock;
- $\beta^{bp}(\ell^s) = \beta^{bp}(\dot{\ell}^s) = \beta(\ell) \wedge y = 0$, $\beta^{bp}(\ell^o) = \beta(\ell)$ for every $\ell \in L$;
- $\Delta^{bp} = \{(\ell_1^s, \lambda, \ell_2^o), (\dot{\ell}_1^s, \lambda, \ell_2^o), (\ell_1^o, \lambda \cup \{y\}, \ell_2^s) \mid (\ell_1, \lambda, \ell_2) \in \Delta\}$ $\cup \{(\ell^o, \{y\}, \dot{\ell}^s), (\ell^s, \emptyset, \ell^o) \mid \ell \in L\}$;
- $\mathcal{F}^{bp} = \{\{\ell^s, \ell^o \mid \ell \in F\} \mid F \in \mathcal{F}\}$.

Intuitively, we create three copies ℓ^s, $\dot{\ell}^s$, ℓ^o of each location ℓ of $\mathcal{A}$ and use the clock y to enforce the desired behaviour. In particular, the 'dotted' locations $\dot{\ell}^s$ are used to deal with the situation where the 'source' interval is right-closed. One can verify that $\mathcal{A}^{bp}$ is bipartite (let $L^s = \{\ell^s, \dot{\ell}^s \mid \ell \in L\}$) and $\mathcal{S}(\mathcal{A}) = \mathcal{S}(\mathcal{A}^{bp})$. ◀

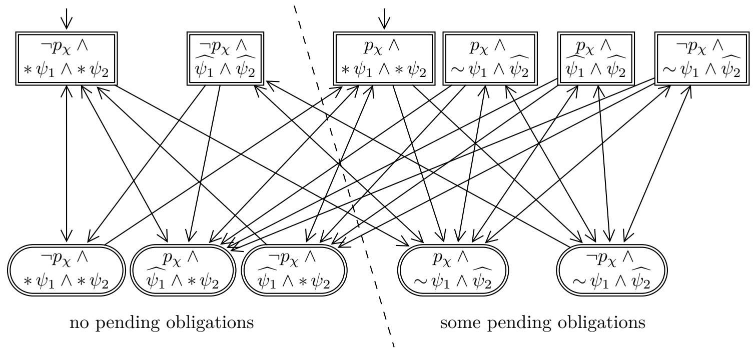

$\neg p_\chi \wedge$ $*\psi_1 \wedge *\psi_2$
$\neg p_\chi \wedge$ $\widehat{\psi_1} \wedge \widehat{\psi_2}$
$p_\chi \wedge$ $*\psi_1 \wedge *\psi_2$
$p_\chi \wedge$ $\sim\psi_1 \wedge \widehat{\psi_2}$
$p_\chi \wedge$ $\widehat{\psi_1} \wedge \widehat{\psi_2}$
$\neg p_\chi \wedge$ $\sim\psi_1 \wedge \widehat{\psi_2}$
$\neg p_\chi \wedge$ $*\psi_1 \wedge *\psi_2$
$p_\chi \wedge$ $\widehat{\psi_1} \wedge *\psi_2$
$\neg p_\chi \wedge$ $\widehat{\psi_1} \wedge *\psi_2$
$p_\chi \wedge$ $\sim\psi_1 \wedge \widehat{\psi_2}$
$\neg p_\chi \wedge$ $\sim\psi_1 \wedge \widehat{\psi_2}$
no pending obligations
some pending obligations

Figure 5 The component SA $\mathcal{C}_\chi$ for $\chi = \psi_1 \mathbf{R} \psi_2$.

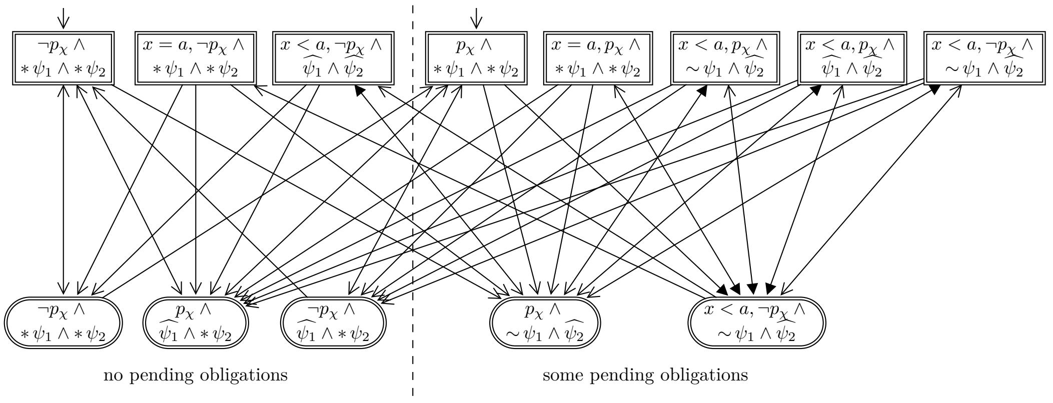

$\neg p_\chi \wedge$ $*\psi_1 \wedge *\psi_2$
$x = a, \neg p_\chi \wedge$ $*\psi_1 \wedge *\psi_2$
$x < a, \neg p_\chi \wedge$ $\widehat{\psi_1} \wedge \widehat{\psi_2}$
$p_\chi \wedge$ $*\psi_1 \wedge *\psi_2$
$x = a, p_\chi \wedge$ $*\psi_1 \wedge *\psi_2$
$x < a, p_\chi \wedge$ $\sim\psi_1 \wedge \widehat{\psi_2}$
$x < a, p_\chi \wedge$ $\widehat{\psi_1} \wedge \widehat{\psi_2}$
$x < a, \neg p_\chi \wedge$ $\sim\psi_1 \wedge \widehat{\psi_2}$
$\neg p_\chi \wedge$ $*\psi_1 \wedge *\psi_2$
$p_\chi \wedge$ $\widehat{\psi_1} \wedge *\psi_2$
$\neg p_\chi \wedge$ $\widehat{\psi_1} \wedge *\psi_2$
$p_\chi \wedge$ $\sim\psi_1 \wedge \widehat{\psi_2}$
$x < a, \neg p_\chi \wedge$ $\sim\psi_1 \wedge \widehat{\psi_2}$
no pending obligations
some pending obligations

Figure 6 The component SA $\mathcal{C}_\chi$ for $\chi = \psi_1 \mathbf{R}_{(0,a)} \psi_2$. The transitions with ▶ reset x.

B The components for 'release' operators

The component $\mathcal{C}_\chi$ for $\chi = \psi_1 \mathbf{R} \psi_2$ (Figure 5) is based on similar ideas as the component for $\psi_1 \mathbf{U} \psi_2$. In this case, an obligation can be satisfied by either $\widehat{\psi_1} \wedge \widehat{\psi_2}$ holding in a singular interval or $\widehat{\psi_1} \wedge *\psi_2$ holding in an open interval.

The component $\mathcal{C}_\chi$ for $\chi = \psi_1 \mathbf{R}_{(0,a)} \psi_2$ is given in Figure 6. In this case, all old obligations are implied by the newest one. We therefore reset the clock x when p_χ becomes false. The component for $\psi_1 \mathbf{R}_{(0,a]} \psi_2$ is similar and hence omitted.

Dynamic Controllability Made Simple

Massimo Cairo[1] and Romeo Rizzi[2]

1 **Department of Mathematics, University of Trento, Trento, Italy; and Department of Computer Science, University of Verona, Verona, Italy**
`massimo.cairo@unitn.it`, `massimo.cairo@univr.it`

2 **Department of Computer Science, University of Verona, Verona, Italy**
`romeo.rizzi@univr.it`

Abstract

Simple Temporal Networks with Uncertainty (STNUs) are a well-studied model for representing temporal constraints, where some intervals (contingent links) have an unknown but bounded duration, discovered only during execution. An STNU is dynamically controllable (DC) if there exists a strategy to execute its time-points satisfying all the constraints, regardless of the actual duration of contingent links revealed during execution.

In this work we present a new system of constraint propagation rules for STNUs, which is sound-and-complete for DC checking. Our system comprises just three rules which, differently from the ones proposed in all previous works, only generate unconditioned constraints. In particular, after applying our sound rules, the network remains an STNU in all respects. Moreover, our completeness proof is short and non-algorithmic, based on the explicit construction of a valid execution strategy. This is a substantial simplification of the theory which underlies all the polynomial-time algorithms for DC-checking.

Our analysis also shows: (1) the existence of late execution strategies for STNUs, (2) the equivalence of several variants of the notion of DC, (3) the existence of a fast algorithm for real-time execution of STNUs, which runs in $O(KN)$ total time in a network with $K \geq 1$ contingent links and $N \geq K$ time points, considerably improving the previous $O(N^3)$-time bound.

1998 ACM Subject Classification I.2.4 Knowledge Representation Formalisms and Methods, I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Simple Temporal Network with Uncertainty, Dynamic controllability

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.8

1 Introduction

A Simple Temporal Network (STN) is a model for representing temporal constraints: it comprises a set of time-points $\{P, Q, R, \dots\}$ and a collection of binary difference constraints of the form $Q \leq P + w$ with $w \in \mathbb{R}$. A planning agent wants to schedule the execution of time-points, i.e., assign a real value to each variable $P, Q, R, \dots$ representing its execution time, so that all the constraints in the network are satisfied.

Simple Temporal Networks with Uncertainty (STNUs) extend STNs to incorporate uncertainty in the duration of some time intervals. In an STNU some of the time-points, called contingent time-points, are not under the control of the planning agent but are executed by the environment. The execution time of a contingent time-point C is regulated by a contingent link (A, l, u, C), whose meaning is that C will be executed some time $\Delta \in [l, u]$ after the time-point A. The value Δ is called the duration of the contingent link, and is unknown to the planner until C is actually executed. An STNU is said to be dynamically controllable (DC) if the agent holds a strategy to execute all the time-points in the network, such that all constraints are satisfied regardless of the duration of the contingent links. This strategy has to be dynamic, in the sense that the execution time of a time-point X can only depend on the duration of contingent links (A, l, u, C) whose contingent time-point C is executed before X.

Two main problems are considered: checking whether a given network is DC and, if this is the case, executing its time-points in real-time, reacting dynamically to the durations revealed by the environment.

Previous work. Remarkably, it is possible to check in polynomial time whether a given network is DC. The best upper bound on the running time, for a network with N time-points, has been improved from pseudo-polynomial [10] (an "incremental" algorithm proposed in [15], as fixed in [11], also runs in pseudo-polynomial time), to $O(N^5)$ [9], $O(N^4)$ [7] (an incremental algorithm in [12] also runs in $O(N^4)$ time) and finally to $O(N^3)$ [8] (an incremental algorithm in [13] also runs in $O(N^3)$ time). The key tool to achieve a polynomial-time algorithm is propagation of binary constraints, i.e., generation of new constraints from existing ones according to some sound rules.

A system of constraint propagation rules has been proposed first by Morris, Muscettola and Vidal [10], and proven to be sound and complete. Soundness means that the system only generates constraints which must be satisfied by any dynamic execution strategy, as a logical consequence of existing constraints. Completeness means that, if, at some point, no tighter constraints can be generated by the rules, then the resulting network is DC. The system, in the revised and simplified version given in [9], consists of four propagation rules which generates both ordinary, unconditioned constraints (i.e., having the same form as input constraints) and conditional constraints marked with labels. A fifth rule transforms labeled constraints into ordinary constraints, when some conditions apply. We refer to this system of rules as MMV, and provide an illustration it in Table 1. The proof of completeness of MMV relies on the description of an algorithm, sketched in [10], improved and fully described in [5], which executes the time-points in the network in real-time.

Being such a simple and flexible model, STNUs have been extended in several ways in the literature (see, e.g., [14, 6, 2, 3, 1]). All these extension rely on the theory of DC-checking and executions developed for STNUs, and this theory, both for DC-checking and real-time execution, hinges on the system of constraint propagation rules. Indeed, to the best of our knowledge, all the existing DC-checking and real-time execution algorithms for STNUs work by applying propagation rules, implicitly or explicitly, and their correctness proof is based on the soundness and completeness of this rule system.

Issues with MMV system. The MMV system, as introduced in [10] and improved in [9], has still many points of weakness. First of all, the system itself is rather complex. The main reason is that it generates, besides ordinary constraints, also labeled constraints: these labeled constraints are not part of the STNU model, hence their interpretation is not immediately clear and requires further definitions and explanations. Moreover, the system comprises an arguably large number of rules: four constraint-propagation rules and one label-removal rule, which combine labeled and unlabeled constraints in several possible ways. Besides the intrinsic complexity of the system, the related theory is also somewhat intricate. Specifically, the proof of completeness is based on the description of an execution algorithm which, even after the improvement given in [5], is still quite involved, manipulating the network in non-trivial ways during its execution. Instead, as a completeness proof, it would be desirable to a have a simple, static description of an execution strategy which witnesses the dynamic controllability of a network, when no more application of the rules is possible.

Table 1 The MMV system of constraint-propagation rules [10, 9], in a graphical representation. In each of the rules, the original edges are drawn solid and the generated edge is drawn dotted.

Rule	**Graph representation**	**Applicability conditions**
No Case	$\begin{tikzcd} P \arrow[r, "v"] \arrow[rr, bend right, dotted, "v+w"'] & Q \arrow[r, "w"] & R \end{tikzcd}$	(none)
Upper Case	$\begin{tikzcd} P \arrow[r, "v"] \arrow[rr, bend right, dotted, "C\colon v+w"'] & Q \arrow[r, "C\colon w"] & A^C \end{tikzcd}$	$P \neq C$
Lower Case	$\begin{tikzcd} A^C \arrow[r, "c\colon l"] \arrow[rr, bend right, dotted, "l+w"'] & C \arrow[r, "w"] & R \end{tikzcd}$	$w \leq 0$, $R \neq C$
Cross Case	$\begin{tikzcd} A^C \arrow[r, "c\colon l"] \arrow[rr, bend right, dotted, "D\colon l+w"'] & C \arrow[r, "D\colon w"] & A^D \end{tikzcd}$	$w \leq 0$, $C \neq D$
Label Removal	$\begin{tikzcd} P \arrow[r, "C\colon z"] \arrow[r, bend right, dotted, "z"'] & A^C \end{tikzcd}$	$z \geq -l^C$

Our contributions. In this paper, we present a new, sound-and-complete system of constraint-propagation rules, called RUL, which solves all the above problems. Our system contains only three rules (called RELAX, UPPER and LOWER), with simple applicability conditions, which only generate new ordinary (i.e., unlabeled) constraints. This means that, with any application of the rules, the network remains an STNU in all respects, so no further explanation is necessary about the interpretation of generated constraints. We also provide a short, non-algorithmic proof of the completeness of our system, exhibiting a valid execution strategy explicitly. An illustration of the RUL system is given in Table 2 for comparison: the notation is explained in the next sections.

We perform a thorough analysis of our system of rules, proving both soundness and completeness in a simple but rigorous way. The proof of soundness is given with a minimal set of assumptions, which strictly extends its realm of applicability beyond the usual notion of DC in STNUs. In particular, our rules are sound even if the environment reveals the duration of a contingent link (A, l, u, C) right away at time $A + l$, i.e., at the lowest possible execution time of C, instead of its actual (contingent) execution time. This makes the soundness result strictly stronger, since any constraint deduced in this setting must apply also in the traditional setting, where the planner is only provided with less information. Moreover, for

Table 2 The RUL system of constraint propagation rules, in a graphical representation. In each of the rules, the original edges are drawn solid and the generated edge is drawn dotted.

Rule	**Graph representation**	**Applicability conditions**
Relax	$\begin{tikzcd} P \arrow[r, "v"] \arrow[rr, dotted, bend right, "v+w"'] & Q \arrow[r, "w"] & R \end{tikzcd}$	(none)
Upper	$\begin{tikzcd} P \arrow[r, "v"] \arrow[rr, dotted, bend right, "\max\{v-u,-l\}"'] & C \arrow[r, shift left, "-u"] & A^C \arrow[l, shift left, "l"] \end{tikzcd}$	(none)
Lower	$\begin{tikzcd} A^C \arrow[r, shift left, "l"] \arrow[rr, dotted, bend right, "l+w"'] & C \arrow[l, shift left, "-u"] \arrow[r, "w"] & R \end{tikzcd}$	$w \leq 0$ for $R \in \mathcal{T}_X$; $w \leq u^R$ for $R \in \mathcal{T}_C \setminus \{C\}$

soundness, we only assume that each duration Δ can take its extremal values l and u; we do not need to take into account intermediate values $l < \Delta < u$. Again, this allows to apply our rules in a strictly more general setting, where the durations Δ are restricted to assume only specific values (if any) beyond the extremal ones. By considering relaxed assumptions, we address in one shot several variants of the notion of DC, and the traditional notion among them as a special case.

As for completeness, we consider the notion of late execution strategy, where each time-point is executed at the latest possible time among all valid dynamic execution strategies which finish within a fixed time horizon. The existence of a valid late execution strategy is non-trivial and, to the best of our knowledge, has not been observed in previous work. The proof of completeness of our rules is obtained by constructing explicitly the late execution strategy of the given STNU, which is used without modifications as a witness of dynamic controllability in all the settings considered for soundness. This not only proves the existence of the late execution strategy, but also proves the equivalence of several variants of the notion of DC for STNUs. Thus, the late executions strategy serves as a unifying theoretical tool for studying STNUs, besides its potential practical usage, being both easier to define and more generally applicable than the early execution strategy. Indeed, despite being possibly more useful in practice, the early strategy requires a more involved construction, and might vary among different but equivalent variants. Actually, there are several notions of "early execution strategies", varying according to the specific variant of controllability, while the late strategy has a unique, simple construction which applies to all the variants and thus proves their equivalence.

Finally, we propose an algorithm for executing the late strategy in real time. Thanks to the simple and explicit definition of late execution strategy, our algorithm is intuitive and amounts to instantiating the definition in a computationally efficient way. With our algorithm, we achieve a running time of $O(KN)$ for executing a network of N time-points and $1 \leq K < N$ contingent links. This is a considerable improvement over the previous best upper bound of $O(N^3)$ time for executing a network, achieved using the early strategy.

Paper outline. We introduce concepts and notations for STNUs in Section 2. Then, in Section 3, we describe our system of rules RUL. We prove soundness in Section 4 and completeness in Section 5. Finally, we show our real-time execution algorithm in Section 6.

2 Preliminaries and notation

An STNU is a tuple $\Gamma = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ where: $\mathcal{T}$ is a finite set of real-valued temporal variables called *time-points* (here denoted with capital letters $P, Q, R, \dots$), $\mathcal{C}$ is a finite set of *constraints* of the form $Q \leq P + w$, for $P, Q \in \mathcal{T}$ and $w \in \mathbb{R}$, and $\mathcal{L}$ is a finite set of *contingent links*, i.e., tuples of the form (A, l, u, C), for $A, C \in \mathcal{T}$ and $l, u \in \mathbb{R}$ with $0 < l \leq u < \infty$. In a contingent link (A, l, u, C), A is the *activation* time-point, l is the *duration lower bound*, u is the *duration upper bound*, and C is the *contingent* time-point. Distinct contingent links have distinct contingent time-points. Given the contingent time-point C of a contingent link $(A, l, u, C) \in \mathcal{L}$ we define $A^C = A$, $u^C = u$, $l^C = l$. The set of contingent time-points is $\mathcal{T}_{\mathrm{C}} := \{C \mid (A, l, u, C) \in \mathcal{L}\}$, while the set of *executable* time-points is $\mathcal{T}_{\mathrm{X}} = \mathcal{T} \setminus \mathcal{T}_{\mathrm{C}}$. In the following we assume that an STNU $\Gamma = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ is given.

A *situation* is a function $s \colon \mathcal{T}_{\mathrm{C}} \to \mathbb{R}$, which assigns a *duration* $\Delta_s^C := s(C) \in [l^C, u^C]$ to each contingent link $(A^C, l^C, u^C, C) \in \mathcal{L}$. The set of all possible situations is $\Omega_{\mathrm{all}} := \prod_{C \in \mathcal{T}_{\mathrm{C}}} [l^C, u^C]$. For increased generality, the following definitions are given with respect to a fixed, non-empty subset of the situations $\Omega \subseteq \Omega_{\mathrm{all}}$. (The usual notions are obtained simply by choosing $\Omega = \Omega_{\mathrm{all}}$.) An *execution strategy* is a function $\sigma \colon (\Omega, \mathcal{T}_{\mathrm{X}}) \to \mathbb{R}$ that assigns an *execution time* $X_s^\sigma := \sigma(s, X)$ to each executable time-point X, in each possible situation $s \in \Omega$. Moreover, we define the execution time of a contingent time-point $C \in \mathcal{T}_{\mathrm{C}}$ to be $C_s^\sigma := A_s^{C\sigma} + \Delta_s^C$. In general, $P_s^\sigma \in \mathbb{R}$ denotes the execution time of the (executable or contingent) time-point $P \in \mathcal{T}$ in the situation s with the strategy σ. The superscript σ is omitted when clear from the context. Also, in some equations we replace the subscript s with $*$, meaning that the specific equation holds for every situation $s \in \Omega$.

An execution strategy σ is *viable* if $Q_* \leq P_* + w$ for every constraint $Q \leq P + w$ in $\mathcal{C}$. It is *dynamic* if, for any two situations $s, r \in \Omega$ and executable time-point $X \in \mathcal{T}_{\mathrm{X}}$, if $\{\langle C, \Delta_s^C \rangle \mid C_s < X_s\} = \{\langle C, \Delta_r^C \rangle \mid C_r < X_s\}$ then $X_r = X_s$. This definition correctly captures the intuitive notion that the execution time of X can only depend on the duration of contingent links whose contingent time-point is executed before X [4]. An STNU is said to be *dynamically controllable* (DC) if it admits an execution strategy which is both dynamic and viable.

We assume without loss of generality to have at most one constraint $Q \leq P + w_{PQ}$ in $\mathcal{C}$ for any two time-points $P, Q \in \mathcal{T}$: to this end, it is sufficient to take $w_{PQ} := \min\{w \mid (Q \leq P + w) \in \mathcal{C}\}$. By convention $w_{PQ} = \infty$ if there are no constraints $(Q \leq P + w) \in \mathcal{C}$.

An STNU is represented as a graph on node set $\mathcal{T}$ with three classes of weighted edges: *ordinary edges* $\mathcal{E}^0 = \{(P, Q, w_{PQ}) \mid w_{PQ} < \infty\}$, *lower bound edges* $\mathcal{E}^- = \{(A^C, C, l^C) \mid C \in \mathcal{T}_{\mathrm{C}}\}$ and *upper bound edges* $\mathcal{E}^+ = \{(C, A^C, -u^C) \mid C \in \mathcal{T}_{\mathrm{C}}\}$. When convenient, this graph-based notation is used to describe the constraints and contingent links in an STNU.

We borrow an example of STNU from [5] to use as a running example for our rule system. The example is shown in Figure 1 with two different graphical notations. In the notation on the left, used by previous work, contingent links are represented with a pair of edges labeled with upper-case and lower-case labels. Since in this work we do not need to generate other labeled edges in any given network, we prefer to use a different notation, as shown on the right. Contingent links are still represented as a pair of parallel edges, one lower bound edge and one upper bound edge: this is convenient in describing the rules, ensuring that rules

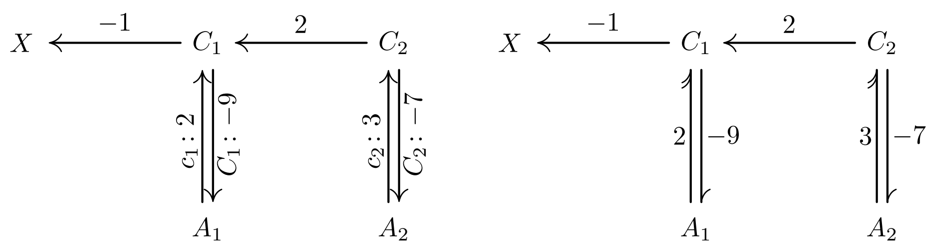


Figure 1 An STNU used as running example (borrowed from [5]). On the left, contingent links are represented as labeled constraints. On the right, they are represented using our notation.

always combine consecutive edges pointing in the right direction. However, the three types of edges, ordinary, lower and upper, are distinguished using a different graphical notation, and not by the addition of labels.

3 The RUL system

We introduce a system of three constraint-propagation rules: Relax, Upper and Lower. Each of these rules takes two consecutive edges (P, Q, v) and (Q, R, w) of the network and, if some conditions are satisfied, generates a new *ordinary* edge from P to R.

The Relax rule takes two ordinary edges (P, Q, v) and (Q, R, w) and generates the ordinary edge $(P, R, v + w)$.

The Upper rule takes an ordinary edge (P, C, v) and an upper bound edge $(C, A^C, -u^C)$ and generates the ordinary edge $(P, A^C, \max\{v - u^C, -l^C\})$.

The Lower rule takes a lower bound edge (A^C, C, l^C) and an ordinary edge (C, R, w) and generates the ordinary edge $(A^C, R, l^C + w)$, if the following preconditions are satisfied: either $R \in \mathcal{T}_X$ and $w \leq 0$, or $R \in \mathcal{T}_C \setminus \{C\}$ and $w \leq u^R$.

This system of constraint-propagation rules is called RUL and is illustrated in Table 2.

Comparison with MMV. The reader can compare the RUL system and the MMV system with the help of Table 1 and Table 2.

The Relax rule in RUL is identical to the No Case rule in MMV, and was renamed only for uniformity.

The Upper rule in RUL can be thought of as a combination of the Upper Case and Label Removal rules in MMV. Indeed, any edge $(P, A^C, \max\{v - u^C, -l^C\})$ obtained by Upper can be obtained with Upper Case and Label Removal in three steps.

1. If $v < u^C - l^C$, then take the unlabeled edge (P, C, v) and transform it into the weaker edge $(P, C, u^C - l^C)$. This is legal since the constraint $C \leq P + v$ subsumes the constraint $C \leq P + (u^C - l^C)$. In general, we end up with the edge $(P, C, \max\{v, u^C - l^C\})$.
2. Apply the Upper Case rule, obtaining the labeled edge $(P, A^C, C\colon \max\{v, u^C - l^C\} - u^C) = (P, A^C, C\colon \max\{v - u^C, -l^C\})$.
3. Apply the Label Removal rule, valid since $\max\{v - u^C, -l^C\} \geq -l^C$, to obtain the desired edge $(P, A^C, \max\{v - u^C, -l^C\})$.

As for the Lower rule, we have two cases. For $R \in \mathcal{T}_X$, it is identical to the Lower Case rule. However, for $R \in \mathcal{T}_C$, it extends strictly the domain of applicability of Lower Case. This is a crucial addition given by the RUL system with respect to MMV, since the edges generated by Lower cannot in general be obtained using MMV only. It is thanks to this addition that we can avoid generating labeled edges with the Upper Case rule, and still achieve completeness.

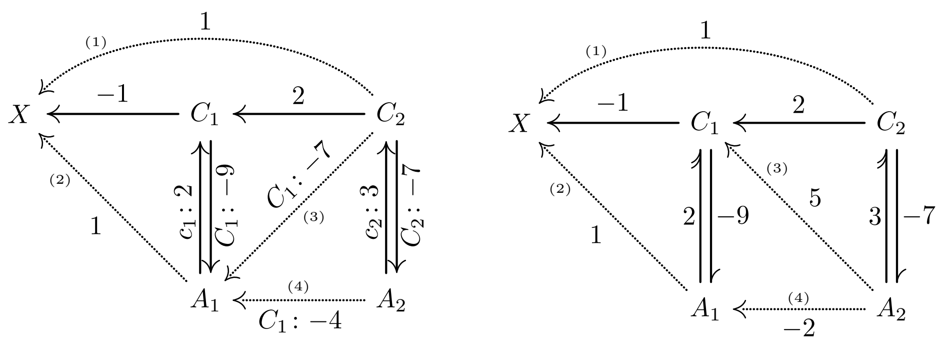


Figure 2 Examples of application of MMV (left) and RUL (right) on the example STNU.

Application examples. In Figure 2, we compare the application of the MMV and RUL systems on the example network.

MMV:

1. The No Case rule is applied to $(C_2, C_1, 2)$ and $(C_1, X, -1)$ to obtain $(C_2, X, 1)$.
2. The Lower Case rule is applied to $(A_1, C_1, c_1: 2)$ and $(C_1, X, -1)$ to obtain $(A_1, X, 1)$.
3. The Upper Case rule is applied to $(C_2, C_1, 2)$ and $(C_1, A_1, C_1: -9)$ to obtain the labeled edge $(C_2, A_1, C_1: -7)$.
4. The Cross Case rule is applied to $(A_2, C_2, c_1: 3)$ and $(C_2, A_1, C_1: -7)$ to obtain $(A_2, A_1, C_1: -4)$.

RUL:

1. The Relax rule is applied to $(C_2, C_1, 2)$ and $(C_1, X, -1)$ to obtain $(C_2, X, 1)$.
2. The Lower rule (in the case $R \in \mathcal{T}_X$) is applied to $(A_1, C_1, 2)$ and $(C_1, X, -1)$ to obtain $(A_1, X, 1)$.
3. The Lower rule (in the case $R \in \mathcal{T}_C$) is applied to $(A_2, C_2, 3)$ and $(C_2, C_1, 2)$ to obtain $(A_2, C_1, 5)$.
4. The Upper rule is applied to $(A_2, C_1, 5)$ and $(C_1, A_1, -9)$ to obtain $(A_2, A_1, \max\{5 - 9, -2\}) = (A_2, A_1, -2)$.

The edges $(C_2, X, 1)$ and $(A_1, X, 1)$ are generated by both systems, respectively using the No Case and Relax rules, and the Lower Case and Lower rules with $R \in \mathcal{T}_X$.

As for the other edges, on one hand, the MMV system generates the labeled edges $(C_2, A_1, C_1: -7)$ and $(A_2, A_1, C_1: -4)$ which are not generated using RUL. On the other hand, the RUL system generates the edge $(A_2, C_1, 5)$, with the Lower rule in the case $R \in \mathcal{T}_C$, which cannot be generated with MMV, and only gets a value -2 on the edge from A_2 to A_1, the strongest bound that can be given unconditionally, instead of the labeled edge $(A_2, A_1, C_1: -4)$. Observe, however, that the edge $(A_2, A_1, C_1: -4)$ could be obtained, *in a single step*, from the edges $(A_2, C_1, 5)$ and $(C_1, A_1, C_1: -9)$, with an application of the Upper Case rule, so it is somewhat implicit in the edge $(A_2, C_1, 5)$. In general, when replacing the MMV system the RUL system, the labeled edges, with head in an activation time-point, are replaced with unlabeled edges with head in the corresponding contingent time-point.

4 Soundness

We commit to prove soundness of all three rules comprising RUL, in the most general sense. Specifically, we partially relax the following assumptions: (1) that $\Omega = \Omega_{\text{all}}$, i.e., that every possible situation $s \in \Omega_{\text{all}}$ can be realized by the environment, and (2) that the duration Δ_s^C

is revealed only at time C_s. Proving soundness under our weaker assumptions is a stronger result, which not only implies soundness in the usual sense, but also helps applying the model of STNUs to a wider range of scenarios.

To express our assumptions precisely, the following notion is needed.

▶ **Definition 1.** Given a situation s, a contingent time-point $C_0 \in \mathcal{T}_\mathrm{C}$ and a value $d \in [l^{C_0}, u^{C_0}]$, define the situation $s[d/\Delta^{C_0}] = r$ where $\Delta_r^{C_0} = d$ and $\Delta_r^C = \Delta_s^C$ for every $C \in \mathcal{T}_\mathrm{C} \setminus \{C_0\}$.

We first state a condition on the set Ω.

▶ **Definition 2.** The set $\Omega \subseteq \Omega_\mathrm{all}$ is *extremal-closed* if, for every $s \in \Omega$, also $s[u^C/\Delta^C] \in \Omega$ and $s[l^C/\Delta^C] \in \Omega$.

For soundness, we need to assume that Ω is extremal-closed. This means that, even if not all situations may be realized by the environment, each duration Δ^C can at least assume the extremal values l^C and u^C, and do so independently of the other durations. Observe that this is the case for $\Omega = \Omega_\mathrm{all}$, where all the situations are possible. However, there are other, interesting examples of sets Ω that are extremal-closed: e.g., $\Omega = \prod_{C \in \mathcal{T}_\mathrm{C}} \{l^C, u^C\}$, where each duration Δ^C can *only* assume the extremal values l^C and u^C. Hence, with this assumption, we strictly increase the generality with respect to the case $\Omega = \Omega_\mathrm{all}$ addressed by previous work.

A relaxed notion of dynamic execution strategy is also introduced, where the duration of a contingent link $(A, l, u, C) \in \mathcal{L}$ in the situation s is revealed at time $A_s + l$ instead of C_s.

▶ **Definition 3.** An execution strategy σ is *upfront-dynamic* if, for any two situations $s, r \in \Omega$ and executable time-point $X \in \mathcal{T}_\mathrm{X}$, we have that $X_r = X_s$ if $\{\langle C, \Delta_s^C \rangle \mid A_s^C + l^C < X_s\} = \{\langle C, \Delta_r^C \rangle \mid A_r^C + l^C < X_s\}$.

By Lemma 4, working with upfront-dynamic strategies yields no loss in generality with respect to dynamic execution strategies.

▶ **Lemma 4.** *Any dynamic execution strategy is upfront-dynamic.*

Proof. The statement is clear if we consider that $C_* \geq A_*^C + l^C$, so upfront-dynamic strategies can only have more information at any time, whence more freedom, with respect to dynamic strategy.

A formal proof is provided for reference. The proof is by contradiction: assume σ is dynamic but not upfront-dynamic. Fix any two scenarios $s, r \in \Omega$ and suppose $X_s \neq X_r$ for some $X \in \mathcal{T}_\mathrm{X}$. Assume without loss of generality that $X_s = t < X_r$ and that t is minimal, i.e., $Y_s = Y_r$ for any $Y \in \mathcal{T}_\mathrm{X}$ such that either $Y_s < t$ or $Y_r < t$.

Since σ is dynamic, we have $\{\langle C, \Delta_s^C \rangle \mid C_s < t\} \neq \{\langle C, \Delta_r^C \rangle \mid C_r < t\}$. Consider any C in the difference. Either $A_s^C < t$ or $A_r^C < t$, then actually[1] $A_s^C = A_r^C$, by minimality of t. Thus, $\Delta_s^C \neq \Delta_r^C$, so $\{\langle C, \Delta_s^C \rangle \mid A_s^C + l^C < X_s\} \neq \{\langle C, \Delta_r^C \rangle \mid A_r^C + l^C < X_s\}$, closing the proof. ◀

Lemma 5 is the last ingredient needed for our soundness proof.

▶ **Lemma 5.** *Consider an upfront-dynamic execution strategy σ and situations $s, r \in \Omega$, where $r = s[d/\Delta^{C_0}]$ for some contingent time-point $C^0 \in \mathcal{T}_\mathrm{C}$ and duration $d \in [l^{C^0}, u^{C^0}]$. Then, $P_r = P_s$ for any $P \in \mathcal{T} \setminus \{C_0\}$ such that $P_s \leq A_s^{C_0} + l^{C_0}$. In particular, $A_s^{C_0} = A_r^{C_0}$.*

[1] For simplicity, here we are assuming $A^C \in \mathcal{T}_\mathrm{X}$, as common in the context of STNUs. The proof can be easily adapted if this assumption does not hold.

Proof. By contradiction. Fixed s and r, choose P so that $P_s \le A_s^{C_0} + l^{C_0}$, $P_s \ne P_r$, and P_s is smallest possible. Then $A_s^C = A_r^C$ for every $C \in \mathcal{T}_\mathrm{C}$ such that $A_s^C < P_s$. Also, $\Delta_r^C = \Delta_s^C$ for every $C \in \mathcal{T}_\mathrm{C} \setminus \{C_0\}$. If $P \in \mathcal{T}_\mathrm{C} \setminus \{C_0\}$, then $P_s = A_s^P + \Delta_s^P = A_r^P + \Delta_r^P = P_r$. Otherwise, if $P \in \mathcal{T}_\mathrm{X}$, then $P_s = P_r$ by Definition 3, since $A_s^{C_0} + l^{C_0} \ge P_s$. ◀

Soundness is proven in Lemma 6, where each of the three rules comprising RUL is proven to be sound, even for upfront-dynamic execution strategies, and for any extremal-closed $\Omega \subseteq \Omega_\mathrm{all}$.

▶ **Lemma 6.** *Assume Ω is extremal-closed. Let σ be a viable upfront-dynamic execution strategy. Consider an edge (P, R, x) generated using RUL. Then, $R_*^\sigma \le P_*^\sigma + x$.*

Proof. Relax. Clearly, $Q_* \le P_* + v$ and $R_* \le Q_* + w$ imply $R_* \le P_* + (v + w)$.

Upper. Assume $C_* \le P_* + v$. Fix any situation $s \in \Omega$: we need to prove $A_s^C \le P_s^C + \max\{v - u^C, -l^C\}$. If $A_s^C < P_s - l^C$ we are done, hence assume $A_s^C \ge P_s - l^C$. Take $r = s[u^C/\Delta^C]$ and observe that $A_r^C = A_s^C$ by Lemma 5. Moreover, $P_s \le A_s^C + l^C$, so also $P_r = P_s$ by Lemma 5. Then,

$$\begin{aligned} A_s^C &= A_r^C && \text{by Lemma 5} \\ &= C_r - u^C && \text{as } C_r = A_r^C + \Delta_r^C = A_r^C + u^C \\ &\le P_r + v - u^C && \text{by assumption } C_* \le P_* + v \\ &= P_s + v - u^C && \text{as } P_s = P_r \text{ by Lemma 5.} \end{aligned}$$

Lower. Assume $R_* \le C_* + w$. Fix any situation $s \in \Omega$: we need to prove that $R_s \le A_s^C + (l^C + w)$. Take $r = s[l^C/\Delta^C]$ and observe that $A_r^C = A_s^C$ by Lemma 5. We now prove that $R_r = R_s$, in two cases.

Case 1: $R \in \mathcal{T}_\mathrm{X}$ and $w \le 0$.
Consider that $R_r \le C_r + w = A_r^C + l^C + w \le A_r^C + l^C$ since $w \le 0$. Hence, $R_s = R_r$ by Lemma 5.

Case 2: $R \in \mathcal{T}_\mathrm{C} \setminus \{C\}$ and $w \le u^R$.
Take $q = r[u^R/\Delta^R]$ and observe that $A_q^R = R_q - u^R \le C_q + w - u^R \le C_q = A_q^C + l^C$, whence[2] $A_s^R = A_r^R = A_q^R$ by Lemma 5 and $R_s = A_s^R + \Delta_s^R = A_r^R + \Delta_r^R = R_r$.

Then,

$$\begin{aligned} R_s &= R_r && \text{proven separately in the two cases} \\ &\le C_r + w && \text{by assumption } R_* \le C_* + w \\ &= A_r^C + l^C + w && \text{as } C_r = A_r^C + \Delta_r^C = A_r^C + l^C \\ &= A_s^C + l^C + w && \text{as } A_s = A_r \text{ by Lemma 5.} \end{aligned}$$ ◀

5 Completeness via late execution strategy

For completeness, we need to assume, as usual in temporal networks, to have a special time-point Z which is due to be executed at time 0 and before any other time-point. Moreover, it will be useful to consider Z as a contingent node for uniformity.

[2] Let us spend some extra words to explain, in a more intuitive way, this step of the proof, which is crucial for our system of rules and arguably is the most involved. In the situation q, where $\Delta_q^C = l^C$ and $\Delta_q^R = u^R$, the activation node A^R for R must be executed before or at time $A_q^C + l_q^C$: this is a direct consequence of the existing constraint $R \le C + w$ with $w \le u^R$. Hence, at the time A_q^R, neither the duration Δ^R (trivially) nor Δ^C are known to the planner. Thus, the time A^R cannot change depending on those durations, i.e., $A_s^R = A_r^R = A_q^R$ as claimed.

Table 3 Extra restrictions for the rules Relax$^+$ and Upper$^+$.

Rule	Extra restrictions		
Relax$^+$	$P \in \mathcal{T}_\text{C}$	$Q \in \mathcal{T}_\text{X}$	$v > 0$
Upper$^+$	$P \in \mathcal{T}_\text{C}$		$v > u^C$

Table 4 RUL$^+$ sub-system of RUL.

Rule	Graph representation	Applicability conditions
Relax$^+$	$\begin{tikzcd} P \arrow[r, "v"] \arrow[rr, dotted, bend right, "v+w"'] & Q \arrow[r, "w"] & R \end{tikzcd}$	$P \in \mathcal{T}_\text{C}$ $Q \in \mathcal{T}_\text{X}$ $v > 0$
Upper$^+$	$\begin{tikzcd} P \arrow[r, "v"] \arrow[rr, dotted, bend right, "v-u"'] & C \arrow[r, shift left, "-u"] & A^C \arrow[l, shift left, "l"] \end{tikzcd}$	$P \in \mathcal{T}_\text{C}$ $v > u^C$
Lower	$\begin{tikzcd} A^C \arrow[r, shift left, "l"] \arrow[rr, dotted, bend right, "l+w"'] & C \arrow[l, shift left, "-u"] \arrow[r, "w"] & R \end{tikzcd}$	$w \leq 0$ for $R \in \mathcal{T}_\text{X}$ $w \leq u^R$ for $R \in \mathcal{T}_\text{C} \setminus \{C\}$

▶ **Definition 7.** An STNU is *pointed* if it contains distinguished time-points $A^Z \in \mathcal{T}_\text{X}$, $Z \in \mathcal{T}_\text{C}$, and a contingent link $(A^Z, 1, 1, Z) \in \mathcal{L}$, with $w_{PZ} < 0$ and $w_{ZP} > 0$ for any $P \in \mathcal{T}_\text{X} \setminus \{A^Z\}$. If also $w_{ZP} < \infty$ for every $P \in \mathcal{T}_\text{X} \setminus \{A^Z\}$, then it is *upper-bounded.*

Any network can be made pointed, without loss of generality, by adding the contingent link $(A^Z, 1, 1, Z) \in \mathcal{L}$ between dummy nodes A^Z and Z, and setting $w_{PZ} = -1$ and $w_{ZP} = \infty$ for every $P \in \mathcal{T}_\text{X} \setminus \{A^Z\}$. To make it upper-bounded, it is sufficient to pick a sufficiently large horizon time $h > 0$, and set $w_{ZP} = h$ for every $P \in \mathcal{T}_\text{X} \setminus \{A^Z\}$.

The completeness of the RUL system is proven by exhibiting an explicit execution strategy for any upper-bounded network Γ which is closed under them. Actually, Γ only needs to be closed under a subset of the RUL system, called RUL$^+$. In the sub-system RUL$^+$, we only apply the Relax and Upper rules when $P \in \mathcal{T}_\text{C}$ is contingent. Moreover, we require $v > 0$ and $Q \in \mathcal{T}_\text{X}$ for the Relax rule, and $v > u^C$ for the Upper rule. The restricted rules, called Relax$^+$ and Upper$^+$, are illustrated in Table 3. Together with the unrestricted rule Lower, they constitute the RUL$^+$ system (Table 4).

The strategy we exhibit is a *late* execution strategy, as opposed to the commonly used early execution strategy.

▶ **Definition 8.** In an upper-bounded network Γ, a viable dynamic execution strategy σ is the *late execution strategy* if, for any other viable dynamic execution strategy τ, $P_s^\sigma \geq P_s^\tau$ for each $s \in \Omega$ and $P \in \mathcal{T}$. (It is implicitly assumed $A_*^Z = -1$ for both σ and τ.) I.e., in the late execution strategy each time-point is executed at the latest possible time over all viable dynamic execution strategies.

It follows from the definition that the late execution strategy, if it exists, is unique. However, it is not at all implied by the definition that a given dynamically controllable network, even if upper-bounded, admits a late execution strategy. Indeed, in principle it could be that the strategy defined as

$$X_s^\sigma := \max\{X_s^\tau \mid \text{dynamic and viable } \tau \text{ with } A_*^Z = -1\},$$

i.e., the only candidate to be the late execution strategy, is not itself dynamic and viable.[3] It is obtained, as a consequence of the analysis of our system of rules, that indeed, for the case of STNUs, the late execution strategy always exists, provided the network is DC and upper-bounded. Working with late execution strategies – it turns out – is very convenient, since we do not have to deal with "wait" (or labeled) edges; this will allow for a short and self-contained non-algorithmic proof of the completeness of RUL.

Notice that a strategy satisfying the following equation would trivially be the late execution strategy, if only it were dynamic and viable:

$$Y_* = \min_{\substack{C \in \mathcal{T}_C \\ w_{CY} > 0}} C_* + w_{CY} \tag{1}$$

for every $Y \in \mathcal{T}_X \setminus \{A^Z\}$, and $A_*^Z = -1$.

▶ **Lemma 9.** *Assume Γ is upper-bounded. Then, there exists a unique execution strategy σ satisfying Equation 1. Moreover, σ is dynamic.*

Proof. The reason why the lemma holds is simple: Equation 1 defines the execution time of each executable time-point in terms of time-points with a strictly lower execution time, hence σ is well-defined and unique. Remarkably, the very same observation also shows that σ is dynamic. Notice that boundedness (i.e., $0 < w_{ZP} < \infty$ for every $P \in \mathcal{T}_X \setminus \{A^Z\}$) is necessary to ensure that the right-hand side of Equation 1 is finite. Also, the assumption $A_*^Z = -1$ is necessary for σ to be unique.

To insist on a formal proof, we can rewrite Equation 1 as follows:

$$Y_s = t \iff \min_{\substack{C \in \mathcal{T}_C \\ w_{CY} > 0 \\ C_s < t}} C_s + w_{CY} = t \qquad \forall Y \in \mathcal{T}_X, s \in \Omega.$$

In this writing, the dependency on the time $t \in [0, \infty)$ is made explicit and allows for an inductive construction of the solution σ for increasing t, which is thus proven to be unique. To prove that σ is also dynamic, suppose $X_s < X_r$ for some $X \in \mathcal{T}_X$ and $s, r \in \Omega$. Also, choose a minimal X_s, i.e., assume $Y_s = t \iff Y_r = t$ for every $t < X_s$ and $Y \in \mathcal{T}_X$. Take $C \in \mathcal{T}_C$ such that $X_s = C_s + w_{CX}$ and $w_{CX} > 0$ (i.e., the arg min of Equation 1). By definition of σ, we have $X_r \le C_r + w_{CX}$, so $C_s + w_{CX} = X_s < X_r \le C_r + w_{CX}$, whence $C_s < C_r$. By minimality of X_s we have[4] $A_s^C = A_r^C$, so it must be $\Delta_s^C < \Delta_r^C$. This proves that $\{\langle C, \Delta_s^C \rangle \mid C_s < t\} \neq \{\langle C, \Delta_r^C \rangle \mid C_r < t\}$ for every $t \ge X_s$, concluding the proof. ◀

▶ **Lemma 10.** *Assume Γ is upper-bounded and closed under RUL$^+$. Then, the strategy σ defined by Equation 1 is viable.*

[3] In some other temporal network models, such as CSTNs, it is possible to construct examples of upper-bounded networks which admit an early execution strategy but no late execution strategy.

[4] For simplicity, here we are assuming $A^C \in \mathcal{T}_X$, as common in the context of STNUs. The proof can be easily adapted if this assumption does not hold.

Table 5 Cases in the proof of Lemma 10.

Constraint $R \leq P + w_{PR}$			**Case**
$P \in \mathcal{T}_{\mathrm{X}}$			Relax$^+$
$P \in \mathcal{T}_{\mathrm{C}}$	$R \in \mathcal{T}_{\mathrm{X}}$	$w_{PR} > 0$	By-Construction
		$w_{PR} \leq 0$	Lower
	$R \in \mathcal{T}_{\mathrm{C}}$	$w_{PR} > u_R$	Upper$^+$
		$w_{PR} \leq u_R$	Lower

Proof. Fix a situation s. We prove that $R_s \leq P_s + w_{PR}$ by induction on the times P_s and R_s.

The following four cases (By-Construction, Relax$^+$, Lower, Upper$^+$) cover all the possibilities, as illustrated in Table 5.

Case By-Construction: $Y \leq C + w_{CY}$ with $C \in \mathcal{T}_{\mathrm{C}}$, $w_{CY} > 0$ and $Y \in \mathcal{T}_{\mathrm{X}}$.
We have $Y_s \leq C_s + w_{CY}$ by Equation 1.

Case Relax$^+$: $R \leq X + w_{XR}$ with $X \in \mathcal{T}_{\mathrm{X}}$.
Take $C \in \mathcal{T}_{\mathrm{C}}$ such that $X_s = C_s + w_{CX}$ with $w_{CX} > 0$. Then,

$$\begin{aligned} R_s &\leq C_s + w_{CR} && \text{by induction, since } C_s < X_s = C_s + w_{CX} \\ &\leq C_s + w_{CX} + w_{XR} && \text{by Relax}^+ \text{ applied to } C, X, R \\ &= X_s + w_{XR} && \text{by assumption } X_s = C_s + w_{CX}. \end{aligned}$$

Case Lower: $R \leq C + w_{CR}$ with $C \in \mathcal{T}_{\mathrm{C}}$, and either $R \in \mathcal{T}_{\mathrm{X}}$ and $w_{CR} \leq 0$, or $R \in \mathcal{T}_{\mathrm{C}} \setminus \{C\}$ and $w_{CR} \leq u_R$.
Then,

$$\begin{aligned} R_s &\leq A^C_s + w_{A^C R} && \text{by induction, as } A^C_s < C_s = A^C_s + \Delta^C_s \\ &\leq A^C_s + (w_{CR} + l^C) && \text{by Lower applied to } C, A^C, R \\ &\leq C_s + w_{CR} && \text{since } C_s = A^C_s + \Delta^C_s \geq A^C + l^C. \end{aligned}$$

Case Upper$^+$: $C \leq P + w_{PC}$, with $C \in \mathcal{T}_{\mathrm{C}}$ and $w_{PC} \geq u^C$.
Then,

$$\begin{aligned} C_s &\leq A^C_s + u^C && \text{as } C_s = A^C_s + \Delta^C_s \text{ and } \Delta^C_s \leq u^C \\ &\leq P_s + w_{PA^C} + u^C && \text{by induction, as } A^C_s < C_s = A^C_s + \Delta^C_s \\ &\leq P_s + (w_{PC} - u^C) + u^C && \text{by Upper}^+ \text{ applied to } P, A^C, C \\ &= P_s + w_{PC}. \end{aligned}$$ ◀

To complete our analysis, we need to show that a DC network admits a closure under RUL$^+$. More precisely:

▶ **Lemma 11.** *If a network admits a viable upfront-dynamic execution strategy, then it admits a closure under RUL.*

This fact is quite intuitive: just continue to apply the rules, as long as possible, until closure is reached. Since the rules are sound, and the network admits a viable and dynamic execution strategy σ, then at some point it should become impossible to deduce stricter and stricter constraints, as they must be satisfied at least by σ. However, to prove that closure is reached in a finite number of steps requires (explicitly or implicitly) the description of an actual algorithm to perform the propagations, which is beyond the scope of this work. The authors verified that the existing DC checking algorithms can be adapted to use the RUL system, and indeed produce the closure under RUL of a given input network, provided it is DC.

We explore here the consequences of Lemma 11.

▶ **Theorem 12.** *Let Γ be an upper-bounded STNU. The following are equivalent:*
1. *Γ is dynamically controllable (admits a viable dynamic execution strategy),*
2. *Γ admits a viable upfront-dynamic execution strategy,*
3. *Γ admits a closure under the RUL system (without negative self-loops),*
4. *Γ admits a closure under the RUL$^+$ system (with only positive edges from Z).*

Proof. Consider the following implications:
- $1 \implies 2$, by Lemma 5,
- $2 \implies 3$, by Lemma 11,
- $3 \implies 4$, trivial,
- $4 \implies 1$, by the construction of the late execution strategy (Lemma 9 and Lemma 10). ◀

▶ **Corollary 13.** *Every dynamically controllable upper-bounded STNU admits a late execution strategy.*

▶ **Corollary 14.** *The property of being dynamically controllable does not change if the duration Δ_s^C of contingent links is revealed at any time between $A_s^C + l^C$ and C_s, nor if the duration Δ_s^C is restricted to assume only the extremal values l^C and u^C.*

Proof. Consider that the property of admitting a closure under the RUL system is unvaried, and the RUL system is sound-and-complete for all these variants. ◀

▶ **Corollary 15.** *DC-checking for an STNU with N time-points, M constraints, and K contingent links admits a certificate of YES of size $O(KN)$, verifiable in $O(K^2N + KM)$ time and logarithmic space.*

Proof. The closure of the network under RUL$^+$ is a certificate of YES. Only $O(KN)$ new edges may be generated, since they all have tail in either a contingent time-point or an activation time-point, and this suffices for the size bound. To verify the closure under the Relax rule, one can guess a constraint $R \le Q + w$ (either original or in the closure) and the time-point $P \in \mathcal{T}_\mathrm{C}$, and verify that the rule does not generate any stronger constraint. This takes $O(K^2N + KM)$ time (a factor $KN + M$ to guess the original or generated constraint, times a factor K to guess the time-point P). Similarly, to verify the closure under the Lower rule, one can guess $C \in \mathcal{T}_\mathrm{C}$ and $R \in \mathcal{T}$, in $O(KN)$ total time, while verifying the Upper rule only require $O(K^2) \le O(KN)$ time to guess $P, C \in \mathcal{T}_\mathrm{C}$. Finally, the space required for verification is clearly logarithmic. ◀

▶ **Corollary 16.** *If the network is integer, i.e., $w_{PQ} \in \mathbb{Z} \cup \{\infty\}$ for every $P, Q \in \mathcal{T}$, there there exists a strategy where each execution time X_s satisfies either:*
- *$X_s \in \mathbb{Z}$ is integer, or*
- *$X_s = C_s + k$ for some $C \in \mathcal{T}_\mathrm{C}$ with integer $k \in \mathbb{Z}$.*

▶ **Corollary 17.** *If the network is integer, i.e., $w_{PQ} \in \mathbb{Z} \cup \{\infty\}$ for every $P, Q \in \mathcal{T}$, and the durations are constrained to be integers, i.e., $l^C, u^C \in \mathbb{Z}$ and $\Omega = \prod_{C \in \mathcal{T}_\mathrm{C}} \{l^C, l^C + 1, \ldots, u^C\}$, then the network admits an integer execution strategy, i.e., where $P_* \in \mathbb{Z}$ for every $P \in \mathcal{T}$.*

Proof. Consider the late execution strategy. ◀

6 Real-time execution

We have shown that a network closed under RUL$^+$ admits a late execution strategy, defined by Equation 1. We next show that the late execution strategy can be also executed in real-time, with a total running time $O(KN)$. We only need to assume that, for every $C \in \mathcal{T}_\mathrm{C}$, we have stored the values w_{CX}, for every $X \in \mathcal{T}_\mathrm{X}$ such that $w_{CX} > 0$, in a list L_C, sorted by increasing numerical value of w_{CX}. These lists L_C, for $C \in \mathcal{T}_\mathrm{C}$, can be compiled in a preprocessing step (by sorting and filtering all the values w_{CX}, $X \in \mathcal{T}_\mathrm{X}$), in $O(KN \log N)$ total time. This cost should not be accounted for in the real-time execution running time: it is better regarded as a light postprocessing of the DC-checking step, having lower asymptotic time and space complexity.

The algorithm works by maintaining the value

$$X_\circ := \min_{\substack{C \in \mathcal{X}_\mathrm{C} \\ w_{CX} > 0}} C_\bullet + w_{CX}$$

for every $X \in \mathcal{T}_X$, where $\mathcal{X}_\mathrm{C}$ is the set of already executed contingent time-points (initially containing only Z), $C_\bullet$ is the execution time of the time-point C, and $X_\circ$ represents the current scheduled time for the time-point X. During the real-time execution of the network, the values $X_\circ$ can only decrease monotonically, and have to be updated whenever a contingent time-point gets executed. The values $X_\circ$ are maintained in a list S sorted by increasing numerical value of $X_\circ$, which can be regarded as an event queue representing the current candidate schedule of executable time-points.

At the beginning $\mathcal{X}_\mathrm{C} = \{Z\}$, and the list S is built simply by copying L_Z, setting $X_\circ = w_{XZ}$ for every $X \in \mathcal{T}_\mathrm{X}$. The algorithm works by repeatedly executing the time-point $\bar{X}$ which appears first in the list S, at time $\bar{X}_\circ$, unless a contingent time-point is executed before that time. When a contingent time-point $\bar{C}$ is executed, the values $X_\circ$ and the list S need to be updated before resuming the execution. This can be done in $O(N)$ time by merging the new candidate values $\bar{C}_\bullet + w_{\bar{C}X}$ with the list S. More precisely, a new list $S_{\bar{C}}$ is built from $L_{\bar{C}}$, containing the values

$$X_\circ^{\bar{C}} := \bar{C}_\bullet + w_{\bar{C}X}$$

for every $X \in \mathcal{T}_\mathrm{X}$ such that $w_{\bar{C}X} > 0$. The list $S_{\bar{C}}$ is constructed preserving the increasing order given by $L_{\bar{C}}$. Observe that the new value of $X_\circ$ is obtained as

$$X_\circ^{\mathrm{new}} := \min\{X_\circ^{\mathrm{old}}, X_\circ^{\bar{C}}\}.$$

To obtain a sorted list with the new values, first merge the lists S and $S_{\bar{C}}$ as in the Merge Sort algorithm. Then, scan the merged list to remove duplicates. For each duplicated time-point X, keep only the first occurrence in the list, i.e., the one with smaller numerical value $\min\{X_\circ^{\mathrm{old}}, X_\circ^{\bar{C}}\}$, and set that value as the new value of $X_\circ$. With this pass we obtain the updated list S^{new}, already sorted, containing the updated values $X_\circ^{\mathrm{new}}$ which take into account the execution of C. Since we pay $O(N)$ time for each of the K contingent time-points and only $O(1)$ for each executable time-point, the total time is $O(KN)$.

7 Conclusions

We have presented a new, sound-and-complete system of constraint propagation rules, called RUL, for checking the dynamic controllability of STNUs. Soundness has been proven, for each of the rules, in its most general version, and holds even if we take into account upfront-dynamic execution strategies besides dynamic execution strategies. As for completeness, we have proven that closure under a subset of the rules, called RUL$^+$, is sufficient to guarantee the existence of a viable dynamic execution strategy. Specifically, we considered the strategy which executes each time-point at the latest time allowed by positive edges from contingent nodes. We defined this strategy explicitly, with a single equation; then, we provided a short proof that it is viable, i.e., it satisfies every other constraint in the network, assuming closure under RUL$^+$. This strategy is the late execution strategy, since moving the execution time of any time-point further in the future would violate at least one constraint. Finally, we showed how to execute the late execution strategy in real-time, paying only $O(KN)$ time in total for a network with N time-points and K contingent links. With the introduction of the RUL system, this paper helps making STNUs not only simpler, and better understood, but also more generally applicable.

In this work, we did not provide any new DC-checking algorithm. However, existing algorithms can be adapted to use the RUL system, and produce the closure under RUL of a given input network, if it is DC, without incurring in any additional complexity cost. Furthermore, the added simplicity of the new rules allows for a cleaner approach to DC-checking, which yields improved algorithms as for both simplicity and efficiency, beyond the scope in this work. We opted for a short and clean description of the RUL system, which is proposed as a new foundation for the theory of DC-checking and real-time execution of STNUs, and their numerous extensions.

References

1 Carlo Combi, Luke Hunsberger, and Roberto Posenato. An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty. In Joaquim Filipe and Ana L. N. Fred, editors, *ICAART 2013 – Proceedings of the 5th International Conference on Agents and Artificial Intelligence, Volume 2, Barcelona, Spain, 15-18 February, 2013*, pages 144–156. SciTePress, 2013.

2 Patrick R. Conrad and Brian Charles Williams. Drake: An efficient executive for temporal plans with choice. *CoRR*, abs/1401.4606, 2014. URL: `http://arxiv.org/abs/1401.4606`.

3 Robert T. Effinger, Brian Charles Williams, Gerard Kelly, and Michael Sheehy. Dynamic controllability of temporally-flexible reactive programs. In Alfonso Gerevini, Adele E. Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. AAAI, 2009. URL: `http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/739`.

4 Luke Hunsberger. Fixing the semantics for dynamic controllability and providing a more practical characterization of dynamic execution strategies. In Carsten Lutz and Jean-François Raskin, editors, *TIME 2009, 16th International Symposium on Temporal Representation and Reasoning, Bressanone-Brixen, Italy, 23-25 July 2009, Proceedings*, pages 155–162. IEEE Computer Society, 2009. `doi:10.1109/TIME.2009.25`.

5 Luke Hunsberger. Efficient execution of dynamically controllable simple temporal networks with uncertainty. *Acta Inf.*, 53(2):89–147, 2016. `doi:10.1007/s00236-015-0227-0`.

6 Lina Khatib, Paul H. Morris, Robert A. Morris, and Francesca Rossi. Temporal constraint reasoning with preferences. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 322–327. Morgan Kaufmann, 2001.

7 Paul Morris. A structural characterization of temporal dynamic controllability. In Frédéric Benhamou, editor, *Principles and Practice of Constraint Programming - CP 2006, 12th

International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings, volume 4204 of *Lecture Notes in Computer Science*, pages 375–389. Springer, 2006. doi:10.1007/11889205_28.

8 Paul Morris. Dynamic controllability and dispatchability relationships. In Helmut Simonis, editor, *Integration of AI and OR Techniques in Constraint Programming – 11th International Conference, CPAIOR 2014, Cork, Ireland, May 19-23, 2014. Proceedings*, volume 8451 of *Lecture Notes in Computer Science*, pages 464–479. Springer, 2014. doi:10.1007/978-3-319-07046-9_33.

9 Paul H. Morris and Nicola Muscettola. Temporal dynamic controllability revisited. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 1193–1198. AAAI Press / The MIT Press, 2005. URL: http://www.aaai.org/Library/AAAI/2005/aaai05-189.php.

10 Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 494–502. Morgan Kaufmann, 2001.

11 Mikael Nilsson, Jonas Kvarnström, and Patrick Doherty. Incremental dynamic controllability revisited. In Daniel Borrajo, Subbarao Kambhampati, Angelo Oddi, and Simone Fratini, editors, *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, ICAPS 2013, Rome, Italy, June 10-14, 2013*. AAAI, 2013. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS13/paper/view/6028.

12 Mikael Nilsson, Jonas Kvarnström, and Patrick Doherty. EfficientIDC: A faster incremental dynamic controllability algorithm. In Steve A. Chien, Minh Binh Do, Alan Fern, and Wheeler Ruml, editors, *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*. AAAI, 2014. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS14/paper/view/7936.

13 Mikael Nilsson, Jonas Kvarnström, and Patrick Doherty. Incremental dynamic controllability in cubic worst-case time. In Amedeo Cesta, Carlo Combi, and François Laroussinie, editors, *21st International Symposium on Temporal Representation and Reasoning, TIME 2014, Verona, Italy, September 8-10, 2014*, pages 17–26. IEEE Computer Society, 2014. doi:10.1109/TIME.2014.13.

14 Francesca Rossi, Kristen Brent Venable, and Neil Yorke-Smith. Uncertainty in soft temporal constraint problems: A general framework and controllability algorithms forthe fuzzy case. *J. Artif. Intell. Res. (JAIR)*, 27:617–674, 2006. doi:10.1613/jair.2135.

15 Julie A. Shah, John Stedl, Brian Charles Williams, and Paul Robertson. A fast incremental algorithm for maintaining dispatchability of partially controllable plans. In Mark S. Boddy, Maria Fox, and Sylvie Thiébaux, editors, *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, pages 296–303. AAAI, 2007. URL: http://www.aaai.org/Library/ICAPS/2007/icaps07-038.php.

Incorporating Decision Nodes into Conditional Simple Temporal Networks

Massimo Cairo[1], Carlo Combi[2], Carlo Comin[2], Luke Hunsberger[4], Roberto Posenato[5], Romeo Rizzi[6], and Matteo Zavatteri[7]

1 **Department of Mathematics, University of Trento, Trento, Italy**
`massimo.cairo@unitn.it`
2 **Department of Computer Science, University of Verona, Verona, Italy**
`carlo.combi@univr.it`
3 **Department of Computer Science, University of Verona, Verona, Italy**
`carlo.comin@univr.it`
4 **Department of Computer Science, Vassar College, Poughkeepsie, NY, USA**
`hunsberger@vassar.edu`
5 **Department of Computer Science, University of Verona, Verona, Italy**
`roberto.posenato@univr.it`
6 **Department of Computer Science, University of Verona, Verona, Italy**
`omeo.rizzi@univr.it`
7 **Department of Computer Science, University of Verona, Verona, Italy**
`matteo.zavatteri@univr.it`

Abstract

A Conditional Simple Temporal Network (CSTN) augments a Simple Temporal Network (STN) to include special time-points, called observation time-points. In a CSTN, the agent executing the network controls the execution of every time-point. However, each observation time-point has a unique propositional letter associated with it and, when the agent executes that time-point, the environment assigns a truth value to the corresponding letter. Thus, the agent observes, but does not control the assignment of truth values. A CSTN is dynamically consistent (DC) if there exists a strategy for executing its time-points such that all relevant constraints will be satisfied no matter which truth values the environment assigns to the propositional letters.

Alternatively, in a Labeled Simple Temporal Network (Labeled STN) – also called a Temporal Plan with Choice – the agent executing the network controls the assignment of values to the so-called choice variables. Furthermore, the agent can make those assignments at any time. For this reason, a Labeled STN is equivalent to a Disjunctive Temporal Network.

This paper incorporates both of the above extensions by augmenting a CSTN to include not only observation time-points but also decision time-points. A decision time-point is like an observation time-point in that it has an associated propositional letter whose value is determined when the decision time-point is executed. It differs in that the agent – not the environment – selects that value. The resulting network is called a CSTN with Decisions (CSTND). This paper shows that a CSTND generalizes both CSTNs and Labeled STNs, and proves that the problem of determining whether any given CSTND is dynamically consistent is PSPACE-complete. It also presents algorithms that address two sub-classes of CSTNDs: (1) those that contain only decision time-points; and (2) those in which all decisions are made before execution begins.

1998 ACM Subject Classification G.2.2 Graph Theory, I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Conditional Simple Temporal Networks with Decisions, Dynamic Consistency, SAT Solver, Hyper Temporal Networks, PSPACE

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.9

1 Introduction

Temporal networks have long been employed for the representation, validation, and execution of plans affected by temporal constraints [1, 5, 8, 10, 11, 17, 23]. A temporal network constains *time-points* and *temporal constraints.* Time-points are real-valued variables; temporal constraints are binary difference constraints that specify lower or upper bounds on the temporal distance between pairs of time-points [16]. The *execution* of a time-point (i.e., the assignment of a real value to it) models the (instantaneous) occurrence of an event. An agent executing a temporal network aims to execute its time-points so that all relevant temporal constraints are satisfied.

A *Simple Temporal Network* (STN) is the most studied and used kind of temporal network due to its simplicity, efficiency, and general applicability [16]. An STN is typically used in planning applications where all time-points must be executed (i.e., must play their role in the plan) and where the agent controls their execution. An STN is *consistent* if the network can be executed in such a way that all of its constraints are satisfied.

Since STNs were proposed, several authors have introduced extensions to STNs to augment their expressiveness. Among them, we mention here: (i) *Simple Temporal Networks with Uncertainty* (STNUs) [24], (ii) *Conditional Simple Temporal Networks* (CSTNs) [22, 25], and (iii) *Conditional Simple Temporal Networks with Uncertainty* (CSTNUs) [21].

A *Simple Temporal Network with Uncertainty* extends an STN by incorporating *contingent links* to model uncontrollable, but bounded temporal durations. A contingent link has the form (A, x, y, C), where A is the *activation* time-point, x and y are real numbers such that $0 < x < y < \infty$, and C is the *contingent* time-point. Typically, the agent controls the execution of A, but once A has been executed, the execution of C is beyond the agent's control because the time of its execution is decided by the environment. Indeed, C may be executed at any time such that $x \leq C - A \leq y$. Thus, the agent merely observes the execution of C when it occurs. The agent aims to execute the time-points under its control such that all constraints will be satisfied no matter how the contingent durations turn out. Crucially, the agent may *react* to the contingent durations it observes, adapting its execution of the remaining unexecuted time-points (i.e., it may use a *dynamic* execution strategy [24]).

A *Conditional Simple Temporal Network* extends an STN in a different direction by specifying the time-points/constraints must be executed/satisfied in various *scenarios.* Each scenario is represented by a conjunction of (positive or negative) propositional literals, where each proposition represents some condition. The agent executing a CSTN aims to execute the time-points relevant to the unfolding scenario, while satisfying all of the relevant constraints. Each propositional letter p has a corresponding *observation time-point* $P?$ When the agent executes $P?$, the environment sets the value of p. Since these values are only observed in real time, the agent aims to satisfy all relevant constraints no matter how the conditions turn out during execution [21, 25]. Hence, for a CSTN, the agent typically uses a *dynamic* execution strategy with respect to the uncontrollable condition values [22].

A *Conditional Simple Temporal Network with Uncertainty* generalizes both STNUs and CSTNs in order to deal with both kinds of uncertainties simultaneously [21].

For temporal networks with uncontrollable features (e.g., STNUs, CSTNs and CSTNUs), dynamic properties, named *dynamic controllability* or *dynamic consistency*, have been studied and some related algorithms have been proposed. These properties specify whether it is possible to execute all time-points under the agent's control while satisfying all relevant constraints, by dynamically reacting to the uncertainties (whether the duration of a contingent link or the setting of a boolean value) as they are revealed in real time.

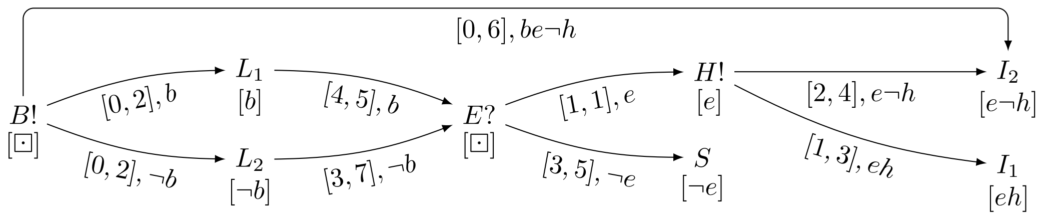


Figure 1 A CSTN with Decisions modeling the example discussed in the text. The propositions b, e and h correspond to the time-points $B!, E?$ and $H!$, respectively.

Conrad and Williams [14] present *Drake,* a dynamic executive for temporal plans that include discrete choices. In Drake, the executive sets the values for the propositional letters – hence the name *choice* – and the goal of the system is to both schedule events and make discrete choices as the execution unfolds. The ability to make discrete choices enriches an executive by offering it the ability to order activities, and choose between alternate methods (sub-plans) for achieving goals. Consistency analysis aims to determine a set of choices that will enable the executive to satisfy the relevant constraints.

Thus, STNUs, CSTNs and CSTNUs address uncontrollable parts, whereas Drake addresses controllable parts only. No temporal network discussed so far has addressed the arising interplay that occurs when controllable and uncontrollable conditions may mutually influence one another. This paper focuses on this issue and makes the following contributions.

1. A new model, *Conditional Simple Temporal Network with Decision* (CSTND), that accommodates contingent propositional variables (conditions) and *controllable propositional variables* (decisions). During execution, the decisions made by the agent, together with the conditions specified by the environment, determine the unfolding scenario.
2. A proof that the decision problem of establishing whether or not any CSTND is dynamically consistent is PSPACE-complete.
3. Algorithms addressing two sub-classes of CSTNDs: (i) those containing decisions only, and (ii) those in which all decisions must be set before starting to execute the network.

2 Motivating Example

Fig. 1 depicts a simplification of an example from the healthcare domain. The nodes in the graph represent time-points; the edges represent constraints. Time-points and constraints are relevant whenever their propositional labels are consistent with the unfolding scenario. Temporal ranges are in minutes. For instance, the annotation $[0,2], b$ on the edge from $B!$ to L_1 represents that in scenarios where b is true, the difference $L_1 - B!$ must be in the interval $[0,2]$ (i.e., L_1 must be executed between zero and two minutes after $B!$).

The plan applies to patients suffering from hematological diseases. It starts by having a patient's blood tested. There are two labs in the hospital, $\texttt{Lab}_1$ and $\texttt{Lab}_2$, but only one of them will analyze the blood sample, to be determined by the value the executing agent assigns to b, when $B!$ is executed. If b is $\top$ (resp., $\bot$), then $\texttt{Lab}_1$ (resp., $\texttt{Lab}_2$) will analyze the sample. The execution of either L_1 or L_2 models this task. Depending on the result of the blood test, a physician (who represents the environment in this example) evaluates whether the patient needs urgent care. This evaluation is represented by the time-point $E?$. The environment setting $e = \bot$ represents that the physician determined that the patient does not need urgent care. In that case, the plan concludes with a standard treatment, represented by the execution of S. However, if $e = \top$, then a hospitalization process is engaged, represented by the time-points $H!, I_1$ and I_2. There are two Intensive Care Units in the hospital, $\texttt{ICU}_1$ and $\texttt{ICU}_2$, but only one of them will be used, depending on the value for h, which is determined when $H!$ is executed. If $h = \top$ (resp., $h = \bot$), then the patient is hospitalized in $\texttt{ICU}_1$ (resp., $\texttt{ICU}_2$). Note that if $b = e = \top$, and $h = \bot$, then I_2 must be executed no more than 6 minutes after $B!$, the time-point modeling the start of the plan.

Our goal is to determine whether such networks are dynamically consistent. Note that if the values of b and h were set by the environment, then the network in Fig. 1 would be inconsistent. For example, in the scenario $be\neg h$, the lower bound of I_2 is 7, which is inconsistent with the constraint $B! \xrightarrow{[0,6],\, be\neg h} I_2$. However, if b and h are set by the agent, then the plan is dynamically consistent. Indeed, there are two significant possibilities: if $b = \bot$ and $e = \top$, then h can be $\top$ or $\bot$ without any problem; but if $b = e = \top$, then h must be $\top$, because the scenario $be\neg h$ would introduce the same inconsistency discussed above.

3 Conditional Simple Temporal Network with Decisions

This section introduces the formal definitions of a *Conditional Simple Temporal Network with Decisions* (CSTND) and the corresponding *dynamic consistency* property. We begin by recalling *Simple Temporal Networks* (STNs) [16], a well-known model for representing and reasoning about temporal constraints. An STN is a pair $(\mathcal{T}, \mathcal{C})$, where $\mathcal{T}$ is a set of real-valued variables, called *time-points,* and $\mathcal{C}$ is a set of binary constraints on those variables, each having the form, $(Y - X \leq \delta)$, where $X, Y \in \mathcal{T}$ and $\delta \in \mathbb{R}$. A constraint $(Y - X = \delta)$ can be represented by the constraints, $(Y - X \leq \delta)$ and $(X - Y \leq -\delta)$. The *Simple Temporal Problem* (STP) is that of determining whether an STN is consistent (i.e., has a solution).

Tsamardinos et al. [25] extended STNs to include time-points and temporal constraints that apply only in certain *scenarios*, where each scenario is represented by a conjunction of propositional literals. In their work, each time-point has a *label* that concisely specifies the scenarios in which that time-point must be executed. During execution, the execution of so-called *observation time-points* non-deterministically generates truth values for the corresponding propositional variables. Thus, the scenario is incrementally revealed. Later, Hunsberger et al. [21, 22] augmented their model to also allow constraints to have labels that specify the scenarios in which they must be satisfied. The result was a *Conditional STN* (CSTN). A CSTN is called *dynamically consistent* if there is a strategy for executing its time-points such that all relevant constraints will be satisfied no matter which scenario is incrementally revealed. They also formalized several well-definedness properties that had been only informally expressed in the earlier work.

This paper generalizes a CSTN by allowing some of the propositional variables to be assigned values not by the environment, but by the agent executing the network.

▶ **Definition 1** (Label). Let $\mathcal{P}$ be a set of propositional letters. A *label* ℓ over $\mathcal{P}$ is a (possibly empty) conjunction, $\ell = l_1 \wedge \cdots \wedge l_k$, of (positive or negative) literals $l_i \in \{p_i, \neg p_i\}$ on distinct variables $p_i \in \mathcal{P}$. The empty label is denoted by $\boxdot$. $\mathcal{P}^*$ denotes the set of all labels over $\mathcal{P}$.

▶ **Definition 2** (CSTND). A *Conditional Simple Temporal Network with Decisions* (CSTND) is a tuple $\Gamma = \langle \mathcal{T}, \mathcal{P}, \mathcal{CP}, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ where:

- $\mathcal{T}$ is a finite set of *temporal variables* or *time-points*;
- $\mathcal{P}$ is a finite set of *propositional letters/variables*;
- $(\mathcal{CP}, \mathcal{DP})$ is a partition of $\mathcal{P}$ into *contingent propositional variables (conditions)* $\mathcal{CP}$ and *controllable propositional variables (decisions)* $\mathcal{DP}$;

- $\mathcal{C}$ is a finite set of *labeled constraints,* each of the form, $(Y - X \leq \delta, \ell)$, where $X, Y \in \mathcal{T}$, $\delta \in \mathbb{R}$, and $\ell \in \mathcal{P}^*$;
- $\mathcal{OT} \subseteq \mathcal{T}$ is the set of *disclosing time-points;* and
- $\mathcal{O} \colon \mathcal{P} \to \mathcal{OT}$ is a bijection that associates each propositional variable $p \in \mathcal{P}$ to a disclosing time-point $\mathcal{O}(p) \in \mathcal{OT}$. If $p \in \mathcal{CP}$, then its disclosing time-point is called an *observation time-point;* but if $p \in \mathcal{DP}$, its disclosing time-point is called a *decision time-point.*

When an observation time-point is executed, the corresponding propositional variable (condition) is assigned a truth value by the environment; however, when a decision time-point is executed, the corresponding variable (decision) is assigned a truth value by the agent. To highlight the correspondence between propositional variables and their disclosing time-points, a decision time-point for any variable p is notated as $P!$, while an observation time-point for any variable q is notated as $Q?$, as illustrated in Fig. 1.

It is worth pointing out that in the definition of CSTND only constraints are labeled, not time-points. This restriction does not limit the expressivity of CSTNDs, as it has been recently shown that labeling constraints is sufficient to represent any possible CSTN [3]. Moreover, since time-points in CSTNDs do not have labels, the well-definedness properties formalized by Hunsberger et al. [22] become vacuous, which simplifies subsequent definitions.

To facilitate defining the dynamic consistency property for CSTNDs, we refine the supporting definitions of scenarios, projections and schedules to distinguish condition variables from decision variables.

▶ **Definition 3** (Scenario). A (combined) *scenario* over $\mathcal{P}$ is a total assignment $s \colon \mathcal{P} \to \{\bot, \top\}$ of truth values to propositional variables. Each scenario s also determines a truth value for each label $\ell \in \mathcal{P}^*$. If $s(\ell) = \top$, we may write $s \vDash \ell$. The set of all scenarios over $\mathcal{P}$ is denoted by $\Sigma_{\mathcal{P}}$. When a scenario is restricted to the subset $\mathcal{CP} \subseteq \mathcal{P}$ of conditions, then it may be called a *condition scenario;* similarly, when a scenario is restricted to the subset $\mathcal{DP} \subseteq \mathcal{P}$ of decisions, it may be called a *decision scenario.* The sets of all condition and decision scenarios are denoted by $\Sigma_{\mathcal{CP}}$ and $\Sigma_{\mathcal{DP}}$, respectively.

A CSTND projection is an STN that contains the constraints applicable in a given scenario.

▶ **Definition 4** (Projection). Let $\Gamma = \langle \mathcal{T}, \mathcal{P}, \mathcal{CP}, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ be any CSTND, and let s be any (combined) scenario. The *projection* of Γ over s is the STN $\Gamma_s = (\mathcal{T}, \mathcal{C}_s)$ where $\mathcal{C}_s = \{Y - X \leq \delta \mid (Y - X \leq \delta, \ell) \in \mathcal{C} \text{ and } s \vDash \ell\}$.

▶ **Definition 5** (Schedule). A *schedule* over a set $\mathcal{T}$ of time-points is a total assignment $\psi \colon \mathcal{T} \to \mathbb{R}$ of real values to those time-points. Hereinafter, the value $\psi(X)$ will be notated as $[\psi]_X$. A schedule ψ over $\mathcal{T}$ is said to be *feasible* for an STN $(\mathcal{T}, \mathcal{C})$ if ψ satisfies all of the constraints in $\mathcal{C}$. The set of schedules over $\mathcal{T}$ is denoted by $\Psi_{\mathcal{T}}$.

The agent executing the network must (1) schedule time-points, and (2) assign values to decision variables. In addition, the agent is able to *react* in real time to conditions set by the environment. Therefore, we define a *two-part* execution *strategy,* as follows.

▶ **Definition 6** (Execution Strategy). A *temporal strategy* for a CSTND Γ is a function $\sigma^{\mathrm{t}} \colon \Sigma_{\mathcal{CP}} \to \Psi_{\mathcal{T}}$ that maps each *condition scenario* $cs \in \Sigma_{\mathcal{CP}}$ to a (complete) schedule $\sigma^{\mathrm{t}}(cs)$ over $\mathcal{T}$. A *decision strategy* for Γ is a function $\sigma^{\mathrm{d}} \colon \Sigma_{\mathcal{CP}} \to \Sigma_{\mathcal{DP}}$ that maps each condition scenario $cs \in \Sigma_{\mathcal{CP}}$ to a decision scenario $ds = \sigma^{\mathrm{d}}(cs) \in \Sigma_{\mathcal{DP}}$. An *execution strategy* is a pair $\sigma = (\sigma^{\mathrm{t}}, \sigma^{\mathrm{d}})$ where σ^{t} is a temporal strategy and σ^{d} is a decision strategy. An *execution strategy* $\sigma = (\sigma^{\mathrm{t}}, \sigma^{\mathrm{d}})$ is *viable* if, for every condition scenario $cs \in \Sigma_{\mathcal{P}}$, letting $ds = \sigma^{\mathrm{d}}(cs)$ and $s = cs \cup ds$, the schedule $\sigma^{\mathrm{t}}(cs)$ is feasible for the projection Γ_s.

To ensure that the schedules and decisions generated by an execution strategy only depend on *past* observations, a *dynamic* execution strategy is subject to restrictions expressed in terms of *condition scenario histories,* as follows.

▶ **Definition 7** (Condition Scenario History). Given a temporal strategy σ^{t}, a condition scenario $cs \in \Sigma_{\mathcal{CP}}$, and a time value $t \in \mathbb{R}$, the *condition scenario history* at t in the condition scenario cs for the temporal strategy σ^{t} – notated as $scHst(t, cs, \sigma^{\mathrm{t}})$ – is the set of contingent variable assignments made by the environment *before* time t according to the schedule $\sigma^{\mathrm{t}}(cs)$: $scHst(t, cs, \sigma^{\mathrm{t}}) = \{(p, cs(p)) \mid p \in \mathcal{CP} \text{ and } [\sigma^{\mathrm{t}}(cs)]_{\mathcal{O}(p)} < t\}$.

▶ **Definition 8** (Dynamic Execution Strategy). A temporal strategy σ^{t} is *dynamic* if for any pair of condition scenarios $cs \in \Sigma_{\mathcal{CP}}$ and $cs' \in \Sigma_{\mathcal{CP}}$, and any time-point $X \in \mathcal{T}$:

let: $t = [\sigma^{\mathrm{t}}(cs)]_X$,
if: $scHst(t, cs, \sigma^{\mathrm{t}}) = scHst(t, cs', \sigma^{\mathrm{t}})$,
then: $[\sigma^{\mathrm{t}}(cs')]_X = t$.

Similarly, a decision strategy σ^{d} is *dynamic* if for any condition scenarios $cs \in \Sigma_{\mathcal{CP}}$ and $cs' \in \Sigma_{\mathcal{CP}}$, and any decision variable $p \in \mathcal{DP}$:

let: $t = [\sigma^{\mathrm{t}}(cs)]_{\mathcal{O}(p)}$,
if: $scHst(t, cs, \sigma^{\mathrm{t}}) = scHst(t, cs', \sigma^{\mathrm{t}})$,
then: $\sigma^{\mathrm{d}}(cs)(p) = \sigma^{\mathrm{d}}(cs')(p)$.

An execution strategy is *dynamic* if its temporal and decision strategies are both dynamic.

Now, it is possible to formally introduce the concept of dynamic consistency for CSTNDs.

▶ **Definition 9** (Dynamic Consistency). A CSTND is *dynamically consistent* (DC) if it admits a dynamic and viable execution strategy. The *CSTND-DC problem* is that of checking whether any given CSTND is dynamically consistent.

4 Computational Complexity of the CSTND-DC Problem

This section shows that the CSTND-DC problem is PSPACE-complete.

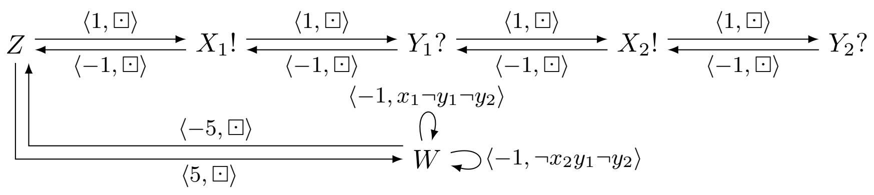
4.1 PSPACE-hardness

Cairo and Rizzi [4] showed that the DC-checking problem for CSTNs is PSPACE-hard. This section presents a simpler proof of that result by providing a direct reduction from the Quantified Boolean Formula (QBF) problem to the CSTND-DC problem.

Consider any quantified boolean formula of the form, $\Phi = \exists x_1 \forall y_1 \cdots \exists x_n \forall y_n\ \varphi$, where φ is a formula in conjunctive normal form, each clause of which is limited to at most three literals over a finite set of propositional variables $x_1, y_1, \ldots, x_n, y_n$ (i.e., φ is a 3SAT formula). More specifically, φ is a conjunction of the form $\bigwedge_{j=1}^{m}(l_{j,1} \vee l_{j,2} \vee l_{j,3})$, where each literal $l_{j,k}$ is either a positive or negative instance of one of the quantified variables.

The reduction involves the construction of a corresponding CSTND instance Γ such that Γ is DC if and only if Φ is true. To begin, define the sets of decision and condition variables to be $\mathcal{DP} = \{x_1, \ldots, x_n\}$ and $\mathcal{CP} = \{y_1, \ldots, y_n\}$, respectively. Thus, the disclosing time-points of the network are given by $\mathcal{OT} = \{X_1!, \ldots, X_n!, Y_1?, \ldots, Y_n?\}$. The only other time-point, W, is discussed below. Thus, $\mathcal{T} = \mathcal{OT} \cup \{W\}$. Finally, the set of constraints is given by:

$$\mathcal{C} = \Big(\bigcup_{i=1,\ldots,n} \{(X_i! - Y_i? = -1, \boxdot), (Y_i? - X_{i+1}! = -1, \boxdot)\}\Big) \cup$$
$$\Big(\bigcup_{i=1,\ldots,n} \{(W - W \leq -1, \neg l_{i,1} \wedge \neg l_{i,2} \wedge \neg l_{i,3})\}\Big) \cup \{X_0! = 1, W = 2n+1\}$$



Z
$\langle 1, \boxdot \rangle$
$\langle -1, \boxdot \rangle$
$X_1!$
$\langle 1, \boxdot \rangle$
$\langle -1, \boxdot \rangle$
$Y_1?$
$\langle 1, \boxdot \rangle$
$\langle -1, \boxdot \rangle$
$X_2!$
$\langle 1, \boxdot \rangle$
$\langle -1, \boxdot \rangle$
$Y_2?$
$\langle -1, x_1 \neg y_1 \neg y_2 \rangle$
$\langle -5, \boxdot \rangle$
W
$\langle -1, \neg x_2 y_1 \neg y_2 \rangle$
$\langle 5, \boxdot \rangle$

Figure 2 The CSTND obtained from the quantified boolean 3SAT formula

$$\Phi \equiv \exists x_1 \forall y_1 \exists x_2 \forall y_2 (\neg x_1 \vee y_1 \vee y_2) \wedge (x_2 \vee \neg y_1 \vee y_2)\,.$$

The constraints, $(X_i! - Y_i? = -1, \boxdot)$ and $(Y_i? - X_{i+1}! = -1, \boxdot)$ for $i \in \{1, \ldots, n\}$, impose the order $X_1! < Y_1? < \ldots < X_n! < Y_n?$ on the disclosing time-points, which mirrors the order of the alternating quantifiers $x_1, y_1, \ldots, x_n, y_n$ in Φ. Furthermore, together with the constraint, $X_0! = 1$, they ensure that any viable temporal strategy must make the *fixed* assignments, $X_0! = 1, Y_0? = 2, \ldots, X_n! = 2n - 1, Y_n? = 2n$, across all (combined) scenarios. Note that any such temporal strategy is trivially dynamic.

The constraints of the form, $(W - W \leq -1, \neg l_{i,1} \wedge \neg l_{i,2} \wedge \neg l_{i,3})$ are negative self-loops on the extra time-point W, each of which has a label that is the logical negation of one of the clauses in the boolean formula φ.

Fig. 2 depicts a simple example of the reduction from a QBF to the corresponding CSTND, where $\Phi \equiv \exists x_1 \forall y_1 \exists x_2 \forall y_2 (\neg x_1 \vee y_1 \vee y_2) \wedge (x_2 \vee \neg y_1 \vee y_2)$.

In any (combined) scenario cs, the labeled negative self-loops at W are satisfied if and only if cs assigns the value $\bot$ to each of their labels. That happens if and only if the formula φ evaluates to $\top$ in the scenario cs. Therefore, Γ admits a viable execution strategy if and only if Φ is true.

Given the fixed ordering of the disclosing time-points, a decision strategy is dynamic if and only if the assignment of each decision variable x_i depends only on the *preceding* condition variables, $y_1, \ldots, y_{i-1}$. This mirrors the semantics of the nested quantifiers, where each x_i is existentially quantified, and each preceding y_j is universally quantified. Thus, Γ has a dynamic execution strategy if and only if Φ is true.

4.2 A Polynomial-Space Algorithm for the CSTND-DC Problem

This section presents a polynomial-space algorithm for the CSTND-DC problem, which extends the algorithm for CSTNs from prior work [4], assuming that time is discretized. As already discussed in [4], such assumption does not limit the generality of the algorithm. Together with the PSPACE-hard result from the preceding section, this proves that the CSTND-DC problem is PSPACE-complete.

We begin by showing that, under certain conditions, a DC CSTND admits a *discrete* execution strategy (i.e., one that only schedules time-points at rational multiples of some *fixed* real number ϵ, where the granularity of the rational factors is bounded). This result, whose proof is in the Appendix, adapts a similar result for CSTNs from prior work [4].

▶ **Lemma 10.** *Suppose that Γ is a CSTND such that, for some $\epsilon \in \mathbb{R}^+$ and $W \in \mathbb{Z}^+$, each constraint $(Y - X \leq w)$ in Γ satisfies $w = k\epsilon$ for some $k \in \mathbb{Z}$, where $-W < k < W$. If Γ is dynamically consistent, then Γ admits a viable and dynamic execution strategy $\sigma = (\sigma^{\mathrm{t}}, \sigma^{\mathrm{d}})$ such that for each scenario cs and each $X \in \mathcal{T}$, $[\sigma^{\mathrm{t}}(cs)]_X = k'\epsilon/K$, for some $k' \in \{0, 1, \ldots, 2K^2W\}$, where $K = |\mathcal{T}| \cdot 2^{|\mathcal{P}|}$.*

Algorithm 1: CSTND Dynamic Consistency checking in polynomial space

```
1  Function DC($\Gamma$)
     Input   : $\Gamma$ is a CSTND satisfying the conditions of Theorem 10 for some values $\epsilon > 0$ and
               $W \in \mathbb{Z}^+$
     Returns : true if $\Gamma$ is dynamic consistent, false otherwise
2    $c_0 \leftarrow (0, \emptyset, \emptyset)$                                        ▷ Initial configuration
3    return DC-From($\Gamma, c_0$)

4  Recursive Function DC-From($\Gamma, c$)
     Input   : $\Gamma$ is a CSTND satisfying the conditions of Theorem 10 for some values $\epsilon > 0$ and
               $W \in \mathbb{Z}^+$, $c = (k\epsilon/K, \psi, h)$ is a configuration with $k \in \{0, 1, \dots, 2K^2W\}$ and
               $K = 2^{|\mathcal{P}|} \cdot |\mathcal{T}|$.
     Returns : true if $\Gamma$ is dynamically consistent from $c$, false otherwise
5    if Dom($\psi$) $= \mathcal{T}$ then return (true if $\psi$ is feasible for $\Gamma_h^+$ else false)   ▷ Base case
6    foreach $\mathcal{T}_{next} \subseteq \mathcal{T} \setminus$ Dom($\psi$) not empty do  ▷ Recursive case. Enumerate all next actions ($\exists$)
7      $\mathcal{CP}_{next} \leftarrow \{p \in \mathcal{CP} \mid \mathcal{O}(p) \in \mathcal{T}_{next}\}$
8      $\mathcal{DP}_{next} \leftarrow \{p \in \mathcal{DP} \mid \mathcal{O}(p) \in \mathcal{T}_{next}\}$
9      foreach $k_{next} \in \{k + 1, \dots, 2K^2W\}$ do              ▷ Enumerate all discrete times ($\exists$)
10       $t_{next} \leftarrow k_{next}\,\epsilon$
11       $\psi' \leftarrow \psi[t_{next}/\mathcal{T}_{next}]$
12       foreach $d\colon \mathcal{DP}_{next} \to \{\top, \bot\}$ do               ▷ Enumerate all decisions ($\exists$)
13         AllObsDC $\leftarrow$ true
14         foreach $o\colon \mathcal{CP}_{next} \to \{\top, \bot\}$ do          ▷ Enumerate all observations ($\forall$)
15           $h' \leftarrow h \cup o \cup d$
16           $c' \leftarrow (t_{next}, \psi', h')$
17           if not DC-From($\Gamma, c'$) then AllObsDC $\leftarrow$ false            ▷ Recursion
18         if AllObsDC then return true
19   return false
```

The execution of a CSTND can be viewed as a two-player game between the agent and the environment. The agent aims to make decisions and schedule time-points to satisfy all relevant constraints; the environment aims to assign values to conditions that will thwart the agent. Our DC-checking algorithm, whose pseudo-code is given in Algorithm 1, recursively explores all possible configurations of the game. Each configuration is a tuple $c = (t, \psi, h)$, where t is the current time, $\psi\colon \mathcal{T}' \to [0, t)$ (with $\mathcal{T}' \subseteq \mathcal{T}$) is a partial schedule, and $h\colon \mathcal{P}' \to \{\top, \bot\}$ (with $\mathcal{P}' = \{p \in \mathcal{P} \mid \mathcal{O}(p) \in \text{Dom}(\psi)\}$) is the partial (combined) scenario known at time t.

By Lemma 10, we may assume that each t satisfies $t = k{\cdot}\epsilon/K$ for some $k \in \{0, 1, \dots, 2K^2W\}$ and $[\psi]_X = k_X \cdot \epsilon/K$, where $k_X \in \{0, \dots, k-1\}$ for every $X \in \mathcal{T}$.

The base case of our recursive procedure corresponds to a final configuration of the game, which occurs when all the time-points have been executed (i.e., $\mathcal{T}' = \mathcal{T}$) and, hence, all propositional variables have been assigned. The agent wins if and only if the total schedule ψ is feasible for the projected network Γ_h^+ over the total combined scenario h.

In the recursive step, the algorithm enumerates all possible moves of the agent and all possible counter-moves of the environment. A move of the agent consists of:

1. the set $\mathcal{T}_{next} \subseteq \mathcal{T} \setminus \text{Dom}(\psi)$ of time-points to execute next,
2. the time $t' > t$ at which these time-points are executed, and
3. the decisions to take at time t' (i.e., an assignment $d\colon \mathcal{DP}_{next} \to \{\top, \bot\}$ of the variables in $\mathcal{DP}_{next} = \{p \in \mathcal{DP} \mid \mathcal{O}(p) \in \mathcal{T}_{next}\}$, which must be decided at time t'.

Thanks to Lemma 10, we can assume $t' = k' \cdot \epsilon/K$ for $k' \in \{k + 1, \dots, 2K^2W\}$. This ensures that there are a finite number of possible moves, and that each move can be described with a polynomial number of bits.

The counter-move of the environment sets the values for the conditions to be revealed to the agent at time t' (i.e., an assignment $o\colon \mathcal{CP}_{next} \to \{\top, \bot\}$ of the propositional variables in $\mathcal{CP}_{next} = \{p \in \mathcal{CP} \mid \mathcal{O}(p) \in \mathcal{T}_{next}\}$ observed at time t'.

Checking the dynamic consistency of a network amounts to determining the winner of the game from the initial configuration $c_0 = (0, \emptyset, \emptyset)$, where the current time is 0 and no time-points have yet been executed.

5 An Algorithm for CSTNDs having no Condition

In this section we propose an algorithm for solving the DC problem for CSTNDs having no contingent propositional variable, i.e., when $\mathcal{CP}$ is empty. Hereinafter we refer to this subclass of CSTND as the class of Simple Temporal Network with Decisions (STND).

▶ **Definition 11** (STND). A *Simple Temporal Network with Decisions* (STND) is a CSTND $\Gamma = \langle \mathcal{T}, \mathcal{P}, \emptyset, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ having $\mathcal{CP} = \emptyset$. An STND Γ is *consistent* if and only if there exists a decision scenario $ds \in \Sigma_{\mathcal{DP}}$ for which the projection STN Γ_{ds} is consistent.

Since contingent propositional variables are not present, it is possible to check the consistency of the network in a static way. The approach we are about to discuss synthesizes such an assignment offline, i.e., before the execution of the STND starts.

Basically, the proposed algorithm (Algorithm 2) checks whether an STND Γ is consistent by looking for one $ds \in \Sigma_{\mathcal{DP}}$ for which the projection STN Γ_{ds} is consistent. This search is performed by maintaining a support CNF ϕ, which, roughly speaking, represents the complementary of the space of all decision scenarios for which Γ is already known to be inconsistent; to satisfy ϕ means to find a decision scenario outside that space.

Therefore, Algorithm 2 (`STND-CC`) takes as input an STND Γ and it employs an approach working in rounds. Throughout the rounds, `STND-CC` maintains a formula ϕ in CNF representing all decision scenarios for which Γ is already known to be inconsistent. `STND-CC` keeps trying to guess a decision scenario $ds \in \Sigma_{\mathcal{DP}}$ until either $ds \not\models \phi$, or $ds \models \phi$ and Γ_{ds} is consistent – in such case it returns YES. If Γ_{ds} is inconsistent for all possible decision scenario, `STND-CC` returns NO.

Initially, ϕ contains no clauses and ds is a random decision scenario; thus, $ds \models \phi$ holds trivially. If Γ_{ds} is consistent, then `STND-CC` outputs `YES` and halts. Otherwise, ds contains at least a "bad decision" making Γ_{ds} inconsistent, i.e., Γ_{ds} contains a negative cycle. Let ρ be such a negative cycle and let ψ be the conjunction of the labels associated to the values making the negative cycle (see Algorithm 3). Then, `STND-CC` (i) augments ϕ adding $\neg(\psi)$ (which is a disjunction of literals) as a new clause (line 9), (ii) determines a new ds that satisfies ϕ using an external SAT solver, and (iii) proceeds to the next round if a suitable ds has been found, outputs `NO` and halts otherwise.

`STND-CC` uses a SAT solver as MiniSat [18] to find a decision scenario ds that satisfies (the augmented) ϕ. We underline that a decision scenario cannot be found when ϕ is unsatisfiable, i.e., when Γ is inconsistent. In this case, the algorithm can stop saying `NO`.

It is not difficult to see that `STND-CC` is sound and complete. As regards time complexity, the following facts hold. Checking the consistency of Γ_{ds} (line 5 of `STN-CC`) requires time $O(|\mathcal{T}| \cdot |\mathcal{C}|)$ using Bellman-Ford algorithm [16]. The time for determining a negative cycle ρ of Γ_{ds} (`CYCLE-CUT`) amounts to that of applying De Morgan's law to $\neg(\psi)$, which is a linear time in $|\mathcal{DP}|$. Finally, the number of invocations to the SAT Solver (line 11 of Algorithm 2) is at most that of all possible considered decision scenarios s, i.e., $2^{|\mathcal{DP}|}$; each of such invocations costs $O(2^{|\mathcal{DP}|})$ time. Therefore, the worst-case time complexity of `STND-CC` is $O(2^{2|\mathcal{DP}|} \cdot (|\mathcal{T}| \cdot |\mathcal{C}| + |\mathcal{DP}|))$.

Algorithm 2: Consistency Checking Algorithm with Decisions

```
Procedure STND-CC($\Gamma$)
    input : An STND $\Gamma = \langle \mathcal{T}, \mathcal{P}, \emptyset, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$.
    output : A feasible schedule of $\Gamma$ or NO.
1   $\phi \leftarrow$ an empty set of clauses on $\mathcal{DP}$;
2   $ds \leftarrow$ any assignment on $\mathcal{DP}$;          ▷ Initialize the assignment $ds$ arbitrarily
3   while TRUE do
4       $\Gamma_{ds} \leftarrow$ the projection of $\Gamma$ over $ds$;
5       $\rho \leftarrow$ STN-CC($\Gamma_{ds}$);          ▷ Check the consistency of $\Gamma_{ds}$
6       if $\rho$ is a feasible schedule of $\Gamma_{ds}$ then
7           return (YES, $\rho$);
8       if $\rho$ is a negative cycle of $\Gamma_{ds}$ then
9           $\psi \leftarrow$ CYCLE-CUT($\Gamma, \rho$);          ▷ Derive a clause $\psi$ expressing cut of $\rho$ in $\Gamma$
10          $\phi \leftarrow \phi \cup \{\psi\}$;          ▷ Add the clause $\psi$ to the CNF $\phi$
11          $ds \leftarrow$ SAT-SOLVE($\phi$);          ▷ Invoke a SAT-Solver on input $\phi$
12          if $ds \not\models \phi$ then
13              return NO;
```

Algorithm 3: Cutting a Cycle with Decisions

```
Procedure CYCLE-CUT($\Gamma, \rho$)
    input : An STND $\Gamma = \langle \mathcal{T}, \mathcal{P}, \emptyset, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ and one of its cycles $\rho$.
    output : A clause $\psi$ on $\mathcal{DP}$ expressing the cut of $\rho$ in $\Gamma$.
1   $\psi \leftarrow \top$;
2   foreach constraint $C$ of $\rho$ do
3       $\ell_C \leftarrow$ the label of $C$ in $\Gamma$;          ▷ $\ell_C$ is a conjunction of literals
4       $\psi \leftarrow \psi \wedge \ell_C$;          ▷ $\psi$ is also a conjunction of literals
5   $\psi \leftarrow$ DeMorgan($\neg\psi$);          ▷ apply De Morgan's law to $\neg\psi$
6   return $\psi$;
```

▶ **Theorem 12.** *The problem of checking whether a given STND $\Gamma = \langle \mathcal{T}, \mathcal{P}, \emptyset, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ is consistent can be solved in singly exponential (w.r.t. $|\mathcal{DP}|$) deterministic time. Moreover, when Γ is consistent, a positive certificate (ds, ρ) (where ds is a decision scenario over $\mathcal{DP}$ and ρ is a feasible schedule of Γ_{ds}) is computable within the same time bound.*

To complete the result of Theorem 12, we prove that STND consistency checking problem is NP-complete. For lack of space, we sketch the proof of a polynomial reduction from 3-SAT to the checking problem. Let $\psi(x_1, \ldots, x_n) = \bigwedge_{j=1}^{m} C_j$ be a 3-CNF formula with n boolean variables $\{x_i\}_{i=1}^{n}$ and with m clauses $\{C_j\}_{j=1}^{m}$. An STND instance can be represented by a triplet $\langle \mathcal{T}, \mathcal{C}, \mathcal{DP} \rangle$, i.e., dropping out $\mathcal{OT}, \mathcal{O}$ and $\mathcal{P}$ $(= \mathcal{DP})$; this compact form already allows a correct consistency checking. Consider STND $\Gamma_\psi = (\mathcal{T} = \{Z\}, \mathcal{C} = \{\mathcal{C}_j\}_{j=1}^{m}, \mathcal{DP} = \{x_i\}_{i=1}^{n})$ having only one time-point Z and m negative self-loop constraints, i.e., $\mathcal{C}_j = \big(Z - Z \leq -1, \neg(C_j)\big)$ where $\neg(C_j)$ can be turned into a label by De Morgan's law. It is not difficult to verify that ψ is SAT if and only if Γ is consistent.

5.1 Hyper Temporal Networks with Decisions

In this subsection we consider the *Hyper Temporal Network with Decisions* (HyTND) model and we prove that Algorithm 2 and 3 can be easily extend to it. *Hyper Temporal Networks* (HyTNs) are a strict generalization of STNs, introduced to partially overcome the limitation of allowing only conjunctions of constraints [12]. Compared to STN distance graphs, HyTNs allow for a greater flexibility in the definition of the temporal constraints meanwhile offering a pseudo-polynomial tractability in the consistency checking of the instances. In turn, HyTNDs extend the HyTN model by labeling each hyper-constraint with a conjunction of literals drawn from the set $\mathcal{DP}$ of controllable propositional variables; then, the consistency problem of HyTNDs asks for the existence of a decision scenario for which the corresponding projection network is consistent.

In order to formally define the model, let us firstly recall (multi-head) hypergraphs.

▶ **Definition 13.** (Hypergraphs) A (multi-head, weighted) *hypergraph* $\mathcal{H}$ is a pair $(\mathcal{T}, \mathcal{HC})$, where $\mathcal{T}$ is a set of nodes and $\mathcal{HC}$ is a set of *hyper-edges*. Each hyper-edge $A = (t_A, H_A, w_A) \in \mathcal{HC}$ has a distinguished node t_A, called the *tail* of A, and a non-empty set $H_A \subseteq \mathcal{T} \setminus \{t_A\}$ comprising the *heads* of A; to each head $v \in H_A$, it is associated a *weight* $w_A(v) \in \mathbb{Z}$. Let W be the maximum absolute weight in $\mathcal{H}$ and $|A| = |H_A \cup \{t_A\}|$. The *size* of $\mathcal{H}$ is $m_{\mathcal{H}} = \sum_{A\in\mathcal{HC}} |A|$, and it is a measure for the encoding length of $\mathcal{H}$. If $|A| = 2$, then $A = (u, v, w)$ is a standard edge; in this way hypergraphs generalize graphs.

We are now in the position to define HyTNDs and related concepts.

▶ **Definition 14.** (HyTND) A *Hyper Temporal Network with Decisions* (HyTND) is a triplet $\Gamma = (\mathcal{T}, \mathcal{HC}, \mathcal{DP})$ where $\mathcal{H} = (\mathcal{T}, \mathcal{HC})$ is a hypergraph and $\mathcal{DP}$ is a set of controllable propositional variables. Nodes $T \in \mathcal{T}$ represent temporal variables (time-points), and each hyper-edge $A = (t_A, H_A, w_A, \ell_A) \in \mathcal{HC}$, where $w_A : H_A \to \mathbb{Z}$ and $\ell_A \in \mathcal{DP}^*$, represents a temporal distance constraint between the tail and the heads (*labeled hyper-constraint* (LHC))

In general, given any $ds \in \Sigma_{\mathcal{DP}}$ and $\psi : \mathcal{T} \to \mathbb{R}$, we say that A is satisfied by (ds, ψ) if and only if the following implication holds: $ds \models \ell_A \Rightarrow \psi(t_A) \geq \min_{v\in H_A} \big\{\psi(v) - w_A(v)\big\}$. Note that, when $\mathcal{DP} = \emptyset$, $(\mathcal{T}, \mathcal{HC})$ becomes an *Hyper Temporal Network* (HyTN), and each hyper-edge $A \in \mathcal{HC}$ represents an (unlabeled) temporal distance which is satisfied by any given $\psi : \mathcal{T} \to \mathbb{R}$ if and only if $\psi(t_A) \geq \min_{v\in H_A}\{\psi(v) - w_A(v)\}$. We recall that the HyTN-Consistency problem asks, given any HyTN $\mathcal{H} = (\mathcal{T}, \mathcal{HC})$, to decide whether there exists a schedule $\psi : \mathcal{T} \to \mathbb{R}$ that satisfies every hyper-constraint $A \in \mathcal{HC}$; if so, $\mathcal{H}$ is said *consistent*. HyTND-Consistency is also a static notion of consistency, i.e., all decisions can be taken offline; so, for ease of notation, it is fine to omit decision time-points in Definition 14.

▶ **Definition 15.** (HyTND Projection, HyTND-Consistency) The *projection* of a HyTND $\Gamma = (\mathcal{T}, \mathcal{HC}, \mathcal{DP})$ over a decision scenario $ds \in \Sigma_{\mathcal{DP}}$ is the HyTN $\Gamma_{ds} = (\mathcal{T}, \mathcal{HC}_{ds})$, where:

$$\mathcal{HC}_{ds} = \Big\{(t_A, H_A, w_A) \mid \exists \ell_A \in \mathcal{DP}^* \text{ s.t. } (t_A, H_A, w_A, \ell_A) \in \mathcal{HC} \text{ and } ds \models \ell_A\Big\}.$$

The HyTND-Consistency problem asks, given any HyTND $\Gamma = (\mathcal{T}, \mathcal{HC}, \mathcal{DP})$, to decide whether there exists a decision scenario $ds \in \Sigma_{\mathcal{DP}}$ such that the projection Γ_{ds} is consistent; if so, Γ is said *consistent* as well.

As a negative cycle is a negative certificate for consistency check in STN, the *generalized negative cycle* is a negative certificate for HyTN and HyTND [12].

▶ **Definition 16** (Generalized (negative) cycle)**.** Given a HyTN $\mathcal{H} = (\mathcal{T}, \mathcal{HC})$, a *cycle* is a pair $(S, \mathcal{C})$ with $S \subseteq \mathcal{T}$ and $\mathcal{C} \subseteq \mathcal{HC}$ such that:

1. $S = \bigcup_{A\in\mathcal{C}} \big(H_A \cup \{t_A\}\big)$ and $S \neq \emptyset$;
2. $\forall v \in S$ there exists an unique $A \in \mathcal{C}$ such that $t_A = v$.

Moreover, we let $a(v)$ denote the unique edge $A \in \mathcal{C}$ with $t_A = v$, as required in above item 2. Every infinite path in a cycle $(S, \mathcal{C})$ contains, at least, one *finite cyclic sequence* $v_i, v_{i+1}, \ldots, v_{i+p}$, where $v_{i+p} = v_i$ is the only repeated node in the sequence. A cycle $(S, \mathcal{C})$ is *negative* if and only if $\sum_{t=1}^{p-1} w_{a(v_t)}(v_{t+1}) < 0$, for *any* finite cyclic sequence $v_1, v_2, \ldots, v_p$.

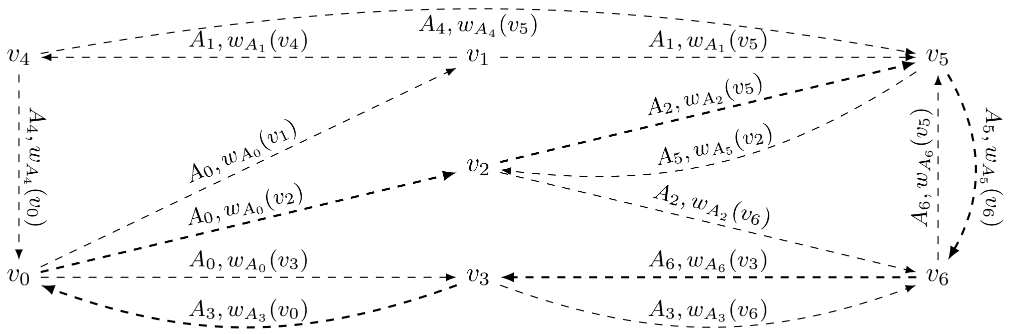


Figure 3 A (generalized) cycle $(S, \mathcal{C})$, where $S = \{v_0, \dots, v_6\}$ and $\mathcal{C} = \{A_0, \dots, A_6\}$.

▶ **Example 17.** An example of a cycle $(S, \mathcal{C})$ is shown in Fig. 3; where $S = \{v_0, \dots, v_6\}$ and $\mathcal{C} = \{A_0, \dots, A_6\}$, provided $t_{A_i} = v_i$ for every $i \in \{0, \dots, 6\}$; and $H_{A_0} = \{v_1, v_2, v_3\}$, $H_{A_1} = \{v_4, v_5\}$, $H_{A_2} = \{v_5, v_6\}$, $H_{A_3} = \{v_0, v_6\}$, $H_{A_4} = \{v_0, v_5\}$, $H_{A_5} = \{v_2, v_6\}$, $H_{A_6} = \{v_3, v_5\}$. Moreover, a finite cyclic sequence, $(v_0, v_2, v_5, v_6, v_3, v_0)$, is highlighted with thickened edges.

As shown in [12], checking HyTN-Consistency can be done in pseudo-polynomial time.

▶ **Theorem 18** ([12]). *Let $\mathcal{H} = (\mathcal{T}, \mathcal{HC})$ be a HyTN. The following propositions hold:*

1. *There exists an $O((|\mathcal{T}| + |\mathcal{HC}|)m_{\mathcal{H}}W)$ pseudo-polynomial time algorithm deciding* HyTN-Consistency *for $\mathcal{H}$;*
2. *There exists an $O((|\mathcal{T}| + |\mathcal{HC}|)m_{\mathcal{H}}W)$ pseudo-polynomial time algorithm such that, given as input any consistent HyTN $\mathcal{H}$, it returns a feasible schedule $s : \mathcal{T}_{\mathcal{H}} \to \mathbb{Z}$ of $\mathcal{H}$;*
3. *There exists an $O((|\mathcal{T}| + |\mathcal{HC}|)m_{\mathcal{H}}W)$ pseudo-polynomial time algorithm such that, given as input any inconsistent HyTN $\mathcal{H}$, it returns a negative cycle $(S, \mathcal{C})$ of $\mathcal{H}$.*

To solve HyTND-Consistency, one may as well apply Algorithm 2 and Algorithm 3, subject to the following simple modifications:

1. at line 5 of Algorithm 2, replace `STN-CC` by the HyTN-Consistency checking algorithm mentioned in Theorem 18 (see [12]);
2. at line 8 of Algorithm 2, replace "*cycle*" with "*generalized cycle*" and observe that checking whether a generalized cycle $(S, \mathcal{C})$ is negative can be done in polynomial time (see e.g., Lemma 3 in [12] for an algorithm);
3. at line 2 of Algorithm 3, replace "*constraint*" with "*hyper-constraint*" and notice that, since by Definition 14 each $A \in \mathcal{HC}$ has a unique label $\ell_A \in \mathcal{DP}^*$, it is still possible to apply De Morgan's law at line 5 of Algorithm 3 for obtaining a clause.

Considering such modification, it is not difficult to verify that the checking algorithm remains correct. Concerning time complexity, the only noticeable overhead is now due to Theorem 18 results. Considering such complexity results, the new worst-case time complexity for the checking algorithm is $O(2^{2^{|\mathcal{DP}|}} \cdot ((|\mathcal{T}| + |\mathcal{HC}|)m_{\mathcal{H}}W + |\mathcal{DP}|))$.

▶ **Theorem 19.** *The problem of checking whether or not a given HyTND $\Gamma = (\mathcal{T}, \mathcal{HC}, \mathcal{DP})$ is consistent can be solved in (pseudo) singly exponential (w.r.t. $|\mathcal{DP}|$) deterministic time. Moreover, when Γ is consistent, a positive certificate (ds, ρ) (where ds is a decision scenario over $\mathcal{DP}$ and ρ is a feasible schedule of Γ_{ds}) is computable within the same time bound.*

As STNDs are special cases of HyTNDs, checking HyTND-Consistency is NP-complete.

6 An Algorithm for CSTNDs having Offline Decisions

In this section we consider a special case of CSTNDs where decisions are made before the execution of the network starts (offline decisions). This allows us to apply the techniques developed in the previous Section 5. In this special case, dynamic consistency of CSTNDs is equivalent to the existence of a decision scenario such that the (partial) projection of the network over this scenario (now, a traditional CSTN) is in turn dynamically consistent.

▶ **Definition 20.** [Offline Decision DC] A CSTND $\Gamma = \langle \mathcal{T}, \mathcal{P}, \mathcal{CP}, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O}\rangle$ has the property of *Offline Decision Dynamic Consistency* (CSTND-od-DC) when there exists some decision scenario $ds \in \Sigma_{\mathcal{DP}}$ such that the projection CSTN Γ_{ds} is dynamically consistent.

We will also also say that a CSTND is CSTND-od-DC if it has CSTND-od-DC property.

To check if a CSTND instance is CSTND-od-DC, it is possible to re-use and adapt the techniques presented in Section 5. There, the problem is reduced to check the consistency of a STN instance (projected network), here the projected network is an CSTN. Therefore, it is necessary to consider a CSTN dynamic consistency checking algorithm as the constraint propagation algorithm for CSTNs proposed in [22] for testing the CSTND-od-DC property. An alternative way to check the CSTN dynamic consistency is to reduce an CSTN instance to an appropriate HyTN one and check the dynamic consistency of the last one [13].

An improvement of such approach, is to reduce the problem of CSTND-od-DC checking to the problem of HyTND-Consistency consistency check. We now present such reduction.

Firstly, we argue that in this prospect any CSTND can be viewed as a succinct representation which can be expanded into an exponentially sized HyTND. The following definition introduce the concept of *expansion* of a CSTND.

▶ **Definition 21** (Expansion $\langle \mathcal{T}^{\mathsf{Ex}}_{\Gamma}, \Lambda^{\mathsf{Ex}}_{\Gamma}\rangle$)**.** For any $\ell \in \mathcal{P}^*$, let us denote by $\ell_{\mathcal{DP}} \in \mathcal{DP}^*$ ($\ell_{\mathcal{CP}} \in \mathcal{CP}^*$) the label comprising all and only those literals of ℓ whose propositional variable lies in $\mathcal{DP}$ ($\mathcal{CP}$, respectively): $\ell = \ell_{\mathcal{DP}} \wedge \ell_{\mathcal{CP}}$ where $\ell_{\mathcal{DP}} \in \mathcal{DP}^*$ and $\ell_{\mathcal{CP}} \in \mathcal{CP}^*$.

For any $s \in \Sigma_{\mathcal{CP}}$, let us consider the *(partial) projection* $\mathcal{C}^+_s$ of $\mathcal{C}$ over s defined as $\mathcal{C}^+_s = \Big\{ \big(X, Y, \delta, \ell_{\mathcal{DP}}\big) \mid (Y - X \leq \delta, \ell) \in \mathcal{C} \text{ and } s \models \ell_{\mathcal{CP}} \Big\}$.

Next, let us consider the family of distinct and disjoint STNDs $(\mathcal{T}_s, \mathcal{C}_s, \mathcal{DP})$, where $\mathcal{T}_s = \{v_s \mid v \in \mathcal{T}, s \in \Sigma_{\mathcal{CP}}\}$, $\mathcal{C}_s = \Big\{ \big(X_s, Y_s, \delta, \ell_{\mathcal{DP}}\big) \,\Big|\, \big(X, Y, \delta, \ell_{\mathcal{DP}}\big) \in \mathcal{C}^+_s \Big\}$, and $v_s = (v, s)$ for every $v \in \mathcal{T}, s \in \Sigma_{\mathcal{CP}}$. For each condition scenario $s \in \Sigma_{\mathcal{CP}}$ there is one such STND.

Now, the *expansion STND* of the CSTND Γ is defined as

$$(\mathcal{T}^{\mathrm{Ex}}_{\Gamma}, \Lambda^{\mathrm{Ex}}_{\Gamma}, \mathcal{DP}), \text{ where } \mathcal{T}^{\mathrm{Ex}}_{\Gamma} = \bigcup_{s \in \Sigma_{\mathcal{CP}}} \mathcal{T}_s \text{ and } \Lambda^{\mathrm{Ex}}_{\Gamma} = \bigcup_{s \in \Sigma_{\mathcal{CP}}} \mathcal{C}_s.$$

Note that $\mathcal{T}_{s_1} \cap \mathcal{T}_{s_2} = \emptyset$ whenever $s_1 \neq s_2$ and that $(\mathcal{T}^{\mathrm{Ex}}_{\Gamma}, \Lambda^{\mathrm{Ex}}_{\Gamma}, \mathcal{DP})$ is an STND with $|\mathcal{T}^{\mathrm{Ex}}_{\Gamma}| \leq |\Sigma_{\mathcal{CP}}| \cdot |\mathcal{T}|$ time-points and size at most $|\Lambda^{\mathrm{Ex}}_{\Gamma}| \leq |\Sigma_{\mathcal{CP}}| \cdot |\mathcal{C}|$.

We show next that the expansion of a CSTND can be enriched with some (extra) multi-head hyper-edges in order to model dynamic consistency, by means of a particular HyTND $\mathcal{H}^{\Gamma}_{\epsilon}$ for some small $\epsilon \in \mathbb{R}_{>0}$. As it was in [13], the actual value of ϵ will turn out to be singly exponentially small in the number of contingent propositional variables (i.e., $\epsilon = \frac{1}{|\Sigma_{\mathcal{CP}}| \cdot |\mathcal{T}|}$).

▶ **Definition 22** (HyTND $\mathcal{H}^{\Gamma}_{\epsilon}$)**.** For any two condition scenarios $s_1, s_2 \in \Sigma_{\mathcal{CP}}$, let us denote by $\Delta(s_1; s_2) = \big\{ (\mathcal{O}_p?)_{s_1} \in \mathcal{T}_{s_1} \mid \mathcal{O}_p? \in \mathcal{OT}, p \in \mathcal{CP}, s_1(p) \neq s_2(p) \big\}$ the set of all nodes $(\mathcal{O}_p?)_{s_1} \in \mathcal{T}_{s_1}$ such that $\mathcal{O}_p? \in \mathcal{OT}$ is an observation time-point of Γ that is executed in s_1 and, considered in pair with respect to s_2, the value of the variable p differs, i.e., $s_1(p) \neq s_2(p)$.

Given any $\epsilon \in \mathbb{R}_{>0}$, HyTND $\mathcal{H}^{\Gamma}_{\epsilon}$ is defined as follows:

- For every two condition scenarios $s_1, s_2 \in \Sigma_{\mathcal{CP}}$ and for every time-point $u \in \mathcal{T}$, define a hyper-edge $\alpha_\epsilon(s_1; s_2; u) = \big(t_\alpha, H_\alpha, w_\alpha, \boxdot\big)$, $\forall\, s_1, s_2 \in \Sigma_{\mathcal{CP}}$ and $u \in \mathcal{T}$, where:
  - $t_\alpha = u_{s_1}$ is the tail of the (multi-head) hyper-edge $\alpha_\epsilon(s_1; s_2; u)$;
  - $H_\alpha = \{u_{s_2}\} \cup \Delta(s_1; s_2)$ is the set of the heads of $\alpha_\epsilon(s_1; s_2; u)$;
  - $w_\alpha(u_{s_2}) = 0$, and $w_\alpha(v) = -\epsilon$ for each $v \in \Delta(s_1; s_2)$.
- Consider the expansion STND $(\mathcal{T}_\Gamma^{\text{Ex}}, \Lambda_\Gamma^{\text{Ex}}, \mathcal{DP})$ of Γ, the HyTND $\mathcal{H}_\epsilon^\Gamma$ is the tuple $\big(\mathcal{T}_\Gamma^{\text{Ex}}, \mathcal{HC}_\epsilon, \mathcal{DP}\big)$, where $\mathcal{HC}_\epsilon = \Lambda_\Gamma^{\text{Ex}} \cup \bigcup_{\substack{s_1, s_2 \in \Sigma_{\mathcal{CP}} \\ u \in \mathcal{T}}} \alpha_\epsilon(s_1; s_2; u)$.

  Notice that each $\alpha_\epsilon(s_1; s_2; u)$ has size $|\alpha_\epsilon(s_1; s_2; u)| = 1 + |\Delta(s_1; s_2)| \leq 1 + |\mathcal{CP}|$.
  Now, based on the results given in [13], the following result can be shown.

▶ **Theorem 23.** *Let $\Gamma = \langle \mathcal{T}, \mathcal{P}, \mathcal{CP}, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ be a CSTND and let $\hat{\epsilon} = \frac{1}{|\Sigma_{\mathcal{CP}}| \cdot |\mathcal{T}|}$. It holds that Γ is* CSTND-OD-DC *if and only if the HyTND $\mathcal{H}_{\hat{\epsilon}}^\Gamma$ is consistent.*

We give the proof of Theorem 23 in the appendix. Given a CSTND Γ, CSTND-OD-DC can be checked by firstly constructing the HyTND $\mathcal{H}_{\hat{\epsilon}}^\Gamma$ and then by relying on Theorem 19 for checking its consistency. Notice that, even though the size of $\mathcal{H}_{\hat{\epsilon}}^\Gamma$ is singly exponential in $|\mathcal{CP}|$ and also a possible negative generalized cycle can be of an exponential size, the corresponding clause ψ that is eventually returned by Algorithm 3 has size at most $|\mathcal{DP}|$. The obtained results of this section are summarized in the following theorem.

▶ **Theorem 24.** *Deciding* CSTND-OD-DC *on a given CSTND can be done in (pseudo) singly exponential (w.r.t. $|\mathcal{P}|$) deterministic time. When the input CSTND is* CSTND-OD-DC*, a viable and dynamic execution strategy is computable within the same time bound.*

7 Related Work

There are many proposals in the literature for ways of extending the expressiveness of the STN model. Below, we summarize the main results about CSTNs and related models.

Tsamardinos et al. [25] defined the Conditional Simple Temporal Problem (CTP) as that of determining whether a given CSTN admits a viable and dynamic execution strategy. (The CSTN acronym was introduced later.) In their work, propositional labels are associated only with time-points, not constraints. They also informally specified some reasonableness properties that any CSTN ought to satisfy. Although they showed how to solve the CTP by encoding it as a meta-level Disjunctive Temporal Problem (DTP) and feeding it to an off-the-shelf solver, that approach is not practical because the CTP-to-DTP encoding has exponential size and, on top of that, the DTP solver runs in exponential time. To our knowledge, this approach has never been implemented or empirically evaluated.

Later, Hunsberger et al. [21, 22] defined CSTNs (separate from the CTP) and formalized the well-definedness properties for CSTNs. In their work, both time-points (nodes) and constraints (edges) of a CSTN can have propositional labels that specify the scenarios in which they are applicable. (Allowing constraints to be labeled was inspired by the work of Conrad et al. [14], discussed below.) They showed that the labels must satisfy the well-definedness properties in order to guarantee the existence of a dynamic execution strategy. They also presented a sound-and-complete DC-checking algorithm for solving the CTP, and empirically demonstrated its practical performance.

Conrad et al. [14] considered a variant of CSTNs, proposing Drake, a dynamic executive for temporal plans with choice. In their work, the constraints of a temporal plan are labeled as in CSTNs, but the values of propositions (choices) are decided by the executive during run-time, not by the environment.

Cimatti et al. [6, 7] presented a different approach to solving a variety of temporal problems (CSTNs included) in which a temporal network is first translated into an equivalent Timed Game Automaton (TGA) and, then, solved by an off-the-shelf TGA solver. Although this approach is interesting because it shows the relationships between TGAs and a variety of temporal networks – including CSTNs – it has not yet been shown to be practical for solving the CTP.

Comin and Rizzi [13] solved the CTP by converting it into a Mean Payoff Game (MPG). They also introduced a variant of dynamic consistency, called ε-DC, where $\varepsilon > 0$ represents the minimum reaction time of the executive in response to observations. They presented (1) a sharp lower-bounding analysis on the critical value of the reaction time where the CSTN changes from being DC to non-DC, (2) a proof that the CTP is coNP-hard, and (3) the first singly-exponential-time algorithm for solving the CTP.

Hunsberger and Posenato [19] showed how their DC-checking algorithm from earlier work [22] can be extended to check the ε-DC property without incurring any performance degradation. They also introduced four benchmarks for testing DC-checking algorithms.

Hunsberger and Posenato [20] presented another optimization of the approach presented by Cimatti et al. in which the CTP is viewed as a two-player game. Its solution is determined by exploring an abstract game tree to find a "winning" strategy, using Monte Carlo Tree Search and Limited Discrepancy Search to guide its search. An empirical evaluation shows that the new algorithm is competitive with the propagation-based algorithm.

Cairo et al. [2] improved the analysis of the ε-DC property. They showed that if $\varepsilon = 0$ (i.e., if the system can react instantaneously), it is necessary to impose a further condition to avoid a form of instantaneous circularity.

Cui and Haslum [15] extend STNU by conditioning temporal constraints on the assignment of controllable discrete variables (decisions) that can be done at any time. In CSTNDs we connect decisions to time-points and thus provide greater expressiveness because we allow a designer to constraint when decisions have to been taken.

Zavatteri [26] defined CSTNUDs (i.e., CSTNUs [9] augmented with decision nodes), using an approach based on Timed Game Automata (TGAs) for both checking the dynamic controllability of CSTNUDs and the synthesizing memoryless execution strategies. Although CSTNUDs are more general than CSTNDs, this paper focused on analyzing the complexity of the CSTND-DC problem and presenting DC-checking algorithms for two special cases of CSTNDs.

8 Conclusions and Future Work

This paper introduced a new kind of temporal network, called a *Conditional Simple Temporal Network with Decisions*, that accommodates both *conditions* that are not under the control of the executing agent, and *decisions* that are under the agent's control. The agent aims to make decisions and schedule time-points so that all relevant constraints are satisfied no matter how the environment assigns values to the conditions in real time. After defining a notion of dynamic consistency for CSTNDs, the paper proved that the CSTND-DC problem is PSPACE-complete. Finally, it introduced some algorithms to deal with two special cases: (1) CSTNDs that contain decisions, but not conditions; and (2) CSTNDs for which all decisions are made prior to executing the network.

As for future work, among the many possible research directions, we mention here the application of our approach to the design of business process models, where not all of the decisions (represented as gateway variables in business process models) are under the control of the process engine. Another potential topic concerns the identification of other special cases of CSTNDs that might yield corresponding DC-checking algorithms.

References

1 Claudio Bettini, Xiaoyang Sean Wang, and Sushil Jajodia. Temporal reasoning in workflow systems. *Distributed and Parallel Databases*, 11(3):269–306, 2002. doi:10.1023/A:1014048800604.

2 Massimo Cairo, Carlo Comin, and Romeo Rizzi. Instantaneous reaction-time in dynamic-consistency checking of conditional simple temporal networks. In *23rd International Symposium on Temporal Representation and Reasoning (TIME 2016)*, pages 80–89, 2016. doi:10.1109/TIME.2016.16.

3 Massimo Cairo, Luke Hunsberger, Roberto Posenato, and Romeo Rizzi. A streamlined model of conditional simple temporal networks. In *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*, volume 90 of *LIPIcs*, pages 10:1–10:19, 2017. doi:10.4230/LIPIcs.TIME.2017.10.

4 Massimo Cairo and Romeo Rizzi. Dynamic controllability of conditional simple temporal networks is PSPACE-complete. In *23rd International Symposium on Temporal Representation and Reasoning (TIME 2016)*, pages 90–99, 2016. doi:10.1109/TIME.2016.17.

5 Susan J. Chinn and Gregory R. Madey. Temporal representation and reasoning for workflow in engineering design change review. *IEEE Transactions on Engineering Management*, 47(4):485–492, 2000. doi:10.1109/17.895343.

6 Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Sound and complete algorithms for checking the dynamic controllability of temporal networks with uncertainty, disjunction and observation. In *21st International Symposium on Temporal Representation and Reasoning (TIME 2014)*, pages 27–36, 2014. doi:10.1109/TIME.2014.21.

7 Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Dynamic controllability via timed game automata. *Acta Informatica*, 53(6-8):681–722, 2016. doi:10.1007/s00236-016-0257-2.

8 Carlo Combi, Mauro Gambini, Sara Migliorini, and Roberto Posenato. Representing business processes through a temporal data-centric workflow modeling language: An application to the management of clinical pathways. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(9):1182–1203, 2014. doi:10.1109/TSMC.2014.2300055.

9 Carlo Combi, Luke Hunsberger, and Roberto Posenato. An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty. In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence (ICAART 2013)*, volume 2, pages 144–156, 2013. doi:10.5220/0004256101440156.

10 Carlo Combi and Roberto Posenato. Controllability in temporal conceptual workflow schemata. In *Business Process Management (BPM 2009)*, volume 5701 of *LNCS*, pages 64–79, 2009. doi:10.1007/978-3-642-03848-8_6.

11 Carlo Combi and Roberto Posenato. Towards temporal controllabilities for workflow schemata. In *17th International Symposium on Temporal Representation and Reasoning (TIME 2010)*, pages 129–136, 2010. doi:10.1109/TIME.2010.17.

12 Carlo Comin, Roberto Posenato, and Romeo Rizzi. Hyper temporal networks – A tractable generalization of simple temporal networks and its relation to mean payoff games. *Constraints*, 22(2):152–190, 2017. doi:10.1007/s10601-016-9243-0.

13 Carlo Comin and Romeo Rizzi. Dynamic consistency of conditional simple temporal networks via mean payoff games: A singly-exponential time dc-checking. In *22nd International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 19–28, 2015. doi:10.1109/TIME.2015.18.

14 Patrick R. Conrad and Brian C. Williams. Drake: An efficient executive for temporal plans with choice. *Journal of Artificial Intelligence Research*, 42(1):607–659, 2011. `doi:10.1613/jair.3478`.

15 Jing Cui and Patrik Haslum. Dynamic controllability of controllable conditional temporal problems with uncertainty. In *27th International Conference on Automated Planning and Scheduling (ICAPS 2017)*, 2017. URL: `https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15738`.

16 Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. `doi:10.1016/0004-3702(91)90006-6`.

17 Johann Eder, Wolfgang Gruber, and Euthimios Panagos. Temporal modeling of workflows with conditional execution paths. In *Database and Expert Systems Applications (DEXA 2000)*, volume 1873 of *LNCS*, pages 243–253, 2000. `doi:10.1007/3-540-44469-6_23`.

18 Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing: 6th International Conference (SAT 2003)*, pages 502–518, 2004. `doi:10.1007/978-3-540-24605-3_37`.

19 Luke Hunsberger and Roberto Posenato. Checking the dynamic consistency of conditional simple temporal networks with bounded reaction times. In *26th International Conference on Automated Planning and Scheduling (ICAPS 2016)*, pages 175–183, 2016. URL: `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13108`.

20 Luke Hunsberger and Roberto Posenato. A new approach to checking the dynamic consistency of conditional simple temporal networks. In *Principles and Practice of Constraint Programming (CP 2016)*, volume 9892 of *LNCS*, pages 268–286, 2016. `doi:10.1007/978-3-319-44953-1_18`.

21 Luke Hunsberger, Roberto Posenato, and Carlo Combi. The dynamic controllability of conditional stns with uncertainty. In *Workshop on Planning and Plan Execution for Real-World Systems (PlanEx) at ICAPS 2012*, pages 1–8, 2012. URL: `http://arxiv.org/abs/1212.2005`.

22 Luke Hunsberger, Roberto Posenato, and Carlo Combi. A sound-and-complete propagation-based algorithm for checking the dynamic consistency of conditional simple temporal networks. In *22st International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 4–18, 2015. `doi:10.1109/TIME.2015.26`.

23 Eleanna Kafeza and Kamalakar Karlapalem. Gaining control over time in workflow management applications. In *Database and Expert Systems Applications: 11th International Conference (DEXA 2000)*, volume 1873 of *LNCS*, pages 232–241, 2000. `doi:10.1007/3-540-44469-6_22`.

24 Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 494–502, 2001.

25 Ioannis Tsamardinos, Thierry Vidal, and Martha E. Pollack. CTP: a new constraint-based formalism for conditional, temporal planning. *Constraints*, 8(4):365–388, 2003. `doi:10.1023/A:1025894003623`.

26 Matteo Zavatteri. Conditional simple temporal networks with uncertainty and decisions. In *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*, volume 90 of *LIPIcs*, pages 23:1–23:17, 2017. `doi:10.4230/LIPIcs.TIME.2017.23`.

A Appendix: Proofs of Lemma 10 and Theorem 23.

Lemma 10. Let $\sigma = (\sigma^{\mathrm{t}}, \sigma^{\mathrm{d}})$ be any dynamic and viable execution strategy. We know from [4] that the statement of the lemma is valid for CSTNs. Therefore, if we do not consider all the controllable propositional variables, and transform the corresponding observation time-points into standard time-points, we can apply the result given in [4] for transforming σ^{t} to a strategy that satisfies the statement of the lemma. In the transformation showed in the proof [4], only the numerical values determined by the strategy for time-points are modified while the relative execution order of them is preserved. Hence, the conditions for σ^{d} to be dynamic are not changed by such transformation. Therefore, the resulting strategy is indeed dynamic and viable. ◀

Theorem 23. By Definition 20, the CSTND Γ is CSTND-OD-DC if and only if there exists some $ds \in \Sigma_{\mathcal{DP}}$ such that the CSTN Γ_{ds} is dynamically consistent. By Theorem 4 and 6 in [13], for any $ds \in \Sigma_{\mathcal{DP}}$, the CSTN Γ_{ds} is dynamically consistent if and only if the HyTN $\mathcal{H}_{\hat{\epsilon}}^{\Gamma_{ds}}$ is consistent provided that $\hat{\epsilon} = \frac{1}{|\Sigma_{\mathcal{CP}}|\cdot|\mathcal{T}|}$. Then, considering the definition of (i) CSTND projection (Definition 4), (ii) HyTN projection (Definition 15), and (iii) that of $\mathcal{H}_{\epsilon}^{\Gamma}$ (Definition 22), it holds that $\mathcal{H}_{\hat{\epsilon}}^{\Gamma_{ds}}$ is HyTN $(\mathcal{H}_{\hat{\epsilon}}^{\Gamma})_{ds}$.

Finally, by Definition 15, there exists some $ds \in \Sigma_{\mathcal{DP}}$ such that the projection HyTN $(\mathcal{H}_{\hat{\epsilon}}^{\Gamma})_{ds}$ is consistent if and only if the HyTND $\mathcal{H}_{\hat{\epsilon}}^{\Gamma}$ is consistent. At this point, by composing all of these logical equivalences, the thesis follows. ◀

A Streamlined Model of Conditional Simple Temporal Networks – Semantics and Equivalence Results

Massimo Cairo[1], Luke Hunsberger[2], Roberto Posenato[3], and Romeo Rizzi[4]

1 **Department of Mathematics, University of Trento, Trento, Italy**
`massimo.cairo@unitn.it`
2 **Department of Computer Science, Vassar College, Poughkeepsie, NY, USA**
`hunsberger@vassar.edu`
3 **Department of Computer Science, University of Verona, Verona, Italy**
`roberto.posenato@univr.it`
4 **Department of Computer Science, University of Verona, Verona, Italy**
`romeo.rizzi@univr.it`

Abstract

A Conditional Simple Temporal Network (CSTN) augments a Simple Temporal Network to include a new kind of time-point, called an observation time-point. The execution of an observation time-point generates information in real time, specifically, the truth value of a propositional letter. In addition, time-points and temporal constraints may be labeled by conjunctions of (positive or negative) propositional letters. A CSTN is called dynamically consistent (DC) if there exists a dynamic strategy for executing its time-points such that no matter how the observations turn out during execution, the time-points whose labels are consistent with those observations have all been executed, and the constraints whose labels are consistent with those observations have all been satisfied. The strategy is dynamic in that its execution decisions may react to observations.

The original formulation of CSTNs included propositional labels only on time-points, but the DC-checking algorithm was impractical because it was based on a conversion of the semantic constraints into an exponentially-sized Disjunctive Temporal Network. Later work added propositional labels to temporal constraints, and yielded a sound-and-complete propagation-based DC-checking algorithm, empirically demonstrated to be practical across a variety of CSTNs.

This paper introduces a streamlined version of a CSTN in which propositional labels may appear on constraints, but not on time-points. This change simplifies the definition of the DC property, as well as the propagation rules for the DC-checking algorithm. It also simplifies the proofs of the soundness and completeness of those rules.

This paper provides two translations from traditional CSTNs to streamlined CSTNs. Each translation preserves the DC property and, for any DC network, ensures that any dynamic execution strategy for that network can be extended to a strategy for its streamlined counterpart.

Finally, this paper presents an empirical comparison of two versions of the DC-checking algorithm: the original version and a simplified version for streamlined CSTNs. The comparison is based on CSTN benchmarks from earlier work. For small-sized CSTNs, the original version shows the best performance, but the performance difference between the two versions decreases as the number of time-points in the CSTN increases. We conclude that the simplified algorithm is a practical alternative for checking the dynamic consistency of CSTNs.

1998 ACM Subject Classification G.2.2 Graph Theory, I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Conditional Simple Temporal Networks, Dynamic Consistency, Temporal Constraints

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.10

1 Introduction

Dechter et al. [9] defined a *Simple Temporal Network (STN)* as a pair $(\mathcal{T}, \mathcal{C})$, where $\mathcal{T}$ is set of real-valued variables, called time-points; and $\mathcal{C}$ is a set of binary difference constraints (a.k.a. temporal constraints) on those time-points. The *Simple Temporal Problem (STP)* is that of determining whether any given STN is *consistent* (i.e., whether there exists a complete assignment to the time-points in $\mathcal{T}$ that satisfies all of the constraints in $\mathcal{C}$). Typically, an STN includes a special time-point, Z, whose value is fixed at zero. Binary constraints involving Z correspond to unary constraints since $X \leq \delta$ is equivalent to $X - Z \leq \delta$; and $X \geq \delta$ is equivalent to $Z - X \leq -\delta$. If an STN does not have a Z time-point, one can be inserted without affecting the consistency of the network [11].

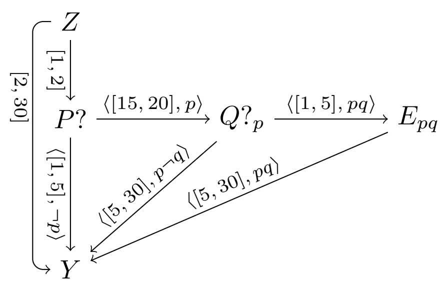
Tsamardinos et al. [18] introduced *Conditional Simple Temporal Networks (CSTNs),* augmenting STNs to include propositional letters, observation time-points, and propositional labels on time-points. Each observation time-point $P?$ has a corresponding propositional letter p, where the execution of $P?$ non-deterministically generates a truth value for p. In addition, any time-point – whether observational or not – may be labeled by a conjunction of (positive or negative) propositional literals, the idea being that only the time-points whose labels are consistent with the incrementally revealed observations need to be executed; and only the constraints among *those* time-points need to be satisfied. A CSTN is called *dynamically consistent (DC)* if there exists a dynamic *strategy* for executing its time-points such that no matter how the observations turn out during execution, the time-points whose labels are consistent with those observations have all been executed, and the constraints among those time-points have all been satisfied. The strategy is dynamic in that its execution decisions may react to observations in real time. They presented an algorithm for checking the DC property – called a DC-checking algorithm – but it was not practical due to its conversion of the semantic constraints into an exponentially-sized Disjunctive Temporal Network.

Hunsberger et al. [15] generalized CSTNs, allowing propositional labels on both time-points *and constraints.* They then introduced rules for propagating labeled constraints, which they used as the basis for a sound-and-complete DC-checking algorithm that was empirically demonstrated to be practical across a variety of CSTNs. To facilitate proving that their propagation rules were sound and complete, they also defined several properties associated with propositional labels (e.g., label *honesty* and label *coherence*); and they formalized a set of *well-definedness* properties that were implicit in the original formulation of CSTNs.

The motivation for this paper began with the observation that proving the soundness and completeness properties for the propagation-based DC-checking algorithm was unnecessarily complicated by the presence of propositional labels on time-points. As this paper shows, no loss of generality results from streamlining CSTNs by allowing propositional labels on constraints, but not on time-points. The streamlined definition of a CSTN simplifies: (1) the definition of the DC property, (2) the definition of the propagation rules, and (3) the soundness and completeness proofs for those rules. The paper proves the equivalence of the streamlined CSTN and the prior formulation. It also empirically demonstrates that the performance of the correspondingly simpler DC-checking algorithm is similar to that of the original DC-checking algorithm, and that the performance difference between the two algorithms decreases as the number of time-points increases.

2 Background

This section reviews the definitions needed for the more general version of CSTN and dynamic consistency presented by Hunsberger et al. [15].


Z
$[1, 2]$
$[2, 30]$
$P?$
$\langle[15, 20], p\rangle$
$Q?_p$
$\langle[1, 5], pq\rangle$
E_{pq}
$\langle[1, 5], \neg p\rangle$
$\langle[5, 30], p\neg q\rangle$
$\langle[5, 30], pq\rangle$
Y

Time Point	Meaning
Z	Begin
$P?$	Do Blood Test (gen. value for p)
$Q?_p$	Repeat Blood Test (only if $p = \top$; generate value for q)
E_{pq}	Do Treatment (only if $p = q = \top$)
Y	End

Figure 1 The graph of a sample CSTN, discussed in the text.

▶ **Definition 1** (Labels). Given a set P of propositional letters:
- a label is a (possibly empty) conjunction of (positive or negative) literals from P. The *empty label* is notated as $\boxdot$.
- for any label ℓ, and any $p \in P$, if $\ell \models p$ or $\ell \models \neg p$, then we say that p *appears* in ℓ.
- for any labels, ℓ_1 and ℓ_2, if $\ell_1 \models \ell_2$ (i.e., if ℓ_1 contains all of the literals in ℓ_2) then ℓ_1 is said to *entail* ℓ_2. If $\ell_1 \wedge \ell_2$ is satisfiable, then ℓ_1 and ℓ_2 are called *consistent.*
- the *label universe* of P, denoted by P^*, is the set of all consistent labels whose literals are drawn from P.

▶ **Definition 2** (CSTN). A Conditional Simple Temporal Network (CSTN) is a tuple, $\langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P\rangle$, where:
- P is a finite set of propositional letters (or propositions);
- $\mathcal{T}$ is a finite set of real-valued variables, called time-points;
- $\mathcal{C}$ is a set of labeled constraints, each having the form, $(Y - X \leq \delta, \ell)$, where $X, Y \in \mathcal{T}$, $\delta \in \mathbb{R}$, and $\ell \in P^*$;
- $L : \mathcal{T} \to P^*$ is a function assigning labels to time-points;
- $\mathcal{OT} \subseteq \mathcal{T}$ is a (finite) set of observation time-points; and
- $\mathcal{O} : P \to \mathcal{OT}$ is a bijection between propositional letters and observation time-points.

For convenience, the observation time-point associated with p may be denoted by $P?$ instead of the more cumbersome $\mathcal{O}(p)$. In a CSTN graph, the time-points serve as the nodes, and each labeled constraint, $(Y - X \leq \delta, \ell)$, is represented by an arrow from X to Y annotated by the labeled value $\langle\delta, \ell\rangle$, as follows: $X \xrightarrow{\langle\delta,\ell\rangle} Y$. (If $\ell = \boxdot$, then the label and angle brackets may be omitted, as follows: $X \xrightarrow{\delta} Y$.) For convenience, an interval constraint such as $(Y - X \in [a, b], \ell)$ may be represented by a single edge from X to Y labeled by $\langle[a, b], \ell\rangle$, although it corresponds to two constraints in the CSTN definition. Finally, since any time-points, X and Y, may participate in multiple constraints of the form, $(Y - X \leq \delta_i, \ell_i)$, each edge in the graph may have multiple labeled values of the form, $\langle\delta_i, \ell_i\rangle$.

Fig. 1 shows the graph of a CSTN for a simple health-care example, originally presented by Hunsberger et al. [15]. It will be used as a running example. The nodes, Z and Y, represent starting and ending time-points, respectively. $P?$ represents the time at which a particular blood test is performed. If this test generates a positive result, represented by $p = \top$, then the test must be repeated at time-point $Q?$, which generates a truth value for q. Since $Q?$ applies only in scenarios where $p = \top$, it is labeled by p. If both tests generate positive results, then an emergency treatment is applied at time-point E, whose label is pq.

The edges in this graph use the compact interval notation. For example, the edge from $P?$ to $Q?$ labeled by $\langle[15, 20], p\rangle$ represents that the difference, $Q? - P?$, must lie within $[15, 20]$

in scenarios where p is true (i.e., the repeated test must be performed between 15 and 20 minutes after the first test). Similarly, the edges from $Q?$ to E, and from E to Y are labeled by pq, indicating that those constraints apply only in scenarios where both p and q are $\top$.

2.1 Well-definedness properties for CSTNs

The following definitions specify properties that any *well defined* CSTN must hold. For example, without the *label coherence* property (Defn. 4), it might happen X is labeled by p, Y is labeled by q, and $\mathcal{C}$ contains the constraint, $(Y - X \leq -2, p)$: $X_p \xrightarrow{\langle -2, p\rangle} Y_q$. Then, in the scenario $p\neg q$, X must be executed and Y must not be executed, but the constraint $(Y - X \leq -2, p)$ must hold, which is absurd. Similar examples can be generated for the other well-definedness properties. In short, CSTNs that are not well defined are of no use.

▶ **Definition 3** (Honest Label)**.** A label ℓ in a CSTN, whether on a time-point or constraint, is called *honest* if for each q that appears in ℓ, $\ell \models L(Q?)$ (i.e., ℓ contains all of the literals from the label of the observation time-point for q).

▶ **Definition 4** (WD1: Label coherence)**.** A CSTN holds property WD1 (i.e., has *coherent labels)* if for each labeled constraint, $(Y - X \leq \delta, \ell)$, the label ℓ is satisfiable and entails $L(X) \wedge L(Y)$ (i.e., contains all of the literals from $L(X)$ and $L(Y)$).

▶ **Definition 5** (WD2)**.** A CSTN holds property WD2 if:
(a) for each time-point $T \in \mathcal{T}$, its label $L(T)$ is honest, and
(b) for each $p \in P$ that appears in $L(T)$, $\mathcal{C}$ contains a constraint, $(P? - T \leq -\epsilon, L(T))$, for some $\epsilon > 0$ (i.e., T is constrained to occur after $P?$).

▶ **Definition 6** (WD3: Constraint Label Honesty)**.** A CSTN holds property WD3 if the label on each of its constraints is honest.

▶ **Definition 7** (Well defined CSTN)**.** A CSTN is called *well defined* if it holds properties WD1, WD2 and WD3.

2.2 Dynamic Consistency for CSTNs

▶ **Definition 8** (Scenario)**.** A *scenario* over a set P of propositional letters is a function, $s : P \to \{\top, \bot\}$, that assigns a truth value to each letter in P. Any such function also provides the truth value for any label $\ell \in P^*$, which is denoted by $s(\ell)$. The set of all scenarios over P is denoted by $\mathcal{I}$.

▶ **Definition 9** (Schedule)**.** A *schedule* for a set of time-points $\mathcal{T}$ is a mapping, $\psi : \mathcal{T} \to \mathbb{R}$, that assigns a real number to each time-point in $\mathcal{T}$. The set of all schedules for any subset of $\mathcal{T}$ is denoted by Ψ.

▶ **Definition 10** (Projection)**.** Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P\rangle$ be any CSTN, and s any scenario over P. The *projection* of S onto s – notated $Prj(\mathcal{S}, s)$ – is the STN, $(\mathcal{T}_s^+, C_s^+)$, where:
- $\mathcal{T}_s^+ = \{T \in \mathcal{T} \mid s \models L(T)\}$; and
- $C_s^+ = \{(Y - X \leq \delta) \mid \text{for some } \ell,\ (Y - X \leq \delta, \ell) \in C \text{ and } s \models \ell\}$.

For convenience, we also define $\mathcal{OT}_s^+ = \mathcal{OT} \cap \mathcal{T}_s^+$ (i.e., the set of observation time-points whose labels are entailed by s).

▶ **Definition 11** (Execution Strategy)**.** Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P\rangle$ be any CSTN. An *execution strategy* for $\mathcal{S}$ is a mapping, $\sigma : \mathcal{I} \to \Psi$, such that for each scenario $s \in \mathcal{I}$, the domain of the schedule $\sigma(s)$ is $\mathcal{T}_s^+$. If, in addition, for each scenario s, the schedule $\sigma(s)$ is a solution to the projection $\mathit{Prj}(\mathcal{S}, s)$, then σ is called *viable.* In any case, the execution time for the time-point X in the schedule $\sigma(s)$ is denoted by $[\sigma(s)]_X$. In addition, $|\sigma|$ denotes the maximum value assigned by σ to any time-point in $\mathcal{T}$ in any scenario $s \in \mathcal{I}$.

▶ **Definition 12** (History). Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P\rangle$ be any CSTN, s any scenario, σ any execution strategy for $\mathcal{S}$, and t any real number. The *history* of t in the scenario s, for the strategy σ – notated $\mathit{Hist}(t, s, \sigma)$ – is the set of observations made before time t according to the schedule $\sigma(s)$: $\mathit{Hist}(t, s, \sigma) = \{(p, s(p)) \mid P? \in \mathcal{OT}_s^+ \text{ and } [\sigma(s)]_{P?} < t\}$.

▶ **Definition 13** (Dynamic Execution Strategy). An execution strategy σ for a CSTN is called *dynamic* if for any scenarios s_1 and s_2, and any time-point $X \in \mathcal{T}_{s_1}^+$:

- If $\mathit{Hist}(t, s_1, \sigma) = \mathit{Hist}(t, s_2, \sigma)$, where $t = [\sigma(s_1)]_X$, then $X \in \mathcal{T}_{s_2}^+$ and $[\sigma(s_2)]_X = t$.

▶ **Definition 14** (Dynamic Consistency). A CSTN is *dynamically consistent* (DC) if there exists an execution strategy for it that is both dynamic and viable.

2.3 Motivations for a new definition

The prior definitions of CSTN are convenient for the designer working in some domain. The designer typically knows the scenarios in which each time-point must be executed and can directly represent that information in the time-point labels. Furthermore, if the designer constructs a CSTN that is not well defined, it can be easily remedied by augmenting the labels on time-points and constraints to make them honest and coherent, and by inserting any missing constraints needed for property WD2.b. However, the presence of labels on time-points needlessly complicates the constraint-propagation rules needed for practical DC checking. For example, Hunsberger et al. [15] introduced a *child-of* relation among propositional letters that derives from cases where observation time-points have non-empty labels. The applicability conditions of their propagation rules are littered with special cases to handle the children of propositional letters. As a consequence, proving that the propagation rules are sound requires dealing with these special cases. However, if a CSTN has no labels on its time-points, then these complexities disappear. Indeed, it is trivial to check that all of the well-definedness properties become vacuous if there are no labels on any time-points.

These considerations motivated the search for an equivalent formulation of CSTNs that does not include labels on time-points. This paper presents such a formulation, called *streamlined CSTN,* and proves that it is equivalent to the ordinary CSTN presented above. The paper presents two alternative translations from ordinary to streamlined CSTNs, each of which preserves the most important properties of a CSTN, including dynamic consistency.

A designer working in some domain may continue to reap the benefits of working with CSTNs having labels on time-points, leaving it to the DC-checking algorithm to convert the CSTN into a streamlined version before carrying out any constraint propagation. Thus, the streamlined CSTN simplifies the theoretical foundations of CSTNs while still allowing users to work with the earlier version should they find it useful to do so.

3 Streamlined Model of CSTNs

This section presents the definition for a streamlined CSTN, which simply removes the assignment of labels to time-points. It also specifies the slight modifications to the sequence of definitions needed for defining the dynamic consistency of CSTNs – namely, that for any scenario s, $\mathcal{T}_s^+ = \mathcal{T}$ and $\mathcal{OT}_s^+ = \mathcal{OT}$, since there are no labels on any time-points.

▶ **Definition 15** (CSTN$_\boxdot$)**.** A *Streamlined Conditional Simple Temporal Network* (CSTN$_\boxdot$) is a tuple, $\langle \mathcal{T}, \mathcal{C}, \mathcal{OT}, \mathcal{O}, P\rangle$, where:

- P is a finite set of propositional letters (or propositions);
- $\mathcal{T}$ is a finite set of real-valued variables, called time-points;
- $\mathcal{C}$ is a set of labeled constraints of the form, $(Y - X \leq \delta, \ell)$, where $X, Y \in \mathcal{T}$, $\delta \in \mathbb{R}, \ell \in P^*$;
- $\mathcal{OT} \subseteq \mathcal{T}$ is a (finite) set of observation time-points; and
- $\mathcal{O} : P \to \mathcal{OT}$ is a bijection between propositional letters and observation time-points.

As previously noted, there is no need for any of the well-definedness properties (Defns. 3-6) for streamlined CSTNs since they become vacuous if there are no labels on time-points. The existing definitions of *scenarios* and *schedules* (Defns. 8 and 9) apply to CSTN$_\boxdot$ without any changes, but Defns. 10-13 must be slightly modified for CSTN$_\boxdot$, as follows.

- **Projection (Defn. 10).** The same, except that for each scenario s, $\mathcal{T}_s^+ = \mathcal{T}$, since there are no labels on any time-points (equivalently, since $s \models \boxdot$). Similarly, $\mathcal{OT}^+ = \mathcal{OT}$.
- **Execution Strategy (Defn. 11).** The same, except that for each scenario s, the domain of $\sigma(s)$ is $\mathcal{T}_s^+ = \mathcal{T}$.
- **History (Defn. 12).** The same, but replace $P? \in \mathcal{OT}_s^+$ by $P? \in \mathcal{OT}$, since $\mathcal{OT}_s^+ = \mathcal{OT}$.
- **Dynamic Execution Strategy (Defn. 13).** The same, but replace $X \in \mathcal{T}_{s_1}^+$ by $X \in \mathcal{T}$, since $\mathcal{T}_{s_1}^+ = \mathcal{T}$, and delete the (now redundant) requirement that $X \in \mathcal{T}_{s_2}^+$.

4 Equivalence of the CSTN and CSTN$_\boxdot$ Models

This section presents two translations from CSTNs to streamlined CSTNs and proves that each translation preserves the property of dynamic consistency. Furthermore, using either translation, any dynamic execution strategy for a DC CSTN can be extended to a dynamic execution strategy for its streamlined counterpart such that for any scenario, the time-points executed by the strategy for the CSTN are executed at the same times by the corresponding strategy for the streamlined CSTN. The first translation inserts constraints that, for each scenario s, require all time-points whose labels are entailed by s to be executed *at or before* a fixed horizon h, while all time-points whose labels are inconsistent with s are constrained to occur *after* h. The second translation does not add any such constraints and, thus, is much simpler; however, it is less explicit about the time-points that are executed in the original CSTN in any given scenario. Both translations are presented here since they illuminate different aspects of the relationships between the CSTN and CSTN$_\boxdot$ models.

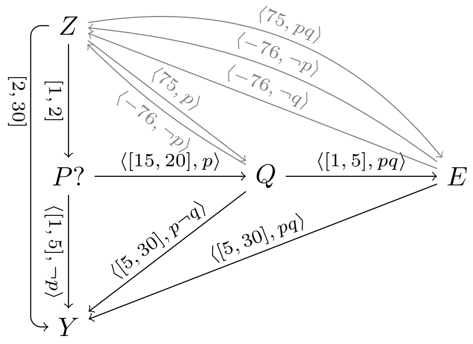
4.1 The First Translation from CSTN to CSTN$_\boxdot$

We begin with some preliminary results.

▶ **Lemma 16** (Upper bound for DC CSTNs)**.** *Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P\rangle$ be any CSTN.*

- *Let $k = |\mathcal{OT}|$ be the number of observation time-points in $\mathcal{S}$;*
- *let $n = |\mathcal{T}|$ the number of time-points;*
- *let $M = \max\{|\delta|$ such that some $(Y - X \leq \delta, \ell) \in \mathcal{C}\}$ be the maximum absolute value of any bound on any constraint in $\mathcal{C}$; and*
- *let $h = Mn(2^k)$ be the* horizon.

If $\mathcal{S}$ is DC, then there exists a viable and dynamic execution strategy σ for $\mathcal{S}$ such that for every scenario s, and every time-point X, $[\sigma(s)]_X \leq h$ (i.e., $|\sigma| \leq h$).


Z
$P?$
Q
E
Y
$[2, 30]$
$[1, 2]$
$\langle [15, 20], p\rangle$
$\langle [1, 5], pq\rangle$
$\langle [1, 5], \neg p\rangle$
$\langle [5, 30], p\neg q\rangle$
$\langle [5, 30], pq\rangle$
$\langle 75, pq\rangle$
$\langle -76, \neg p\rangle$
$\langle -76, \neg q\rangle$
$\langle 75, p\rangle$
$\langle -76, \neg p\rangle$

The horizon value (cf. Lemma 17) is:
$h = Mn = 15 \cdot 5 = 75$.
$\mathcal{C}'_1 = \{(Q? \leq 75, p), (E \leq 75, pq),$
$(P? \leq 75, \boxdot), (Y \leq 75, \boxdot)\}$.
$\mathcal{C}'_2 = \{(Q? \geq 76, \neg p), (E \geq 76, \neg p),$
$(E \geq 76, \neg q)\}$.
The third and fourth constraints in $\mathcal{C}'_1$ are not shown in the graph since they would be redundant.

Figure 2 The CSTN$_\boxdot$ derived from the sample CSTN using the first method of translation.

Proof. Comin and Rizzi [7] (their Theorem 6) proved a correspondence between CSTNs and Hyper Temporal Networks (HyTNs) such that: (1) the corresponding HyTN has at most $(2^k)n$ time-points; and (2) the CSTN is DC if and only if the corresponding HyTN is consistent. Furthermore, Comin [6] (his Theorem 2.3) showed that an HyTN is consistent if and only if it has no negative cycles (his Defn. 2.5). Now, a negative cycle in an HyTN consists of hyperarcs. A finite cyclic path is obtained from a negative cycle by selecting at most one ordinary arc from each hyperarc such that the selected arcs form a cycle in which each node appears at most once. A negative cycle in an HyTN is characterized by each finite cyclic path having negative length. Therefore, if $\mathcal{S}$ is a DC CSTN, there can be no negative cycle in its corresponding HyTN. Furthermore, inserting constraints of the form $X \leq h$ into the CSTN for each X could not introduce a negative cycle into the HyTN because: (i) all such edges emanate from Z, thus only one can appear in any finite cyclic path; (ii) there can be no more than $(2^k)n$ pre-existing edges in each finite cyclic path; and (iii) the absolute value of each weight on the pre-existing edges can be no more than M. As a result, each finite cyclic path that includes an edge of length $h = Mn(2^k)$ cannot have negative length. ◀

▶ **Lemma 17** (Tighter upper bound; rational weights). *Let $\mathcal{S}$ be a DC CSTN with n time-points and* rational *weights. If M is the maximum absolute value of any* negative *edge in $\mathcal{S}$, then the network obtained by constraining every time-point in $\mathcal{S}$ to occur before time Mn is DC.*

The proof of Lemma 17 is in the Appendix.

▶ **Definition 18** (Reducing a CSTN to a CSTN$_\boxdot$). Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P\rangle$ be a well defined CSTN. The reduction of $\mathcal{S}$ is the CSTN$_\boxdot$, $\mathcal{S}_\boxdot = \langle \mathcal{T}, \mathcal{C} \cup \mathcal{C}'_1 \cup \mathcal{C}'_2, \mathcal{OT}, \mathcal{O}, P\rangle$, where:

- some $h \geq Mn(2^k)$ serves as the *horizon* (cf. Lemma 16);*
- $\mathcal{C}'_1 = \bigcup_{X \in \mathcal{T}} \{(X - Z \leq h, L(X))\} = \bigcup_{X \in \mathcal{T}} \{(X \leq h, L(X))\}$;† and
- $\mathcal{C}'_2 = \bigcup_{X \in \mathcal{T}, \rho \in L(X)} \{(Z - X \leq -h - 1, \neg\rho)\} = \bigcup_{X \in \mathcal{T}, \rho \in L(X)} \{(X \geq h + 1 > h, \neg\rho)\}$.

The constraints in $\mathcal{C}'_1$ force each X to be executed at or before h in scenarios that entail X's label. And the constraints in $\mathcal{C}'_2$ force each X to be executed after h in scenarios that do not entail X's label.

Fig. 2 shows the streamlined version of the running example from Fig. 1.

The first method of translation generates an equivalent streamlined CSTN, as follows.

* If the weights in $\mathcal{S}$ are rational, then any $h \geq Mn$ can serve as the horizon.
† In $\mathcal{C}'_2$, ρ represents a positive or negative literal, and $\neg\rho$ the literal with the opposite polarity of ρ.

▶ **Theorem 19** (Equivalence of CSTN and CSTN$_\boxdot$ Models)**.** *Let $\mathcal{S}$ be any well defined CSTN. Then $\mathcal{S}_\boxdot$ (i.e., the reduction of $\mathcal{S}$ to a CSTN$_\boxdot$) is equivalent to $\mathcal{S}$ in the sense that $\mathcal{S}$ is DC if and only if $\mathcal{S}_\boxdot$ is DC. Furthermore, in the case where $\mathcal{S}$ and $\mathcal{S}_\boxdot$ are both DC, if σ is any viable and dynamic (V & D) strategy for $\mathcal{S}$ for which $|\sigma| \leq h$, then there is an equivalent V & D strategy $\sigma_\boxdot$ for $\mathcal{S}_\boxdot$ in the sense that for each scenario s and any $X \in \mathcal{T}$:*

- *If $X \in \mathcal{T}_s^+$, then $[\sigma_\boxdot(s)]_X = [\sigma(s)]_X \leq h$; otherwise, $[\sigma_\boxdot(s)]_X \geq h+1 > h$.*

Proof. Let $\mathcal{S}$ be any well defined CSTN; and let $\mathcal{S}_\boxdot$ be the corresponding CSTN$_\boxdot$.

Part 1: $\mathcal{S}$ is DC $\Rightarrow$ $\mathcal{S}_\boxdot$ is DC

By Lemma 16 there exists a V & D execution strategy σ for $\mathcal{S}$ such that $|\sigma| \leq h$. For any such σ, define $\sigma_\boxdot : \mathcal{I} \to \Psi$ as follows. For any scenario $s \in \mathcal{I}$, let $\sigma(s) : \mathcal{T} \to \mathbb{R}$ be the schedule that is the same as $\sigma(s)$ on time-points in $\mathcal{T}_s^+$, but that maps time-points not in $\mathcal{T}_s^+$ to $h+1$ (i.e., just over the horizon). In other words:

- For each $X \in \mathcal{T}_s^+$, let $[\sigma_\boxdot(s)]_X = [\sigma(s)]_X \leq h$. (1)
- For each $X \in \mathcal{T}\backslash\mathcal{T}_s^+$, let $[\sigma_\boxdot(s)]_X = h+1$. (2)

Note that $|\sigma_\boxdot| \leq h+1$. Next, we show that:

- for any $t \leq h+1$, and any scenario s, $\mathit{Hist}(t,s,\sigma) = \mathit{Hist}_\boxdot(t,s,\sigma_\boxdot)$. ($\star$)

To see this, let p be any propositional letter. If $(p, s(p))$ is in $\mathit{Hist}_\boxdot(t,s,\sigma_\boxdot)$, it follows that $[\sigma_\boxdot(s)]_{P?} < t \leq h+1$ and, hence, by (1) and (2), that $[\sigma_\boxdot(s)]_{P?} \leq h$, in which case, $P? \in \mathcal{T}_s^+$ and $[\sigma(s)]_{P?} = [\sigma_\boxdot(s)]_{P?} < t$. Therefore, $(p, s(p))$ is also in $\mathit{Hist}(t,s,\sigma)$. On the other hand, if $(p, s(p))$ is in $\mathit{Hist}(t,s,\sigma)$, it follows that $P? \in \mathcal{T}_s^+$ and thus $[\sigma_\boxdot(s)]_{P?} = [\sigma(s)]_{P?} < t$, in which case $(p, s(p))$ also appears in $\mathit{Hist}_\boxdot(t,s,\sigma_\boxdot)$.

Next we aim to show that $\sigma_\boxdot$ is a *dynamic* strategy. Toward that end, let s_1 and s_2 be any scenarios, let X be any time-point in $\mathcal{T}$, let $t = [\sigma_\boxdot(s_1)]_X$, and suppose that $\mathit{Hist}_\boxdot(t,s_1,\sigma_\boxdot) = \mathit{Hist}_\boxdot(t,s_2,\sigma_\boxdot)$. We must show that $[\sigma_\boxdot(s_2)]_X = t$. First, note that since $t \leq h+1$ and $\mathit{Hist}_\boxdot(t,s_1,\sigma_\boxdot) = \mathit{Hist}_\boxdot(t,s_2,\sigma_\boxdot)$, it follows from ($\star$) that:

- $\mathit{Hist}(t,s_1,\sigma) = \mathit{Hist}_\boxdot(t,s_1,\sigma_\boxdot) = \mathit{Hist}_\boxdot(t,s_2,\sigma_\boxdot) = \mathit{Hist}(t,s_2,\sigma)$. (†)

There are two cases to consider.

1. **$X \in \mathcal{T}_{s_1}^+$.**
   By (1), $t = [\sigma_\boxdot(s_1)]_X = [\sigma(s_1)]_X \leq h$. Therefore, (†) together with the dynamicity of σ implies that $X \in \mathcal{T}_{s_2}^+$ and $[\sigma(s_2)]_X = t \leq h$, whence $[\sigma_\boxdot(s_2)]_X = [\sigma(s_2)]_X = t$.
2. **$X \notin \mathcal{T}_{s_1}^+$.**
   In this case, it follows from (2) that $t = [\sigma_\boxdot(s_1)]_X = h+1$. Next, let p be any propositional letter for which $(p, s(p))$ is in the histories, $\mathit{Hist}_\boxdot(t,s_1,\sigma_\boxdot) = \mathit{Hist}_\boxdot(t,s_2,\sigma_\boxdot)$. Then $[\sigma_\boxdot(s_1)]_{P?} < t = h+1$ which, by (1) and (2) above, implies that $[\sigma_\boxdot(s_1)]_{P?} \leq h$ and $P? \in \mathcal{T}_{s_1}^+$. Similarly, $[\sigma_\boxdot(s_2)]_{P?} \leq h$ and $P? \in \mathcal{T}_{s_2}$. Furthermore, by (2), any observation time-point $Q?$ that does not appear in those histories must be executed at time $h+1$, which implies that $Q? \notin \mathcal{T}_{s_1}^+$ and $Q? \notin \mathcal{T}_{s_2}^+$. Thus, those histories contain *exactly* the observation time-points in $\mathcal{T}_{s_1}^+ \cap \mathcal{T}_{s_2}^+$. Since X is not in $\mathcal{T}_{s_1}^+$, it follows that $s_1 \not\models L(X)$. Now, if $X \in \mathcal{T}_{s_2}^+$ (i.e., $s_2 \models L(X)$), then (without loss of generality) there must be some $p \in L(X)$ such that $s_1 \not\models p$, but $s_2 \models p$. Since $L(X)$ is honest (by WD2), $L(X) \models L(P?)$. Therefore, $s_2 \models L(X) \models L(P?)$ and, hence, $P? \in \mathcal{T}_{s_2}^+$, which implies that $[\sigma_\boxdot(s_2)]_{P?} = [\sigma(s_2)]_{P?} \leq h$. But then p must appear in $\mathit{Hist}_\boxdot(h+1, s_2, \sigma_\boxdot)$. However, since p yields different outcomes in s_1 and s_2, that contradicts that the histories for s_1 and s_2 are the same. Therefore, it must be that $X \notin \mathcal{T}_{s_2}^+$, in which case, $[\sigma_\boxdot(s_2)] = h+1 = t$.

By cases 1 and 2, it follows that $\sigma_\boxdot$ is dynamic. It remains to show that $\sigma_\boxdot$ is viable. Toward that end, let s be any scenario, and let $(Y - X \leq \delta, \ell)$ be any constraint in $\mathcal{S}_\boxdot$ whose label ℓ is entailed by s. Now, if this constraint is in $\mathcal{S}$, then by WD1, $\ell \models L(X) \wedge L(Y)$. Hence, $s \models \ell \models L(X) \wedge L(Y)$, which implies that $X, Y \in \mathcal{T}_s^+$. Thus, σ and $\sigma_\boxdot$ execute X and Y at the same times in s. And, since $s \models \ell$, this constraint is in $\mathcal{C}_s^+$; thus, the viability of σ ensures that it is satisfied by $\sigma_\boxdot(s)$. On the other hand, if this constraint is not in $\mathcal{S}$, then it must be one of the constraints, $(X - Z \leq h, L(X))$ from $\mathcal{C}_1'$, or $(Z - X \leq -h-1, \neg\rho)$ from $\mathcal{C}_2'$, for some $\rho \in L(X)$. Now if $s \models L(X)$, then (1) requires that $[\sigma_\boxdot(s)]_X \leq h$, which implies that the first constraint is satisfied, and $s \models L(X) \models \rho$ implies that the second constraint is trivially satisfied. On the other hand, if $s \not\models L(X)$, then the first constraint is trivially satisfied and, by (2), $[\sigma_\boxdot(s)]_X = h+1$, which implies that the second constraint is satisfied. Therefore, since the choice of constraint was arbitrary, it follows that $\sigma_\boxdot(s)$ must be a solution to the projection $Prj_\boxdot(\mathcal{S}_\boxdot, s)$. And since s was arbitrary, $\sigma_\boxdot$ must be viable.

Part 2: $\mathcal{S}_\boxdot$ is DC $\Rightarrow$ $\mathcal{S}$ is DC

Let $\sigma_\boxdot$ be any V & D strategy for $\mathcal{S}_\boxdot$. We will construct a similar V & D strategy σ for $\mathcal{S}$.

First, consider any scenario s, and any time-point X. The viability of $\sigma_\boxdot$ ensures that it satisfies the constraints in $\mathcal{C}_1'$ and $\mathcal{C}_2'$. Therefore, if $X \in \mathcal{T}_s^+$ (i.e., if $s \models L(X)$), then $[\sigma_\boxdot(s)]_X \leq h$; otherwise $[\sigma_\boxdot(s)]_X \geq h+1 > h$. In short, $[\sigma_\boxdot(s)]_X \leq h \Leftrightarrow s \models L(X)$.

Next, define the strategy σ for $\mathcal{S}$, as follows. For each scenario s, and each $X \in \mathcal{T}_s^+$, let $[\sigma(s)]_X = [\sigma_\boxdot(s)]_X$ (i.e., $\sigma(s) = \sigma_\boxdot(s)|_{\mathcal{T}_s^+}$). Note that, by the preceding remarks, $|\sigma| \leq h$.

Viability of σ. Let s be any scenario, and $Prj(\mathcal{S}, s) = (\mathcal{T}_s^+, \mathcal{C}_s^+)$ the corresponding projection. By WD1, the constraints in $\mathcal{C}_s^+$ only involve time-points in $\mathcal{T}_s^+$. Thus, the endpoints of each constraint in $\mathcal{C}_s^+$ are executed at the same times by σ and $\sigma_\boxdot$ in s. Since $\sigma_\boxdot$ is viable, it satisfies each constraint in that projection; hence, so does σ. Thus, σ is viable.

Dynamicity of σ. Let s_1 and s_2 be any scenarios, let X be any time-point in $\mathcal{T}_{s_1}^+$, let $t = [\sigma(s_1)]_X \leq h$, and suppose that $Hist(t, s_1, \sigma) = Hist(t, s_2, \sigma)$. Since $X \in \mathcal{T}_{s_1}^+$, $[\sigma(s_1)]_X = [\sigma_\boxdot(s_1)]_X$. Next, let p be any letter appearing in $L(X)$; and let $P?$ be the corresponding observation time-point. By WD1, $L(X)$ is honest; hence, $L(X) \models L(P?)$; whence, $s \models L(X) \models L(P?)$. Therefore, $[\sigma(s_1)]_{P?} = [\sigma_\boxdot(s_1)]_{P?}$. Next, by WD2.b, $\mathcal{C}$ includes a constraint of the form $(P? - X \leq -\epsilon, L(X))$ and, since $\sigma_\boxdot$ is viable, it follows that $[\sigma_\boxdot(s_1)]_{P?} < [\sigma_\boxdot(s_1)]_{P?} + \epsilon \leq [\sigma_\boxdot(s_1)]_X = t$. Then, since $[\sigma(s_1)]_{P?} = [\sigma_\boxdot(s_1)]_{P?} < t$, it follows that $(p, s_1(p))$ appears in $Hist(t, s_1, \sigma)$. Since this holds for each p in $L(X)$, it follows that $Hist(t, s_1, \sigma) \models L(X)$. Since $Hist(t, s_2, \sigma) = Hist(t, s_1, \sigma)$, it follows that $Hist(t, s_2, \sigma) \models L(X)$ and, hence, that $s_2 \models L(X)$ (i.e., $X \in \mathcal{T}_{s_2}^+$). Therefore, $[\sigma(s_2)]_X = [\sigma_\boxdot(s_2)]_X$. Now, if $Hist_\boxdot(t, s_1, \sigma_\boxdot) \neq Hist(t, s_1, \sigma)$, there must be some observation time-point $P? \notin \mathcal{T}_{s_1}^+$ that $\sigma_\boxdot$ executes in scenario s_1 at some time *before* $t \leq h$. But the lower-bound constraints in $\mathcal{C}_2'$ ensure that this can only happen if $s_1 \models L(P?)$, which contradicts that $P? \notin \mathcal{T}_{s_1}^+$. Thus, $Hist_\boxdot(t, s_1, \sigma_\boxdot) = Hist(t, s_1, \sigma)$. Similarly, $Hist_\boxdot(t, s_2, \sigma_\boxdot) = Hist(t, s_2, \sigma)$. But then the dynamicity of $\sigma_\boxdot$ ensures that $[\sigma_\boxdot(s_2)]_X = t$ and, hence, that $[\sigma(s_2)]_X = t$. ◀

4.2 The Second Translation from CSTN to CSTN$_\boxdot$

Unlike the first translation from CSTN to CSTN$_\boxdot$, the second translation, defined below, does not insert any extra edges. For convenience, we use the same notation as in the preceding section (i.e., in this section, CSTN$_\boxdot$ refers to the following definition).

▶ **Definition 20** (Reducing a CSTN to a CSTN$_\boxdot$)**.** Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P\rangle$ be a well defined CSTN. The reduction of $\mathcal{S}$ is the CSTN$_\boxdot$, $\mathcal{S}_\boxdot = \langle \mathcal{T}, \mathcal{C}, \mathcal{OT}, \mathcal{O}, P\rangle$.

For a well defined CSTN, we can define a partial order among propositional letters, as follows. For any $p, q \in P$, we write $p \prec_L q$ if p (or $\neg p$) appears in $L(Q?)$, where L is the function that assigns labels to time-points. In addition, if, for each observation time-point $P?$, the label $L(P?)$ is honest, then we say that L is honest. Using this notation, it follows that if L is honest and $p \prec_L q$, then $L(Q?) \models L(P?)$. In what follows, it is necessary to show that the $\prec_L$ relation is acyclic for well defined CSTNs for which $\mathcal{S}_\boxdot$ is DC.

▶ **Lemma 21.** *If $\mathcal{S}$ is well defined and $\mathcal{S}_\boxdot$ is DC, then $\prec_L$ is acyclic.*

Proof. Suppose that $p_1 \prec_L \cdots \prec_L p_k = p_1$ is a cycle in $\prec_L$. By WD2.a, L is honest and, therefore, $L(P_k?) \models L(P_{k-1}?) \models \ldots \models L(P_2?) \models L(P_1?)$. Since $p_k = p_1$, it follows that $L(P_1?) = \ldots = L(P_k?)$. For convenience, let $\ell = L(P_1?) = \ldots = L(P_k?)$. By WD2.b, $\mathcal{S}$ (and hence $\mathcal{S}_\boxdot$) contains constraints of the form, $(P_{i+1}? - P_i? \leq -\epsilon_i, \ell)$, for every $i = 1, \ldots, k-1$, forming a negative cycle with the consistent label ℓ. Since such a cycle cannot be satisfied in any scenario s for which $s \models \ell$, $\mathcal{S}_\boxdot$ must not be DC, which is a contradiction. ◀

For a CSTN $\mathcal{S}$, it may be that in some scenarios some propositional variables are not observed because their corresponding observation time-points are not executed. For example, in the CSTN from Fig. 1, $Q?$ is not executed in either of the scenarios $\neg pq$ or $\neg p\neg q$ because $L(Q?) = p$. In general, in such cases, there may be a family of scenarios that are equivalent in that they differ only in the values they assign to propositional letters that are not observed. Below, we define a *canonical scenario* to be a unique representative for such a family of scenarios. The canonical scenario is the unique scenario from the family that assigns a value of $\perp$ (i.e., false) to each propositional letter that is not observed when executing $\mathcal{S}$ in scenarios from that family. For example, the canonical scenario for $\{\neg pq, \neg p\neg q\}$ is $\neg p\neg q$.

▶ **Definition 22** (Canonical Scenario)**.** For any scenario s, define the *canonical* scenario $\hat{s}$ as follows. For any $p \in P$, if $s \models L(P?)$, let $\hat{s}(p) = s(p)$; otherwise, let $\hat{s}(p) = \perp$.

Note that if s and $\hat{s}$ disagree on some p, then $s \not\models L(P?)$ (i.e., s and $\hat{s}$ differ only on variables that cannot be observed in the scenario s). However, the converse need not hold (i.e., it may happen that $s \not\models L(P?)$, yet s and $\hat{s}$ happen to agree on p).

▶ **Lemma 23.** *If $\prec_L$ is acyclic and L is honest, then $s \models L(P?)$ if and only if $\hat{s} \models L(P?)$.*

Proof. There are two cases to consider.

1. **$\hat{s} \not\models L(P?) \Rightarrow s \not\models L(P?)$.**
   Suppose that $\hat{s} \not\models L(P?)$, but $s \models L(P?)$. Then s and $\hat{s}$ disagree on some $q \in L(P?)$. By the honesty of $L(P?)$, it follows that $L(P?) \models L(Q?)$ and, therefore, that $s \models L(P?) \models L(Q?)$. However, since s and $\hat{s}$ disagree on q, the definition of $\hat{s}$ implies that $s \not\models L(Q?)$, which is a contradiction.
2. **$\hat{s} \models L(P?) \Rightarrow s \models L(P?)$.**
   Suppose that $\hat{s} \models L(P?)$, but $s \not\models L(P?)$, where $P?$ (and hence p) is chosen minimally with this property with respect to the ordering $\prec_L$. Let $q \in L(P?)$ be arbitrary. Thus, $q \prec_L p$. By the honesty of $L(P?)$, it follows that $L(P?) \models L(Q?)$. Therefore, $\hat{s} \models L(P?) \models L(Q?)$ and, hence, $\hat{s} \models L(Q?)$. But, then, by the minimality of p, it follows that $s \models L(Q?)$. Since $q \in L(P?)$ was chosen arbitrarily, we have that $s \models L(P?)$, which is a contradiction.

◀

The following lemma states that, if a CSTN $\mathcal{S}$ is well defined, then any scenario s and its corresponding canonical scenario $\hat{s}$ determine the same projection.

▶ **Lemma 24.** *If $\mathcal{S}$ is a CSTN for which $\prec_L$ is acyclic and all labels on time-points and constraints are honest, then for any scenario s and any label ℓ, s and $\hat{s}$ must assign the same truth value to ℓ. As a result, $Prj(\mathcal{S}, s) = Prj(\mathcal{S}, \hat{s})$ (i.e., $\mathcal{T}_s^+ = \mathcal{T}_{\hat{s}}^+$ and $C_s^+ = C_{\hat{s}}^+$).*

Proof. Let ℓ be any label on a time-point or constraint such that $s(\ell) \neq \hat{s}(\ell)$. Then there exists some p that appears in ℓ such that $s(p) \neq \hat{s}(p)$. Then, by the definition of $\hat{s}$, $s \not\models L(P?)$. Therefore, by Lemma 23, $\hat{s} \not\models L(P?)$. But the honesty of ℓ implies that $\ell \models L(P?)$. Therefore, it follows that $s \not\models \ell$ and $\hat{s} \not\models \ell$. But then $s(\ell) = \bot = \hat{s}(\ell)$, which is a contradiction. ◀

It is now possible to apply the obtained results to show the relationship between the dynamic consistency of a CSTN and that of its corresponding streamlined CSTN$_{\boxdot}$.

▶ **Lemma 25.** *If $\mathcal{S}$ is a well defined CSTN, and $\mathcal{S}_{\boxdot}$ is DC, then $\mathcal{S}$ is DC.*

Proof. Let $\sigma_{\boxdot}$ be any viable and dynamic strategy for $\mathcal{S}_{\boxdot}$. Let the strategy σ for $\mathcal{S}$ be defined as follows. For any scenario $s \in \mathcal{I}$ and any time-point $X \in \mathcal{T}_s^+$, let $[\sigma(s)]_X = [\sigma_{\boxdot}(\hat{s})]_X$. We need only show that σ is viable and dynamic for $\mathcal{S}$.

Viability. First, note that the conditions of Lemma 21 hold; therefore, $\prec_L$ must be acyclic. Next, since $\mathcal{S}$ is well defined, the labels on all time-points and constraints in $\mathcal{S}$ must be honest; hence, the conditions of Lemma 24 are satisfied. Thus, s and $\hat{s}$ must agree on the truth value of each label on any time-point or constraint in $\mathcal{S}$ (and hence in $\mathcal{S}_{\boxdot}$).

Suppose that σ violates some constraint, $(Y - X \leq \delta, \ell)$. Then there is some scenario s such that $[\sigma(s)]_Y - [\sigma(s)]_X > \delta$ and $s \models \ell$. But then the definition of σ implies that $[\sigma_{\boxdot}(\hat{s})]_Y - [\sigma_{\boxdot}(\hat{s})]_X > \delta$; and Lemma 24 gives that $\hat{s} \models \ell$. Together, these contradict that $\sigma_{\boxdot}$ is viable.

Dynamicity. Suppose that $X \in \mathcal{T}_{s_1}^+$ (i.e., $s_1 \models L(X)$), $t = [\sigma(s_1)]_X$, and $Hist(s_1, t, \sigma) = Hist(s_2, t, \sigma)$. We must show that $X \in \mathcal{T}_{s_2}^+$ (i.e., $s_2 \models L(X)$) and $[\sigma(s_2)]_X = t$.

Toward that end, let p be any letter that appears in $L(X)$. Since L is honest, it follows that $L(X) \models L(P?)$. Thus, $s_1 \models L(P?)$ (i.e., $P? \in \mathcal{T}_{s_1}^+$); hence, by Lemma 24, $\hat{s}_1 \models L(P?)$. Next, by WD2.b, $\mathcal{S}$ (and hence $\mathcal{S}_{\boxdot}$) must contain a constraint of the form, $(P? - X \leq -\epsilon, L(X))$. And, since $\sigma_{\boxdot}$ is viable, it follows that $[\sigma(s_1)]_{P?} = [\sigma_{\boxdot}(\hat{s}_1)]_{P?} < [\sigma_{\boxdot}(\hat{s}_1)]_X = [\sigma(s_1)]_X = t$. Since $s_1 \models L(P?)$, it must be that p appears in $Hist(s_1, t, \sigma) = Hist(s_2, t, \sigma)$. And, since p was chosen arbitrarily in $L(X)$, it follows that $Hist(s_1, t, \sigma) = Hist(s_2, t, \sigma) \models L(X)$. Therefore, $s_2 \models L(X)$ (i.e., $X \in \mathcal{T}_{s_2}^+$).

Finally, suppose that $Hist(\hat{s}_1, t, \sigma_{\boxdot}) \neq Hist(\hat{s}_2, t, \sigma_{\boxdot})$. But then there must be some time $t' < t$ at which one of the following holds: (1) one of the schedules, $\sigma_{\boxdot}(\hat{s}_1)$ or $\sigma_{\boxdot}(\hat{s}_2)$, executes some observation time-point $Q?$ at t', while the other does not; or (2) both schedules execute some observation time-point $Q?$ at t', but yield different values for q. Without loss of generality, choose t' to be the earliest time at which one of the above conditions hold. Then (1) is impossible, because $Hist(\hat{s}_1, t', \sigma_{\boxdot}) = Hist(\hat{s}_2, t', \sigma_{\boxdot})$ and $\sigma_{\boxdot}$ is dynamic. To show that property (2) is impossible, first consider the case where q appears in $L(X)$. Then $s_1(q) = s_2(q)$, which implies that $\hat{s}_1(q) = \hat{s}_2(q)$, contradicting the choice of $Q?$. But if q does not appear in $L(X)$, then $\hat{s}_1(q) = \bot = \hat{s}_2(q)$ by definition of $\hat{s}_1$ and $\hat{s}_2$, another contradiction. Therefore, $Hist(\hat{s}_1, t, \sigma_{\boxdot}) = Hist(\hat{s}_2, t, \sigma_{\boxdot})$ which, by the dynamicity of $\sigma_{\boxdot}$ implies that $[\sigma_{\boxdot}(\hat{s}_2)]_X = t$, which in turn implies that $[\sigma(s_2)]_X = t$. ◀

Now, it is possible to show the main result of the section.

▶ **Theorem 26.** *Let $\mathcal{S}$ be any well defined CSTN. Then $\mathcal{S}_{\boxdot}$ (i.e., the reduction of $\mathcal{S}$ to a CSTN$_{\boxdot}$) is equivalent to $\mathcal{S}$, in the sense that $\mathcal{S}$ is DC if and only if $\mathcal{S}_{\boxdot}$ is DC.*

Proof. The $\Longrightarrow$ direction is already proven in Theorem 19. Indeed, using the first translation requires *more* constraints to be satisfied in $\mathcal{S}_{\boxdot}$. The $\Longleftarrow$ direction is given by Lemma 25. ◀

5 Empirical Evaluation

When applied to Streamlined Conditional Simple Temporal Networks, the constraint-propagation rules used by the DC-checking algorithm of Hunsberger et al. [15, 12] become simpler. That algorithm checks whether an input network is DC by exhaustively propagating labeled constraints and then verifying that the propagated network does not contain a negative cycle with a consistent label. However, the applicability conditions for the constraint-propagation rules, and the labels generated by those rules, depend on time-point labels and the $\prec_L$ relation. Therefore, if the input CSTN has no labels on its time-points (i.e., if it is a CSTN$_{\boxdot}$), then the algorithm can avoid dealing with such complications.

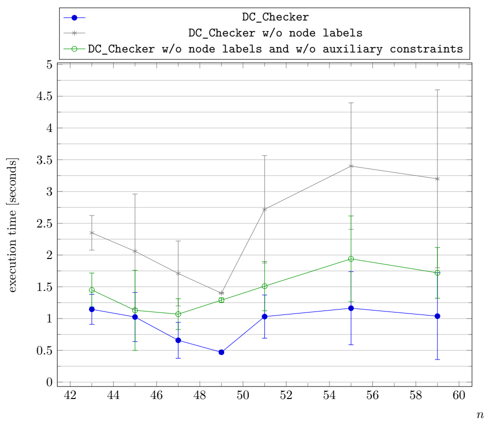
- In this section, the original DC-checking algorithm that applies to any CSTN $\mathcal{S}$ shall be called `DC_Checker`, and the "simplified" algorithm that applies only to a CSTN$_{\boxdot}$ shall be called `DC_CheckerWONodeLabels`.

This section presents an empirical comparison of the performance of `DC_Checker` and `DC_CheckerWONodeLabels` on instances of the benchmarks proposed in prior work [12]. For each CSTN $\mathcal{S}$, `DC_Checker` is run on $\mathcal{S}$, while `DC_CheckerWONodeLabels` is run on the streamlined CSTN $\mathcal{S}_{\boxdot}$, using the simplified propagation rules. Recall that two translations from $\mathcal{S}$ to $\mathcal{S}_{\boxdot}$ were introduced in Section 4, one of which involves the auxiliary constraints in the sets, $\mathcal{C}'_1$ and $\mathcal{C}'_2$. This section reports results from running `DC_CheckerWONodeLabels` on both versions of $\mathcal{S}_{\boxdot}$.

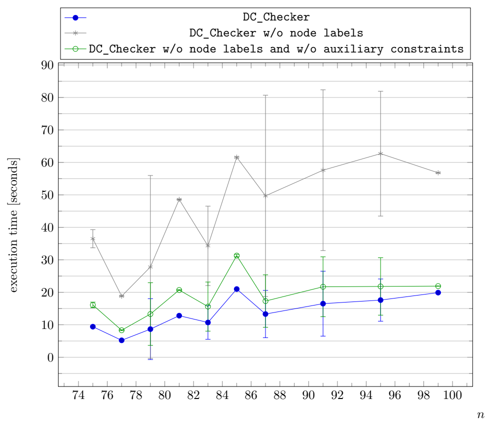
First, we briefly recall that the benchmarks contain CSTN instances obtained from random workflow schemata generated by the ATAPIS toolset [16]. For each $N \in \{10, 20, 30, 40\}$, a class of at least 120 workflow graphs were randomly generated by setting the number of activities to N, the probability for parallel branches to 0.2, the probability for conditional branches to 0.2, and the maximum duration of activities or delays between activities to 50. As a result, all edge-weights were at most 10^4. Then, each workflow graph was translated into an equivalent CSTN as proposed by Combi et al. [5]. It is worth noting that different workflow graphs with the same number of activities may translate into CSTNs of different sizes due to different numbers of connector nodes in the workflows. However, it is not hard to verify that a workflow with N activities translates into a CSTN having n nodes, where $(2N + 2) \leq n \leq (5N + 2)$. There are 4 benchmarks, each containing at least 60 dynamically consistent CSTNs and 60 non-dynamically consistent CSTNs, relating to workflow graphs of the same class. In order to simplify the comparison, for each benchmark, the number of observation time-points in the network, $|\mathcal{P}|$, has been fixed with respect to the class of workflows, as follows.

N:	10	20	30	40
$\lvert\mathcal{P}\rvert$:	3	5	7	9

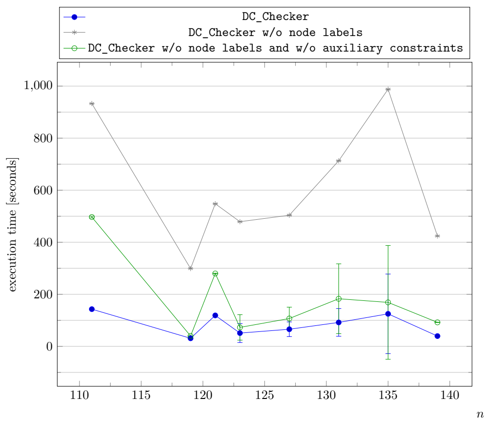
Since non-DC networks were regularly solved one to two orders of magnitude faster than similarly sized DC networks, the rest of this section focuses on the results for the DC networks.



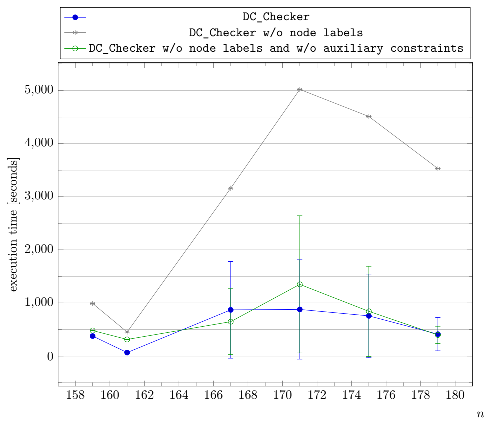
(a) Benchmark $N = 10, |\mathcal{P}| = 3$.



(b) Benchmark $N = 20, |\mathcal{P}| = 5$.



(c) Benchmark $N = 30, |\mathcal{P}| = 7$.
For `DC_Checker w/o node labels` values, the standard deviation has been omitted to have a better scale of the diagram.



(d) Benchmark $N = 40, |\mathcal{P}| = 9$.
For `DC_Checker w/o node labels` values, the standard deviation has been omitted to have a better scale of the diagram.

Figure 3 Execution time vs. number of time-points n.

Algorithms and procedures necessary for this evaluation were implemented in Java and executed on a JVM 8 in a Linux machine with two AMD Opteron 4334 CPUs and 64GB of RAM. The code is freely available [17].

The results shown in Figure 3 demonstrate that, in general, the original CSTN DC-checking algorithm `DC_Checker` has the best performance in almost all instances. Indeed, taking node labels into account allows the algorithm to avoid the propagation of some auxiliary values (in the case of streamlined CSTNs with auxiliary constraints from the sets $\mathcal{C}'_1$ and $\mathcal{C}'_2$) or non-coherent or non-honest ones (in the case of streamlined CSTNs without auxiliary constraints).

We have verified that these kinds of values can be quite numerous and that, for some instances, when they contain the auxiliary constraints from $\mathcal{C}'_1$ and $\mathcal{C}'_2$, the execution time of `DC_CheckerWONodeLabels` can be two or three orders of magnitude greater than the execution time of `DC_Checker` on the corresponding CSTNs.

On the other hand, the performance difference between `DC_CheckerWONodeLabels` and `DC_Checker` decreases as the number of nodes increases. We verified that the original algorithm continues to propagate fewer labeled values than the streamlined version, but such differences become smaller. Therefore, the time required by the original algorithm to stop non-coherent or non-honest labeled values must become more or less equal to the time required to propagate them as done by the simpler algorithm.

We have also evaluated a different implementation of the streamlined version for checking if it was possible to avoid the propagation of useless labeled values. In this new implementation, part ot the information given by node labels is rebuilt dynamically and exploited to “clean” some labels. We verified that while the number of useless labeled values can be reduced, the computation time still remains the same due to the extra time required by the added code. In other words, the overall performance of that implementation is no better than that of `DC_CheckerWONodeLabels`.

6 Related Work

There are many proposals in the literature for ways of extending the expressiveness of the STN model. Below, we summarize the main results about CSTNs and related models.

Tsamardinos et al. [18] defined the Conditional Simple Temporal Problem (CTP) as that of determining whether a given CSTN admits a viable and dynamic execution strategy. (The CSTN acronym was introduced later.) In their work, propositional labels are associated only with time-points, not constraints. They also informally specified some reasonableness properties that any CSTN ought to satisfy. Although they showed how to solve the CTP by encoding it as a meta-level Disjunctive Temporal Problem (DTP) and feeding it to an off-the-shelf solver, that approach is not practical because the CTP-to-DTP encoding has exponential size and, on top of that, the DTP solver runs in exponential time. To our knowledge, this approach has never been implemented or empirically evaluated.

Later, Hunsberger et al. [14, 15] defined CSTNs (separate from the CTP) and formalized the well-definedness properties for CSTNs. In their work, both time-points (nodes) and constraints (edges) of a CSTN can have propositional labels that specify the scenarios in which they are applicable. (Allowing constraints to be labeled was inspired by the work of Conrad et al. [8], discussed below.) They showed that the labels must satisfy the well-definedness properties in order to guarantee the existence of a dynamic execution strategy. They also presented a sound-and-complete DC-checking algorithm for solving the CTP, and empirically demonstrated its practical performance.

Conrad et al. [8] considered a variant of CSTNs, proposing Drake, a dynamic executive for temporal plans with choice. In their work, the constraints of a temporal plan are labeled as in CSTNs, but the values of propositions (choices) are decided by the executive during run-time, not by the environment.

Cimatti et al. [3, 4] presented a different approach to solving a variety of temporal problems (CSTNs included) in which a temporal network is first translated into an equivalent Timed Game Automaton (TGA) and, then, solved by an off-the-shelf TGA solver. Although this approach is interesting because it shows the relationships between TGAs and a variety of temporal networks – including CSTNs – it has not yet been shown to be practical for solving the CTP.

Comin and Rizzi [7] solved the CTP by converting it into a Mean Payoff Game (MPG). They also introduced a variant of dynamic consistency, called ε-DC, where $\varepsilon > 0$ represents the minimum reaction time of the executive in response to observations. They presented (1) a sharp lower-bounding analysis on the critical value of the reaction time where the CSTN changes from being DC to non-DC, (2) a proof that the CTP is coNP-hard, and (3) the first singly-exponential-time algorithm for solving the CTP.

Hunsberger and Posenato [12] showed how their DC-checking algorithm from earlier work [15] can be extended to check the ε-DC property without incurring any performance degradation. They also introduced four benchmarks for testing DC-checking algorithms.

Hunsberger and Posenato [13] presented another optimization of the approach presented by Cimatti et al. in which the CTP is viewed as a two-player game. Its solution is determined by exploring an abstract game tree to find a "winning" strategy, using Monte Carlo Tree Search and Limited Discrepancy Search to guide its search. An empirical evaluation shows that the new algorithm is competitive with the propagation-based algorithm.

Cairo et al. [1] improved the analysis of the ε-DC property. They showed that if $\varepsilon = 0$ (i.e., if the system can react instantaneously), it is necessary to impose a further condition to avoid a form of instantaneous circularity. In particular, they (1) proposed a new extension of dynamic consistency, called π-DC, suitable for systems that can react instantaneously, (2) showed by a counter-example that π-DC is not equivalent to 0-DC, and (3) proposed a sound-and-complete algorithm for checking the π-DC property having a (pseudo) singly-exponential time complexity in the number of propositional letters.

Cario and Rizzi [2] showed that the CTP is PSPACE-complete.

7 Conclusions and Future Work

This paper presented a new version of CSTNs, named *streamlined Conditional Simple Temporal Networks*, in which propositional labels may appear on constraints, but not on time-points. This change simplifies the definition of the DC property and the specification of propagation rules for the DC-checking algorithm. It also makes proving the soundness and completeness of those rules simpler.

The paper proves that traditional CSTNs can be translated into streamlined CSTNs while preserving the dynamic consistency property. Two translations from CSTNs to streamlined CSTNs were presented. The first generates an equivalent streamlined CSTN in which the information contained in time-point labels is preserved in the form of auxiliary constraints that force time-points in certain scenarios to be executed either before or after a fixed horizon, depending on whether they would be executed or not in the original CSTN. The second translation does not preserve the information in the time-point labels, but provides a simpler, equivalent streamlined CSTN. The drawback of the second translation is that some time-points can be executed in the streamlined CSTN even if they would not be in the original CSTN.

Finally, the paper provided an experimental comparison of two versions of the DC-checking algorithm due to Hunsberger et al. [15]: the original version and a simplified version for streamlined CSTNs. For small CSTNs, the original algorithm shows the best performance; however, the difference in performance between the two versions decreases as the number of time-points increases. During the tests, we verified that the static information given by the time-point labels can limit the propagation of non-coherent/non-honest labels in a significant way making the the DC checking faster. However, that advantage decreases as the number of nodes increases.

It appears that simple heuristics such as one that tries to rebuild dynamically the information given by time-point labels would not be successful for improving the performance of the simplified version of the DC-checking algorithm. Our future work will investigate other methods for improving the performance of the algorithm for streamlined CSTNs in order to make it competitive with the original algorithm on small CSTNs, too.

References

1 Massimo Cairo, Carlo Comin, and Romeo Rizzi. Instantaneous reaction-time in dynamic-consistency checking of conditional simple temporal networks. In *23rd International Symposium on Temporal Representation and Reasoning, TIME 2016*, pages 80–89, 2016. doi:10.1109/TIME.2016.16.

2 Massimo Cairo and Romeo Rizzi. Dynamic controllability of conditional simple temporal networks is PSPACE-complete. In *23rd International Symposium on Temporal Representation and Reasoning, TIME 2016*, pages 90–99, 2016. doi:10.1109/TIME.2016.17.

3 Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Sound and complete algorithms for checking the dynamic controllability of temporal networks with uncertainty, disjunction and observation. In *21st International Symposium on Temporal Representation and Reasoning, TIME 2014*, pages 27–36, 2014. doi:10.1109/TIME.2014.21.

4 Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Dynamic controllability via timed game automata. *Acta Informatica*, 53(6-8):681–722, 2016. doi:10.1007/s00236-016-0257-2.

5 Carlo Combi, Mauro Gambini, Sara Migliorini, and Roberto Posenato. Representing business processes through a temporal data-centric workflow modeling language: An application to the management of clinical pathways. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(9):1182–1203, 2014. doi:10.1109/TSMC.2014.2300055.

6 Carlo Comin. *Complexity in Infinite Games on Graphs and Temporal Constraint Networks*. PhD thesis, University of Trento and Universite Paris-Est, 2017.

7 Carlo Comin and Romeo Rizzi. Dynamic consistency of conditional simple temporal networks via mean payoff games: A singly-exponential time dc-checking. In *22nd International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 19–28, 2015. doi:10.1109/TIME.2015.18.

8 Patrick R. Conrad and Brian C. Williams. Drake: An efficient executive for temporal plans with choice. *Journal of Artificial Intelligence Research*, 42(1):607–659, 2011. doi:10.1613/jair.3478.

9 Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. doi:10.1016/0004-3702(91)90006-6.

10 Alfonso Gerevini, Anna Perini, and Francesco Ricci. Incremental algorithms for managing temporal constraints. Technical Report IRST-9605-07, IRST, 1996.

11 Luke Hunsberger. Efficient execution of dynamically controllable Simple Temporal Networks with Uncertainty. *Acta Informatica*, 53(2):89–147, 2015. doi:10.1007/s00236-015-0227-0.

12 Luke Hunsberger and Roberto Posenato. Checking the dynamic consistency of conditional simple temporal networks with bounded reaction times. In *26th International Conference on Automated Planning and Scheduling (ICAPS 2016)*, pages 175–183, 2016. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13108.

13 Luke Hunsberger and Roberto Posenato. A new approach to checking the dynamic consistency of conditional simple temporal networks. In *Principles and Practice of Constraint Programming (CP 2016)*, volume 9892 of *LNCS*, pages 268–286, 2016. doi:10.1007/978-3-319-44953-1_18.

14 Luke Hunsberger, Roberto Posenato, and Carlo Combi. The dynamic controllability of conditional stns with uncertainty. In *Workshop on Planning and Plan Execution for Real-World Systems (PlanEx) at ICAPS 2012*, pages 1–8, June 2012. URL: http://arxiv.org/abs/1212.2005.

15 Luke Hunsberger, Roberto Posenato, and Carlo Combi. A sound-and-complete propagation-based algorithm for checking the dynamic consistency of conditional simple temporal

networks. In *22nd International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 4–18, 2015. doi:10.1109/TIME.2015.26.

16 Andreas Lanz and Manfred Reichert. Enabling time-aware process support with the ATAPIS toolset. In *Proceedings of the BPM Demo Sessions 2014*, volume 1295 of *CEUR Workshop Proceedings*, pages 41–45, 2014.

17 Roberto Posenato. A CSTN(U) consistency check algorithm implementation in Java, June 2017. URL: http://profs.scienze.univr.it/~posenato/software/cstnu.

18 Ioannis Tsamardinos, Thierry Vidal, and Martha E. Pollack. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints*, 8(4):365–388, 2003. doi:10.1023/A:1025894003623.

A Appendix: Proof of Lemma 17

▶ **Lemma 17** (Tighter upper bound; rational weights). *Let $\mathcal{S}$ be a DC CSTN with n time-points and* rational *weights. If M is the maximum absolute value of any* negative *edge in $\mathcal{S}$, then the network obtained by constraining every time-point in $\mathcal{S}$ to occur before time Mn is DC.*

Proof. First, since there are at most $(n^2)(2^k)$ edge weights, each edge weight can be expressed as a fraction involving the least common denominator among the edge weights. As a result, without loss of generality, we may henceforth assume that all edge weights are integers.

Let $\mathcal{C}^*$ be the set of labeled edges obtained by exhaustively applying the sound-and-complete constraint-propagation rules presented by Hunsberger et al. [15]. Given the assumptions that (1) all edge weights are integers, and (2) the network is DC, the constraint propagation must terminate. Hence, $\mathcal{C}^*$ is well defined and contains only finitely many edges. Furthermore, each edge in $\mathcal{C}^*$ can be derived by a finite number of applications of the constraint-propagation rules. For convenience, we shall refer to $\mathcal{C}$ as the set of *original* edges, and $\mathcal{C}^*$ as the set of *derived* edges.

Fix $Z = 0$ and let $\mathcal{U} = \mathcal{T}\backslash\{Z\}$ be the set of as-yet-unexecuted time-points. Prior to executing the time-points in $\mathcal{U}$, there have been no observations and, thus, the *initial partial scenario* is the empty scenario, represented by $\boxdot$. For each $Y \in \mathcal{U}$, its *effective lower bound* (ELB) with respect to the empty scenario, defined by Hunsberger et al., is given by: $ELB(Y, \boxdot) = \max\{\delta \mid (Y \geq \delta, \ell) \in \mathcal{C}^*\}$. Let $\lambda = \min\{ELB(Y, \boxdot) \mid Y \in \mathcal{U}\}$ be the minimum ELB of any as-yet-unexecuted time-point. Let $X \in \mathcal{U}$ be any time-point such that $ELB(X, \boxdot) = \lambda$. (It does not matter if there happen to be multiple such time-points.) We aim to show that $\lambda \leq M$. Therefore, we assume that $\lambda > M$ and seek a contradiction.

By construction, $\lambda \leq ELB(Y, \boxdot)$ for each $Y \in \mathcal{U}$. In addition, the *Spreading Lemma* (from Hunsberger et al.) ensures that for each $Y \in \mathcal{U}$, there is an edge from Y to Z labeled by $\langle -\delta_Y, \boxdot\rangle$, for some $\delta_Y \geq \lambda$.

Given the definition of M, the value, $ELB(X, \boxdot) = \lambda > M$, cannot be due to an *original* edge from X to Z of length $-\lambda < -M$. Instead, it must be due to an edge that has been *derived* by one or more applications of the various constraint-propagation rules. Among all of the derivations used to generate the edges in $\mathcal{C}^*$, generated in some arbitrary order, let D be the *first* derivation that results in an edge from X to Z whose weight equals $-\lambda$.

The following argument focuses on the rule applications in the derivation D that involve the zero time-point Z. (There may be rule applications in D that do not involve Z, but they will not be relevant to the argument that follows.) In this narrow setting, the six constraint-propagation rules presented by Hunsberger et al. (LP, R_0, R_3^*, qLP, qR_0 and qR_3^*) can be represented by the three rules shown in Table 1.[‡]

[‡] The labels shown in Table 1 do not play a big role in the proof. However, for completeness, they are

Table 1 Constraint-propgation rules used in the proof of Lemma 17.

(LP/qLP)	$\begin{tikzcd} A \arrow[r, "\langle u,\alpha\rangle"] \arrow[rr, dashed, bend right, "\langle u+v,\gamma\rangle"'] & B \arrow[r, "\langle v,\beta\rangle"] & Z \end{tikzcd}$	$\alpha\beta$ consistent or $(u < 0$ and $v < 0)$; $\gamma = (\alpha \star \beta_p)'$.
(R_0/qR_0)	$\begin{tikzcd} P? \arrow[r, "\langle w,\alpha\rangle"] \arrow[r, dashed, bend right, "\langle w,(\alpha_p)'\rangle"'] & Z \end{tikzcd}$	$w < 0$.
(R_3^*/qR_3^*)	$\begin{tikzcd} P? \arrow[r, "\langle w,\alpha\rangle"] & Z & C \arrow[l, "\langle v,\beta\rangle"'] \arrow[l, dashed, bend left, "\langle m,\gamma\rangle"] \end{tikzcd}$	$w < 0$, $m = \max\{w, v\}$, and $\gamma = (\alpha \star \beta_p)'$.

Claim: No finite sequence of rule applications involving any of the six constraint-propagation rules from Hunsberger et al. can generate a *shortest* edge from any time-point $Y \in \mathcal{U}$ to Z whose weight is less than or equal to $-\lambda$.

Proof of Claim. For the base case, we note that the definition of M ensures that no *original* edge in $\mathcal{C}$ can have weight less than or equal to $-\lambda < -M$. For the inductive case, consider an arbitrary edge from Y to Z whose label is $\langle u, \alpha\rangle$. Suppose that this edge is a shortest edge among all edges from Y to Z whose label is α (or more general than α). Finally, suppose that all prior edges encountered during the derivation of this edge satisfy the claim. Note that the final rule application that generates the edge from Y to Z must be one of the three rules shown in Table 1. We address each in turn.

- (LP/qLP). In this case, A in the top row of Table 1 plays the role of Y, and $y = u + v \leq -\lambda < -M$ is the weight of a shortest edge from A to Z among those edges labeled by γ (or some more general label). First consider the possibility that $u \geq 0$. In that case, the weight v of the edge from B to Z satisfies: $v \leq u + v = -\lambda < -M$. By the inductive hypothesis, this cannot be a shortest edge from B to Z labeled by β (or some more general label). But then the same rule application, using a shorter edge from B to Z, would generate a shorter edge from Y to Z whose label is α (or more general than α), contradicting that the first edge from Y to Z was shortest. On the other hand, if $u < 0$, then $-\lambda = u + v < v$. In that case, the Spreading Lemma ensures that there is an edge from B to Z labeled by some $\langle -\delta, \boxdot\rangle$, where $-\delta \leq -\lambda$. But then the same rule application, using this stronger edge from B to Z would generate an edge from Y to Z whose weight is $-\delta + u \leq -\lambda + u < -\lambda$, another contradiction.
- (R_0/qR_0). In this case, $P?$ in the middle row of Table 1 plays the role of Y and $y = w \leq -\lambda < -M$ is the weight of a shortest edge from $P?$ to Z among those labeled by $(\alpha_p)'$ (or some more general label). By the inductive hypothesis, the edge from $P?$ to Z labeled by $\langle w, \alpha\rangle$, which also has the weight $y = w \leq -\lambda < -M$, cannot be a shortest edge from $P?$ to Z labeled by α (or some more general label). But then replacing this

shown in full detail. The notation in the table is a slight simplification of that used by Hunsberger et al. First, for any label ℓ, the label ℓ' is that obtained by removing from ℓ any *children* of any q-literals that appear in ℓ. Second, for any propositional letter p, and any label ℓ, the label ℓ_p is that obtained by removing any occurrence of p or any of its children from ℓ. Finally, the $\star$ operator is a *commutative* operator that extends the conjunction of literals as follows. For any propositional letter p, $p\star\neg p = p?\star p = p?\star\neg p = p?\star p? = p?$. The $\star$ operator is then extended to labels by applying it in pairwise fashion to like literals from each operand. For example, $(abcd) \star (a(\neg b)(c?)(\neg e)) = a(b?)(c?)d(\neg e)$.

edge with a shorter one would generate a shorter edge from $P?$ to Z labeled by $(\alpha_p)'$, a contradiction.

- (R_3^*/qR_3^*). In this case, C in the bottom row of Table 1 plays the role of Y and $y = m = \max\{w, v\} \leq -\lambda < -M$ is the weight of a shortest edge from C to Z among those labeled by γ or some more general label. There are three cases to consider:
  1. $w < v$. Here, $m = \max\{w, v\} = v \leq -\lambda < -M$. By the inductive hypothesis, the edge from C to Z labeled by $\langle v, \beta\rangle$ cannot be a shortest such edge. But then replacing it with a shorter edge, say one labeled by $\langle v - \epsilon, \beta\rangle$, where $\epsilon > 0$, would cause the corresponding application of R_3^*/qR_3^* to generate a shorter edge from C to Z labeled by γ (or some more general label), a contradiction. (The weight of the resulting edge would be $v - \epsilon'$, where $\epsilon' = \min\{\epsilon, v - w\} > 0$.)
  2. $v < w$. Here, $m = \max\{w, v\} = w \leq -\lambda < -M$. By the inductive hypothesis, the edge from $P?$ to Z labeled by $\langle w, \alpha\rangle$ cannot be a shortest such edge. But then replacing it with a shorter edge, say one labeled by $\langle w - \epsilon, \alpha\rangle$, where $\epsilon > 0$, would cause the corresponding application of R_3^*/qR_3^* to generate a shorter edge from C to Z labeled by γ (or some more general label), a contradiction. (The weight of the resulting edge would be $w - \epsilon'$, where $\epsilon' = \min\{\epsilon, w - v\} > 0$.)
  3. $w = v$. Here, $m = \max\{w, v\} = w = v \leq -\lambda < -M$. By the inductive hypothesis, neither the edge from $P?$ to Z labeled by $\langle w, \alpha\rangle$, nor the edge from C to Z labeled by $\langle w, \beta\rangle$, can be shortest such edges. Thus, each can be replaced by a corresponding edge with a shorter weight, say, by using the labeled values $\langle w - \epsilon_1, \alpha\rangle$ and $\langle v - \epsilon_2, \beta\rangle$, respectively. But then the corresponding application of R_3^*/qR_3^* would generate an edge from C to Z whose weight was $v - \epsilon < v$, where $\epsilon = \min\{\epsilon_1, \epsilon_2\}$. That is a contradiction.

Thus, the claim is proven. ◀

From the claim, it follows that no finite sequence of rule applications can generate an edge from X to Z of length $-\lambda < -M$, which contradicts the choice of X. Therefore, it must be that $\lambda \leq M$. Thus, the earliest-first strategy executes X at time λ. Equivalently, we may introduce the constraints, $X - Z \leq \lambda$ and $Z - X \leq -\lambda$ (i.e., $X = \lambda$).

At this point, the time-points X and Z form a *rigid component* [10]. As a result, by re-orienting all edges involving Z to instead involve X (adjusting the weights accordingly), Z can be effectively removed from the network. Afterward, rename X as Z and observe that the resulting network is DC with the same (or smaller) value of M, and one fewer time-point. By induction we get a network where each successive time-point is executed no more than M after the previous one, which gives the desired result. ◀

Note: We conjecture that Lemma 17 also holds for real-valued weights.

Evaluation of Temporal Datasets via Interval Temporal Logic Model Checking*

Dario Della Monica[1], David de Frutos-Escrig[2], Angelo Montanari[3], Aniello Murano[4], and Guido Sciavicco[5]

1 **Dept. Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Madrid, Spain; and**
Dept. of Electrical Engineering and Information Technology, University "Federico II"of Naples, Italy
ddellamo@ucm.es, dario.dellamonica@unina.it

2 **Dept. Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Madrid, Spain**
defrutos@sip.ucm.es

3 **Dept. of Mathematics, Computer Science, and Physics, University of Udine, Udine, Italy**
angelo.montanari@uniud.it

4 **Dept. of Electrical Engineering and Information Technology, University "Federico II" of Naples, Naples, Italy**
murano@na.infn.it

5 **Dept. of Mathematics and Computer Science, University of Ferrara, Ferrara, Italy**
guido.sciavicco@unife.it

Abstract

The problem of *temporal dataset evaluation* consists in establishing to what extent a set of temporal data (histories) complies with a given temporal condition. It presents a strong resemblance with the problem of model checking enhanced with the ability of *rating* the compliance degree of a model against a formula. In this paper, we solve the temporal dataset evaluation problem by suitably combining the outcomes of model checking an interval temporal logic formula against sets of histories (finite interval models), possibly taking into account domain-dependent measures/criteria, like, for instance, sensitivity, specificity, and accuracy. From a technical point of view, the main contribution of the paper is a (deterministic) polynomial time algorithm for interval temporal logic model checking over finite interval models. To the best of our knowledge, this is the first application of a (truly) interval temporal logic model checking in the area of temporal databases and data mining rather than in the formal verification setting.

1998 ACM Subject Classification F.4.1 Mathematical Logic, I.2.6 Learning, H.2.1 Logical Design

Keywords and phrases Dataset Evaluation; Temporal Databases; Model Checking; Interval Temporal Logics

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.11

* This work was partially supported by the Italian INdAM-GNCS project *Logics and Automata for Interval Model Checking.* In addition, Dario Della Monica acknowledges the financial support from a Marie Curie INdAM-COFUND-2012 Outgoing Fellowship.

1 Introduction

Temporal databases keep track of the temporal evolution of information by associating one or more temporal dimensions with stored data [12, 21, 40, 41]. One of the fundamental temporal dimensions is *valid* time that associates with each stored fact the time interval at which it is true in the modeled reality. In this paper, we focus on temporal databases featuring such a dimension (valid-time temporal databases). A valid-time temporal database consists of a set of temporal instances, or *histories*. Borrowing the terminology from the field of machine learning, a set of histories is often referred to as a *temporal dataset* [44].

The *temporal dataset evaluation problem* can be (roughly) defined as the problem of establishing how many histories comply with a given temporal formula or constraint. The evaluation of temporal datasets plays a key role in at least three different application domains: (i) *temporal query processing* [40], where the set of histories that satisfy a given condition must be returned, (ii) *temporal constraint checking* [41], where the set of histories that violate a given constraint must be identified, and (iii) *rule evaluation* [44], where one must determine how many histories, and to which extent, comply with a certain temporal rule. In this paper, we formulate (and solve) the temporal dataset evaluation problem as a model checking problem.

Interval temporal logic model checking. In its standard formulation, *model checking* is the problem of verifying whether or not a given formula of some logical language is satisfied by a certain model [11]. One of its most successful applications is the verification of a point-based temporal logic specification (expressed, for instance, by a formula of Linear Temporal Logic [34]) against some reactive system description [35]. It is well known that query evaluation and constraint checking in relational databases can be naturally expressed as model checking problems (see, for instance, [43]). The applicability of model checking techniques for the retrieval and the verification of temporal data has also been explored in the literature, e.g., [5, 38].

Time intervals are commonly used to represent temporal information in (valid-time) temporal databases. On the one hand, they allow one to compactly represent the time periods over which data are valid in the modeled domain [42]; on the other hand, they make it possible to suitably represent inherently interval-based temporal information such as telic facts and temporal aggregations [4, 20]. Accordingly, temporal queries, constraints, and rules can be naturally formulated as formulas of an interval temporal logic to be evaluated over temporal datasets represented as finite interval models. The problem of evaluating a temporal dataset can thus be reduced to the model checking problem for interval temporal logic formulas, making it possible to exploit techniques and tools from logic and formal methods to address and solve problems in temporal databases and data mining (see, for instance, the three aforementioned application domains).

As a matter of fact, there is a little mismatch between the expected outcomes of the two problems: while model checking is a decision problem, which returns "yes" when the model meets the specification and "no" otherwise, the problem of temporal dataset evaluation directly or indirectly returns a set of histories—in rule evaluation, it determines to what extent each single history (resp., the whole set of histories) complies with a certain temporal rule. An actual solution to the problem of temporal dataset evaluation can be obtained by representing a temporal dataset as a set of finite interval models and by suitably combining the outcomes of model checking each single model against the formula. In the case of rule evaluation, some domain-dependent measures for *rating* the compliance degree of the interval model with respect to the formula, like, for instance, sensitivity, specificity, and accuracy [44], are also needed.

Our contribution. The contribution of the present paper is twofold. From a methodological perspective, it proposes interval temporal logic model checking as a viable logical tool for temporal dataset evaluation (in particular, for rule evaluation in valid-time temporal databases), thus establishing a formal connection between the two problems. From a technical point of view, it provides an efficient solution to the problem of model checking an interval temporal logic formula against a finite model (also referred to as path or history). More precisley, we solve the problem of model checking a finite path/history for the well-known Halpern and Shoham's interval temporal logic (HS for short) [17], which features one modality for each Allen relation [2], by devising a deterministic model checking algorithm that runs in polynomial time, thus proving that the problem is in PTIME.

Related work. In the last years, the model checking problem for interval temporal logic has received an increasing attention as an alternative to the traditional (point-based) temporal logic model checking, which can be recovered as a special case. Model checking for full HS, interpreted over finite Kripke structures according to the *state-based semantics* (we refer here to the terminology introduced in [7]), has been studied in [29, 33]. The authors showed that, under the homegeneity assumption, which constrains a proposition letter to hold over an interval if and only if it holds over each component state, the problem is non-elementarily decidable (EXPSPACE-hardness has been later shown in [6]). Since then, the attention was brought to HS fragments, which are often computationally much better [6, 8, 30, 31, 32]. The model checking problem for some HS fragments extended with epistemic operators has been investigated in [24, 25]. The semantic assumptions for these epistemic HS fragments differ from those of [29, 33] in two important respects, making it difficult to compare the two families of logics: formulas are interpreted over the unwinding of the Kripke structure (*computation-tree-based semantics*, borrowing the terminology from [7]) and interval labeling takes into account only the endpoints of intervals. The latter assumption has been later relaxed by making it possible to use regular expressions to define the labeling of proposition letters over intervals in terms of the component states [26].

A common feature of the application of (interval) model checking to the verification of temporal properties of a reactive system is the encoding of all its possible executions by a finite-state transition system, that is, by a finite Kripke structure, which provides an abstract representation of possibly infinitely many interval models. On the contrary, (valid-time) temporal databases commonly assume a given structure of time. Hence, when used for the evaluation of temporal datasets, interval model checking is applied to finite, concrete interval models and makes no restrictive assumptions on interval labeling such as, for instance, the homogeneity assumption. In fact, representing temporal datasets as suitable Kripke structures over which to evaluate formulas is in principle possible. However, it would be both artificial and computationally inconvenient. As we will show, interval model checking for temporal dataset evaluation behaves computationally much better than interval model checking for system verification: we devise a polynomial model checking algorithm for full HS, while the complexity of model checking HS fragments against Kripke structures goes from coNP-complete to non-elementary, depending on the particular fragment of HS under consideration [6, 8, 24, 25, 26, 29, 30, 31, 32, 33]. As a matter of fact, the closest analogue of our model checking problem is the problem of model checking a path in Linear Temporal Logic (LTL) worked out by Markey and Schnoebelen in [27].

HS	Allen's relations	Graphical representation
$\langle A \rangle$	$[x, y]R_A[x', y'] \Leftrightarrow y = x'$	
$\langle L \rangle$	$[x, y]R_L[x', y'] \Leftrightarrow y < x'$	
$\langle B \rangle$	$[x, y]R_B[x', y'] \Leftrightarrow x = x', y' < y$	
$\langle E \rangle$	$[x, y]R_E[x', y'] \Leftrightarrow y = y', x < x'$	
$\langle D \rangle$	$[x, y]R_D[x', y'] \Leftrightarrow x < x', y' < y$	
$\langle O \rangle$	$[x, y]R_O[x', y'] \Leftrightarrow x < x' < y < y'$	

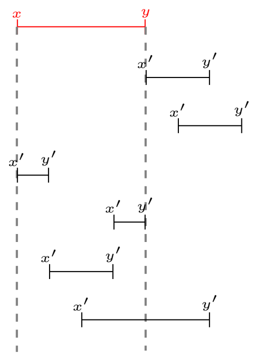


Figure 1 Allen's interval relations and HS modalities.

Organization of the paper. In Section 2, we introduce the logic HS and define the interval model checking problem on a single path/history. Then, in Section 3, we illustrate a range of possible applications of temporal dataset evaluation. Finally, in Section 4, we prove that the interval model checking problem on a single path/history is in PTIME. Conclusions provide an assessment of the work done and outline future research directions.

2 Preliminaries

Let $\mathbb{D} = \langle D, < \rangle$ be a linear order. A *strict* (resp., *non-strict*) interval over $\mathbb{D}$ is an ordered pair $[x, y]$, where $x, y \in D$ and $x < y$ (resp., $x \leq y$). As it is usually the case with the recent literature, we adopt the *strict semantics*, which admits strict intervals only. Such a choice conforms to the definition of interval given by Allen in [2], but it differs from the one by Halpern and Shoham [17]. Even though most results can be easily rephrased in *non-strict semantics*, which also admits intervals of the form $[x, x]$ (*point intervals*), the strict one is definitely cleaner for at least two reasons: first, a number of representation paradoxes arise when the non-strict semantics is adopted, due to the presence of point intervals, as pointed out in [2]; second, when point intervals are included there seems to be no intuitive semantics for interval relations that makes them both pairwise disjoint and jointly exhaustive. Moreover, adopting strict semantics is coherent with recent developments in temporal logic that consider points and intervals on a par as different semantic entities (see, e.g., [3]).

We denote by $\mathbb{I}(\mathbb{D})$ the set of (strict) intervals over a linear order $\mathbb{D}$. If we exclude the identity relation, there are 12 different relations between two intervals in a linear order, often called *Allen's relations* [2]: the six relations R_A (adjacent to), R_L (later than), R_B (begins), R_E (ends), R_D (during), and R_O (overlaps), depicted in Figure 1, and their inverses, that is, $R_{\overline{X}} = (R_X)^{-1}$, for each $X \in \mathcal{A}$, where $\mathcal{A} = \{A, L, B, E, D, O\}$. We associate a universal modality $[X]$ and an existential one $\langle X \rangle$ with each Allen relation R_X. For each $X \in \mathcal{A}$, the *transposes* of the modalities $[X]$ and $\langle X \rangle$ are respectively the modalities $[\overline{X}]$ and $\langle \overline{X} \rangle$, corresponding to the inverse relation $R_{\overline{X}}$ of R_X, and vice versa.

The logic HS ([17]). Halpern and Shoham's HS can be viewed as a multi-modal logic whose formulas are built from a finite, non-empty set $\mathcal{AP}$ of *atomic propositions* (also referred to as *proposition letters*), the classical Boolean connectives, and a modality for each Allen relation:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle X \rangle \varphi \mid \langle \overline{X} \rangle \varphi,$$

where $p \in \mathcal{AP}$ and $X \in \mathcal{A}$. The other Boolean connectives and the logical constants, e.g., $\rightarrow$ and $\top$, as well as the universal modalities $[X]$, can be defined in the standard way.

As shown in [17], the modalities $\langle A \rangle$, $\langle B \rangle$, and $\langle E \rangle$, along with their transposes, are sufficient to express all the other modalities through a formula of polynomial size. Thus, w.l.o.g., hereafter we restrict ourselves to HS formulas over the set of modalities $\{\langle A \rangle, \langle \overline{A} \rangle, \langle B \rangle, \langle \overline{B} \rangle, \langle E \rangle, \langle \overline{E} \rangle\}$.

The semantics of HS formulas is given in terms of *interval models* $M = \langle \mathbb{D}, V \rangle$, where $\mathbb{D}$ is a linear order and $V : \mathcal{AP} \rightarrow 2^{\mathbb{I}(\mathbb{D})}$ is a *valuation function* which assigns to each atomic proposition $p \in \mathcal{AP}$ the set of intervals $V(p)$ on which p holds. (With an abuse of notation, we indistinctly treat valuation functions as functions from atomic propositions to sets of intervals, that is, $V : \mathcal{AP} \rightarrow 2^{\mathbb{I}(\mathbb{D})}$, and as functions from intervals to sets of atomic propositions, that is, $V : \mathbb{I}(\mathbb{D}) \rightarrow 2^{\mathcal{AP}}$.) In this work, we are interested in finite structures and thus we restrict our attention to linear orders over finite domains. Any finite linear order $\mathbb{D}$ of size n can be compactly represented by $[n] = \{x \in \mathbb{N} \mid x \leq n\}$. In the following, we will make use of such a representation.

The *truth* of a formula φ on a given interval $[x, y]$ in an interval model M is defined by structural induction on formulas as follows:

- $M, [x, y] \Vdash p$ iff $[x, y] \in V(p)$, for $p \in \mathcal{AP}$;
- $M, [x, y] \Vdash \neg\psi$ iff $M, [x, y] \nVdash \psi$;
- $M, [x, y] \Vdash \psi \vee \xi$ iff $M, [x, y] \Vdash \psi$ or $M, [x, y] \Vdash \xi$;
- $M, [x, y] \Vdash \langle X \rangle \psi$ iff there exists $[w, z]$ such that $[x, y] R_X [w, z]$ and $M, [w, z] \Vdash \psi$.

We denote the modal depth of an HS formula φ by $md(\varphi)$. In [1], it has been shown that, over finite temporal domains, an HS formula φ can be translated into an equivalent one, say it φ', that involves neither $\langle A \rangle$ nor $\langle \overline{A} \rangle$. On the one hand, since the size of φ' may be exponential in the size of φ, we cannot give up modalities $\langle A \rangle$ and $\langle \overline{A} \rangle$ when addressing complexity issues; on the other hand, since $md(\varphi')$ is only linearly larger than $md(\varphi)$, such a translation will come in handy when proving Lemma 1 (bisimulation lemma) in Section 4, allowing us to ignore modalities $\langle A \rangle$ and $\langle \overline{A} \rangle$ in that context.

In the following, we will make use of the *global* modality $[U]$, that allows one to assert a property φ on the entire model. HS is powerful enough to express such a modality as follows:

$$[U]\varphi \overset{def}{\equiv} \varphi \wedge \bigwedge_{X \in \mathcal{A}} ([X]\varphi \wedge [\overline{X}]\varphi).$$

The HS model checking problem against a finite path/history. Given a pair $\mathcal{I} = (M, \varphi)$, where M is a finite interval model and φ is an HS formula, the problem of *model checking* the HS formula φ against a finite path/history M (MC for short) consists in deciding whether $M, [0, 1] \Vdash \varphi$. The pair $\mathcal{I} = (M, \varphi)$ is called an *instance* of MC. With a little abuse of the notation, for a given instance (M, φ) of MC, we write $(M, \varphi) \in$ MC to indicate that $M, [0, 1] \Vdash \varphi$. In such a case, we say that MC applied to (M, φ), denoted by MC(M, φ), returns true. We say that two instances $\mathcal{I}, \mathcal{I}'$ of MC are *equivalent*, denoted by $\mathcal{I} \equiv_{\mathsf{MC}} \mathcal{I}'$ whenever MC$(\mathcal{I})$ returns true if and only if MC$(\mathcal{I}')$ does. Finally, we say that two HS formulas φ_1 and φ_2 are *equivalent*, denoted by $\varphi_1 \equiv \varphi_2$, if $(M, \varphi_1) \equiv_{\mathsf{MC}} (M, \varphi_2)$ for every interval model M.

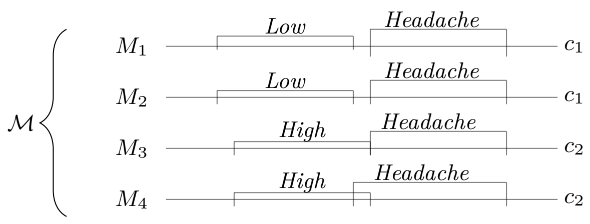


Figure 2 A classification model with four patients partitioned into two classes (c_1 and c_2).

3 Temporal Dataset Evaluation

The problem of *temporal dataset evaluation* can be defined as the problem of evaluating how many histories comply with a given temporal formula. A history can be seen as the entire (finite) flow of information relative to a given entity over a bounded time span. As an example, in the medical context, a history is the *medical history* of a patient, that is, the collection of all relevant pieces of information about tests, results, symptoms, and hospitalizations of the patient that occurred during the entire observation period. In Figure 2, we graphically depict (a simplified version of) the history of four patients.

As far as qualitative reasoning is concerned, it turns out to be convenient to assume that all the meaningful events have been suitably *discretized*. As a concrete example, having a fever can be represented by the propositional letter *Low*—meaning lower than 40 degrees—or *High*—meaning higher than or equal to 40 degrees. Similarly, the proposition letter *Headache* can be used to indicate the presence of a headache.

The case of fever is an illuminating example of the fact that information about histories is naturally *interval-based* and quite often *non-homogeneous*. Suppose, indeed, that a certain patient is experiencing low fever in an interval $[x, y]$, say, a day, and that during just one hour of that day, that is, over the interval $[w, z]$ strictly contained in $[x, y]$, he/she has an episode of high fever. A natural choice is to represent such a situation by labeling the interval $[x, y]$ with *Low* and its sub-interval $[w, z]$ with *High*. Notice that such a representation is compatible with a general consistency requirement such as $[U](Low \to \neg High)$. On the other hand, representing the same pieces of information as three intervals $[x, w], [w, z], [z, y]$ respectively labeled with *Low*, *High*, and *Low*, which would be the case with a point-based representation (or with an interval-based representation under the homogeneity assumption), would be unnatural, and it would entail hiding a potentially important information like: "*the patient presented low fever during the entire day, except for a (brief) episode of high fever*".

Let us denote a temporal dataset by $\mathcal{D} = \{M_1, \ldots, M_r\}$, where each history M_i, with $1 \leq i \leq r$, is, in fact, an interval model as defined in Section 2. Temporal dataset evaluation can be naturally formulated as a model checking problem for an interval temporal logic like HS. In the following, we briefly elaborate on its concrete application to temporal query processing, temporal constraint checking, and machine learning (rule evaluation).

Evaluating a query over a temporal database corresponds to extracting the set of histories that satisfy the conditions of the query. Temporal queries are typically expressed in a temporal query language such as TSQL (Temporal SQL) [40], which features operators for Allen's relations that make it possible to check all possible temporal (interval) relationships between pairs of events. (Another approach to query evaluation based on temporal formalisms with an interval flavor can be found in [10], where the language ATSQL is used; however, the underlying semantics is not truly interval based, as homogeneity is assumed.) The interval temporal logic HS is powerful enough to capture all TSQL operators, which can actually be expressed as suitable disjunctions of HS modalities. To answer a temporal query, an evaluation of the whole dataset of histories is, in general, needed, that returns precisely those histories on which the model checking problem is positively solved. Therefore, it can be naturally viewed as a dataset evaluation problem. As an example, determining the patients that have experienced a headache immediately after an episode of high fever amounts to looking for those histories M_i such that:

$$M_i, [0,1] \Vdash \langle L\rangle(\mathit{High} \wedge \langle A\rangle \mathit{Headache}).$$

In the example of Figure 2, only M_3 makes the formula true.

As far as *temporal constraint checking* is concerned, in a (temporal) database one may impose various different types of (temporal) constraint at design time [23]. They range from very simple *domain* constraints like "each value of a given attribute must belong to a specific domain", to *key* constraints, that is, existence and uniqueness of the values of the key attributes, and, more generally, *functional dependencies*, which require some attributes to functionally depend on other ones (in the context of temporal databases, interval-based temporal functional dependencies have been studied in [13]). More advanced temporal constraints, that involve more complex relationships among the values of each history, can be expressed as logical formulas, and checking such constraints corresponds to solving the temporal dataset evaluation problem and identifying precisely those histories for which the model checking problem returns false. Let us consider again the simple example reported in Figure 2. In order to check that data about the values of the fever parameter over time, for any given patient, are consistent, that is, to exclude that there exists a patient such that both *High* and *Low* hold in the same time interval, it suffices to check that the number of histories M_i such that:

$$M_i, [0,1] \Vdash [U](\mathit{High} \wedge \mathit{Low})$$

is equal to 0 (this is actually the case with the temporal dataset in Figure 2).

Finally, one of the most interesting and extensively studied problems in *machine learning* is supervised classification. In such a problem, each history is assigned to a *class* c from a finite set $\mathcal{C}$ with the aim of devising a module, usually called *classifier*, that, given a new (unclassified) history, sets its class with an acceptable degree of correctness. In our running example (see Figure 2), each patient is either class c_1 or class c_2, which can be possibly interpreted as "the patient is cured (at the end of the treatment) or not".

There are several well-known (not always comparable) classifier learning methods. They can be broadly categorized into learners based on *trees*, on *functions*, and on *rule sets*. *Decision tree learners*, such as C4.5 [37], fall into the first category, while a *logistic regressor* [22] is an example of function-based classifier learner. Methods based on rule sets, that is, *rule extraction*, are further partitioned into indirect and direct methods. When an indirect method is adopted, rules are synthesized from an already existing classifier [9, 18, 28], while when a direct method is followed, rules are directly learned [19, 44].

Let us focus on the latter methods (direct rule extraction). A classification *rule* has the form $\varphi \Rightarrow c$, where φ is a logical formula and $c \in \mathcal{C}$. Rules are not implicative formulas (we use the symbol $\Rightarrow$, instead of $\rightarrow$, to stress this fact). Direct methods include methods that pair inductive reasoning and programming languages, such as *inductive logic programming*, and randomized methods, like *evolutionary algorithms*. Direct rule extraction via evolutionary algorithms is a simple, and yet very promising, methodology. Intuitively, a set of random rules is initially produced. Then, at each iteration, the current set is evaluated and, as a result, a new (better) set of rules is built from the old one that takes into account the result of the evaluation. Evolutionary algorithms produce a new solution from the current one by means of *evolutionary operators*, that may be generic and/or specific to the problem at hand. *Multi-objective* evolutionary algorithms take into account more than one evaluation measure of the current solution. As an example, the current set of rules can be evaluated by the accuracy of the rules *and* by their succinctness—see, for instance, [14]. While different approaches based on this idea may differ in the way in which they represent rules, in the set of evolutionary operators, and in the selection strategy, the key element, common to all of them, is the evaluation of a solution, which, in the context of evolutionary algorithms, is called *fitting* function. Such a problem of evaluating the current solution can naturally be viewed as a dataset evaluation problem.

Although there is not a unique, commonly accepted definition, a fairly natural way of computing the fitting degree of a set of rules $R_1, \dots, R_p$, where R_i is a generic rule of the form $\varphi_i \Rightarrow c_i$, for $1 \leq i \leq p$, is by suitably combining the quantities $\Sigma_{i=1}^{p} \frac{Rec(i)}{C(i)}$ (known as *accuracy*) and $\Sigma_{i=1}^{p} \frac{Rec(i)}{\Phi(i)}$ (*recall*), where $C(i)$ is the number of histories in the class c_i, $\Phi(i)$ is the number of histories that comply with φ_i, and $Rec(i)$ is the number of histories in the class c_i that comply with φ_i [36]. Both accuracy and recall clearly depend on the number of histories M_j such that $M_j, [0,1] \Vdash \varphi_i$. In our running example (Figure 2), two rules can be naturally devised:

$$\langle L\rangle(Low \wedge \langle L\rangle Headache) \Rightarrow c_1;$$
$$\langle L\rangle(High \wedge (\langle A\rangle Headache \vee \langle O\rangle Headache)) \Rightarrow c_2.$$

To summarize, we have shown that the temporal dataset evaluation is a relevant problem that comes into play in a variety of application domains including temporal query processing, temporal constraint checking, and rule evaluation. An efficient solution to the model checking problem for interval temporal logic over finite structures turns out to be a fundamental step towards its effective treatment. In the next section we provide such a solution.

4 Model Checking

The input of classic (point-based) model checking consists of a formula and a Kripke structure generally represented by (a suitable encoding of) the set of its states, along with their labels, and the set of its transitions—for the model checking of very huge structures, other kinds of solution are used, e.g., on-the-fly or bounded model checking. Classic model checking is infinite in nature: infinitely many, possibly infinite paths, which are finitely encoded in the input structure. On the contrary, in order to model check a finite path/history, that is, a finite labeled interval, one can represent the model simply by specifying the size of the interval (number of points in the model) and then listing, for each proposition letter, the set of sub-intervals over which it is true (see Figure 3).

The fundamental difference between the two frameworks is that, in the classic model checking problem, frame information, i.e., states and transitions, must be explicitly represented in the input, while, in finite interval models, frame information, i.e., intervals and their relations, is implicit in the size of the temporal domain (this is because relations among intervals are induced by the underlying linear order). As a consequence, while the size of the representation of a Kripke structure is typically polynomial in the number of states and labels, the size of the representation of a finite interval model may be logarithmic in the number of intervals. As an example, consider an interval model $M = \langle [n], V\rangle$ over $\mathcal{AP} = \{p\}$,

n
$p : [0,1], [0,2], [0,3], [1,3], [2,3], \dots$
$q : [0,1], [0,3], \dots$
$\dots$

Figure 3 A succinct representation of a finite interval model for HS: the first line specifies the size of the history (number of points in the model); the next lines encode the valuation function V.

where $V(p) = \{[n-1, n]\}$. Its representation consists of the number n (which takes space $O(log(n))$) and the mapping $p \mapsto \{[n-1, n]\}$ (which takes space $O(log(n))$ as well).

We will capture situations like the above one by means of the concept of *sparse* MC *instance*. Intuitively, an instance (M, φ) of MC is sparse if it can be represented by logarithmic space (the notion will be formalized later). Now, given an MC instance (M, φ), a model checking procedure can be obtained by a straightforward adaptation of the CTL model checking algorithm by Emerson and Clarke [15] to the interval setting; such a procedure labels each world, i.e., interval, of M with the set of sub-formulas of φ which are true on it. The application of such an algorithm to sparse instances immediately shows its exponential complexity. As an example, checking the formula $\langle A\rangle\langle A\rangle p$ against the model $M = \langle [n], V\rangle$, where $V(p) = \{[n-1, n]\}$, would require labeling with $\langle A\rangle p$ all intervals $[x, n-1]$, with $1 \leq x \leq n-2$, whose number is linear in n, and thus exponential in the size of the representation of M.

In the following, we describe a model checking algorithm that runs in polynomial time on every instance, thus avoiding the above problem.

The model checking algorithm. To keep the complexity under control, the algorithm we are going to describe first executes a preprocessing of a sparse instance, which may be represented in logarithmic space, and generates a non-sparse one; then, it basically applies Emerson and Clarke's solution of the model checking problem.

Hereafter, we fix an HS formula φ over the set of modalities $\{\langle A\rangle, \langle\overline{A}\rangle, \langle B\rangle, \langle\overline{B}\rangle, \langle E\rangle, \langle\overline{E}\rangle\}$ and we let $k = 4 \cdot md(\varphi)$. In addition, let $M = \langle [n], V\rangle$ be a model and let $c \in [n]$ be an element of the domain. We define a transformation $\tau_c(M) = \langle [n'], V'\rangle$, where $n' = n-1$ and $V' : \mathbb{I}([n']) \to 2^{\mathcal{AP}}$ is defined as follows:

$$V'([w,z]) = \begin{cases} V([w,z]) & \text{if } z < c, \\ V([w,z+1]) & \text{if } w < c \leq z, \\ V([w+1,z+1]) & \text{if } w \geq c. \end{cases}$$

Given an interval model $M = \langle [n], V\rangle$ and a point $a \in [n]$, we say that a is a *useless point* if $V([a,y]) = V([x,a]) = \emptyset$ for all $x, y \in [n]$, with $x < a < y$. Moreover, we say that an interval $[a,b] \in \mathbb{I}(\mathbb{D})$, with $b - a > 2 \cdot (k+1)^2 + 1$, is a *gap* in $M = \langle [n], V\rangle$ if each $x \in [n]$, with $a \leq x \leq b$, is a useless point in M, while $a-1$ and $b+1$ are not. Finally, given a gap $[a,b]$, we call the point $c = a + (k+1)^2 + 1$ the *center* of the gap.

Given an interval model M, we denote by n_M the cardinality of its domain and by u_M the number of useless points in it. We now show that every instance with a gap can be transformed into an equivalent one, of smaller size, that has no gaps. The proof is based on the notion of bisimulation and the invariance of modal logics under bisimulations.

For every $m \in \mathbb{N}$ and each set $\mathcal{S}$ of HS modalities, an *m-bisimulation for $\mathcal{S}$* between two interval models $M_1 = \langle [n_1], V_1\rangle$ and $M_2 = \langle [n_2], V_2\rangle$ is a sequence $\langle Z_m, Z_{m-1}, \dots, Z_1, Z_0\rangle$

of binary relations between the intervals in $\mathbb{I}([n_1])$ and those in $\mathbb{I}([n_2])$, that is, $Z_i \subseteq \mathbb{I}([n_1]) \times \mathbb{I}([n_2])$ for $i \in \{0, 1, \ldots, m\}$, such that:

- $V_1([x, y]) = V_2([w, z])$, for all $i \in \{0, 1, \ldots, m\}$ and $([x, y], [w, z]) \in Z_i$ (*local condition*);
- for all $i \in \{1, \ldots, m\}$, $([x, y], [w, z]) \in Z_i$, and $X \in \mathcal{S}$:
  - if $[x, y] R_X [x', y']$, then there is $[w', z']$ such that $([x', y'], [w', z']) \in Z_{i-1}$ and $[w, z] R_X [w', z']$ (*forward condition*), and
  - if $[w, z] R_X [w', z']$, then there is $[x', y']$ such that $([x', y'], [w', z']) \in Z_{i-1}$ and $[x, y] R_X [x', y']$ (*backward condition*).

Two intervals $[x, y]$ and $[w, z]$ are *m-bisimilar under $\mathcal{S}$* if there is an m-bisimulation $\langle Z_m, Z_{m-1}, \ldots, Z_1, Z_0\rangle$ for $\mathcal{S}$ such that $([x, y], [w, z]) \in Z_m$.

The next lemma is based on the well-known *invariance* of modal logics under bisimulations [16], which can be stated as follows: if $\mathcal{L}$ is a modal logic over a set of modalities $\mathcal{S}$ and there exists an m-bisimulation for $\mathcal{S}$ between two models M_1 and M_2 of $\mathcal{L}$, then, for every pair of m-bisimilar worlds w_1 and w_2, it holds that $M_1, w_1 \Vdash \psi$ if and only if $M_2, w_2 \Vdash \psi$, for every formula ψ of $\mathcal{L}$ with $md(\psi) \leq m$.

▶ **Lemma 1.** *If (M, φ) is an instance of* MC *featuring a gap $[a, b]$ in M, then $(M, \varphi) \equiv_{\mathsf{MC}} (\tau_c(M), \varphi)$, where c is the center of $[a, b]$.*

Proof. Let $M = \langle [n], V\rangle$ and $M' = \tau_c(M) = \langle [n'], V'\rangle$, and recall that $n' = n - 1$. Even though φ is built over the set of modalities $\{\langle A\rangle, \langle \overline{A}\rangle, \langle B\rangle, \langle \overline{B}\rangle, \langle E\rangle, \langle \overline{E}\rangle\}$, it is known from [1] that it can be translated into an equivalent formula φ' over the set of modalities $\{\langle B\rangle, \langle \overline{B}\rangle, \langle E\rangle, \langle \overline{E}\rangle\}$, with $md(\varphi')$ linear in $md(\varphi)$ (in fact, $md(\varphi') \leq k = 4 \cdot md(\varphi)$).

We provide a k-bisimulation for $\{\langle B\rangle, \langle \overline{B}\rangle, \langle E\rangle, \langle \overline{E}\rangle\}$ between M and M' such that $([0, 1], [0, 1]) \in Z_k$. The claim immediately follows from the aforesaid invariance property.

Let $f : \{0, 1, \ldots, k\} \to \mathbb{N}$ be the mapping defined as follows: $f(i) = (1 + k - i) \cdot (k + 1)$. For each $i \in \{0, 1, \ldots, k\}$ and $([x, y], [w, z]) \in \mathbb{I}([n]) \times \mathbb{I}([n'])$, we state that $([x, y], [w, z]) \in Z_i$ if and only if one of the following conditions hold:

(1) $w = x \wedge z = y \wedge y \leq c + f(i)$, or
(2) $w = x \wedge z = y - 1 \wedge x \leq c + f(i) \wedge y \geq c - f(i) \wedge z - w > i$, or
(3) $w = x - 1 \wedge z = y - 1 \wedge x \geq c - f(i)$.

We first observe that f is monotonically decreasing as $f(i - 1) - f(i) = k + 1 > i$ for all $i \in \{1, \ldots, k\}$. Moreover, it holds that $f(0) = (k + 1)^2$ and thus, by the definitions of gap and center of a gap, we have that:

$$c - a = (k + 1)^2 + 1 > (k + 1)^2 = f(0)$$

and

$$b - c = b - a - (k + 1)^2 - 1 > 2 \cdot (k + 1)^2 + 1 - (k + 1)^2 - 1 = (k + 1)^2 = f(0).$$

Therefore, it holds that $a \leq c - f(0) - 1$ and $c + f(0) + 1 \leq b$, and thus, for each $i \in \{0, 1, \ldots, k\}$ and each d in between $c - f(i) - 1$ and $c + f(i) + 1$, that is, $c - f(i) - 1 \leq d \leq c + f(i) + 1$, d is a useless point.

We now show that $\langle Z_k, Z_{k-1}, \ldots, Z_1, Z_0\rangle$ satisfies local, forward, and backward conditions.

Local condition. Let $([x,y],[w,z]) \in Z_i$ for some $i \in \{1,\dots,k\}$. We have to show that $V([x,y]) = V'([w,z])$. We must deal with the three possible cases.

- Case (1). If $w = x$, $z = y$, and $y \leq c + f(i)$, then $x < c + f(i)$ and:
  - if $c \leq w < c + f(i)$, then w $(= x)$ is useless and so is $w+1$, and thus $V'([w,z]) = V([w+1, z+1]) = \emptyset = V([x,y])$;
  - if $z < c$, then $V'([w,z]) = V([w,z]) = V([x,y])$;
  - if $w < c$ and $c \leq z \leq c + f(i)$, then z $(= y)$ is useless and so is $z+1$, and thus $V'([w,z]) = V([w,z+1]) = \emptyset = V([x,y])$.
- Case (2). If $w = x$, $z = y - 1$, $x \leq c + f(i)$, $y \geq c - f(i)$, and $z - w > i$, then:
  - if $c \leq w \leq c + f(i)$, then w $(= x)$ is useless and so is $w+1$, and thus $V'([w,z]) = V([w+1, z+1]) = \emptyset = V([x,y])$;
  - if $c - f(i) - 1 \leq z < c$, then z is useless and so is y $(= z+1)$, and thus $V'([w,z]) = V([w,z]) = \emptyset = V([x,y])$;
  - if $w < c$ and $z \geq c$, then $V'([w,z]) = V([w,z+1]) = V([x,y])$.
- Case (3). If $w = x - 1$, $z = y - 1$, and $x \geq c - f(i)$, then $y > c - f(i)$ and:
  - if $c - f(i) \leq z < c$, then z is useless and so is y $(= z+1)$, and thus $V'([w,z]) = V([w,z]) = \emptyset = V([x,y])$;
  - if $w \geq c$, then $V'([w,z]) = V([w+1, z+1]) = V([x,y])$;
  - if $z \geq c$ and $c - f(i) - 1 \leq w < c$, then w is useless and so is x $(= w+1)$, and thus $V'([w,z]) = V([w,z+1]) = \emptyset = V([x,y])$.

Forward condition. Let $([x,y],[w,z]) \in Z_i$, for some $i \in \{1,2,\dots,k\}$, and $[x,y] R_X [x',y']$, for some X. We show that there exists $[w',z']$ such that $([x',y'],[w',z']) \in Z_{i-1}$ and $[w,z] R_X [w',z']$. We proceed case by case, taking into account the value of X and the relationship that holds between $[x,y]$ and $[w,z]$:

- Case (1). If $w = x$, $z = y$, and $y \leq c + f(i)$, then $x < c + f(i)$ and:
  - if $X = B$, $X = E$, or $X = \overline{E}$, then we set $w' = x'$ and $z' = y'$; the new points x', y', w', and z' satisfy (1) with respect to $i - 1$, that is, $w' = x'$, $z' = y'$, and $y' \leq c + f(i-1)$, meaning that $[x',y'] Z_{i-1} [w',z']$;
  - if $X = \overline{B}$, then $x' = x$ and $y' > y$; we set $w' = x'$ and, in order to set the value for z', we distinguish two cases:
    * if $y' \leq c + f(i-1)$, then we set $z' = y'$ and thus x', y', w', and z' satisfy (1) with respect to $i - 1$;
    * if $y' > c + f(i-1)$, then we set $z' = y' - 1$ and thus x', y', w', and z' satisfy (2) with respect to $i - 1$; in particular, to see that $z' - w' > i - 1$, observe that $z' - w' = y' - 1 - x > c + f(i-1) - c - f(i) - 1 = k > i - 1$, and to see that $z' > z$, and thus $[w,z] R_{\overline{B}} [w',z']$, observe that $y' - 1 > c + f(i-1) - 1 > c - f(i) \geq y$, which implies $z' = y' - 1 > y = z$.
- Case (2). If $w = x$, $z = y - 1$, $x \leq c + f(i)$, $y \geq c - f(i)$, and $z - w > i$, then:
  - if $X = \overline{B}$ or $X = \overline{E}$, then we set $w' = x'$ and $z' = y' - 1$, and thus x', y', w', and z' satisfy (2) with respect to $i - 1$;
  - if $X = B$, then $x' = x$ and $y' < y$; we set $w' = x'$ and, in order to set the value for z', we distinguish the following cases:
    * if $y' \geq c - f(i-1)$ and $y' - x' > i$, then we set $z' = y' - 1$, and thus x', y', w', and z' satisfy (2) with respect to $i - 1$; in particular, it holds that $z' - w' = y' - 1 - x' \geq c - f(i-1) - 1 - c - f(i) = k > i - 1$;

  - ∗ if $y' < c - f(i-1)$, then we set $z' = y'$, and thus x', y', w', and z' satisfy (1) with respect to $i-1$; in particular, to see that $z' < z$, and thus $[w, z]R_B[w', z']$, observe that $z' = y' < c - f(i-1) < c - f(i) - 1 \leq y - 1 = z$;
  - ∗ if $y' - x' \leq i$, then we set $z' = y'$, and thus x', y', w', and z' satisfy (1) with respect to $i-1$; in particular, it holds that $y' \leq x' + i \leq c + f(i) < c + f(i-1)$.
- Case (3). If $w = x - 1$, $z = y - 1$, and $x \geq c - f(i)$, then $y > c - f(i)$ and:
  - if $X = B$, $X = \overline{B}$, or $X = E$, then we set $w' = x' - 1$ and $z' = y' - 1$; the new points x', y', w', and z' satisfy (3) with respect to $i-1$;
  - if $X = \overline{E}$, then $x' < x$ and $y' = y$; we set $z' = z = y' - 1$ and, in order to set the value for w', we distinguish two cases:
    - ∗ if $x' \geq c - f(i-1)$, then we set $w' = x' - 1$, and thus x', y', w', and z' satisfy (3) with respect to $i-1$;
    - ∗ if $x' < c - f(i-1)$, then we set $w' = x'$, and thus x', y', w', and z' satisfy (2) with respect to $i-1$; in particular, to see that $z' - w' > i - 1$, observe that $z' - w' = y - 1 - x' > c - f(i) - 1 - c + f(i-1) = k > i - 1$, and to see that $w' < w$, and thus $[w, z]R_{\overline{E}}[w', z']$, observe that $w' = x' < c - f(i-1) < c - f(i) - 1 \leq x - 1 = w$.

The backward condition can be proved in a very similar way.

Therefore, the sequence $\langle Z_k, Z_{k-1}, \ldots, Z_1, Z_0\rangle$ is a k-bisimulation for $\{\langle B\rangle, \langle\overline{B}\rangle, \langle E\rangle, \langle\overline{E}\rangle\}$ between M and M'. In addition, we have that $([0,1],[0,1]) \in Z_k$, and since φ is equivalent to φ' and $md(\varphi') \leq k$, we have that:

$$M, [0,1] \Vdash \varphi \Leftrightarrow M, [0,1] \Vdash \varphi' \Leftrightarrow M', [0,1] \Vdash \varphi' \Leftrightarrow M', [0,1] \Vdash \varphi,$$

and thus the thesis. ◀

We are now ready to formalize the notion of sparse MC instance. We say that an instance $\mathcal{I} = (M, \varphi)$ of MC is *sparse* if

$$u_M > \frac{2 \cdot (k+1)^2 + 2}{2 \cdot (k+1)^2 + 3} \cdot n_M + 1.$$

By making use of Lemma 1, it is possible to transform a sparse instance of MC into an equivalent non-sparse one, as formally stated by the following lemma.

▶ **Lemma 2.** *For every sparse instance of* MC*, there exists an equivalent non-sparse one that is computable in deterministic polynomial time.*

Proof. As a preliminary step, we show that if $\mathcal{I} = (M, \varphi)$ is a sparse instance of MC, then it features a gap $[a, b]$ in M. Assume, towards a contradiction, that $\mathcal{I}$ features no gap and consider the partition of $[n_M]$ into intervals $[(i-1)\cdot\big(2\cdot(k+1)^2+3\big)+1, i\cdot\big(2\cdot(k+1)^2+3\big)]$, with $1 \leq i \leq \lfloor\frac{n_M}{2\cdot(k+1)^2+3}\rfloor$, plus the interval $[\Big(\lfloor\frac{n_M}{2\cdot(k+1)^2+3}\rfloor\Big)\cdot\big(2\cdot(k+1)^2+3\big)+1, n_M]$. Since the length of each interval $[(i-1)\cdot\big(2\cdot(k+1)^2+3\big)+1, i\cdot\big(2\cdot(k+1)^2+3\big)]$, with $1 \leq i \leq \lfloor\frac{n_M}{2\cdot(k+1)^2+3}\rfloor$, is equal to $2\cdot(k+1)^2+2$, any such interval must contain a non-useless point, otherwise there would be a gap in M. It immediately follows that there are at least $\lfloor\frac{n_M}{2\cdot(k+1)^2+3}\rfloor$ non-useless points in M, and thus it holds that:

$$u_M \leq n_M - \left\lfloor\frac{n_M}{2\cdot(k+1)^2+3}\right\rfloor \leq n_M - \frac{n_M}{2\cdot(k+1)^2+3} + 1 = \frac{2\cdot(k+1)^2+2}{2\cdot(k+1)^2+3}\cdot n_M + 1,$$

which is in contradiction with $\mathcal{I}$ being sparse.

Algorithm 1 Transforming a sparse instance into an equivalent non-sparse one.

```
1:  function DE-SPARSIFY($M, \varphi$)
2:      while ($M = \langle [n], V\rangle, \varphi$) is sparse do
3:          let $[a,b]$ be a gap in $M$ and let $b' = a + 2 \cdot (k+1)^2 + 1$
4:          for each $p \in \mathcal{AP}$ do $V'(p) \leftarrow \emptyset$
5:          for each $p \in \mathcal{AP}$ and $[w,z] \in V(p)$ do
6:              if $z < a$ then
7:                  $V'(p) \leftarrow V'(p) \cup \{[w,z]\}$
8:              else if $w < a$ and $z > b$ then
9:                  $V'(p) \leftarrow V'(p) \cup \{[w, z-(b-b')]\}$
10:             else if $w > b$ then
11:                 $V'(p) \leftarrow V'(p) \cup \{[w-(b-b'), z-(b-b')]\}$
12:         $M \leftarrow \langle [n-(b-b')], V'\rangle$
13:     return ($M, \varphi$)
```

Algorithm 1 computes a non-sparse MC instance that is equivalent to the one given in input. To this end, it iteratively applies a suitable transformation $\tau_{[a,b]}$ (defined below) to each gap $[a,b]$, until a non-sparse MC instance is obtained. Such transformation produces the same result as multiple application of transformation τ_c. However, it is important to notice that executing $(b-a-2\cdot(k+1)^2-1)$ times the transformation τ_c for each gap $[a,b]$, instead of executing $\tau_{[a,b]}$ only once, would result in an algorithm whose execution time is exponential when the instance features exponentially large gaps.

Given an interval model M and a gap $[a,b]$ in it, the transformation $\tau_{[a,b]}(M)$ returns the pair $\langle [n'], V'\rangle$, where $n' = n - (b-a-2\cdot(k+1)^2-1)$ and $V' : \mathbb{I}([n']) \to 2^{\mathcal{AP}}$ is such that $V'([w,z])$ is equal to:

$$\begin{cases} V([w,z]) & \text{if } z < a \\ V([w, z+(b-a-2\cdot(k+1)^2-1)]) & \text{if } w < a \leq z \\ V([w+(b-a-2\cdot(k+1)^2-1), z+(b-a-2\cdot(k+1)^2-1)]) & \text{if } w \geq a. \end{cases}$$

It can be easily checked that the model $\tau_{[a,b]}(M)$ returned by one application of $\tau_{[a,b]}$ is equivalent to the one returned by $(b-a-2\cdot(k+1)^2-1)$ applications of the transformation τ_c, where c is the center of the gap $[a,b]$. Any such application of τ_c produces a model where (the current configuration of the interval) $[a,b]$ is shrunk into the interval $[a,b']$, where $b' = b-1$ (and thus $b'-a = b-a-1$). Hence, after $(b-a-2\cdot(k+1)^2-1)-1$ applications of τ_c, $[a,b]$ is reduced to the interval $[a,b']$, with $b' = b-(b-a-2\cdot(k+1)^2-2)$ and $b'-a = b-b+a+2\cdot(k+1)^2+2-a = 2\cdot(k+1)^2+2 > 2\cdot(k+1)^2+1$, meaning that $[a,b']$ is still a gap in the resulting model. By Lemma 1, the model returned by an application of τ_c is equivalent to the input model, and thus we have that $(M,\varphi) \equiv_{\mathsf{MC}} (\tau_{[a,b]}(M), \varphi)$.

Termination of the algorithm is guaranteed by the fact that $\tau_{[a,b]}$, applied to M, produces a model M' with a reduced number of gaps: a gap $[a,b]$ in M is shrunk into an interval $[a,b']$, which is not a gap in M', as $b'-a = 2\cdot(k+1)^2+1$.

As for the computational complexity, it is not difficult to check that the algorithm runs in polynomial time. Let N be the size of the input. The number of gaps is bounded by $n_M - u_M + 1$ ($n_M - u_M$ is the number of non-useless points—observe that there is at least one non-useless point between any two gaps), thus implying that the body of the outermost loop (line 2) is executed at most N times. Both innermost loops (lines 4 and 5) are clearly executed at most N times as well, giving an overall time complexity of $O(N^2)$. ◀

A non-sparse instance (M, φ) can be represented in space polynomial in n_M. To see that, it suffices to show that $n_M \leq p(N)$, for a polynomial p, where N is the size of the representation of (M, φ). First, we observe that

$$n_M = u_M + e_M$$

where e_M is the number of non-useless points, that is, those points that occur *explicitly* in the model. Since non-useless points are explicitly represented in the model, it clearly holds that $e_M \leq N$. By definition of (non-)sparse instance, we have that (recall that $k \leq 4 \cdot md(\varphi) \leq 4 \cdot N$):

$$\begin{aligned}
& u_M \leq \tfrac{2\cdot(k+1)^2+2}{2\cdot(k+1)^2+3} \cdot (u_M + e_M) + 1 \Leftrightarrow \\
\Leftrightarrow \quad & u_M - \tfrac{2\cdot(k+1)^2+2}{2\cdot(k+1)^2+3} \cdot u_M \leq \tfrac{2\cdot(k+1)^2+2}{2\cdot(k+1)^2+3} \cdot e_M + 1 \Leftrightarrow \\
\Leftrightarrow \quad & \tfrac{1}{2\cdot(k+1)^2+3} \cdot u_M \leq \tfrac{2\cdot(k+1)^2+2}{2\cdot(k+1)^2+3} \cdot e_M + 1 \Leftrightarrow \\
\Leftrightarrow \quad & u_M \leq (2 \cdot (k+1)^2 + 2) \cdot e_M + (2 \cdot (k+1)^2 + 3) \leq \\
& \quad \leq (2 \cdot (4 \cdot N + 1)^2 + 2) \cdot N + (2 \cdot (4 \cdot N + 1)^2 + 3) = \\
& \quad = 32 \cdot N^3 + 48 \cdot N^2 + 20 \cdot N + 5,
\end{aligned}$$

which means that

$$n_M = u_M + e_M \leq 32 \cdot N^3 + 48 \cdot N^2 + 21 \cdot N + 5.$$

Therefore the number $|\mathbb{I}([n_M])|$ of intervals in M is also bounded by a polynomial in N ($O(N^6)$), thus making it possible to adapt Emerson and Clarke's algorithm to obtain a polynomial model checking algorithm for non-sparse instances.

Algorithm 2 implements such an adaptation. Let us assume φ to be represented as a binary tree and M to be represented as in Figure 3. Moreover, for each sub-formula ψ of φ, let $L(\psi)$ be the set of all intervals of M where ψ holds. For every node of the tree representing φ (corresponding to a sub-formula ψ of φ), the algorithm computes the corresponding set of intervals $L(\psi)$. Initially, we set $L(\psi) = \emptyset$, for each sub-formula ψ of φ which is not a proposition letter, and we set $L(p) = V(p)$ for each proposition letter p. Modalities $\langle \overline{A} \rangle$, $\langle \overline{B} \rangle$, and $\langle \overline{E} \rangle$ are not dealt with by the algorithm as they are specular to the other ones.

▶ **Lemma 3.** *If $\mathcal{I} = (M, \varphi)$ is a non-sparse instance of* MC*, then Algorithm 2 returns* **true** *if and only if $M, [0, 1] \Vdash \varphi$. Moreover, it runs in polynomial time.*

Proof. It is immediate to see that Algorithm 2 is sound and complete. In order to show that it runs in polynomial time, we proceed as follows. Let N be the size of the representation of the input $\mathcal{I}$. Since $\mathcal{I}$ is non-sparse, the number of intervals in M is polynomial in N, thus providing a polynomial upper bound to the cardinality of $L(\psi)$, for each sub-formula ψ of φ.

The body of the loop at line 7 is executed at most N times (as $|\varphi| \leq N$). Whenever ψ is a Boolean formula, computing $L(\psi)$ takes a linear time in the number $|\mathbb{I}([n_M])|$ of intervals in M. The remaining cases can be efficiently implemented (in $O(N^6)$) by using a symbolic representation. As an example, in order to store the set of intervals on which $\langle A \rangle \tau$ holds, knowing that τ holds on an interval $[x, y]$, it suffices to store the number x, with the intended meaning that it represents all intervals ending at x. By suitably adapting the representation of $L(\psi)$, one is able to guarantee the complexity of these cases to be at most linear in the number of intervals as well. This allows us to conclude that Algorithm 2 runs in $O(N^7)$ time, and thus it is deterministic polynomial. ◀

Algorithm 2 Checking a non-sparse model.

```
function CHECK($M, \varphi$)
    for each $\psi$ sub-formula of $\varphi$ do
        if $\psi = p$ then
            $L(\psi) = V(p)$
        else
            $L(\psi) = \emptyset$
    for each $\psi$ sub-formula of $\varphi$ (ordered by increasing size) do
        if $\psi = \neg\tau$ then
            $L(\psi) = \mathbb{I}([n]) \setminus L(\tau)$
        else if $\psi = \tau \vee \xi$ then
            $L(\psi) = L(\tau) \cup L(\xi)$
        else if $\psi = \langle A\rangle\tau$ then
            for $[x,y] \in L(\tau)$ and for $z < x$ do
                $L(\psi) = L(\psi) \cup \{[z,x]\}$
        else if $\psi = \langle B\rangle\tau$ then
            for $[x,y] \in L(\tau)$ and for $z > y$ do
                $L(\psi) = L(\psi) \cup \{[x,z]\}$
        else if $\psi = \langle E\rangle\tau$ then
            for $[x,y] \in L(\tau)$ and for $z < x$ do
                $L(\psi) = L(\psi) \cup \{[z,y]\}$
    if $[0,1] \in L(\varphi)$ then
        return True
    else
        return False
```

▶ **Theorem 4.** *The HS model checking problem against a finite history can be solved by a deterministic algorithm that runs in polynomial time in the size of the input.*

We conclude the section with a short comparison with other classic model checking problems [39]. CTL model checking is quadratic (therefore, more efficient than ours), while LTL and CTL* model checking are PSPACE-complete (therefore, much less efficient than ours). The complexity of interval model checking over Kripke structures ranges from coNP-complete to non-elementary, depending on the particular fragment of HS under consideration [29]. However, as already pointed out, the most appropriate comparison is that with LTL model checking of a single (finite or ultimately periodic) path, which has been proved to be in PTIME [27].

5 Conclusions

In this paper, we formally defined the problem of temporal dataset evaluation, and we highlighted the role that model checking of finite interval structures plays in it. We also showed that the problem of temporal dataset evaluation has several applications, that range from temporal query answering to temporal constraint checking and rule evaluation, the last one being a key element in various machine learning processes.

We identified the model checking problem for the interval temporal logic HS over finite paths/histories as the main problem to be solved in this perspective, and we devised an efficient (deterministic polynomial) algorithm for it.

We are currently working on an implementation of the developed model checking procedure, which uses symbolic techniques to obtain better performances, and we plan to integrate such a procedure in an existing module for (temporal) rule extraction [19], based on an evolutionary algorithm, to compute a suitable *fitting function* of a set of temporal rules.

References

1 L. Aceto, D. Della Monica, A. Ingólfsdóttir, A. Montanari, and Guido Sciavicco. On the expressiveness of the interval logic of Allen's relations over finite and discrete linear orders. In *Proc. of the 14th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 8761 of *LNAI*, pages 267–281, 2014.

2 J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

3 P. Balbiani, V. Goranko, and G. Sciavicco. Two sorted point-interval temporal logic. In *Proc. of the 7th Workshop on Methods for Modalities (M4M)*, volume 278 of *ENTCS*, pages 31–45. Springer, 2011.

4 M. H. Böhlen, J. Gamper, and C. S. Jensen. How would you like to aggregate your temporal data? In *Proc. of the 13th International Symposium on Temporal Representation and Reasoning (TIME)*, pages 121–136. IEEE Computer Society, 2006.

5 A. Bottrighi, L. Giordano, G. Molino, S. Montani, P. Terenziani, and M. Torchio. Adopting model checking techniques for clinical guidelines verification. *Artificial Intelligence in Medicine*, 48(1):1–19, 2010.

6 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Interval temporal logic model checking: The border between good and bad HS fragments. In *Proc. of the 8th International Joint Conference on Automated Reasoning (IJCAR)*, volume 9706 of *LNCS*, pages 389–405. Springer, 2016.

7 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Interval vs. point temporal logic model checking: an expressiveness comparison. In *Proc. of the 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 65 of *LIPIcs*, pages 26:1–26:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.FSTTCS.2016.26.

8 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Model checking the logic of allen's relations meets and started-by is p^{np}-complete. In *Proc. of the 7th International Symposium on Games, Automata, Logics and Formal Verification (GandALF)*, volume 226 of *EPTCS*, pages 76–90, 2016.

9 A. Chaves, M. Vellasco, and R. Tanscheit. Fuzzy rules extraction from support vector machines for multi-class classification. *Neural Computing and Applications*, 22(7-8):1571–1580, 2013.

10 Jan Chomicki, David Toman, and Michael H. Böhlen. Querying atsql databases with temporal logic. *ACM Trans. Database Syst.*, 26(2):145–178, June 2001. doi:10.1145/383891.383892.

11 E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, 2002.

12 C. Combi and A. Montanari. Data models with multiple temporal dimensions: Completing the picture. In *Proc. of the 13th International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 2068 of *LNCS*, pages 187–202. Springer, 2001.

13 C. Combi and P. Sala. Mining approximate interval-based temporal dependencies. *Acta Informatica*, 53(6-8):547–585, 2016.

14 K. Deb. *Multi-objective optimization using evolutionary algorithms.* Wiley, London, UK, 2001.

15 E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, volume B: formal models and semantics, pages 995–1072. Elsevier MIT Press, 1990.

16 V. Goranko and M. Otto. Model theory of modal logic. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, pages 249–329. Elsevier, 2007.

17 J. Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962, 1991.

18 Y. Hayashi, S. Nakano, and S. Fujisawa. Use of the recursive-rule extraction algorithm with continuous attributes to improve diagnostic accuracy in thyroid disease. *Informatics in Medicine Unlocked*, 1:1–8, 2015.

19 F. Jiménez, G. Sánchez, and J. M. Juárez. Multi-objective evolutionary algorithms for fuzzy classification in survival prediction. *Artificial Intelligence in Medicine*, 60(3):197–219, 2014.

20 V. Khatri, S. Ram, R. T. Snodgrass, and P. Terenziani. Capturing telic/atelic temporal data semantics: Generalizing conventional conceptual models. *IEEE Trans. Knowl. Data Eng*, 26(3):528–548, 2014.

21 K. Kulkarni and J. E. Michels. Temporal features in SQL:2011. *ACM SIGMOD Record*, 41(3):34–43, 2012.

22 S. le Cessie and J. C. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

23 L. Liu and M. T. Özsu, editors. *Encyclopedia of Database Systems*. Springer NY, 2nd edition, 2017.

24 A. Lomuscio and J. Michaliszyn. An epistemic halpern-shoham logic. In *Proc. of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1010–1016, 2013.

25 A. Lomuscio and J. Michaliszyn. Decidability of model checking multi-agent systems against a class of EHS specifications. In *Proc. of the 21st European Conference on Artificial Intelligence (ECAI)*, pages 543–548, 2014.

26 A. Lomuscio and J. Michaliszyn. Model checking multi-agent systems against epistemic HS specifications with regular expressions. In *Proc. of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 298–308. AAAI Press, 2016.

27 N. Markey and P. Schnoebelen. Model checking a path. In *Proc. of the 14th International Conference on Concurrency Theory (CONCUR)*, pages 251–265. Springer, 2003.

28 M. Mashayekhi and R. Gras. Rule extraction from random forest: the RF+HC methods. In *Proc. of the 28th Canadian Conference on Advances in Artificial Intelligence*, pages 223–237, 2015.

29 A. Molinari, A. Montanari, A. Murano, G. Perelli, and A. Peron. Checking interval properties of computations. *Acta Informatica*, 53(6-8):587–619, 2016.

30 A. Molinari, A. Montanari, and A. Peron. Complexity of ITL model checking: Some well-behaved fragments of the interval logic HS. In *Proc. of the 22nd International Symposium on Temporal Representation and Reasoning (TIME)*, pages 90–100. IEEE Computer Society, 2015.

31 A. Molinari, A. Montanari, and A. Peron. A model checking procedure for interval temporal logics based on track representatives. In *Proc. of the 24th EACSL Annual Conference on Computer Science Logic (CSL)*, volume 41 of *LIPIcs*, pages 193–210, 2015.

32 A. Molinari, A. Montanari, A. Peron, and P. Sala. Model checking well-behaved fragments of HS: the (almost) final picture. In *Proc. of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 473–483. AAAI Press, 2016.

33 A. Montanari, A. Murano, G. Perelli, and A. Peron. Checking interval properties of computations. In *Proc. of the 21st International Symposium on Temporal Representation and Reasoning (TIME)*, pages 59–68. IEEE Computer Society, 2014.

34 A. Pnueli. The temporal logic of programs. In *Proc. of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, 1977.

35 A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13(1):45–60, 1981.

36 D. M. W. Powers. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

37 J. R. Quinlan. *C45: Programs for Machine Learning.* Morgan Kaufmann, 1992.

38 E. Quintarelli. *Model-Checking Based Data Retrieval, An Application to Semistructured and Temporal Data*, volume 2917 of *LNCS*. Springer, 2004.

39 P. Schnoebelen. The complexity of temporal logic model checking. In *Proc. of the 4th Conference on Advances in Modal Logic*, pages 393–436, 2002.

40 R. T. Snodgrass, editor. *The TSQL2 Temporal Query Language.* Kluwer Academic Publishers, 1995.

41 R. T. Snodgrass, I. Ahn, G. Ariav, D.S. Batory, J. Clifford, C. E. Dyreson, R. Elmasri, F. Grandi, C. S. Jensen, W. Käfer, N. Kline, K. Kulkarni, T. Y. C. Leung, N. Lorentzos, J. F. Roddick, A. Segev, M. D. Soo, and S. M. Sripada. TSQL2 language specification. *SIGMOD Record*, 23(1):65–86, 1994.

42 A. U. Tansel, J. Clifford, S. K. Gadia, S. Jajodia, A. Segev, and R. T. Snodgrass, editors. *Temporal Databases: Theory, Design, and Implementation.* Benjamin/Cummings, 1993.

43 M. Y. Vardi. Model checking for database theoreticians. In *Proc. of the 10th International Conference on Database Theory (ICDT)*, volume 3363 of *LNCS*, pages 1–16. Springer, 2005.

44 I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann Publishers Inc., 3rd edition, 2011.

Time Expressions Recognition with Word Vectors and Neural Networks

Mathias Etcheverry[1] **and Dina Wonsever**[2]

1 **Universidad de la República, Facultad de Ingeniería, Instituto de Computación, Montevideo, Uruguay**
`mathiase@fing.edu.uy`

2 **Universidad de la República, Facultad de Ingeniería, Instituto de Computación, Montevideo, Uruguay**
`wonsever@fing.edu.uy`

Abstract

This work re-examines the widely addressed problem of the recognition and interpretation of time expressions, and suggests an approach based on distributed representations and artificial neural networks. Artificial neural networks allow us to build highly generic models, but the large variety of hyperparameters makes it difficult to determine the best configuration. In this work we study the behavior of different models by varying the number of layers, sizes and normalization techniques. We also analyze the behavior of distributed representations in the temporal domain, where we find interesting properties regarding order and granularity. The experiments were conducted mainly for Spanish, although this does not affect the approach, given its generic nature. This work aims to be a starting point towards processing temporality in texts via word vectors and neural networks, without the need of any kind of feature engineering.

1998 ACM Subject Classification I.2.7 Natural Language Processing

Keywords and phrases Natural Language Processing, Time Expressions, Word Embeddings, Neural Networks

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.12

1 Introduction

Detecting and interpreting the linguistic expressions that we use to refer to the physical time we live in (e.g. "21 September", "2001" or "yesterday") poses an interesting problem of natural language processing. Time expressions or timexes are a sub-language made up of specific lexicon and with an interpretation linked to the calendar system that we use, numbers, conditions and relations with entities that are external to the expression. Additionally, as many other NLP tasks, the analysis of timexes presents ambiguous situations where contextual information is needed.

Several approaches have been proposed to detect and interpret time expressions. Rule-based systems like HeidelTime [35] and SUTime [9] yield very good results. Also systems based on machine learning techniques like Support Vector Machines [3] or Conditional Random Fields [1] produce good results without the cost of defining the rules, though it is necessary to define specific characteristics and to use external resources.

Furthermore, vector representations of words, obtained from large text collections, have produced interesting results regarding association and analogies between words [25]. Word vectors can be seen as a set of artificial micro features that may be part of the input of a machine-learning algorithm.

Artificial neural networks organized in layers can be interpreted as successive sequent transformations of a representation to obtain an expected result. Using word vectors as input enables us to resolve problems without having to specify rules or machine-learning features with information about the problem. The information about the problem used to define the models considered in this work is limited to the output layer and the way the model is used.

In this work we study the behavior of neural models that use as input distributed representations of words to detect and classify the time expressions occurring in a text. Word representations make associations between similar words or semantically related words, which gives the model the capacity to consider cases with lexical entries that do not exist in the training data. We consider feedforward and LSTM models, and analyze the effect of considering different dimensions for word representations, for hidden layers and normalization techniques such as noise in representations, dropout, L1 and L2. We conducted experiments using a small corpus for Spanish obtaining interesting results. We present results relative to word vectors that suggest the possibility of interpreting time expressions without defining rules or adding knowledge about the temporal domain. Finally, we include experiments in English with encouraging results, and in the different, yet related, task of events detection for Spanish and English.

2 Related Work

2.1 In Time Expressions

Extensive work has been done in the detection and interpretation of time expressions. Furthermore, neural networks and vector representations of words have made great progress recently. However, to our knowledge, no studies using neural models to resolve problems with time expressions have been presented.

Regarding existing rule-based systems, [24] presents a system that resolves recognition and interpretation through manually developed and machine-learned rules. [28] tackles the problem by processing the input text, where information about expressions is cumulatively added in each stage through heuristics and rules. [16] resolves recognition and interpretation with a formal set of rules on the morphosyntactic information of the input. [13] builds a system based on the *TRIOS* system [38], adding pre and post processing stages to improve their results.

The *HeidelTime* system [34] uses regular expressions and resources from a temporal lexicon to recognize and interpret the expressions, reaching the best results at *TempEval-2*. These results were later improved by the *SUTime* system [9]. At *TempEval-3*, the system that obtained the best results was a new version of *HeidelTime* [35]. [8] outperforms this result with a system based on a manually developed context-free grammar.

Furthermore, traditional machine learning methods are essentially classifiers based on conveniently defined features. It is from annotated examples that mechanisms are set to determine the desired information in arbitrary entries. This type of method is suitable to identify and classify time expressions, but its application to interpretation is not direct.

These systems are usually based on features such as: words part-of-speech tags in a context window, belonging to word classes that are manually specified, and restrictions on a dependency or constituents analysis, among many others. The variety of possible features is unlimited, and the results depend mainly on features engineering. It is also important to note that some features may require considerable computing time, thus making their use questionable.

We could mention many existing machine learning based systems. [1] detects time expressions with *Conditional Random Fields (CRF)*, as does [2]. Later on, [3] uses *Support*

Vector Machines (SVM) as classifiers and simplifies the rules for interpretation through a classifiers cascade.

Semi-supervised approaches may include *bootstrapping* techniques to improve recognition [27]. [21] expands positive cases using *WordNet*. The *ManTIME* system [14] runs *CRF* for detection, considering attributes derived from *WordNet*, but does not reach significant improvements.

The*ClearTK-TimeML* system [7] trains multiple supervised classifiers to identify and classify time expressions, events and relations. Different methods are made to compete in the system (*CRF*, *SVM* and logistic regression), with a specific tune of hyperparameters.

Hybrids that include machine learning techniques and a formal set of rules have produced good recognition results. As for interpretation, rules are used relatively naturally given their compositional properties. Some approaches combine the advantages of formal sets of rules and annotated examples to interpret expressions. [4] inferred a probabilistic context-free grammar on the expressions. This system can be easily applied to different languages [5].

[22] uses combinatory categorial grammar system to detect and interpret time expressions. The work considers 287 manually designed entries, as well as automatically generated entries (such as numbers and formats of dates), obtaining 83.1% F-score for detection, 85.4% for classification and 82.4% for interpretation in the evaluation data of *TempEval-3*; this is the current state-of-the-art.

2.2 In Neural Networks

Artificial neural networks currently play a major role in the artificial intelligence community and natural language processing isn't an exception[10, 11]. This has increased with the progress made in the construction of vector representations of words from the contexts where they occur [25, 26].

As for the use of word representations with neural network models to solve NLP tasks, many studies with significants results can be mentioned. Among them, paraphrase detection [32], parsing [31] and sentiment analysis [33]. [18] uses recurrent models and word embeddings to resolve the issue of opinion mining. This work shares ideas with [17] that uses a feedforward network and word2vec embeddings for time expression recognition in English for clinical domain.

3 Time Related Words and Distributed Representations

To study the quality of information provided by words vector representations in the temporal domain we reduced the dimension of the representations to be able to represent their relative positions graphically. In turn, the ordering of the cosine distances respect to a specific word were considered. We used word representations inferred from Wikipedia in Spanish through GloVe [26], presented by [12].

3.1 Clustering

It is relevant to study the structure of the space of vector representations for the words that refer to temporal information. Figure 1 shows the representations of a selection of time related words after reduction to 2 dimensions with *t-sne* [40] for its graphical representation. The presented 2-d representation shows how semantically related words tend to form clusters. The following clusters are formed: days of the week, months, years, adverbs and low numbers

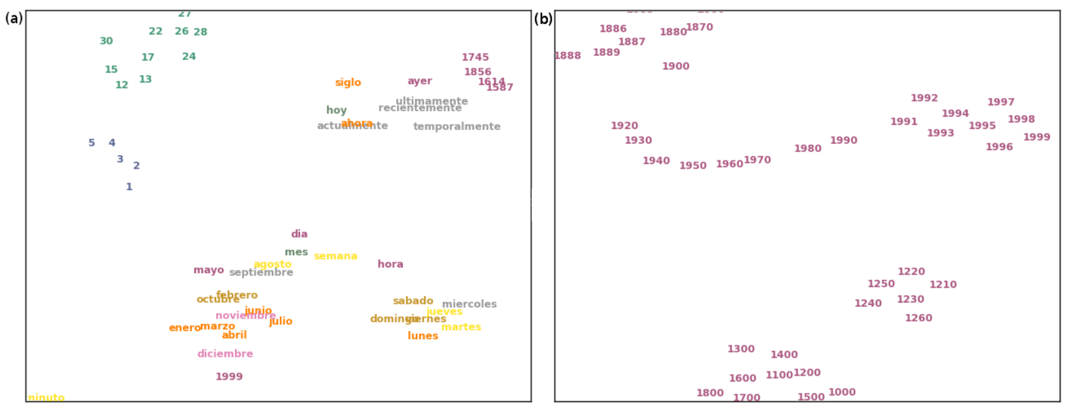


Figure 1 (a) Representation in 2 dimensions using *t-sne* of 200 dimension representations of a time related words selection. (b) Representation in 2 dimensions using *t-sne* of 200 dimension representations of numbers.

Table 1 Table with the closest terms (to the heading) ordered by distance to the time expressions representations.[1]

Amanecer	Neolítico	Comienzo	Antes	Repentinamente	Apresuradamente
atardecer	paleolítico	inicio	después	súbitamente	marchar
mañana	mesolítico	dio	tras	muere	replegarse
noche	calcolítico	antes	ya	falleció	precipitadamente
día	neolítico	final	días	murió	desecaba
medianoche	datan	dando	luego	prematuramente	mudarse
anochece	pleistoceno	llegada	ese	trágicamente	periódicamente
mediodía	precerámico	finales	meses	tempranamente	dirigiera
ocaso	epipaleolítico	principio	tiempo	...	...
madrugada	bronce	momento	comenzar		
...	...	...	...		

used in days of the month. In tasks related to time expressions, this enables the generalization to cases that are not included in the training corpus.

Besides the days of the week and the months of the year, it is interesting to observe, for example, the behavior of the words denoting times of day (e.g. sunrise), prehistoric times (e.g. Neolithic) or time adverbs. Table 1 shows the closest words to a given term ordered by distance.

3.2 Ordering and Granularity

In the examples presented, besides the formation of clusters we notice that the words that follow a sequential order (common in time domain), such as the days of the week, months, etc., tend to hold the order in terms of the representations. For instance, the representation of the word *miércoles* (Wednesday) occurs closer to that of *martes* (Tuesday) and of *jueves* (Thursday) than to other week day names. This shows that representations tend to preserve the chronological order of the terms.

[1] There is an English translated version in the Appendix B, Table 22.

Table 2 Table with the closest terms (to the heading) ordered by distance to the representations of ordinal and numerical terms.[2]

Primero	**Segundo**	**Vigésimo**	**1853**	**1850**	**1700**	**1999**
luego	tercer	trigésimo	1855	1840	1600	1998
segundo	primer	décimo	1854	1849	1800	1995
mismo	cuarto	cuadragésimo	1856	1870	1500	1997
último	último	noveno	1852	1860	1400	1996
primer	quinto	quincuagésimo	1851	1880	1200	2002
posteriormente	primero	octavo	1865	1851	1100	2003
después	tercero	quinto	1849	1830	1300	1994
...	...	...	...	...	...	...

Besides this tendency to preserve the order, ordinal terms and numbers present a granularity issue. When considering terms such as *primero* (first), *segundo* (second) and *tercero* (third), the term primero is close to terms like *segundo* and also to *último* (last). Then, when considering the terms close to *segundo*, terms like *tercero* (third), *cuarto* (fourth) and *quinto* (fifth) prevail. Additionally, next to the term *vigésimo* (twentieth) we find terms like *désimo* (tenth) and *trigésimo* (thirtieth) (see Table 2).

Similarly, granularity is considered in the vector representation of numerical terms. For instance, if the number is in hundreds granularity, for example 1700, other numbers from the hundreds like 1600 or 1800 are found close. Note as well that these terms are sequentially the previous and next ones to 1700 in the hundreds granularity. Something similar happens with numbers from the tens such as 1850, with close numbers like 1840, 1860 and 1870; and if 1853 is considered, its representation is close to that of 1855 and 1854.

Figure 1(b) shows the representations of a selection of numerical terms after reduction to 2 dimensions with t-sne. We can see clusters that reflect the granularity and order properties. It is also interesting to note how the sequence 1920, 1930, ..., 1970, 1980, 1990 connects the cluster close to 1900 (1888, 1889, ...) with the one close to 1990 (1991, 1992, ...). These order and granularity phenomena of the terms are potentially useful to interpret time expressions. Note that these properties are inferred only from the local contexts where words occur.

3.3 Regression on Years vectors

Experiments were conducted on neural network models trained to infer their respective numerical value from the vector representation of a numerical term. A sampling of 300 random numbers was considered in the range from 1000 to 2000 with their respective vector representations. The set was used to train a regression feedforward network with one 100-size hidden layer, and the lowest absolute error was used as target function for training. Though deeper studies are needed, the results obtained are encouraging. The experiments include results such as 1985.21; 1986.84; 1986.72; 1988.58 and 1989.02 for the 1985 – 1989 sequence, where only 1989 was in the training data. However, not all the results were as accurated, although, they were close to the expected value.

[2] There is an English translated version in the Appendix B, Table 23.

Table 3 Labeling scheme used to classify time expressions.

Type	**Label**
date (da)	L_{da}, U_{da}
time (t)	L_t, U_t
duration (du)	L_{du}, U_{du}
set (s)	L_s, U_s
other	B, I, O

4 Timex Detection and Classification

Detecting time expressions involves identifying the expression and indicating its extension. The classification of time expressions means to determine the type of the expressions identified using a predefined set of types.

With a focus on the TimeML annotation system of time expressions and events [29], the *date* type was considered for dates (e.g. "three years before"), *time* for the times of day (e.g. "at 3 pm"), *duration* for durations (e.g. "5 minutes") and *set* for the expressions that represent frequencies and sets (e.g. "every Monday").

Detection and classification are resolved simultaneously by formulating the problem as the labeling of the words in the text. Each word is assigned a label indicating if a time expression applies and which type of expression it is. The labels from the *BILOU* scheme were used, as it has shown better results than the *BIO* scheme for the extraction of named entities [30]. The original scheme determines for each word if it is the beginning (B), the inside (I) or the last (L) token of an expression; if it is a single word expression (U) or a word that does not belong to any expression (O).

To consider classification we distinguish the last word in the expression adding an $\mathbf{L}_i$ and $\mathbf{U}_i$ label for each type, the other labels remain unchanged (see Table 3).

4.1 Model

To resolve the detection and classification of time expressions we consider models organized in layers to label words according to the *BILOU* variant presented. The model is applied sequentially to each word giving the word label on each activation.

The input layer receives word vectors, the information is transformed along the hidden layers to the output layer, where the *softmax* function is used. We consider *feedforward* and recurrent models with different numbers and size of hidden layers and regularization techniques.

We include local contextual information concatenating vector representation of fixed size window to both sides from the word to label (window context).

For example, if a size 2 left window context and 3 size right context is considered, the input received by the neural network would be $x = [w_{i-2}w_{i-1}w_iw_{i+1}w_{i+2}w_{i+3}]$; the concatenation vector of the word representations in the input text, where w_i is the representation of the word to classify. The disadvantage of this type of context is that it has a rigid size regardless of the case presented to the model. This might be inappropriate for selective context considerations or to capture long-range dependencies.

Table 4 Information about time expressions in the used corpus.

Name	Words	Date	Duration	Time	Set	Total
		Expresiones Temporales				
TEval13_es_train	46.687	585	215	49	29	878
TEval13_es_test	12.197	164	36	8	8	216

4.2 Corpus and Training

Training and evaluations were conducted with the data for Spanish included in the TempEval-3 task [39]. As the evaluation data is not available, the training data was divided into a training set and an evaluation set. The training set has **878** time expressions, and the evaluation set has **216** of them. Table 4 shows information from the corpora used.

The limited size of the corpus is a setback for supervised learning approaches in contrast to those that include rule-based knowledge. However, it is a good scene to test the unsupervised word vector representations as a generalization tool when there is limited data. A further disadvantage is that the reduced size of the evaluation set affects the quality of the evaluation and reduces the impact of small variations on the experiment results.

As for training, all the models considered were trained with *RMSprop* [37], a variant of backpropagation. In all cases we used a learning rate of 1×10^{-4} and a moment value of 0.9. The stopping criterion is that the improvement of the target function does not exceed 1×10^{-5} per 30 epochs. The experiments were conducted with *Theanets* package [19] implemented over *Theano* [36].

The evaluation is made through precision and recall at the expression level of the output of each model against the evaluation data. The global evaluation measure is taken through the F-score with the same balance for both.

4.3 Experiments

There follow the experiments conducted and the results obtained. The experiments focus on the dimensions of words and internal representations, number of layers, regularization techniques and variations on local context.

4.3.1 Dimension

[25] and [26] show that by increasing the dimension, vector quality improves. This tendency seems not to be unlimited, although there are no known results. Furthermore, compact vectors have desirable characteristics and benefits regarding computational cost.

To observe the impact of dimension empirically on the detection task we considered several dimensions representations under similar conditions. Models were trained avoiding to alter the rest of the environment.

The models used were three-layer feedforward models with three words of symmetric context. To preserve the proportion between layers, we adjusted the size of the hidden layer to three times the word dimension. We observed a steady increase in the recall, but the precision began to decrease. The results are shown in Table 5.

We observed the behavior that takes place when halving the size of the models considered for dimensions 150 and 200. In both cases we observed a slight improvement in the results, mainly contributed by the increase in precision, thus preserving global results. The results are shown in Table 6.

Table 5 Comparison of detection results in similar environments varying the dimension or the word representations using feedforward models with one hidden layer and three words (to each side) of symmetric context were used.

Dim	Train Acc	Prec	Rec	F
25	1x10-6	72.97	50.00	59.34
50	1x10-6	76.14	62.04	68.37
100	396x10-4	**82.35**	64.81	72.54
150	396x10-4	80.79	66.20	72.77
200	396x10-4	78.61	**68.06**	**72.95**

Table 6 Detection and classification results when halving the hidden layer dimensions in the most significant models in Table 5. We present between brackets the classification results.

Dim	Hid	Prec	Rec	F
150	450	80.79 (77.46)	66.20 (62.04)	72.77 (68.89)
	225	**82.28** (75.00)	66.67 (62.50)	**73.66** (68.18)
200	600	78.61 (77.97)	68.06 (**63.89**)	72.95 (**70.22**)
	300	79.14 (**78.61**)	**68.52** (62.96)	73.45 (69.92)

4.3.2 Detection with/without Classification

As the models that resolve the classification of expressions also conduct detection tasks, it would be interesting to know how classification affects detection. Table 7 shows the comparison of the detection results between models with output layers for detection and the same model but changing the output layer for classification. In general, detection precision was improved in the models that also classify the expression. This might be due to the fact that the classification information is used in detection.

As for lexical generalization to cases that are not included in the training data, we observed at least one positive case with the word *semestre* (semester). This word does not occur in the training data, but it does appear twice in the test data. A few of the models can detect at least one of the two occurrences. This might be the case because the model was able to generalize from words like *trimestre* (trimester) and *cuatrimestre* (four-month period) that do occur in the training data.

4.3.3 Hidden Layer Size

Hidden layers are a fundamental part of the model that build intermediate representations to resolve the task. The sizes of hidden layers correspond to the dimensions of the spaces of intermediate representations. Hidden layers that are too small might hinder the right resolution of the task, while a very large size is more inefficient and might lead the model to overfit the training data.

We previously observed that the size of hidden layers (although up to now only models with one hidden layer have been considered) has an impact on results. To observe the effect of considering different sizes in the hidden layer, we trained a sequence of one-layer models to classify expressions varying its size (Table 8).

As for the number of hidden layers, we conducted a initial experiment where an additional 100-size layer before output was considered for a 300-size layered model. The results mainly improved the recall, with 79.67 precision and 67.13 recall, which entails an F-score of 72.86.

Table 7 Comparison of results in detection in models trained for classification with the same model but with detection output layer (between brackets).

Dim	Hid	Prec	Rec	F	
150	450	82.66 (80.79)	66.20 (66.20)	73.52 (72.77)	+0.75
	225	80.00 (82.28)	66.67 (66.67)	72.73 (73.66)	-0.93
200	600	83.05 (78.61)	**68.06 (68.06)**	**74.81 (72.95)**	+1.86
	300	**83.81 (79.14)**	67.13 (68.52)	74.55 (73.45)	+1.10

Table 8 Results in the classification of expressions with feedforward models on 200-dimension words, with three words of symmetric context, varying the size of the unique hidden layer.

Hid	P	R	F
100	74.58	61.11	67.18
200	75.28	62.04	68.02
300	78.61	62.96	69.92
400	77.27	62.96	69.39
500	76.95	63.42	69.54
600	79.21	**65.27**	**71.57**
700	**79.31**	63.89	70.77
1000	76.40	62.96	69.04
2000	79.19	63.43	70.44

This looks promising in the deeper consideration of models. We will come back to this point below.

4.3.4 Window Context Size

The local context is extremely important for time expressions. We tested various context window configurations. Models with isolated left and right contexts were considered, and also models with symmetric contexts. The model that served as base had a hidden layer whose size is defined according to context length. The inclusion of any context produced better results in all cases than the context-less version, however, large contexts can negatively affect the results. Table 9 shows the results.

As expected, considering the symmetric context yielded far better results than considering only the left or right context. Although the best global result was achieved with two context words (two left and two right), the best recall result was reached with three words whose F-score is also next to the maximum result. The results also show that the right context seems to provide more information than the left one for this task.

4.3.5 Regularizations

Regularization techniques often can improve the way the neural network model is trained giving as result a best generalizatión of unseen cases. We consider input and hidden noise, dropout, l1 and l2 regularizations. We try different values for each regularization technique considered and here we present the conclusions obtained. The details of the results obtained are in Appendix A.

We try input and hidden noise with positive results mainly in the first. Input noise improves substantially the recall (in comparison with the same model without any noise)

Table 9 Comparison of classification results when increasing the left context without considering the right (left value), left (center) and symmtric context (right value). The base model is feedforward of 200 word dimension, and according calculated hidden size.

Context	Hid	Prec	Rec	F
0	100	60.64	26.39	36.77
1	100	67.31 / **64.97** / 72.78	32.41 / 47.22 / 56.94	43.75 / 54.69 / 63.90
2	200	**69.83** / 63.64 / **80.84**	**37.50** / 48.61 / 62.50	**48.79** / 55.12 / **70.50**
3	300	67.00 / 62.42 / 76.92	31.02 / 45.37 / **64.81**	42.40 / 52.55 / 70.35
4	400	69.81 / 62.42 / 76.86	34.26 / 43.06 / 43.06	45.96 / 50.96 / 55.19

Table 10 Results of the classification of expressions using BLSTM models based on 200-dimension words with no window context an one single hidden layer.

Hid	Steps	P	R	F
150	100	54.20	32.87	40.92
200	100	60.75	30.09	40.25
600	100	63.16	**33.33**	**43.64**
300	15	64.36	30.09	41.25
600	15	64.44	27.31	38.43
1000	15	**71.44**	30.56	42.85
2000	15	71.25	26.39	38.51
600	3	64.00	14.09	24.06

and minor improvement in precision. The grade of input noise that gives the best results was 0.2. Respect to hidden noise, a much lesser improvement was detected with its best configuration in 0.05.

Dropout also improved precision and recall if a very low value was considered, the value that gives the best results was 0.01. Respect l1 (sparsity) and l2 (weight decay) a minor improvement was noticed in both cases. In the case of l2, the results fluctuate and for that reason is hard to determine which configuration is better. For l1, the best results were obtained with 0.01 affecting positively to the recall of the model.

4.3.6 Recurrent Networks

A non-exclusive alternative to the window context is the context considered by recurrent models. Previously established activations are considered through feedback in the hidden layer, allowing the sequential application of the network to consider the context that corresponds to the inputs previously applied.

Context considerations of the recurrent models are more flexible than the explicit information provided by the window context. However, its interpretation is more complex, and the experiments conducted did not produce good results without the additional consideration of the window context.

As the results improve significantly when considering both left and right contexts, bidirectional models are considered, especially *Bidirectional Long-Short Term Memory* (BLSTM) [15].

In the experiments conducted with BLSTMs, the results obtained were lower than those of the feedforward models with a window context. Table 10 shows the results. Different sizes for the recurrent layer and depths of recurrence were considered.

Table 11 Results of the classification of expressions using feedforward models on 200-dimension words with 3 words of symmetric window context varying hidden layers number and sizes.

h1	h2	h3	h4	P	R	F
300	–	–	–	78.61	62.96	69.92
300	100	–	–	**79.67**	**67.13**	**72.86**
300	200	100	–	74.07	64.81	69.13
600	400	200	100	64.25	57.41	60.64

Table 12 Results of the classification of expressions using BLSTM models on 200-dimension words with no window context, varying hidden (recurrent) layers number and sizes.

Hid1	Hid2	Hid3	Prec	Rec	F
200	–	–	60.75	30.09	40.25
300	150	–	**68.42**	**54.17**	**60.46**
300	200	100	61.15	45.83	52.52

4.3.7 Network Depth

The number of hidden layers is a key aspect when defining layered neural models. It is known that the training of networks with several hidden layers presents difficulties. We experiment with models up to four hidden layers empirically.

Different depths were considered in homogeneous models, that is, with all layers of the same type. The first experiment consisted in adding an extra hidden layer between the existing hidden layer and the output layer to the model previously used as a base. The inclusion of the additional layer substantially improved the results, especially regarding recall (see Table 11).

Upon observing the favorable effect of considering a model with two hidden layers (versus the single hidden layer model), we trained and assessed models with three and four hidden layers. In this case, results show that considering more than two hidden layers the model was not adequately trained.

Regarding recurrent models, even though their results were far lower than those of the feedforward models with a context window, we studied the effect of considering more hidden layers, also formed by BLSTMs. As in the previous case, considering one additional layer resulted in a considerable improvement and the F measure decreased when more than two hidden layers were considered (Table 12). These models were trained considering 100 recurrence steps.

4.3.8 Combining Variations

The results yielded by different variations of neural models, including feedforward and recurrent BLSTM models, were shown above. We tested different structural configurations and regularization techniques.

A general observation regarding the experiments conducted is that the random initialization of the weights of the model can cause different instances to produce different results. To reduce the impact of this situation, we repeated some of the experiments in questionable situations and others randomly, and we found no high differences between the different instances of the same experiment in most cases.

Regarding the structural considerations of the model, considering two hidden layers, instead of one, had a positive effect. This behavior was not sustained for greater depths.

Table 13 Results of the classification of best models presented before (top), combinations of which show positive effects independently (middle) and good combinations with an improved word vectors set.

h1	h2	noise$_I$	dropout	L1	L2	P	R	F
300	–	–	–	–	–	78.61	62.96	69.92
600	–	–	–	–	–	79.21	65.27	71.57
300	100	–	–	–	–	79.67	67.13	72.86
300	–	**0.20**	–	–	–	**80.66**	**67.59**	**73.55**
300	–	–	0.01	–	–	80.57	65.28	72.12
300	–	–	–	0.01	–	78.89	65.74	71.72
300	–	–	–	–	0.001	80.11	65.28	71.94
300	–	0.20	–	–	–	80.66	67.59	73.55
700	–	0.2	–	–	–	81.03	65.28	72.31
700	400	0.2	–	–	–	**81.36**	66.67	73.28
400	150	0.2	0.1	–	–	80.35	64.35	71.46
350	120	0.1	0.1	–	–	80.32	**68.06**	**73.68**
600	–	0.1	–	–	–	81.21	**68.06**	**74.05**
450	200	0.1	–	–	–	**81.92**	67.13	73.79

The size of the hidden layers fluctuated for sizes over 300, gradually decreasing for lower values. Input and hidden layer noise, dropout, L1 and L2 were considered as regularization techniques. The most significant improvement was observed with the consideration of noise in the input. Table 13 shows the best results obtained for each family of experiments.

In order to see how different techniques which yielded good results perform when combined, we considered experiments with two hidden layers, input noise and dropout simultaneously. Middle of Table 13 shows the results and bottom of the table includes results of models trained using 300-dimension vectors provided by [6].

4.4 Comparison with SVM

We compare the obtained results with Support Vector Machines (SVM) in order to empirically validate the advantage of considering neural models versus other alternatives. We consider SVM classifier with a word dimension of 200 and 3 words with a local context in both directions yelding 60.8 of F measure in time expressions recognition and classification. This result indicates, at least initially, that neural models make a better use of word embeddings and local context to resolve this task.

4.5 Comparison with Other Works

It is not possible to compare this to other works because we have not been able to access the same evaluation data. Nevertheless, we think it is useful to at least include some comparative values. As we explained before, the training set was split to have an evaluation set. In essence, the model was trained with a part of the training set and evaluated with an evaluation set of similar characteristics, but which was different.

The comparison is made based on the model that produced the best results in detecting time expressions. The model considered has a single 300-size hidden layer, with 3 words with symmetric contexts, a 200-dimension word and a variance value of 0.2 for the input noise.

Table 14 Results of the classification for Spanish in the tempeval-3 task. ANNTime refers to the best results obtained in this work. ANNTime* refers to the same model applying heuristics to rectify inconsistent labels. ANNTime-nabu and ANNTime-nabu* correspond to the best models using the vectors of [6].

	P(r)	R(r)	F_1(r)	F_1(s)
HeidelTime	**96.0**	**84.9**	**90.1**	**85.3**
TIPSemB-F	93.7	81.9	87.4	82.6
FSS-TimEx	86.6	52.3	65.2	49.5
ANNTime	92.8	77.8	84.6	78.6
ANNTime*	91.2	81.5	86.1	79.2
ANNTime-nabu	91.7	76.8	83.6	79.6
ANNTime-nabu*	91.4	83.3	87.2	79.9

Table 15 Information of the TempEval 2013 corpus for English.

Name	Words	Date	Duration	Time	Set	Total
TEval13_en_silver	718.746	11.133	1.346	192	68	12.739
TEval13_en_platinum	7.003	96	34	4	4	138

Table 14 shows the comparison of the results. We also included the results obtained with the model that produced the best results using the vectors of [6].

For each model we include the result of applying the model and, subsequently, a basic heuristics to correct inconsistent labels (this is marked with an asterisk in Table 14). Heuristics consists of correcting an incorrect label in three situations. First, if an O is followed by an I, the I is replaced by B or U_{da} accordingly[3]. Second, if an O is preceded by a B or I and followed by an I or L, said label is replaced by I. Third and last, if an I is followed by an O, said I is replaced by L_{da} o U_{da} accordingly. The application of heuristics improved results by almost 4% regarding the recall of the model, and reduced precision by 1.6%, thus resulting in a global improvement (F) of 1.5%, in the case of overlapping for basic vectors and a 3.6% improvement for 300-dimension vectors.

4.6 Time expressions in English

The results using same neural models, but applied to resolve the task in English, are shown, taking into account the lessons learned throughout the experiments conducted for Spanish. We use the data conveyed for English in TempEval 2013 (see Table 15). The representations presented by [26] of 200 dimensions, built with a six billion word corpus, are used.

The approach used is applied directly because it does not include specific knowledge of the language or domain. We trained a feedforward model with three words with symmetric contexts, based on the 200-dimension representations, with a single, 300-size hidden layer and a noise level of 0.2 in the input, which was the model that produced the best results in detection for Spanish. We also consider two hidden layers models detecting faster convergence. Table 16 shows the results obtained.

Regarding the classification of expressions, in all cases models showed an F-score for classification lower than the one in the work of [22], which is to be expected because this

[3] The date type was used because it is the most frequent.

Table 16 Results of the detection and classification of expressions for English.

h1	**h2**	**noise$_I$**	**P$_{det}$**	**R$_{det}$**	**F$_{det}$**	**Acc$_{class}$**	**F$_{class}$**
300	–	0.20	93.88	66.67	77.97	93.47	72.88
300	100	–	95.83	66.67	78.63	**95.65**	75.21
900	200	–	**96.80**	65.94	78.45	95.60	75.00
600	100	0.20	95.88	**67.39**	**79.15**	93.54	74.04
[22]			86.1	80.4	83.1	93.4	**85.4**

Table 17 Top: Results of events detection model in Spanish with and without time expressions detection. Middle: Results of events detection model in English with time expressions detection. Bottom: Time expressions detection in English including events detection.

	P	**R**	**F**
Events (without timexes) (600x200-0.2) (SP)	81.16	79.40	80.27
Events (with timexes) (600x200-0.2) (SP)	84.31	79.08	81.61
CRF+Morph+SRL+WNet [23]	83.43	79.54	81.40
Events (with timexes) (600x200-0.2) (EN)	79.31	79.62	79.46
ATT1(MaxEnt+Syn+Sem) [20]	81.44	80.67	81.05
Timexes (with Events) (600x200-0.2) (EN)	98.99	71.01	82.70
Semantic Parsing [22]	86.1	80.4	83.1

value is influenced by detection. Regarding accuracy, we obtained better results in all cases[4], but it should be noted that this value is positively affected by the lower result in detection.

5 Events Detection

The model presented only uses supervised training data and word representations built through self-supervised methods. This means that the approach is rapidly adaptable to other languages, as it was observed for English. Similarly, it can be considered to deal with tasks that have a compatible formulation, for instance, events detection. The main difference is that, as opposed to events, time expressions often include words such as the prepositions "in" and "during", days of the week, names of months, etc., which can act as indicators that aid their detection.

It is interesting to notice that in events detection part-of-speech information becomes extremely useful because of the abundance of events denoted as verbs. In this approach any part-of-speech is included, just word embeddings. This could be interpreted as analogous to the human common understanding of a language, where the knowledge of what is a verb or a noun it is not particularly needed to understand. Another interesting observation is that the jointly detection of time expressions and events performs better that each isolated task (Table 17).

[4] This measure was calculated with $AttrAccuracy = AttrF1/EntityExtractionF1$ [39].

6 Conclusion

This work tackles the problem of detecting and classifying time expressions with approaches based purely on distributed representations of words and artificial neural networks, and it presents interesting results for Spanish with a relatively small dataset.

Initial results that support the possibility of interpreting time expressions purely from data were presented. We found that word representations tend to contain information that refers to the sequential nature of units such as the names of months, days and numbers, among others. This trait is potentially useful for interpreting expressions. Regarding numerical and ordinal entities, we found that they take into account granularity information in the vector representations.

We showed the behavior of neural network models in detecting and classifying expressions, as well as the effect of varying the definition of the model. Results show the relevance of the local linguistic context. Recurrent models did not produce significant improvements in the experiments conducted, when compared to simple feedforward models, and the latter have the advantage that their training is much less costly. We also showed the effect of applying regularization techniques, especially emphasizing the benefit of considering noise in the input data and dropout.

Based on the previous results, we trained models for the problem in English and we also added the events detection problem. The results for English are encouraging, and they provide an example of the approach behavior when more data is available, both for supervised data to train models and for unsupervised data used to build the word embeddings. Regarding events detection, it is of special interest because it is not strongly based on specific lexicon, as is the case of time expressions. The results obtained were encouraging and the inclusion of said task, together with the detection and classification of time expressions brings an improvement in the results as compared to the isolated treatment presented before.

References

1 Sisay Fissaha Adafre and Maarten de Rijke. Feature engineering and post-processing for temporal expression recognition using conditional random fields. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, FeatureEng'05, pages 9–16, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL: `http://www.aclweb.org/anthology/W05-0402`.

2 David Ahn, Sisay Fissaha Adafre, and Maarten de Rijke. Towards task-based temporal extraction and recognition. In Graham Katz, James Pustejovsky, and Frank Schilder, editors, *Annotating, Extracting and Reasoning about Time and Events*, number 05151 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

3 David Ahn, Joris Rantwijk, and Maarten Rijke. A cascaded machine learning approach to interpreting temporal expressions. In *in Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2007*. Citeseer, 2007.

4 Gabor Angeli, Christopher D. Manning, and Daniel Jurafsky. Parsing time: Learning to interpret time expressions. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 3-8, 2012, Montréal, Canada*, pages 446–455, 2012. URL: `http://www.aclweb.org/anthology/N12-1049`.

5 Gabor Angeli and Jakob Uszkoreit. Language-independent discriminative parsing of temporal expressions. In *Proceedings of the 51st Annual Meeting of the Association for Compu-*

tational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers, pages 83–92, 2013. URL: `http://www.aclweb.org/anthology/P/P13/P13-1009.pdf`.

6 Agustín Azzinnari and Alejandro Martínez. Representación de Palabras en Espacios de Vectores. Proyecto de grado, Universidad de la República, Uruguay, 2016.

7 Steven Bethard. Cleartk-timeml: A minimalist approach to tempeval 2013. *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, 2013.

8 Steven Bethard. A synchronous context free grammar for time normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 821–826, 2013. URL: `http://www.aclweb.org/anthology/D/D13/D13-1078.pdf`.

9 Angel X. Chang and Christopher Manning. Sutime: A library for recognizing and normalizing time expressions. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

10 R. Collobert and J. Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. *In Proceedings of ICML, pages 160–167*, 2008.

11 R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *JMLR, 12:2493–2537*, 2011.

12 Mathias Etcheverry and Dina Wonsever. Spanish word vectors from wikipedia. *Language Resource Conference (LREC 2016)*, 2016.

13 Michele Filannino. Temporal expression normalisation in natural language texts. *CoRR*, abs/1206.2010, 2012. URL: `http://arxiv.org/abs/1206.2010`.

14 Michele Filannino, Gavin Brown, and Goran Nenadic. Mantime: Temporal expression identification and normalization in the tempeval-3 challenge. *CoRR*, abs/1304.7942, 2013. URL: `http://arxiv.org/abs/1304.7942`.

15 A. Graves and J. Schmidhuber. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks*, 18(5–6):602–610, 2005.

16 Claire Grover, Richard Tobin, Beatrice Alex, and Kate Byrne. Edinburgh-LTG: TempEval-2 System Description. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval'10, pages 333–336, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL: `http://dl.acm.org/citation.cfm?id=1859664.1859738`.

17 Naman Gupta, Aditya Joshi, and Pushpak Bhattacharyya. A temporal expression recognition system for medical documents by taking help of news domain corpora. *12th International Conference on Natural Language Processing (ICON)*, 2015.

18 Ozan Irsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

19 Leif Johnson, Majid alDosari, Filip Juricek, John, Kyle Kastner, Yoav Goldberg, talbaumel, Yu Yang, mhr, Eben Olson, and Sergey Romanov. theanets: v0.6.1, July 2015. `doi:10.5281/zenodo.19930`.

20 Hyuckchul Jung and Amanda Stent. ATT1: temporal annotation using big windows and rich syntactic and semantic features. In *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, Georgia, USA, June 14-15, 2013*, pages 20–24, 2013.

21 Oleksandr Kolomiyets and Marie-Francine Moens. Kul: Recognition and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval'10, pages 325–328, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL: http://www.aclweb.org/anthology/S10-1072.

22 Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. *Context-dependent semantic parsing for time expressions*, volume 1, pages 1437–1447. Association for Computational Linguistics (ACL), 2014.

23 H. Llorens, E. Saquete, and B. Navarro-Colorado. Timeml events recognition and classification: Learning crf models with semantic roles. In *In COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, page 725–733, 2010.

24 Inderjeet Mani and D. George Wilson. Robust temporal processing of news. In *38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, China, October 1-8, 2000.*, 2000. URL: http://www.aclweb.org/anthology/P00-1010.

25 Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *In Proceedings of Workshop at ICLR*, 2013.

26 Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

27 Jordi Poveda, Mihai Surdeanu, and Jordi Turmo. An analysis of bootstrapping for the recognition of temporal expressions. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, SemiSupLearn'09, pages 49–57, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL: http://dl.acm.org/citation.cfm?id=1621829.1621836.

28 Georgiana Puscasu. A framework for temporal resolution. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*, 2004. URL: http://www.lrec-conf.org/proceedings/lrec2004/pdf/664.pdf.

29 James Pustejovsky, José M. Castaño, Robert Ingria, Roser Sauri, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. Timeml: Robust specification of event and temporal expressions in text. In *New Directions in Question Answering, Papers from 2003 AAAI Spring Symposium, Stanford University, Stanford, CA, USA*, pages 28–34, 2003.

30 Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. *Proceedings of the Thirteenth Conference on Computational Natural Language Learning , CoNLL '09, pages 147–155, Stroudsburg, PA, USA.*, 2009.

31 Richard Socher, John Bauer, Christopher D, Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. *Association for Computational Linguistics 2013 Conference (ACL 2013)*, 2013.

32 Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in Neural Information Processing Systems (NIPS 2011)*, 2011.

33 Richard Socher, Alex Perelygin, Jason Chuang Jean Wu, Chris Manning, Andrew Ng, and Chris Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, 2013.

34 Jannik Strötgen and Michael Gertz. HeidelTime: High Quality Rule-based Extraction and Normalization of Temporal Expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval'10, pages 321–324, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL: http://dl.acm.org/citation.cfm?id=1859664.1859735.

35 Jannik Strötgen and Michael Gertz. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298, 2013. doi:10.1007/s10579-012-9179-y.

36 Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL: http://arxiv.org/abs/1605.02688.

37 Tijmen Tieleman and Geoffrey E. Hinton. Lecture 6.5 – rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

38 Naushad UzZaman and James F. Allen. TRIPS and TRIOS System for TempEval-2: Extracting Temporal Information from Text. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval'10, pages 276–283, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL: http://www.aclweb.org/anthology/S10-1062.

39 Naushad UzZaman, Hector Llorens, James F. Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. Tempeval-3: Evaluating events, time expressions, and temporal relations. *CoRR*, abs/1206.5333, 2012. URL: http://arxiv.org/abs/1206.5333.

40 L. J. P. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research 9(Nov):2579-2605*, 2008.

A Result Tables

A.1 Noise

We tested the effects of applying noise to the input. We slightly modified the input, according to Gaussian distribution centered at zero and using several variance values, in the input data and intermediate representations. The variance regulates the noise level injected.

Including noise substantially improved the recall, and it slightly improved the precision for noise values from 0.1 (Table 18). The results for noise values under 0.1 were slightly lower. The best results were obtained when considering 0.2 injected noise variance, with significant degradation when considering a 0.3 variance. The hidden layer noise has a less substantial impact and is more sensitive to the amount of noise injected.

We trained a model with the best noise levels in the input and in the hidden layer to observe the interaction of both cases. It seems that the effects add up degrading the precision.

As for expression detection, as well as in classification, we obtained the best result when considering a 0.2 noise value at the input and with no noise in the hidden layer. This was the same case that produced the best results in classification. The best results achieved

Table 18 Results in the classification of expressions in feedforward models on 200-dimension words, with three words of symmetric context and a 300-size hidden layer, with varying noise levels at the input and hidden layer. The results between brackets correspond to cases with noise in the hidden layer.

Noise	Prec	Rec	F
0.00	78.61	62.96	69.92
0.01	77.20 (**79.41**)	61.11 (62.50)	68.22 (69.95)
0.05	77.40 (77.60)	63.43 (**65.74**)	69.72 (**71.18**)
0.10	79.33 (77.40)	65.74 (63.43)	71.90 (69.72)
0.20	**80.66** (77.01)	**67.59** (62.04)	**73.55** (68.72)
0.30	78.65 (–)	64.81 (–)	71.06 (–)

Table 19 Results in the classification of expressions from feedforward models on 200-dimension words, with three words of symmetric context and a 300-size hidden layer, with varying levels of dropout in the hidden layer.

Dropout	Prec	Rec	F
0.00	78.61	62.96	69.92
0.01	**80.57**	**65.28**	**72.12**
0.05	78.29	63.43	70.08
0.10	75.82	63.89	69.35
0.20	77.14	62.50	69.05

Table 20 Results in the classification of expressions from feedforward models on 200-dimension words, with three words of symmetric context and a 300-size hidden layer, with varying levels of L2 regularization.

L2	P	R	F
0.00	78.61	62.96	69.92
0.0001	78.21	64.81	70.89
0.001	80.11	65.28	**71.94**
0.01	77.58	62.50	69.23
0.05	79.21	**65.43**	71.57
0.10	**81.21**	62.04	70.34
0.20	78.82	62.04	69.43

for detection were 78.59 F-score for the strict case and 81.42 for the relaxed case, where displacements of a word were allowed in the extension of the expression detected.

A.2 Dropout

This section shows the behavior of dropout (or multiplicative mask) in the hidden layer. This technique randomly turns to zero some entries in intermediate layers. It can be seen as a partial network where some components are completely eliminated. The dropout value is the portion of units set to zero.

We conducted an experiment with varying dropout levels for the hidden layer of a feedforward (Table 19). Regarding detection, a dropout value of 0.01 also yielded the best result in the hidden layer, with a 75.70 F-score in the strict case and of 81.84 for the relaxed case.

A.3 L1 and L2 regularizations

Overfitting can be reflected on high values in the weights learned, so training the model avoiding high values in the learned weights may help to prevent it. This technique is called *weight decay* (or L2 regularization). The technique consist on adding the term $\lambda_{L2}||\theta||_2$ to the target function, where $||.||_2$ is the L2 norm, and θ is the weights vector to fit. This aims to a reduction in the weights magnitude besides the function to optimize.

Many instances of a base model was trained with different values for L2 regularization each (Table 20). Results improve when considering $\lambda_{L2} = 0.001$. Anyways, the results fluctuate and for that reason is hard to determine which was the best configuration.

Furthermore, besides the tendency towards low values in learned parameters, there might be a positive effect on sparse representations. A way to tend to sparse parameters is to

Table 21 Results in the classification of expressions from feedforward models on 200-dimension words, with 3 words of symmetric context and a 300-size hidden layer, with varying levels of L1.

L1	P	R	F
0.00	78.61	62.96	69.92
0.001	**80.00**	62.96	70.47
0.01	78.89	**65.74**	**71.72**
0.05	74.85	59.26	66.15
0.10	74.30	61.57	67.34
0.20	77.21	56.50	65.24

include the term $\lambda_{L1}\|\theta\|_1$, where $\|.\|_1$ is the L1 norm. Including this term reduces some components if the result is not affected.

Table 21 shows the results obtained in the detection and classification of expressions. We can see that values higher than 0.05 for L1 regularization significantly degrade the results. The best result was obtained with 0.01, which showed an improvement of 2 points in the F-score.

B English Translated Tables

Table 22 Table with the closest terms (to the heading) ordered by distance to the time expressions representations.

Dawn	Neolithic	Start	Before	Impromptu	Hurriedly
sunset	paleolithic	initiation	after	suddenly	march
morning	mesolithic	gave	behind	die	retract
night	chalcolitic	before	already	perish	hastily
day	neolthic	final	days	died	dried out
midnight	date	giving	then	prematurely	move
nightfall	pleistocene	arrival	that	tragically	periodically
midday	preceramic	endings	months	early	direct
twilight	epipaleolytic	beginning	time	...	...
early morning	bronze	moment	begin		
...	...	...	...		

Table 23 Table with the closest terms (to the heading) ordered by distance to the representations of ordinal and numerical terms.

First	Second	Twentieth	1853	1850	1700	1999
then	third	thirtieth	1855	1840	1600	1998
second	first	tenth	1854	1849	1800	1995
same	fourth	fortieth	1856	1870	1500	1997
last	last	nineth	1852	1860	1400	1996
first	fifth	fiftieth	1851	1880	1200	2002
later	first	eighth	1865	1851	1100	2003
after	third	fifth	1849	1830	1300	1994
...	...	...	...	...	...	...

Models and Algorithms for Chronology

Gilles Geeraerts[1], Eythan Levy[2], and Frédéric Pluquet[3]

1 **Université libre de Bruxelles, Computer Science Department, Brussels, Belgium**
`gigeerae@ulb.ac.be`
2 **Tel Aviv University, Archaeology Department, Tel Aviv, Israel**
`eythan.levy@gmail.com`
3 **École Supérieure d'Informatique (HE2B-ESI), Brussels, Belgium**
`fpluquet@he2b.be`

Abstract

The last decades have seen the rise of many fundamental chronological debates in Old World archaeology, with far-reaching historical implications. Yet, outside of radiocarbon dating – where Bayesian formal tools and models are applied – these chronological debates are still relying on non-formal models, and dates are mostly derived by hand, without the use of mathematical or computational tools, albeit the large number of complex constraints to be taken into account. This article presents formal models and algorithms for encoding archaeologically-relevant chronological constraints, computing optimal chronologies in an automated way, and automatically checking for chronological properties of a given model.

1998 ACM Subject Classification J.2 [Physical Sciences and Engineering] Archaeology

Keywords and phrases Chronology, Algorithms, Archaeology, Formal methods

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.13

1 Introduction

1.1 Motivation

Reconstructing ancient chronologies based on archaeological data is a complex task, which has haunted researchers since the start of the discipline. Current chronologies of given regions result from an integration of diverse data such as historical records (providing regnal dynasties for example), pottery sequences (providing relative chronologies for a given region), stratigraphic sequences (providing synchronisms through stratified pottery and artefacts, both local and imported), ancient inscriptions (providing synchronisms between different kings for example) and laboratory methods (providing date-ranges for a given event, through radiocarbon dating for example). In most cases, these data constitute a complex web of intricate information, connecting archaeological data of neighbouring sites and regions in a subtle way. Hence, any change in chronological hypotheses at one end of the network (say, changing the historical dates of a given king for example) can have far-reaching chronological repercussions throughout the network (say for example that inscriptions of this king have been found in a specific archaeological layer, bearing specific pottery types). How can we model such constraints formally, and how can we use our formal model to automatically derive a global chronology for a region of interest, through the integration of several scattered local chronological constraints? We have observed that, outside of carbon dating, where Bayesian statistical tools are used to formally derive date estimates [15], traditional chronological research still uses no formal model or algorithm to integrate sets of local chronological

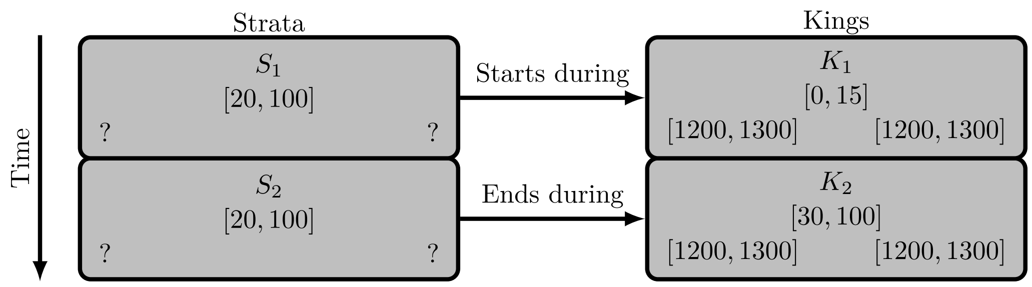


Figure 1 Our running example: Chronoland. Each period is represented by a rectangle containing the period's name. The bounds on each period's duration, start date and end date are displayed in the centre, bottom left corner and bottom right corner respectively.

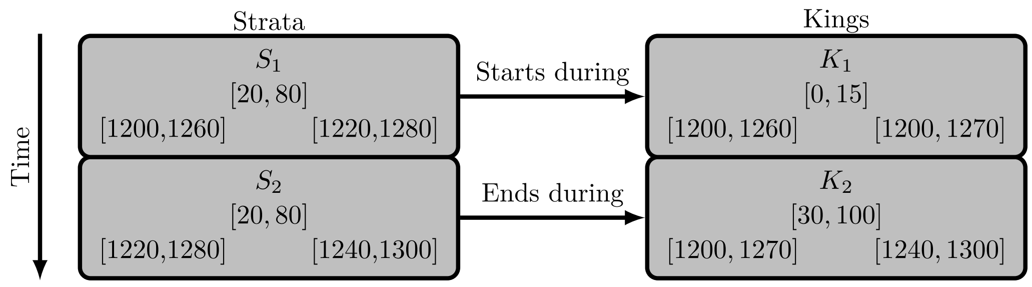


Figure 2 Optimal date and duration estimates for the Chronoland example.

constraints into a global chronology. In most cases (outside of radiocarbon dating), the date estimates proposed by researchers are derived by hand, thus certainly obtaining non-optimal results, since manual treatment of any set of chronological constraints of non-trivial size is almost impossible. The purpose of this paper is to present a model that allows one to formally model archaeologically-relevant chronological constraints, to compute optimal chronologies through specific algorithms, and to automatically check chronological properties of a given model. Applications of such models are numerous, since the past decades have seen the rise of many important – and still unsettled – chronological debates in Old World archaeology, with far-reaching historical implications. Important recent examples of such debates include the Thera Eruption debate, where two opposing chronologies differ by *more than a century*, with important implications regarding the chronology of the Aegean Bronze Age [8]. A second well-known example is the Iron Age Low Chronology debate for the southern Levant, where two chronologies differ as to whether the first full-fledged territorial states in the region are to be dated to the 10th or the 9th century BCE [13].

1.2 Running example

As a didactic example of the kind of data and constraints that typically appear in chronological debates, we will start with a small test case, in order to show that even small examples, with far less constraints than any real archaeological case study, are impractical to treat manually and require a formal and algorithmic approach. The test case is called Chronoland, and is presented hereinafter.

The story of Chronoland. In the kingdom of Chronoland, Kings K_1 and K_2 reigned in succession. We do not know their precise reign dates, but both reigns are known to have occurred between 1200 and 1300 CE. Furthermore, K_1's reign did not exceed 15 years, since we know he died at a rather young age, and King K_2 reigned for at least 30 complete years, since we have a monument dated to his 31st year[1]. We can also safely assume that K_2's reign did not exceed 100 years. Now Chronocity, the capital city of Chronoland, was excavated and only two strata were found, namely strata S_1 and S_2. We know that stratum S_1 was built by King K_1 since he thus claims in a stela found there in-situ, and we also know that Stratum S_2 ended during the reign of King K_2, since ancient annals tell us that the city was destroyed during his reign and never reoccupied. Finally, we assume that each one of our strata has a duration of at least 20 years and at most 100 years. The data of the example are given in schematic form in Figure 1, and several natural questions can be asked about it, such as:

1. What are the most precise estimates (ranges) we can gather for the start date, end date and duration of each king and stratum of Chronoland?
2. Has King K_1 built stratum S_2? (Recall that the data only asserts he has built S_1.)

Observe that the second question could admit three different answers because of the uncertainties on the dates: (i) 'yes', K_1 has surely built S_2, in all possible scenarios that fit with the constraints on the dates and durations; or (ii) 'no', K_1 has surely not built S_2 (again in all possible scenarios); or (iii) 'maybe', i.e. some dates that respect the constraints, imply that K_1 has built S_2, and some don't.

Chronology computation. We start by discussing the first question (optimal ranges). These optimal estimates are shown in Figure 2. One can easily see that they are not straightforward and require a close look at the model. Here are the steps that one could follow to obtain these results:

1. **Initialisation.** The date ranges of K_1 and K_2 can be improved using the duration estimates. The 30 years minimum duration of K_2 imply that K_2 starts in [1200,1270] and that it ends in [1230,1300].
2. **K_1–K_2 sequence.** The end of K_1 equals the start of K_2, hence it is in [1200,1270]. The start of K_1 is thus also in [1200,1270], since a start date cannot exceed its corresponding end date.
3. **S_1–K_1 synchronism.** The 'S_1 starts during K_1' synchronism implies that the start of S_1 is in [1200,1270]. The [20,100] duration range now implies that the end of S_1 is in [1220,1370].
4. **S_1–S_2 sequence.** The start of S_2 equals the end of S_1, hence it is in [1220,1370]. The [20,100] duration range now implies that the end of S_2 is in [1240,1470].
5. **S_2–K_2 synchronism.** The 'S_2 ends during K_2' synchronism implies that the end of S_2 is in [1240,1300]. The [20,100] duration range now implies that the start of S_2 is in [1220,1280]. The duration range can now be reduced to [20,80] since the earliest start of S_2 is 1220 and its latest end is 1300.
6. **S_1–S_2 sequence.** The end of S_1 equals the start of S_2, hence it is in [1220,1280]. The [20,100] duration range now implies that the start of S_1 is in [1200,1260]. The duration range can now be reduced to [20,80] since the earliest start of S1 is 1200 and its latest end is 1280.

[1] We use a cautious bound of 30 years instead of 31 years because we do not know if the king completed his 31st year of reign (i.e. he might have died in the course of his 31st year).

7. **S_1–K_1 synchronism.** We need to come back to the 'S_1 starts during K_1' synchronism. Our new upper bound of 1260 on S_1's start needs to be propagated back to K_1, implying a new improved range of [1200,1260] on K_1's start, instead of the former [1200,1270].
8. **S_2–K_2 synchronism**. Finally, we need to consider anew the 'S_2 ends during K_2' synchronism. Our new lower bound of 1240 on S_2's end needs to be propagated back to K_2, implying a new improved range of [1240,1300] on K_2's end, instead of the former [1230,1300].

Note the difficulty of manually solving this simple problem, especially the final 'retroaction' step, where K_1's start date and K_2's end date get further refined by 10 years thanks to their synchronisms with S_1 and S_2 respectively. This final step is somewhat unexpected since S_1 and S_2 had no a priori absolute chronology of their own (they had only duration estimates), and thus derived their absolute dates from K_1 and K_2. Now, let us come back to the question whether king K_1 has built stratum S_2. It turns out that this is impossible, because K_1's reign lasts at most 15 years, but at least 20 years separate the respective start dates of S_1 and S_2, since S_1 lasts at least 20 years. However, again, deriving this information from Figure 1 or Figure 2 is not straightforward. Also note that this example is of small dimensions compared to real archaeological data for which manual treatment of the information is practically impossible.

1.3 Contribution

As can be seen from the Chronoland example, rigorous and automatic techniques for reasoning about chronological problems are needed. To the best of our knowledge, no such techniques are available today, and the kind of reasoning we exhibit in the Chronoland example are performed 'by hand' by archaeologist, using restricted data sets. In this paper, **our first contribution is a formal constraint language (Section 2) that can express most relationships between periods which are needed for practical cases of archaeology**. More precisely, our language can express: (i) lower and upper bounds on the start dates, end dates and durations of periods; (ii) sequences (such as period S_2 follows directly S_1); and (iii) different kinds of synchronisms between periods (such as 'S_1 starts during K_1', etc). Then, **our second contribution consists of several algorithmic techniques (Section 3) to manipulate such constraints and extract information that is meaningful to archaeologists**, namely: (i) what are the *tightest bounds* on the start dates, end dates and durations of all periods that one can infer from a given set of constraints? (ii) is the set of constraints *satisfiable*. That is, is it possible to assign dates to the starts and ends of all periods that satisfy all the constraints? (iii) do the constraints *imply* that two given periods have a non-empty intersection? That is, is the intersection guaranteed to exist, for all choices of start and end dates of the periods that satisfy the constraints? We call this problem the *sure-contemporaneity* problem; and, finally (iv) do the constraints *allow* a non-empty intersection between two periods? That is, does there *exist* a choice of dates for the starts and ends of all periods that satisfy the constraints and where the two given periods intersect? We call this problem the *possible-contemporaneity* problem. Observe that both the sure- and possible-contemporaneity problems can be invoked to answer our question: 'Has K_1 built S_2?'

Following [17], we translate our constraints in directed weighted graphs, and reduce the computation of the tightest bounds to an all-pair shortest paths computation in this graph (and, as particular case, we obtain an algorithm for the satisfiability problem which amounts to detecting negative cycles in the graph). Such computation can be carried out *in polynomial time* (in the number of periods) using classical algorithms [12], which is a clear strong point of approach, enabling scaling to examples of big dimensions. Since all-pairs shortest paths algorithms also detect negative cycle, we can also test satisfiability in polynomial time[2]. Finally, we show that, after computation of the shortest paths, the sure- and possible-contemporaneity problems can be solved *in constant time.*

1.4 Related works

The best known use of computer-assisted techniques in Archaeology is in a setting which is different from ours, namely the calibration of radiocarbon dates [15] thanks to the OxCal software. The aim of Oxcal is to refine estimates of dates computed from radiocarbon samples, by taking into account extra information such as the order of strata. Due to the stochastic nature of radiocarbon decay, these methods are probabilistic (they rely on Bayes's theorem) while our methods are not.

Concerning relative chronology, other formal approaches exist, based on the Harris matrix [7] or the generalised schemes of Sharon [16] and Holst [9]. These works differ from ours in several way. First, they aim at at obtaining a feasible sorting of the archaeological features, but do not estimate their absolute time-frame, while our method provides both *absolute* and *relative* information on the periods. Second, the underlying computational problem they address is NP-complete (hence, they rely on heuristics), while we have identified problems that can be solved in polynomial time. In fact, an early paper by Kromholz [11] already combined some relative and absolute dating elements, though in a limited way, through the application of business-oriented tools (such as Pert charts) to chronological problems. We are not aware however of any later paper that took on this novel approach.

On the other hand, the constraints we rely on are actually a special case of the *zone* data structure developed in the framework of *timed automata* [1, 6, 4] that form the cornerstone of efficient tools for the analysis of timed automata such as Uppaal [2] and TiAMo [5]. In our case, however, variables take natural values, instead of real values in the case of timed automata.q Our constraints are also a particular case of general classes of constraints developed in the artificial intelligence community for temporal reasoning, see for instance [14]. Those constraints are more expressive than ours and basic problems about them are already NP-hard [14] while we propose polynomial-time algorithms.

2 Modelling chronology problems

In this section, we introduce our model for the chronology problems we have sketched in the introduction. Throughout the paper, we denote the set of natural and integers numbers by $\mathbb{N}$ and $\mathbb{Z}$ respectively. We consider closed interval on $\mathbb{N}$ (i.e., convex subsets of $\mathbb{N}$), with natural endpoints. We use the usual notation $[a, b]$ to denote intervals, and for $I = [a, b]$, we let $\ell(I) = a$ and $u(I) = b$.

2.1 Periods and chronologies

Periods. We fix a finite set $\mathcal{P}$ of *periods*. We use the general term 'period' to speak about continuous periods of time characterised by a start date, an end date and a duration (defined

[2] Similar techniques are used in the setting of timed automata, see related works.

as the difference between the end and start dates)[3]. Examples of periods include historical eras (‘The Middle Ages’), archaeological strata, king’s reigns and ceramic periods, among others.

Chronologies. A *chronology* $\mathcal{C}$ on a set of periods $\mathcal{P}$ is function that associates, to each period $p \in \mathcal{P}$, an interval $\mathcal{C}(p) = [a_p, b_p] \subset \mathbb{N}$ with natural endpoints a_p and b_p. For all periods p, the interval $\mathcal{C}(p)$ represents the whole duration of the period, in the sense that its endpoints represent the start and the end *dates* of the period. Throughout the paper, we assume that all dates are given simply as years, but a finer granularity can be used if need be (for example, dates can represent days). Observe that, for historical events, start and end dates could be negative, but we assume that there is a lower bound on all dates[4] (i.e., an ‘origin of time’), which we identify with 0 (hence we have $a_e, b_e \geq 0$ for all $e \in \mathcal{P}$). A chronology $\mathcal{C}$ for our running example could be s.t. $\mathcal{C}(K_1) = [1210, 1222]$, which is compatible with the constraints in Figure 1.

2.2 Constraints

Common chronological constraints. Our goal is to formalise a large set of chronological constraints relevant to archaeology and history. In these fields, the most common constraints can be grouped in three families:

Bounds. The first family consists of lower and upper bounds on the start date, end date and duration of a period. For example, in Figure 1, the $[20, 100]$ years constraint on S_1’s duration, or the $[1200, 1300]$ constraint on K_1’s start date belong to this family.

Sequences. The second family expresses that a certain period starts where the preceding period ends, as in the K_1, K_2 sequence, and the S_1, S_2 sequence.

Synchronisms. Finally, the third family expresses diverse sorts of *synchronisms*, such as contemporaneity (two periods having a non-empty intersection, as in two contemporary kings), ‘starts during’ (as in ‘S_1 starts during K_1) and ‘ends during’ (as in ‘S_2 ends during K_2’).

Let us now discuss a formal constraint language that allows us to describe all these constraints.

A formal model of constraints. To all finite sets of periods $\mathcal{P} = \{p_1, \ldots, p_n\}$, we associate a set of *variables* $\mathcal{V}(\mathcal{P}) = \{z_0, \mathsf{beg}(p_1), \mathsf{end}(p_1), \ldots \mathsf{beg}(p_n), \mathsf{end}(p_n)\}$, interpreted over the integers. For each period p_i, variables $\mathsf{beg}(p_i)$ and $\mathsf{end}(p_i)$ represent respectively the beginning and end of p_i. Variable z_0 is a special variable that is assumed to be always equal to 0 and its purpose will become clear later. Then, an atomic constraint on $\mathcal{P}$ is an expression of one of the following forms: either $x - y \sim c$ or $x \sim c$, where x, y are variables from $\mathcal{V}(\mathcal{P})$, $c \in \mathbb{Z} \cup \{+\infty\}$, and $\sim$ is either $\leq$ or $\geq$ or $=$. Finally, a *constraint* is a finite *conjunction* of atomic constraints[5]. For an atomic constraint ψ and a constraint φ, we write $\psi \in \varphi$ iff ψ is a conjunct of φ.

[3] Punctual events could be seen as special cases of periods have equal start and end dates, and null duration.
[4] In practice, this is not restrictive since events can always be associated to an epoch, even if this is very broad.
[5] Observe that neither disjunction nor negation are allowed.

Semantics. Let us now define the semantics of constraints in terms of chronologies. Intuitively, a constraint is meant to define a set of *possible chronologies*, which are all those that are compatible with the constraint. Formally, a chronology $\mathcal{C}$ (on set of periods $\mathcal{P}$) *satisfies* an atomic constraint $x - y \sim c$ (respectively, $x \sim c$) iff $\nu(x) - \nu(y) \sim c$ (respectively, $\nu(x) \sim c$), where $\nu : \mathcal{V}(\mathcal{P}) \to \mathbb{N}$ is the function associating to each variable x a *valuation* according to $\mathcal{C}$, i.e., for all $x \in \mathcal{V}(\mathcal{P})$: (i) $\nu(x) = 0$ if $x = z_0$; (ii) $\nu(x) = \ell(\mathcal{C}(e))$ if $x = \mathsf{beg}(e)$; and (iii) $\nu(x) = u(\mathcal{C}(e))$ if $x = \mathsf{end}(e)$. When a chronology $\mathcal{C}$ satisfies an atomic constraint ψ, we write $\mathcal{C} \models \psi$. We extend this notion of satisfaction to constraints: $\mathcal{C}$ satisfies a constraint $\varphi = \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_n$ (noted $\mathcal{C} \models \varphi$) iff $\mathcal{C}$ satisfies all conjuncts of φ, i.e., $\mathcal{C} \models \psi_i$ for all $1 \leq i \leq n$. We denote by $[\![\varphi]\!]$ the set of all chronologies $\mathcal{C}$ s.t. $\mathcal{C} \models \varphi$. Remark that this set could be empty, for instance if we have specified constraints that are not satisfiable. Note also that two different constraints can encode the same chronologies, e.g. when there are redundant atomic constraints. For example, $\varphi = x \geq 0 \wedge x \leq 1 \wedge y \geq 0 \wedge y \leq 1$ encodes the same chronologies as $\varphi' = \varphi \wedge x - y \leq 5$, (i.e., $[\![\varphi]\!] = [\![\varphi']\!]$) because φ *implies* $x - y \leq 5$.

Expressiveness of the model. While the language of constraints we have just defined might seem very restrictive, we claim that it allows one to define most of the relevant constraints in archaeology, as defined at the beginning of this section:

Terminus post quem. A *Terminus post quem* is defined as a lower bound B on a given start or end date. Such constraints can be expressed by $\mathsf{beg}(p) \geq B$ and $\mathsf{end}(p) \geq B$, respectively.

Terminus ante quem. Symmetrically, a *Terminus ante quem* is defined as an upper bound B on a given start or end date. Such constraints correspond to $\mathsf{beg}(p) \leq B$ and $\mathsf{end}(p) \leq B$, respectively.

Date range. Ranges on dates are the conjunction of a *terminus post* and *ante quem*. In the example of Figure 1, the constraint on the start of K_1 is expressed as: $\mathsf{beg}(K_1) \geq 1200 \wedge \mathsf{beg}(K_1) \leq 1300$.

Duration constraints. Since the duration of a period p can be computed as $\mathsf{end}(p) - \mathsf{beg}(p)$, constraints (lower bounds, upper bound or ranges) on the duration of a period also fit our model. In the example of Figure 1, the range on the duration of K_1 is expressed as: $\mathsf{end}(K_1) - \mathsf{beg}(K_1) \geq 0 \wedge \mathsf{end}(K_1) - \mathsf{beg}(K_1) \leq 15$.

Sequence. A sequence of periods p and q means that q follows immediately after p. Thus, the end date of p is the start date of q, which is formalised as: $\mathsf{end}(p) - \mathsf{beg}(q) = 0$.

'Contemporaneity' synchronism. Periods p and q are contemporary, i.e. there is a non-empty intersection between the intervals $I_p = [\mathsf{beg}(p), \mathsf{end}(p)]$ and $I_q = [\mathsf{beg}(q), \mathsf{end}(q)]$. To understand how to model this, we consider the opposite statement: the intersection between I_p and I_q is empty iff either I_p follows strictly I_q or I_q follows strictly I_p. That is, $I_p \cap I_q = \emptyset$ iff $\mathsf{end}(p) < \mathsf{beg}(q) \vee \mathsf{end}(q) < \mathsf{beg}(p)$. This can be expressed by our constraints by taking the negation: $I_p \cap I_q \neq \emptyset$ iff $\mathsf{end}(p) \geq \mathsf{beg}(q) \wedge \mathsf{end}(q) \geq \mathsf{beg}(p)$.

'Starts during' synchronism. Period p starts during period q, i.e. the start of p is included in the interval $[\mathsf{beg}(q), \mathsf{end}(q)]$. This is formalised as: $\mathsf{beg}(p) \geq \mathsf{beg}(q) \wedge \mathsf{beg}(p) \leq \mathsf{end}(q)$.

'Ends during' synchronism. Period p ends during period q, i.e. the end of p is included in the interval $[\mathsf{beg}(q), \mathsf{end}(q)]$. This is formalised as: $\mathsf{end}(p) \geq \mathsf{beg}(q) \wedge \mathsf{end}(p) \leq \mathsf{end}(q)$.

'Inclusion'. Period p is included in period q, which can be formalised in our model as: $\mathsf{beg}(p) - \mathsf{beg}(q) \geq 0 \wedge \mathsf{end}(p) - \mathsf{end}(q) \leq 0$.

In addition to these constraints that come from the archaeological data, we *assume* from now on that all our constraints *imply* that all periods must start before they end. This can be achieved by taking the conjunction of any constraint with: $\bigwedge_{p \in \mathcal{P}} \mathsf{beg}(p) \leq \mathsf{end}(p)$.

Observe that one type of requirement that our constraints cannot express is *non-contemporaneity*, i.e. that the intersection between two periods p and q is empty. Indeed, non-contemporaneity means that *either* the end of p occurs strictly before the beginning of q *or* end of q occurs strictly before the beginning of p. However, our constraint language does not allow one to express disjunction.

2.3 Normalisation of constraints

In order to make our subsequent discussions easier, we will, from now on, consider a *normal form* for constraints, where constraints contain atomic constraints of the form $x - y \leq c$ only. Let φ be a constraint. We obtain $\mathsf{Norm}(\varphi)$, the normal form of φ by applying the following steps:

1. First, we make sure that there exists at least one atomic constraint for each pair of variables x and y, by taking the conjunction of φ with:

$$\bigwedge_{x \in \mathcal{V}(\mathcal{P})} \big(x - x \leq 0 \wedge x - z_0 \leq +\infty \wedge z_0 - x \leq 0\big) \wedge \bigwedge_{x,y \in \mathcal{V}(\mathcal{P}) \setminus \{z_0\}} x - y \leq +\infty\,.$$

   It is easy to check that the resulting constraint accepts the same set of chronologies than φ. Indeed, since we assume that z_0 is always null: $x - x \leq 0$ is equivalent to $x \leq x$; $x - z_0 \leq +\infty \wedge z_0 - x \leq 0$ is equivalent to $0 \leq x \leq +\infty$; and $x - y \leq +\infty$ should hold for all x, y since they take finite values.

2. We turn all atomic constraints into constraints of the form $x - y \leq c$. That is, we replace all atomic constraints $x \sim c$ by $x - z_0 \sim c$; and all constraints $x - y \geq c$ by $y - x \leq -c$.

3. Finally, whenever there are two different atomic constraints of the form $x - y \leq c$ and $x - y \leq c'$ in the resulting constraint, we keep the strongest one, i.e. $x - y \leq c$ if $c < c'$ and $x - y \leq c'$ otherwise.

The resulting is a normalised constraint $\mathsf{Norm}(\varphi)$. Observe that a normalised constraint always contain *exactly* $(2n+1)^2$ *atomic constraints of the form* $x - y \leq c$, where n is the number of periods; one for each pair of variables x and y in $\mathcal{V}(\mathcal{P})$ – this is actually the point of normalising constraints, even if the normalisation step might introduce some trivial atomic constraints. *From now on, we assume that all constraints are normalised.* We abuse notations and denote normalised constraint by non-normalised ones, writing, for instance, $x - y \leq 1$ instead of $\mathsf{Norm}(x - y \leq 1)$ which has 9 conjuncts.

Observe that this particular class of linear constraints has been studied before by Shostak [17]. They also form a special case of zones [1], and the normalised version of the constraint correspond to the Difference Bound Matrix [6] encoding the corresponding zone.

► **Example 1.** Consider again the example in Figure 1. In order to keep our example legible, we will consider only stratum S_1, king K_1 and the 'starts during' relationship between them. The following normalised constraint expresses exactly all the information from Figure 1 about S_1 and K_1, assuming $\mathcal{X} = \{z_0, \mathsf{beg}(S_1), \mathsf{beg}(K_1), \mathsf{end}(S_1), \mathsf{end}(K_1)\}$. Observe that the five last lines (marked 'Norm.') are here for normalisation purpose only.

$$
\begin{aligned}
&\mathsf{end}(S_1) - \mathsf{beg}(S_1) \leq 100 \wedge \mathsf{beg}(S_1) - \mathsf{end}(S_1) \leq -20 && \text{(Length } S_1\text{)}\\
\wedge\ &\mathsf{end}(S_1) - \mathsf{beg}(S_1) \leq 15 \wedge \mathsf{beg}(S_1) - \mathsf{end}(S_1) \leq 0 && \text{(Length } K_1\text{)}\\
\wedge\ &\mathsf{beg}(K_1) - z_0 \leq 1300 \wedge z_0 - \mathsf{beg}(K_1) \leq -1200 && \text{(Start } K_1\text{)}\\
\wedge\ &\mathsf{end}(K_1) - z_0 \leq 1300 \wedge z_0 - \mathsf{end}(K_1) \leq -1200 && \text{(End } K_1\text{)}\\
\wedge\ &\mathsf{beg}(S_1) - z_0 \leq +\infty \wedge \mathsf{beg}(S_1) - \mathsf{beg}(K_1) \leq +\infty \wedge \mathsf{beg}(S_1) - \mathsf{end}(K_1) \leq +\infty && \text{(Norm.)}\\
\wedge\ &\mathsf{beg}(K_1) - \mathsf{beg}(S_1) \leq +\infty \wedge \mathsf{beg}(K_1) - \mathsf{end}(S_1) \leq +\infty && \text{(Norm.)}\\
\wedge\ &\mathsf{end}(S_1) - z_0 \leq +\infty \wedge \mathsf{end}(S_1) - \mathsf{beg}(K_1) \leq +\infty \wedge \mathsf{end}(S_1) - \mathsf{end}(K_1) \leq +\infty && \text{(Norm.)}\\
\wedge\ &\mathsf{end}(K_1) - \mathsf{beg}(S_1) \leq +\infty \wedge \mathsf{end}(K_1) - \mathsf{end}(S_1) \leq +\infty && \text{(Norm.)}\\
\wedge\ &\bigwedge_{x\in\mathcal{X}} x - x \leq 0 \wedge z_0 - \mathsf{beg}(S_1) \leq 0 \wedge z_0 - \mathsf{end}(S_1) \leq 0 && \text{(Norm.)}
\end{aligned}
$$

3 Algorithmic manipulation of constraints

In this section, we show how the constraints from Section 2 can be manipulated algorithmically to answer the questions we have highlighted in introduction. We start by defining four meaningful problems on constraints, then show how the constraints can be expressed by means of weighted directed graphs, and finally give polynomial-time algorithms to solve those problems on the graphs. The main ideas of these techniques have been presented by Shostak [17], but our techniques for solving the sure- and possible- contemporaneity problems (that are motivated by the archaeological setting) are, as far as we know, original (and hence require dedicated proofs).

3.1 Four basic problems

Based on the motivations from the introduction, we focus on the four following problems.

Satisfiability. First, the *satisfiability* problem asks whether there is some chronology that satisfies a given constraint. If not, then the constraint contains a contradiction, for example: the constraint entails that some punctual period A should occur strictly before B, and, at the same time, that B should occur before A. Thus, the definition of this problem is as follows:

▶ **Problem 1.** *Given a constraint φ, the* satisfiability problem *asks whether* $[\![\varphi]\!] \neq \emptyset$*?*

If yes, we say that the constraint φ is *satisfiable*.

Tightening. Second, the *tightening* problem asks, given a constraint φ, to compute the tightest constraint φ' that represents the same set of chronologies. Intuitively, φ' represents the most precise information one can deduce from φ. Let us first define formally these notions.

Given two atomic constraints $\psi_1 = x - y \leq c_1$ and $\psi_2 = x - y \leq c_2$ (on the same variables x and y), we say that ψ_1 is *tighter* than ψ_2 (denoted $\psi_1 \preceq \psi_2$) iff $c_1 \leq c_2$. Intuitively, a constraint is *tighter* than another if it imposes a more stringent limitation on the possible values of the variables than the other (hence, the upper bound c_1 is smaller than or equal to c_2). Then, given two (normalised) constraints φ_1 and φ_2 on $\mathcal{P}$, we say that φ_1 is *tighter* than φ_2 (denoted $\varphi_1 \preceq \varphi_2$) iff each atomic constraint of φ_1 is tighter than the corresponding constraint in φ_2. Formally, $\varphi_1 \preceq \varphi_2$ iff for all x, y in $\mathcal{V}(\mathcal{P})$: $\psi_1 = x - y \leq c_1 \in \varphi_1$ and $\psi_2 = x - y \leq c_2 \in \varphi_2$ implies that $\psi_1 \preceq \psi_2$.

Observe that $\preceq$ is a partial order on constraints, which is not total. For instance, $x_1 - y_1 \leq 1 \wedge x_2 - y_2 \leq 2$ and $x_1 - y_1 \leq 2 \wedge x_2 - y_2 \leq 1$ are not comparable. Our definition of the order on constraints *implies* the intuition on the sets of chronologies they represent, i.e., for all constraints φ_1 and φ_2: $\varphi_1 \preceq \varphi_2$ implies $[\![\varphi_1]\!] \subseteq [\![\varphi_2]\!]$. However, the converse is not true. For instance, consider: $\varphi_1 = \mathsf{Norm}\,(x - y \leq 1 \wedge y - z \leq 1)$, and $\varphi_2 = \mathsf{Norm}\,(x - y \leq 1 \wedge y - z \leq 1 \wedge x - z \leq 2)$. It is easy to check that $[\![\varphi_1]\!] = [\![\varphi_2]\!]$ since the $x - z \leq 2$ atomic constraint of φ_2 is implied by its two other atomic constraints. Hence, in particular $[\![\varphi_1]\!] \subseteq [\![\varphi_2]\!]$, but, clearly $\varphi_1 \not\preceq \varphi_2$, because φ_1 constraints $x - z$ to be $\leq +\infty$, which is strictly weaker than $x - z \leq 2$. We can now define precisely the *tightening problem*:

▶ **Problem 2.** *Given a constraint φ, the* tightening problem *asks to compute the tightest (i.e., minimal wrt $\preceq$) constraint φ' s.t. $[\![\varphi']\!] = [\![\varphi]\!]$. Such a constraint φ' is called* tight.

Remark that this constraint φ' is necessarily unique. Indeed, assume it is not the case, and there are two constraints φ_1' and φ_2' s.t. $[\![\varphi_1']\!] = [\![\varphi_2']\!] = [\![\varphi]\!]$; $\varphi_1' \preceq \varphi$; $\varphi_2' \preceq \varphi$, but φ_1' and φ_2' are not comparable by $\preceq$ (that is, neither is tighter than the other). Then, we can consider instead the constraint ξ computed as follows: for each pair of variables x and y, we have in ξ the atomic constraint $x - y \leq \min(c_1, c_2)$, where c_1 and c_2 are the constants occurring in the atomic constraints on $x - y$ in φ_1' and φ_2' respectively (i.e., $x - y \leq c_1 \in \varphi_1'$ and $x - y \leq c_2 \in \varphi_2'$). By definition $\xi \preceq \varphi_1'$ and $\xi \preceq \varphi_2'$. Hence, $\xi \preceq \varphi$. Moreover, since $[\![\varphi_1']\!] = [\![\varphi_2']\!]$, we also have $[\![\xi]\!] = [\![\varphi_1']\!] = [\![\varphi_2']\!]$, hence $[\![\xi]\!] = [\![\varphi]\!]$. Thus, ξ is an even tighter constraint that can be used instead of φ_1' and φ_2'.

Sure-contemporaneity. Third, the *sure-contemporaneity problem* asks whether two given periods p_1 and p_2 do *certainly* have a non-empty intersection given a constraint φ:

▶ **Problem 3.** *The* Sure-Contemporaneity Problem *asks, given a constraint φ (on set of periods $\mathcal{P}$) and two periods p_1 and p_2 in $\mathcal{P}$, whether φ* guarantees *that p_1 and p_2 intersect, i.e., whether $\mathcal{C}(p_1) \cap \mathcal{C}(p_2) \neq \emptyset$ for all $\mathcal{C} \in [\![\varphi]\!]$*

Possible-contemporaneity. Fourth, the *Possible-Contemporaneity problem* asks whether two given periods p_1 and p_2 can *possibly* have a non-empty intersection given a constraint φ:

▶ **Problem 4.** *The* possible-contemporaneity problem *asks whether a given constraint φ* does not exclude *a contemporaneity between two given periods p_1 and p_2, i.e. whether there is $\mathcal{C} \in [\![\varphi]\!]$ s.t. $\mathcal{C}(p_1) \cap \mathcal{C}(p_2) \neq \emptyset$.*

3.2 Graph-based algorithms

Let us now present polynomial time algorithms for solving the four problems highlighted in the previous section. The core of our approach consists, following Shostak [17], in translating each constraint φ into a directed weighted graph G_φ. Roughly speaking, the set of nodes in G_φ is the set of variables of φ, and each constraint $x - y \leq c$ is translated by a directed edge[6] from x to y, labelled by c.

More precisely, in our setting, a graph $G = \langle V, E, w \rangle$ is made up of a finite set of vertices V, a finite set of (directed) edges $E \subseteq V \times V$ and a weight function $w : E \to \mathbb{Z}$. Given a constraint φ on a set of periods $\mathcal{P}$, we build the graph $G_\varphi = \langle V, E, w \rangle$ as follows:

[6] Observe that this definition is consistent with the classical definitions in the literature on timed automata [6], but that the edges are reversed wrt the definitions generally used in the literature on constraint graphs [14].

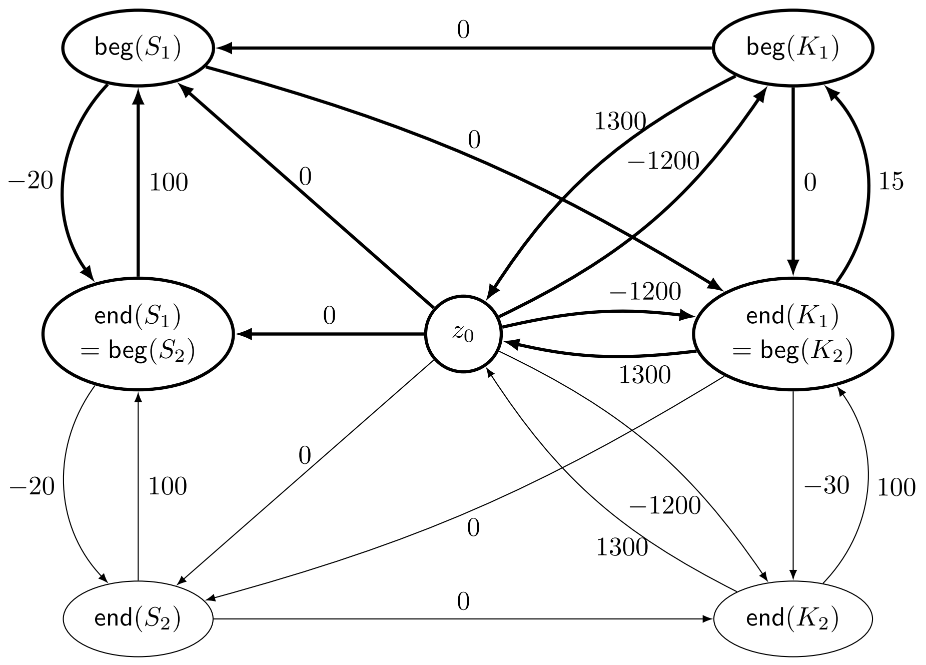


Figure 3 The graph Chrono for the constraint in Figure 1. To simplify presentation, nodes end(K_1) and beg(K_2) (respectively end(S_1) and beg(S_2)) have been merged and 0-labelled self-loops on the all nodes are not displayed. The bold part of the graph corresponds to the constraints on S_1 and K_1 only (see Example 1).

- $V = \mathcal{V}(\mathcal{P})$, i.e. there is a vertex for each variable in the constraint;
- $E = \{(x, y) \mid x - y \leq c \in \varphi \text{ and } c \neq +\infty\}$; and
- for all $(x, y) \in E$: $w(x, y) = c$ iff $x - y \leq c \in \varphi$. That is, there is an edge from x to y, labelled by c every time φ contains a non-trivial atomic constraint $x - y \leq c$ (by 'non-trivial', we mean that c is not $+\infty$). Thus, the edges encode exactly the set of φ's atomic constraints.

▶ **Example 2.** The graph corresponding to the full constraint modelling Chronoland (Figure 1) is given in Figure 3. The bold part of the graph corresponds to the constraint given in Example 1 (ranging on S_1 and K_1 only).

Now, we introduce our algorithms for our four problems given above.

Satisfiability checking and tightening. We address these two problems together as satisfiability can clearly be checked from the tightening of the constraint: clearly, φ is satisfiable iff the tightening of φ yields a constraint ξ s.t. $[\![\xi]\!] \neq \emptyset$. As said before, we rely on previous works for satisfiability and tightening. We start by recalling classical notions on graphs. Given a directed weighted graph $G = \langle V, E, w \rangle$, a *path* (from v_1 to v_k) is a finite sequence $\pi = v_1, v_2, \ldots, v_k$ of vertices ($v_i \in V$ for all i) s.t. $(v_i, v_{i+1}) \in E$ for all $1 \leq i \leq k - 1$. A *cycle* is a path $v_1, v_2, \ldots, v_k$ s.t. $v_1 = v_k$. The *weight* $w(\pi)$ of a path $\pi = v_1, v_2, \ldots, v_k$ is the sum of its edge weights, i.e. $\sum_{i=1}^{k-1} w(v_i, v_{i+1})$. A path π (and, in particular, a cycle) is *negative* iff $w(\pi) < 0$. A path $\pi = v_1, v_2, \ldots, v_k$ is called a *shortest path* from v_1 to v_k iff there is no other path π' from v_1 to v_k s.t. $w(\pi') < w(\pi)$ (observe that there could be several shortest paths from v_1 to v_k, but all of them have necessarily the same weight). In a graph that contains no negative cycle, it is well-known that there exists always a shortest path between two pairs of nodes, provided that there exists a path between those nodes. In such graphs G, we note $sp_G(x, y)$ the weight $w(\pi)$ of any shortest path π from x to y (and we let $sp_G(x, y) = +\infty$ if there is no path from x to y in G). The problem asking to compute $sp_G(x, y)$ for all pairs of nodes (x, y) (or to declare the value $sp_G(x, y)$ as undefined when the graph contains a negative cycle) is known in the literature as the *all-pairs shortest path problem* [12]. This problem allows us to solve the satisfiability and tightening problems. In [17], the author shows that:

▶ **Theorem 3** ([17]). *A constraint φ is satisfiable iff its graph G_φ contains no* negative cycle*.*

Moreover, the tightening of constraints has been considered in the setting of timed systems, and has been shown equivalent to the computation of all-pairs shortest paths:

▶ **Theorem 4** ([6]). *Given a* satisfiable *constraint φ (on set of variables $\mathcal{V}$ and with corresponding graph G_φ), the tightest constraint φ' s.t. $[\![\varphi]\!] = [\![\varphi']\!]$ is the constraint:*

$$\bigwedge_{x,y\in\mathcal{V}} x - y \leq sp_{G_\varphi}(x, y).$$

The reduction to shortest paths applies only when the constraint is satisfiable. Otherwise, the graph contains a negative cycle (by Theorem 3) and the notion of shortest path makes no sense.

Thus, in order to solve both satisfiability and tightening in practice, one can rely on one of the algorithms for the all-pairs shortest path problem from the literature (see [12] for a survey). All of these algorithms run in polynomial time. For example, one could use Johnson's algorithm [10], which runs in time $O(|V|^2 \log(|V|) + |V||E|)$ and detects negative cycles before computing all-pairs shortest paths, if the graph contains no negative cycle.

▶ **Example 5.** Let us come back to the Chronoland example. The graph in Figure 3 contains no negative cycle, hence the overall constraint φ of Chronoland is satisfiable. We now come to tightening. The result of the all-pairs shortest path computation is given in appendix (Figure 4) for reference. The most relevant results of computing all-pairs shortest paths in the graph of Figure 3 are summarised in Figure 2. For instance, the upper bound of 1260 on $\mathsf{beg}(S_1)$ is obtained by considering the atomic constraint of the form $\mathsf{beg}(S_1) - z_0 \leq sp_G(\mathsf{beg}(s_1), z_0)$ in the tightened constraint. The value $sp_G(\mathsf{beg}(s_1), z_0)$ is obtained by considering the path $\mathsf{beg}(S_1), \mathsf{end}(S_1), \mathsf{beg}(S_2), \mathsf{end}(S_2), \mathsf{end}(K_2), z_0$ of total weight $-20 + -20 + 1300 = 1260$. Other values are obtained similarly.

Checking sure-contemporaneity. Let us now explain how sure-contemporaneity (Problem 3) can be checked against the graph G_φ (corresponding to constraint φ) *in constant time*, provided that φ is tight. We start by defining the Inclusion Checking problem, which will be useful to this end. This problem checks whether the set of chronologies $[\![\varphi_1]\!]$ represented by a constraint φ_1 is included into the set of chronologies $[\![\varphi_2]\!]$ represented by another constraint φ_2:

▶ **Problem 5.** *The* Inclusion *problem asks, given two constraints φ_1 and φ_2, whether $[\![\varphi_1]\!] \subseteq [\![\varphi_2]\!]$.*

We have already seen in an example above that $\varphi_1 \preceq \varphi_2$ implies $[\![\varphi_1]\!] \subseteq [\![\varphi_2]\!]$, but that, in general, the reverse implication is not true. It becomes, however, true, when the constraints have been tightened. As a matter of fact, having φ_1 tight is sufficient (see [3] for a reference):

▶ **Proposition 6.** *For all pairs of constraints φ_1 and φ_2, the two following statements hold:*

$$\varphi_1 \preceq \varphi_2 \text{ implies } [\![\varphi_1]\!] \subseteq [\![\varphi_2]\!],$$
$$\big([\![\varphi_1]\!] \subseteq [\![\varphi_2]\!] \text{ and } \varphi_1 \text{ is tight}\big) \text{ implies } \varphi_1 \preceq \varphi_2.$$

We now come back to the sure-contemporaneity problem, and show that it can be reduced to the inclusion problem. The definition of the sure-contemporaneity problem means that, for all chronologies $\mathcal{C} \in [\![\varphi]\!]$: p_1 and p_2 intersect in $\mathcal{C}$, i.e., $\ell(\mathcal{C}(p_1)) \leq u(\mathcal{C}(p_2))$ and $\ell(\mathcal{C}(p_2)) \leq u(\mathcal{C}(p_1))$. Clearly, this holds iff $[\![\varphi]\!] \subseteq [\![\varphi^{\text{sync}}_{p_1,p_2}]\!]$, where:

$$\varphi^{\text{sync}}_{p_1,p_2} = \mathsf{beg}(p_1) \leq \mathsf{end}(p_2) \wedge \mathsf{beg}(p_2) \leq \mathsf{end}(p_1). \tag{1}$$

However, $\varphi^{\text{sync}}_{p_1,p_2}$ is equivalent to $\mathsf{beg}(p_1) - \mathsf{end}(p_2) \leq 0 \wedge \mathsf{beg}(p_2) - \mathsf{end}(p_1) \leq 0$. Thus, using Proposition 6, and assuming φ is tight, we deduce that $[\![\varphi]\!] \subseteq [\![\varphi^{\text{sync}}_{p_1,p_2}]\!]$ iff φ constraints $\mathsf{beg}(p_1) - \mathsf{end}(p_2)$ and $\mathsf{beg}(p_2) - \mathsf{end}(p_1)$ to be non-positive. Thus, we obtain a constant time procedure to check the sure-contemporaneity of two periods p_1 and p_2 on the graph G_φ of a tight constraint φ:

▶ **Proposition 7.** *For all tight constraint φ on $\mathcal{P}$ (with corresponding graph $G_\varphi = (V, E, w)$), for all pairs of periods p_1 and p_2, there is a sure-contemporaneity between p_1 and p_2 in φ iff $w(\mathsf{beg}(p_1), \mathsf{end}(p_2)) \leq 0$ and $w(\mathsf{beg}(p_2), \mathsf{end}(p_1)) \leq 0$ (assuming $w(x, y) = +\infty$ when $(x, y) \notin E$).*

▶ **Example 8.** To answer our question 'has K_1 built S_2?' from the introduction, we can check whether there is a sure-contemporaneity between K_1 and S_2. By proposition 7, there is *no sure-contemporaneity* iff either $sp_{\mathsf{Chrono}}(\mathsf{beg}(K_1), \mathsf{end}(S_2)) > 0$ or $sp_{\mathsf{Chrono}}(\mathsf{beg}(S_2), \mathsf{end}(K_1)) > 0$ (see Figure 3 for Chrono). While the path $\mathsf{beg}(K_1), \mathsf{beg}(S_1), \mathsf{end}(S_1), \mathsf{beg}(S_2), \mathsf{end}(S_2)$ is indeed negative, one can check that there all paths from $\mathsf{beg}(S_2)$ to $\mathsf{end}(K_1)$ are positive (actually, $sp_{\mathsf{Chrono}}(\mathsf{beg}(S_2), \mathsf{end}(K_1)) = 80$, see Appendix A). Hence, there is no sure-contemporaneity between K_1 and S_2, so the available archaeological data does not allow to say for sure that K_1 built S_2.

Checking Possible-Contemporaneity. As with the sure-contemporaneity problem, we first rephrase the definition of the possible-contemporaneity problem using the $\varphi^{\text{sync}}_{p_1,p_2}$ constraint from equation (1). Since there must be only one chronology compatible with the constraint φ in which p_1 and p_2 intersect, and since the set of such chronologies is characterised by $\varphi^{\text{sync}}_{p_1,p_2}$ from equation (1), we have:

▶ **Lemma 9.** *For all constraints φ on set of periods $\mathcal{P}$, and all pairs of periods $p_1, p_2 \in \mathcal{P}$: there is a possible-contemporaneity between p_1 and p_2 iff $\varphi \wedge \varphi^{sync}_{p_1,p_2}$ is satisfiable.*

From this characterisation, we obtain a simple algorithm to check possible-contemporaneity on tight constraints (in the spirit of Proposition 7 for sure-contemporaneity):

▶ **Proposition 10.** *For all tight constraints φ on $\mathcal{P}$ (with corresponding graph $G_\varphi = (V, E, w)$), for all pairs of periods p_1 and p_2, there is a possible-contemporaneity between p_1 and p_2 in φ iff $w(\mathsf{end}(p_2), \mathsf{beg}(p_1)) \geq 0$ and $w(\mathsf{end}(p_1), \mathsf{beg}(p_2)) \geq 0$ (assuming $w(x, y) = +\infty$ when $(x, y) \notin E$).*

▶ **Example 11.** We come back again to the Chronoland example and our question asking whether K_1 has built S_2. We check whether there is a possible-contemporaneity between K_1 and S_2 (remember from Example 8 that there is no *sure*-contemporaneity between them).

By Proposition 10, there is *no* possible-contemporaneity iff $sp_{\mathsf{Chrono}}(\mathsf{end}(S_2), \mathsf{beg}(K_1)) < 0$ or $sp_{\mathsf{Chrono}}(\mathsf{end}(K_1), \mathsf{beg}(S_2)) < 0$. The latter holds since the path $\mathsf{end}(K_1), \mathsf{beg}(K_1), \mathsf{beg}(S_1)$, $\mathsf{end}(S_1), \mathsf{beg}(S_2)$ has weight $15 + -20 = -5$ (see Appendix A). Hence, there is *no possible-contemporaneity* between K_1 and S_2. This result is stronger than the one from Example 8 and allows one to rule out for sure that K_1 built stratum S_2.

The result of Proposition 10 is not straightforward and requires a proof. We start by giving some intuitions. First observe that we consider a constraint φ, which we assume to be satisfiable (otherwise there is trivially no possible-contemporaneity), and tight. Then, the proof is based on the following observation: taking the conjunction of φ and $\varphi^{sync}_{p_1,p_2}$ (Lemma 9) amounts to computing the graph $G' = (V, E', w')$ from $G_\varphi = (V, E, w)$ as follows (assuming $w(x, y) = +\infty$ if $(x, y) \notin E$):

- $E' = E \cup \{(\mathsf{beg}(p_2), \mathsf{end}(p_1)), (\mathsf{beg}(p_1), \mathsf{end}(p_2))\}$;
- $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) = \min\{w(\mathsf{beg}(p_2), \mathsf{end}(p_1)), 0\}$;
- $w'(\mathsf{beg}(p_1), \mathsf{end}(p_2)) = \min\{w(\mathsf{beg}(p_1), \mathsf{end}(p_2)), 0\}$;
- $w'(e) = w(e)$ for all $e \in E\{(\mathsf{beg}(p_2), \mathsf{end}(p_1)), (\mathsf{beg}(p_1), \mathsf{end}(p_2))\}$.

Thus, G' is obtained from G_φ by setting the weights of $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$ and $(\mathsf{beg}(p_1), \mathsf{end}(p_2))$ to 0 if they had non-negative weights in G_φ. One can check that the constraint corresponding to G' is equivalent to $\varphi \wedge \varphi^{\text{sync}}_{p_1,p_2}$, so there is a possible-contemporaneity between p_1 and p_2 iff there is no negative cycle in G', by Lemma 9.

Now assume that $\varphi \wedge \varphi^{\text{sync}}_{p_1,p_2}$ is *not satisfiable* and thus G' contains a negative cycle. Since φ is tight and satisfiable, G_φ contains no negative cycle, hence the negative cycle in G' comes necessarily from the modification we have performed on $(\mathsf{beg}(p_1), \mathsf{end}(p_2))$ and $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$. Thus, there is, in G', at least one negative cycle that contains $(\mathsf{beg}(p_1), \mathsf{end}(p_2))$ or $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$. For the sake of the discussion, let us assume the negative cycle contains $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$ and not $(\mathsf{beg}(p_1), \mathsf{end}(p_2))$, hence, the only reason for this negative cycle to exist in G' is because we have set $w(\mathsf{beg}(p_2), \mathsf{end}(p_1))$ to 0, and thus, there is a negative *path* from $\mathsf{end}(p_1)$ to $\mathsf{beg}(p_2)$ in G' and G_φ. Since φ is tight, this implies that $(\mathsf{end}(p_1), \mathsf{beg}(p_2))$ exists and has negative weight in G_φ. Conversely, if $w(\mathsf{end}(p_1), \mathsf{beg}(p_2)) < 0$, then, there is necessarily a negative cycle containing $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$ in G', since $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) \leq 0$, by construction. This explains intuitively why checking that both $w(\mathsf{end}(p_2), \mathsf{beg}(p_1))$ and $w(\mathsf{end}(p_1), \mathsf{beg}(p_2))$ are non-negative is necessary and sufficient to check possible-contemporaneity. This can be performed in $O(1)$ on *tight* (and satisfiable) constraints. A formal proof of the proposition is given in Appendix B.

4 Conclusion and future works

This paper presents a theoretical framework for the modelling of chronological constraints relevant to archaeological research. Within this framework, algorithms have been presented to solve four basic chronological problems (satisfiability, tightening, sure- and possible-contemporaneity) that are usually addressed in a non-formal way by the archaeological community. As shown here for a toy example featuring only two kings and two archaeological strata (Figure 1), solving these problems manually is tedious and error-prone, especially when it comes to obtaining optimal bounds on dates and duration (tightening). On real-life archaeological cases, featuring dozens of periods and synchronisms, obtaining reliable and optimal results is virtually impossible without the help of a formal computational approach as the one advocated for here.

We further contend that the application of our algorithms to real-life archaeological test-cases might provide significant advances to current chronological debates, through the detection of (yet unnoticed) unsatisfiable sets of chronological constraints, and through the computation of improved chronological estimates through our tightening procedure. An automated technique also allows archaeologist to test quickly the implications of new hypotheses on chronological models.

The next steps of this research are therefore both practical and theoretical. On the *practical* side, we wish to develop a comprehensive methodology for chronological research in archaeology, including a user-friendly tool that will allow archaeologists to specify and manipulate their own chronological models, through a dedicated high-level constraint language. We also want to apply this methodology to concrete case studies from the archaeological literature. On the *theoretical* side, it is clear that the development of concrete case studies will raise new theoretical questions. One such questions we can already mention is 'how to provide the archaeological user with a meaningful *witness* of non-satisfiability?' when a constraint is found to be unsatisfiable. In such as case, the user would wish to understand why the system is unsatisfiable, and also be provided with hints as to how the system could be rendered satisfiable through the removal (or relaxing) of a limited number of constraints.

Acknowledgements. Eythan Levy was supported by the Center for Absorption in Science, the Ministry of Absorption, the State of Israel.

References

1 Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994. `doi:10.1016/0304-3975(94)90010-8`.

2 Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on Uppaal. In *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, September 2004.

3 Johan Bengtsson. *Clocks, DBM, and States in Timed Systems.* PhD thesis, Uppsala University, 2002.

4 Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer, 2003. `doi:10.1007/978-3-540-27755-2_3`.

5 Patricia Bouyer, Maximilien Colange, and Nicolas Markey. Symbolic optimal reachability in weighted timed automata. In *CAV'16*, volume 9779 of *LNCS*, pages 513–530. Springer, 2016. `doi:10.1007/978-3-319-41528-4`.

6 David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems, International Workshop, Grenoble, France, June 12-14, 1989, Proceedings*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer Verlag, 1989. `doi:10.1007/3-540-52148-8_17`.

7 E. C. Harris. *Principles of Archaeological Stratigraphy.* Academic Press, New York. NY, 1979.

8 Felix Höflmayer. The date of the Minoan Santorini eruption: Quantifying the "offset". *Radiocarbon*, 54:435–448, 2012.

9 Mads Kähler Holst. Complicated relations and blind dating: Formal analysis of relative chronological structures. In Caitlin E. Buck and Andrew R. Millard, editors, *Tools for Constructing Chronologies*, pages 129–147. Springer, London, 2004.

10 Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1):1–13, 1977. `doi:10.1145/321992.321993`.

11 Alfred H. Kromholz. Business and industry in archaeology. In Paul Ahström, editor, *High, Middle or Low? (Part 1)*, pages 119–137. Paul Ahströms Förlag, Gothenburg, 1987.

12 A. Madkour, W. G. Aref, F. U. Rehman, M. Abdur Rahman, and S. Basalamah. A Survey of Shortest-Path Algorithms. Technical Report CoRR abs/1705.02044, Cornell University Library, arXiv.org, 2017. URL: `https://arxiv.org/abs/1705.02044`.

13 Amihai Mazar. The debate over the chronology of the Iron Age in the southern Levant. In *The Bible and Radiocarbon Dating, Archaeology, Text and Science*, pages 15–30, 2005.

14 Itay Meiri. Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence*, 87(1):343–385, 1996. `doi:10.1016/0004-3702(95)00109-3`.

15 Christopher Bronk Ramsey. Radiocarbon calibration and analysis of stratigraphy: the OxCal program. *Radiocarbon*, 37(2):425–430, 1995.

16 Ilan Sharon. Partial order scalogram analysis of relations – a mathematical approach to the analysis of stratigraphy. *Journal of Archaeological Science*, 22:751–767, 1995.

17 Robert E. Shostak. Deciding linear inequalities by computing loop residues. *J. ACM*, 28(4):769–779, 1981. `doi:10.1145/322276.322288`.

A Shortest paths in the Chronoland example

The all-pairs shortest path matrix of the Chronoland example (graph in Figure 3) is given in Figure 4. The entry in row i column j gives the weight of the shortest path from i to j.

B Proof of Proposition 10

Statement of Proposition 10: *For all tight constraints φ on $\mathcal{P}$ (with corresponding graph $G_\varphi = (V, E, w)$), for all pairs of periods p_1 and p_2, there is a possible-contemporaneity between p_1 and p_2 in φ iff $w(\mathsf{end}(p_2), \mathsf{beg}(p_1)) \geq 0$ and $w(\mathsf{end}(p_1), \mathsf{beg}(p_2)) \geq 0$ (assuming $w(x, y) = +\infty$ when $(x, y) \notin E$).*

Proof. We prove both directions of the iff. First let us show that a possible-contemporaneity between p_1 and p_2 implies $w(\mathsf{end}(p_2), \mathsf{beg}(p_1)) \geq 0$ and $w(\mathsf{end}(p_1), \mathsf{beg}(p_2)) \geq 0$. We prove the contraposition, i.e., if either $w(\mathsf{end}(p_2), \mathsf{beg}(p_1)) < 0$ or $w(\mathsf{end}(p_1), \mathsf{beg}(p_2)) < 0$, then there is no possible-contemporaneity between p_1 and p_2.

Assume that $w(\mathsf{end}(p_2), \mathsf{beg}(p_1)) < 0$ (the case $w(\mathsf{end}(p_1), \mathsf{beg}(p_2)) < 0$ is symmetrical), and consider the graph $G' = \langle V, E', w' \rangle$ obtained, as described above. By definition of G',

$$\begin{array}{r|ccccccccc|}
 & z_0 & \mathsf{beg}(S_1) & \mathsf{end}(S_1) & \mathsf{beg}(S_2) & \mathsf{end}(S_2) & \mathsf{beg}(K_1) & \mathsf{end}(K_1) & \mathsf{beg}(K_2) & \mathsf{end}(K_2) \\
z_0 & 0 & -1,200 & -1,220 & -1,220 & -1,240 & -1,200 & -1,200 & -1,200 & -1,240 \\
\mathsf{beg}(S_1) & 1,260 & 0 & -20 & -20 & -40 & 15 & 0 & 0 & -40 \\
\mathsf{end}(S_1) & 1,280 & 80 & 0 & 0 & -20 & 80 & 80 & 80 & -20 \\
\mathsf{beg}(S_2) & 1,280 & 80 & 0 & 0 & -20 & 80 & \mathbf{80} & 80 & -20 \\
\mathsf{end}(S_2) & 1,300 & 100 & 80 & 80 & 0 & \boxed{100} & 100 & 100 & 0 \\
\mathsf{beg}(K_1) & 1,260 & 0 & -20 & -20 & \mathbf{-40} & 0 & 0 & 0 & -40 \\
\mathsf{end}(K_1) & 1,270 & 15 & -5 & \boxed{-5} & -25 & 15 & 0 & 0 & -30 \\
\mathsf{beg}(K_2) & 1,270 & 15 & -5 & -5 & -25 & 15 & 0 & 0 & -30 \\
\mathsf{end}(K_2) & 1,300 & 100 & 80 & 80 & 60 & 100 & 100 & 100 & 0
\end{array}$$

Figure 4 The all-pairs shortest paths for the Chronoland example. **Bold** numbers highlight the values referred to in Example 8 where sure-contemporaneity between K_1 and S_2 is checked (and found not to hold). $\boxed{\text{Boxed}}$ numbers highlight the values referred to in Example 11, where possible-contemporaneity between K_1 and S_2 is checked (and found not to hold either).

and since $w(\mathsf{end}(p_2), \mathsf{beg}(p_1)) < 0$: $w'(\mathsf{end}(p_2), \mathsf{beg}(p_1)) = w(\mathsf{end}(p_2), \mathsf{beg}(p_1)) < 0$. Let us show that the cycle

$$\mathsf{end}(p_2), \mathsf{beg}(p_1), \mathsf{end}(p_1), \mathsf{beg}(p_2), \mathsf{end}(p_2)$$

is negative in G'. Indeed, we have:

$$\begin{aligned}
&w'(\mathsf{beg}(p_1), \mathsf{end}(p_1)) \leq 0 && \text{True in all constraints: the beginning occurs before the end}\\
&w'(\mathsf{beg}(p_2), \mathsf{end}(p_2)) \leq 0 && \text{Same argument}\\
&w'(\mathsf{end}(p_1), \mathsf{beg}(p_2)) \leq 0 && \text{By construction of } G'\\
&w'(\mathsf{end}(p_2), \mathsf{beg}(p_1)) < 0 && \text{By the above arguments.}
\end{aligned}$$

Hence, the weight of the $\mathsf{end}(p_2), \mathsf{beg}(p_1), \mathsf{end}(p_1), \mathsf{beg}(p_2), \mathsf{end}(p_2)$ cycle is indeed negative. However, by construction G' is the graph that corresponds to $\varphi \wedge \varphi^{\text{sync}}_{p_1,p_2}$, hence this constraint is not satisfiable. By Lemma 9, there is no possible-contemporaneity between p_1 and p_2 in φ.

For the other direction, let us show that:

$$w(\mathsf{end}(p_2), \mathsf{beg}(p_1)) \geq 0 \text{ and } w(\mathsf{end}(p_1), \mathsf{beg}(p_2)) \geq 0 \tag{2}$$

implies that there is a possible-contemporaneity between p_1 and p_2 in φ. Following Lemma 9, we show that (2) implies that $\varphi \wedge \varphi^{\text{sync}}_{p_1,p_2}$ is satisfiable. To show this, we rely again on the graph G' described above: we proceed by contradiction and assume that (2) holds but that G' contains a negative cycle. We consider several cases:

1. The negative cycle contains neither the $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$, nor the $(\mathsf{beg}(p_1), \mathsf{end}(p_2))$ edge. Since these two edges are the only ones that have been modified when building G' from G_φ, we conclude that the negative cycle is already present in G_φ. This is a contradiction since we have assumed that φ is satisfiable.
2. The negative cycle contains only edge among $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$ and $(\mathsf{beg}(p_1), \mathsf{end}(p_2))$. Wlog, we assume that the negative cycle of G' contains $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$, i.e. it is of the form:

$$\mathsf{beg}(p_2) \xrightarrow{w'(\mathsf{beg}(p_2),\mathsf{end}(p_1))} \underbrace{\mathsf{end}(p_1) \xrightarrow{w_1} v_1 \xrightarrow{w_2} \cdots \xrightarrow{w_n} v_n \xrightarrow{w_{n+1}} \mathsf{beg}(p_2)}_{\pi}$$

where $w_1 = w'(\mathsf{end}(p_1), v_1)$, $w_i = w'(v_{i-1}, v_i)$ for all $2 \leq i \leq n$, $w_{n+1} = w'(v_n, \mathsf{beg}(p_2))$, and neither $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$, nor $(\mathsf{beg}(p_1), \mathsf{end}(p_2))$ occur in π. Since this cycle is negative in G':

$$w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) + \sum_{i=1}^{n+1} w_i < 0.$$

Recall that, by construction of G': $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) \leq 0$. We consider two further sub-cases:

   a. If $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) = 0$, then $\sum_{i=1}^{n} w_i < 0$, i.e., the total weight of π is non-positive. However, since all edges occurring in π occur with the same weight in G_φ, and since π starts in $\mathsf{end}(p_1)$ and ends in $\mathsf{beg}(p_2)$, we conclude that the shortest path between $\mathsf{end}(p_1)$ and $\mathsf{beg}(p_2)$ is < 0 in G_φ. Since φ is assumed to be tight, this implies that $w(\mathsf{end}(p_1), \mathsf{beg}(p_2)) < 0$, which is a contradiction with our hypothesis (2).
   b. If $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) < 0$, then by construction of G', this edge had the same weight in G_φ, i.e., $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) = w(\mathsf{beg}(p_2), \mathsf{end}(p_1)) < 0$. Since all the edges in π were also present in G_φ with the same weight, we conclude that the negative cycle we have identified in G' is also present in G_φ. This is a contradiction since we have assumed that φ is satisfiable.

3. The negative cycle contains both $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$ and $(\mathsf{beg}(p_1), \mathsf{end}(p_2))$, i.e. it is of the form:

$$\mathsf{beg}(p_1) \xrightarrow{w'(\mathsf{beg}(p_1),\mathsf{end}(p_2))} \underbrace{\mathsf{end}(p_2) \xrightarrow{w_1} v_1 \xrightarrow{w_2} \cdots \xrightarrow{w_n} v_n \xrightarrow{w_{n+1}} \mathsf{beg}(p_2)}_{\pi_1}$$
$$\xrightarrow{w'(\mathsf{beg}(p_2),\mathsf{end}(p_1))} \underbrace{\mathsf{end}(p_1) \xrightarrow{w'_1} v'_1 \xrightarrow{w'_2} \cdots \xrightarrow{w'_\ell} v'_\ell \xrightarrow{w'_{\ell+1}} \mathsf{beg}(p_1)}_{\pi_2}$$

where $w_1 = w'(\mathsf{end}(p_2), v_1)$; for all $2 \le i \le n$: $w_i = w'(v_{i-1}, v_i)$; $w_{n+1} = w'(v_n, \mathsf{beg}(p_2))$; $w'_1 = w'(\mathsf{end}(p_1), v'_1)$; for all $1 \le i \le \ell$: $w'_i = w'(v'_{i-1}, v'_i)$; $w'_{\ell+1} = w'(v'_\ell, \mathsf{beg}(p_1))$; and $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$, $(\mathsf{beg}(p_1), \mathsf{end}(p_2))$ occur neither in π_1 nor in π_2. Since this cycle is negative in G', we have:

$$w'(\mathsf{beg}(p_1), \mathsf{end}(p_2)) + \sum_{i=1}^{n+1} w_i + w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) + \sum_{i=1}^{\ell+1} w'_i < 0 \tag{3}$$

Since, by construction of G': $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) \le 0$ and $w'(\mathsf{beg}(p_1), \mathsf{end}(p_2)) \le 0$, we consider four further sub-cases:

a. First, $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) = w'(\mathsf{beg}(p_1), \mathsf{end}(p_2)) = 0$. Then, (3) yields:

$$\sum_{i=1}^{n+1} w_i + \sum_{i=1}^{\ell+1} w'_i < 0.$$

Hence, one of these two sums must be < 0. Wlog, let us assume $\sum_{i=1}^{n+1} w_i < 0$, i.e., the weight of π_1 is non-positive (the arguments carry on when the overall weight of π_2 is non-positive instead). Since all the edges of π_1 were already present in G with the same weights, we conclude that the shortest path from $\mathsf{end}(p_2)$ to $\mathsf{beg}(p_2)$ is < 0 in G. Since we have assumed that φ is tight, this implies that $w(\mathsf{end}(p_2), \mathsf{beg}(p_2)) < 0$, hence, φ contains an atomic constraint of the form $\mathsf{end}(p_2) - \mathsf{beg}(p_2) \le c$ from some $c < 0$. However, this renders φ unsatisfiable, because all constraints imply that $\mathsf{beg}(p) - \mathsf{end}(p) \le 0$ for all periods p. Contradiction.

b. Second $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) < 0$ and $w'(\mathsf{beg}(p_1), \mathsf{end}(p_2)) = 0$. Then, (3) yields:

$$\sum_{i=1}^{n+1} w_i + w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) + \sum_{i=1}^{\ell+1} w'_i < 0. \tag{4}$$

However, by construction of G', $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) < 0$ implies that the edge $(\mathsf{beg}(p_2), \mathsf{end}(p_1))$ was already in G with the same weight, i.e. $w(\mathsf{beg}(p_2), \mathsf{end}(p_1)) < 0$. Moreover, all edges in π_1 and π_2 are also present in G with the same weight as in G'. Thus, we conclude from (4) that the

$$\mathsf{end}(p_2), v_1, \ldots, v_n, \mathsf{beg}(p_2), \mathsf{end}(p_1), v'_1, \ldots, v'_\ell, \mathsf{beg}(p_1)$$

path exists in G with non-positive weight. Hence, the shortest path from $\mathsf{end}(p_2)$ to $\mathsf{beg}(p_1)$ in G has weight < 0. Since we have assumed that φ is tight, we conclude that $w(\mathsf{end}(p_2), \mathsf{beg}(p_1)) < 0$, which contradict our hypothesis (2). Contradiction.

c. Third, $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) = 0$ and $w'(\mathsf{beg}(p_1), \mathsf{end}(p_2)) < 0$ is treated as the previous case.

d. Finally, when $w'(\mathsf{beg}(p_2), \mathsf{end}(p_1)) < 0$ and $w'(\mathsf{beg}(p_1), \mathsf{end}(p_2)) < 0$, we conclude that all the edges in the negative cycle we have identified are already present in G with the same weight, i.e., the $\mathsf{beg}(p_1), \mathsf{end}(p_2), v_1, \ldots, v_n, \mathsf{beg}(p_2), \mathsf{end}(p_1), v'_1, \ldots, v'_\ell, \mathsf{beg}(p_1)$ cycle has negative weight in G, hence φ is not satisfiable. Contradiction. ◀

CTL with Finitely Bounded Semantics*

Valentin Goranko[1], Antti Kuusisto[2], and Raine Rönnholm[3]

1 **Stockholm University, Stockholm, Sweden; and**
University of Johannesburg, Johannesburg, South Africa†
`valentin.goranko@philosophy.su.se`
2 **University of Bremen, Bremen, Germany**
`kuusisto@uni-bremen.de`
3 **University of Tampere, Tampere, Finland**
`raine.ronnholm@uta.fi`

Abstract

We consider a variation of the branching time logic CTL with non-standard, "finitely bounded" semantics (FBS). FBS is naturally defined as game-theoretic semantics where the proponent of truth of an eventuality must commit to a time limit (number of transition steps) within which the formula should become true on all (resp. some) paths starting from the state where the formula is evaluated. The resulting version $\mathrm{CTL}_{\mathbf{FB}}$ of CTL differs essentially from the standard one as it no longer has the finite model property.

We develop two tableaux systems for $\mathrm{CTL}_{\mathbf{FB}}$. The first one deals with infinite sets of formulae, whereas the second one deals with finite sets of formulae in a slightly extended language allowing explicit indication of time limits in formulae. We prove soundness and completeness of both systems and also show that the latter tableaux system provides an EXPTIME decision procedure for it and thus prove EXPTIME-completeness of the satisfiability problem.

1998 ACM Subject Classification F.4.1 Mathematical Logic, I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases CTL, finitely bounded semantics, tableaux, decidability

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.14

1 Introduction

The branching time logic CTL ([4]) is interpreted in transition systems and its language involves quantification over all infinite paths (computations) starting at the current state. It is well known that in order to determine truth of any CTL formula in a *finite* transition system, it suffices to consider only sufficiently long finite prefixes of paths starting at the current state, in the sense that any existential (resp. universal) eventuality is satisfied at that state iff it is satisfied on some (resp. all) of these finite prefixes. Furthermore, satisfiability/validity in CTL has the finite model property. It is natural, therefore, to consider a finitary version of the semantics of CTL, restricted on all models to finite paths. Such a version has recently emerged as a compositional counterpart of the *finitely bounded game-theoretic semantics* (FBS), developed for the multi-agent extension ATL of CTL in [6] (see also [7] for an extended and revised version), where the length of the evaluation game is constrained by finite time limits which the players must set and decrease after every transition move in the game.

* The work of Valentin Goranko was supported by a research grant 2015-04388 of the Swedish Research Council. The work of Antti Kuusisto was supported by the ERC grant 647289 "CODA."
† Visiting professorship.

Intuitively, the main distinctive feature of the FBS is that a player who claims an eventuality formula to be true must commit to a time bound (number of transition steps) within which she can defend that claim by fulfilling that eventuality. Technically, the FBS for CTL can be obtained by changing the definition of temporal operators so that a uniform bound on the number of transition steps needed to fulfill a given eventuality is imposed.

In this paper we study the alternative version $\text{CTL}_{\mathbf{FB}}$ of CTL, defined by the FBS. $\text{CTL}_{\mathbf{FB}}$ differs essentially from CTL with standard semantics. In particular, it falsifies the fundamental fixed-point characterizations of the operators EG and AU. Based on this we show that $\text{CTL}_{\mathbf{FB}}$ lacks the finite model property and that the set of validities of $\text{CTL}_{\mathbf{FB}}$ is a proper subset of the validities of CTL.

We develop two equivalent versions of tableaux for $\text{CTL}_{\mathbf{FB}}$. The first tableaux deals with infinite sets of CTL formulae, while the second one involves only finite sets of formulae, but in a suitably extended language which allows explicit indication of *time limits* by corresponding *indexing parameters*. Essentially, the parameters enable encoding infinitary formulae by finite ones. We prove soundness and completeness of both tableau systems and also show that the latter system provides a decision procedure for $\text{CTL}_{\mathbf{FB}}$. We thereby establish that satisfiability in that logic is decidable and EXPTIME-complete.

We note that even though not being the actual motivation for the present study, a major independent justification for considering bounded semantics for CTL is (the conceptual idea behind) *bounded model checking* [2], which provides the main link with related previous work. Other versions of bounded semantics for CTL, based on evaluation of formulae on finite paths, have been considered in the context of bounded model checking in e.g. [8], [9], [10].

2 Preliminaries: CTL with finitely bounded semantics

Here we only provide brief preliminaries on CTL. For further details see e.g. [5, Ch.7].

2.1 The standard semantics of CTL

▶ **Definition 1.** A **transition system** is a tuple $\mathcal{T} = (\mathrm{S}, \mathrm{R})$, where S is a nonempty set of **states** and $\mathrm{R} \subseteq \mathrm{S} \times \mathrm{S}$ a **transition relation**. We also assume that R is *serial*, i.e. for every $s \in \mathrm{S}$ there is $s' \in \mathrm{S}$ such that $(s, s') \in \mathrm{R}$. A **path** in $\mathcal{T}$ is a sequence $\lambda : \mathbb{N} \to \mathrm{S}$ of states such that $(\lambda(n), \lambda(n{+}1)) \in \mathrm{R}$ for every $n \in \mathbb{N}$.

An **interpreted transition system** (ITS) over $\mathcal{T}$ is a tuple $\mathcal{M} = (\mathrm{S}, \mathrm{R}, \Phi, \ell)$, where Φ a set of **proposition symbols** and $\ell : \mathrm{S} \to \mathcal{P}(\Phi)$ is a **state description function** defining for every state s the set of atomic propositions true at that state.

The syntax CTL is given by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{EX}\,\varphi \mid \mathsf{E}(\varphi\,\mathsf{U}\,\varphi) \mid \mathsf{A}(\varphi\,\mathsf{U}\,\varphi)\,.$$

We also use the following abbreviations $\mathsf{AX}\,\varphi := \neg\mathsf{EX}\,\neg\varphi$, $\mathsf{EF}\,\varphi := \mathsf{E}(\top\,\mathsf{U}\,\varphi)$, $\mathsf{AF}\,\varphi := \mathsf{A}(\top\,\mathsf{U}\,\varphi)$, $\mathsf{EG}\,\varphi := \neg\mathsf{AF}\,\neg\varphi$ and $\mathsf{AG}\,\varphi := \neg\mathsf{EF}\,\neg\varphi$.

▶ **Definition 2.** Let $\mathcal{M} = (\mathrm{S}, \mathrm{R}, \Phi, \ell)$ be an interpreted transition system, $s \in \mathrm{S}$ and φ a CTL-formula. **Truth of** φ **at** s **in** $\mathcal{M}$, denoted by $\mathcal{M}, s \models \varphi$, is defined as follows:

- $\mathcal{M}, s \models p$ iff $p \in \ell(s)$.
- $\mathcal{M}, s \models \neg\varphi$ iff $\mathcal{M}, s \not\models \varphi$.
- $\mathcal{M}, s \models \varphi \vee \psi$ iff $\mathcal{M}, s \models \varphi$ or $\mathcal{M}, s \models \psi$.
- $\mathcal{M}, s \models \mathsf{EX}\,\varphi$ iff there is a state s' such that $(s, s') \in \mathrm{R}$ and $\mathcal{M}, s' \models \varphi$.

- $\mathcal{M}, s \models \mathsf{E}(\varphi \,\mathsf{U}\, \psi)$ iff there is a path λ starting from s and $i \geq 0$ such that $\mathcal{M}, \lambda(i) \models \psi$ and $\mathcal{M}, \lambda(j) \models \varphi$ for every $j < i$.
- $\mathcal{M}, s \models \mathsf{A}(\varphi \,\mathsf{U}\, \psi)$ iff for every path λ starting from s, there is $i \geq 0$ such that $\mathcal{M}, \lambda(i) \models \psi$ and $\mathcal{M}, \lambda(j) \models \varphi$ for every $j < i$.

We define the following operators on formulae, where $\mathsf{Q} \in \{\mathsf{E}, \mathsf{A}\}$:

$$\mathbf{G}_{\mathsf{Q};\theta}(\varphi) := \theta \wedge \mathsf{QX}\,\varphi; \quad \mathbf{U}_{\mathsf{Q};\psi,\theta}(\varphi) := \theta \vee (\psi \wedge \mathsf{QX}\,\varphi)\,.$$

It is well-known that, for each $\mathsf{Q} \in \{\mathsf{E}, \mathsf{A}\}$, in terms of their semantics:

- $\mathsf{QG}\,\theta$ is the greatest fixpoint of the operator $\mathbf{G}_{\mathsf{Q};\theta}$, i.e. $\mathsf{QG}\,\theta \equiv \nu Z.\mathbf{G}_{\mathsf{Q};\theta}(Z)$,
- $\mathsf{Q}(\psi \,\mathsf{U}\, \theta)$ the least fixpoint of the operator $\mathbf{U}_{\mathsf{Q};\psi,\theta}$, i.e. $\mathsf{Q}(\psi \,\mathsf{U}\, \theta) \equiv \mu Z.\mathbf{U}_{\mathsf{Q};\psi,\theta}(Z)$.

Now, we define recursively on $n \in \mathbb{N}$ the respective iterations of these operators:

- $\mathbf{G}^0_{\mathsf{Q}}(\theta) := \theta;\ \ \mathbf{G}^{n+1}_{\mathsf{Q}}(\theta) := \mathbf{G}_{\mathsf{Q};\theta}(\mathbf{G}^n_{\mathsf{Q}}(\theta))$
- $\mathbf{U}^0_{\mathsf{Q}}(\psi, \theta) := \theta;\ \ \mathbf{U}^{n+1}_{\mathsf{Q}}(\psi, \theta) := \mathbf{U}_{\mathsf{Q};\psi,\theta}(\mathbf{U}^n_{\mathsf{Q}}(\psi, \theta))$.

2.2 Game-Theoretic semantics for CTL

In [7] we have defined game-theoretic semantics (GTS) for the alternating time temporal logic ATL ([1], [5, Ch.9]) by using *evaluation games* between two players, Abelard and Eloise. Since CTL can be regarded as a (1-agent) fragment of ATL, a simple GTS for CTL can be obtained from the mentioned GTS for ATL in a straightforward (but not immediate) way.

▶ **Definition 3.** Let $\mathcal{M} = (\mathrm{S}, \mathrm{R}, \Phi, \ell)$ be an ITS, $s_{in} \in \mathrm{S}$ and φ a CTL-formula. The **(unbounded) evaluation game** $\mathcal{G}(\mathcal{M}, s_{in}, \varphi)$ is defined as follows. A **position** of the game is a tuple $(\mathbf{P}, s, \psi)$ where $\mathbf{P} \in \{\mathrm{Abelard}, \mathrm{Eloise}\}$, $s \in \mathrm{S}$ and ψ is a subformula of φ. The **initial position** of the game is $(\mathrm{Eloise}, s_{in}, \varphi)$. The evaluation game proceeds according to the following rules.

1. A position of the form $(\mathbf{P}, s, p)$, where $p \in \Phi$, is called an **ending position**. If $p \in \ell(s)$, then $\mathbf{P}$ wins the evaluation game. Else the **opposing player of P**, denoted by $\overline{\mathbf{P}}$, wins.
2. In $(\mathbf{P}, s, \neg\psi)$ the game moves to the next position $(\overline{\mathbf{P}}, s, \psi)$.
3. In $(\mathbf{P}, s, \psi \vee \theta)$ the player $\mathbf{P}$ decides whether the next position is $(\mathbf{P}, s, \psi)$ or $(\mathbf{P}, s, \theta)$.
4. In $(\mathbf{P}, s, \mathsf{EX}\,\psi)$ the player $\mathbf{P}$ may choose any state s' such that $(s, s') \in \mathrm{R}$ and the next position is $(\mathbf{P}, s', \psi)$.

For the rules for the formulae $\mathsf{E}(\psi \,\mathsf{U}\, \theta)$ and $\mathsf{A}(\psi \,\mathsf{U}\, \theta)$, we define the **embedded game** $\mathbf{G} := \mathbf{g}(\mathbf{V}, \mathbf{L}, s_0, \psi_{\mathbf{V}}, \psi_{\overline{\mathbf{V}}})$, where $\mathbf{V}, \mathbf{L} \in \{\mathit{Abelard}, \mathit{Eloise}\}$, s_0 is a state, and $\psi_{\mathbf{V}}$ and $\psi_{\overline{\mathbf{V}}}$ are formulae. The player $\mathbf{V}$ is called the **verifier** (of the embedded game) and $\mathbf{L}$ the **leader**. These players may, but need not be, the same. We let $\overline{\mathbf{V}}$ and $\overline{\mathbf{L}}$ denote the opponents of $\mathbf{V}$ and $\mathbf{L}$, respectively. The embedded game $\mathbf{G}$ starts from the **initial state** s_0 and proceeds from any state s according to the following rules until an **exit position** is reached.

(i) $\mathbf{V}$ may end the game at the exit position $(\mathbf{V}, s, \psi_{\mathbf{V}})$.
(ii) $\overline{\mathbf{V}}$ may end the game at the exit position $(\mathbf{V}, s, \psi_{\overline{\mathbf{V}}})$.
(iii) $\mathbf{L}$ may select any state s' such that $(s, s') \in \mathrm{R}$ and then $\mathbf{G}$ is continued from s'.

If the embedded game $\mathbf{G}$ continues an infinite number of rounds, the verifier $\mathbf{V}$ loses the entire evaluation game. The rest of the rules for the evaluation game are defined as follows:

5. In $(\mathbf{P}, s, \mathsf{E}(\psi \,\mathsf{U}\, \theta))$ the game is continued from the exit position of $\mathbf{g}(\mathbf{P}, \mathbf{P}, s, \theta, \psi)$.
6. In $(\mathbf{P}, s, \mathsf{A}(\psi \,\mathsf{U}\, \theta))$ the game is continued from the exit position of $\mathbf{g}(\mathbf{P}, \overline{\mathbf{P}}, s, \theta, \psi)$.

In GTS, truth of a formula is defined as existence of a winning strategy for Eloise in the corresponding evaluation game. By [7] we obtain the following equivalence:

$$\mathcal{M}, s \models \varphi \text{ iff } \ \text{Eloise has a winning strategy in } \mathcal{G}(\mathcal{M}, s, \varphi)\,.$$

2.3 Finitely bounded semantics for CTL

Note that unbounded evaluation games may continue infinitely long before a winner is determined. In order to avoid this, we have also defined **bounded evaluation games** for ATL in [7]. Such a game is obtained by modifying the unbounded game simply by attaching ordinal values, called **time limits**, to the embedded games. A time limit is announced by the verifier in the beginning of the embedded game and the verifier has to decrease it after every transition. Since ordinals are *well-founded*, it is guaranteed that the whole bounded evaluation game ends in a finite number of moves – even in infinite models.

If players are allowed to announce arbitrarily large time limits in the game, then unbounded and bounded evaluation games become equivalent[1]. A particularly interesting and natural variant of a bounded evaluation game is when only *finite* ordinals may be used by the players. We call the corresponding game-theoretic semantics **finitely bounded (FBS)** and denote its truth condition by $\models_{\mathrm{fb}}$. As shown in [7], the FBS for ATL differs from the standard compositional semantics but corresponds to a natural variant of it, to be discussed further. In the special case of CTL, this semantics modifies only the truth conditions of AU and EU so that a uniform bound on the number of transition steps needed to fulfill a given eventuality is imposed.

($\mathsf{AU}_{\mathrm{fb}}$) $\mathcal{M}, s \models_{\mathrm{fb}} \mathsf{A}(\varphi \,\mathsf{U}\, \psi)$ iff there is $n \in \mathbb{N}$ such that for every path λ starting from s, there is $i \leq n$ such that $\mathcal{M}, \lambda(i) \models_{\mathrm{fb}} \psi$ and $\mathcal{M}, \lambda(j) \models_{\mathrm{fb}} \varphi$ for every $j < i$.

($\mathsf{EU}_{\mathrm{fb}}$) $\mathcal{M}, s \models_{\mathrm{fb}} \mathsf{E}(\varphi \,\mathsf{U}\, \psi)$ iff there is $n \in \mathbb{N}$, a path λ starting from s and $i \leq n$ such that $\mathcal{M}, \lambda(i) \models_{\mathrm{fb}} \psi$ and $\mathcal{M}, \lambda(j) \models_{\mathrm{fb}} \varphi$ for every $j < i$. (We note that since existential quantifiers commute, ($\mathsf{EU}_{\mathrm{fb}}$) is in fact equivalent to the standard truth definition of EU.)

Thus, the FBS of formulae of the type $\mathsf{E}(\varphi \,\mathsf{U}\, \psi)$, is standard, even though they will not be treated here as (existential) eventualities usually are, see Section 3.1.1. Furthermore, it is easy to see that the FBS given to $\mathsf{A}(\varphi \,\mathsf{U}\, \psi)$ by ($\mathsf{AU}_{\mathrm{fb}}$) is not equivalent to the standard one, and in fact, such formulae *do not behave like universal eventualities*, as they would in standard CTL. Respectively, the derived FBS for AG is equivalent to the standard one, while for EG we obtain the following non-equivalent version.

($\mathsf{EG}_{\mathrm{fb}}$) $\mathcal{M}, s \models_{\mathrm{fb}} \mathsf{EG}\,\varphi$ iff for every $n \in \mathbb{N}$, there is a path λ_n starting from s such that $\mathcal{M}, \lambda_n(i) \models_{\mathrm{fb}} \varphi$ for every $i \leq n$. (Note that the path λ_n depends on n.)

By replacing the truth condition for AU (and EG) with the ones above, we obtain CTL **with finitely bounded semantics**, denoted by $\mathrm{CTL}_{\mathbf{FB}}$.

For a set of formulae Γ, by $\mathcal{M}, s \models_{\mathrm{fb}} \Gamma$ we denote the claim that $\mathcal{M}, s \models_{\mathrm{fb}} \varphi$ for all $\varphi \in \Gamma$. Satisfiability and validity of $\mathrm{CTL}_{\mathbf{FB}}$ formulae are defined and denoted as usual.

2.4 Some properties of $\mathrm{CTL}_{\mathbf{FB}}$

All observations made for the finitely bounded semantics of ATL in [7] apply directly to $\mathrm{CTL}_{\mathbf{FB}}$. Here are the most important and interesting ones.

1. On all *image finite models* (where every state has finitely many immediate successors) $\mathrm{CTL}_{\mathbf{FB}} = \mathrm{CTL}$, i.e., truth of CTL-formulae is independent of which semantics is used.
2. $\mathrm{CTL} \neq \mathrm{CTL}_{\mathbf{FB}}$ in models that have infinite branchings. In particular, the fixed point properties of the operators F and G fail since the implications $\mathsf{EG}\,p \to (p \wedge \mathsf{EX}\,\mathsf{EG}\,p)$ and (dually) $(p \vee \mathsf{AX}\,\mathsf{AF}\,p) \to \mathsf{AF}\,p$ are valid in CTL but not in $\mathrm{CTL}_{\mathbf{FB}}$. For an explicit model where the former implication fails, see the ITS in Figure 4 in Section 3.4.

[1] It suffices to use ordinals that have the same cardinality as the model. For more details, see [7].

3. Since CTL has the finite model property, the two facts above imply that $\mathrm{CTL}_{\mathbf{FB}}$ *does not have the finite model property*, as these implications cannot fail in (image-)finite models. It is thus not immediately obvious that satisfiability for $\mathrm{CTL}_{\mathbf{FB}}$ should be decidable.
4. Consequently, the set of validities of $\mathrm{CTL}_{\mathbf{FB}}$ is properly included in the set of validities of CTL. Indeed, every non-validity of CTL is falsified in a finite model and thus, by fact 1, it is a non-validity of $\mathrm{CTL}_{\mathbf{FB}}$, too.

On the one hand, the lack of finite model property of $\mathrm{CTL}_{\mathbf{FB}}$ can be regarded as an increase of the semantic complexity and richness in comparison with standard CTL. But on the other hand, the semantics of $\mathrm{CTL}_{\mathbf{FB}}$ can be seen as simpler than that of CTL, in the sense that it only requires one to consider finite paths and does not involve dealing with universal eventualities.

Note the conceptual parallels between FBS and *for-loops* on the one side, and between the standard semantics and *while-loops* on the other. For more on this, see Section 5.2 of [7].

Finally, we list in the lemma below some validities in $\mathrm{CTL}_{\mathbf{FB}}$ used further. These can be verified easily by using the respective fixpoint characterizations.

▶ **Lemma 4.** *For every ITS $\mathcal{M}$ and $s \in \mathcal{M}$ the following hold, for $\mathsf{Q} \in \{\mathsf{E}, \mathsf{A}\}$:*
1. $\mathcal{M}, s \models_{\mathrm{fb}} \mathsf{QG}\,\varphi$ *iff* $\mathcal{M}, s \models_{\mathrm{fb}} \mathbf{G}^n_{\mathsf{Q}}(\varphi)$ *for every* $n \in \mathbb{N}$.
2. $\mathcal{M}, s \models_{\mathrm{fb}} \mathsf{Q}(\varphi\,\mathsf{U}\,\psi)$ *iff* $\mathcal{M}, s \models_{\mathrm{fb}} \mathbf{U}^n_{\mathsf{Q}}(\varphi, \psi)$ *for some* $n \in \mathbb{N}$.
3. $\models_{\mathrm{fb}} \mathsf{AG}\,\varphi \to \mathbf{G}^n_{\mathsf{A}}(\varphi)$ *and* $\models_{\mathrm{fb}} \mathbf{U}^n_{\mathsf{E}}(\varphi, \psi) \to \mathsf{E}(\varphi\,\mathsf{U}\,\psi)$ *for every* $n \in \mathbb{N}$.
4. $\models_{\mathrm{fb}} \mathbf{G}^n_{A}(\theta) \to \mathbf{G}^m_{A}(\theta)$ *and* $\models_{\mathrm{fb}} \mathbf{U}^m_{A}(\psi, \theta) \to \mathbf{U}^n_{A}(\psi, \theta)$ *for all* $m, n \in \mathbb{N}$ *such that* $m < n$.

3 Infinitary tableaux for $\mathrm{CTL}_{\mathbf{FB}}$

We only provide a detailed sketch of the (infinitary) tableaux-building procedure for $\mathrm{CTL}_{\mathbf{FB}}$ here. For further details that are essentially the same as in the tableaux method for the standard CTL, see [5, Chapter 13], the style of which we closely follow here.

3.1 Preliminaries

3.1.1 Types and components of formulae

In this section we will regard each of $\top, \bot, \neg, \wedge, \mathsf{EX}, \mathsf{AX}, \mathsf{EG}, \mathsf{AG}, \mathsf{EU}, \mathsf{AU}$ as primitive connectives in the language, while $\vee, \to, \leftrightarrow, \mathsf{EF}, \mathsf{AF}$ will be regarded as abbreviations. We will distinguish five *types of formulae*: **literals, conjunctive, disjunctive, existential successor and universal successor** formulae. Literals are atomic propositions and negations of these, and $\top$, $\bot$. Successor formulae are those beginning with EX (existential) and AX (universal). For each of the latter three types of formulae listed above we define their respective **components** as in Figure 1. Literals have no components. We write $\mathit{scomp}(\chi)$ to denote the successor component of the formula χ. Note that formulae of the type EU and $\neg\mathsf{AG}$, even though having standard semantics, are not treated here as existential eventualities are treated in standard tableaux for CTL. This is mainly for the sake of uniformity with the finitary tableaux for $\mathrm{CTL}_{\mathbf{FB}}$ presented further.

▶ **Lemma 5.** *For every ITS $\mathcal{M}$, $s \in \mathcal{M}$ and a formula φ of $\mathrm{CTL}_{\mathbf{FB}}$ the following hold:*
1. *If φ is any conjunctive formula, then $\mathcal{M}, s \models_{\mathrm{fb}} \varphi$ iff $\mathcal{M}, s \models_{\mathrm{fb}} \psi$ for every conjunctive component ψ of φ.*
2. *If φ is any disjunctive formula, then $\mathcal{M}, s \models_{\mathrm{fb}} \varphi$ iff $\mathcal{M}, s \models_{\mathrm{fb}} \psi$ for some disjunctive component ψ of φ.*

successor formula	successor component	conjunctive formula	conjunctive components	disjunctive formula	disjunctive components
$\mathsf{EX}\,\varphi$ (exist.)	φ	$\neg\neg\varphi$	φ		
$\mathsf{AX}\,\varphi$ (univ.)	φ	$\varphi \wedge \psi$	$\varphi,\ \psi$	$\neg(\varphi \wedge \psi)$	$\neg\varphi,\ \neg\psi$
$\neg\mathsf{AX}\,\varphi$ (exist.)	$\neg\varphi$	$\mathsf{AG}\,\varphi$	$\{\varphi, \mathsf{AX}\,\mathsf{AG}\,\varphi\}$	$\neg\mathsf{AG}\,\varphi$	$\{\neg\mathbf{G}^n_{\mathsf{A}}(\varphi)\}_{n\in\mathbb{N}}$
$\neg\mathsf{EX}\,\varphi$ (univ.)	$\neg\varphi$	$\mathsf{EG}\,\varphi$	$\{\mathbf{G}^n_{\mathsf{E}}(\varphi)\}_{n\in\mathbb{N}}$	$\neg\mathsf{EG}\,\varphi$	$\{\neg\mathbf{G}^n_{\mathsf{E}}(\varphi)\}_{n\in\mathbb{N}}$
		$\neg\mathsf{E}(\varphi\,\mathsf{U}\,\psi)$	$\{\neg\psi, \neg\varphi \vee \neg\mathsf{EX}\,\mathsf{E}(\varphi\,\mathsf{U}\,\psi)\}$	$\mathsf{E}(\varphi\,\mathsf{U}\,\psi)$	$\{\mathbf{U}^n_{\mathsf{E}}(\varphi,\psi)\}_{n\in\mathbb{N}}$
		$\neg\mathsf{A}(\varphi\,\mathsf{U}\,\psi)$	$\{\neg\mathbf{U}^n_{\mathsf{A}}(\varphi,\psi)\}_{n\in\mathbb{N}}$	$\mathsf{A}(\varphi\,\mathsf{U}\,\psi)$	$\{\mathbf{U}^n_{\mathsf{A}}(\varphi,\psi)\}_{n\in\mathbb{N}}$

Figure 1 Types and components of formulae in $\mathrm{CTL}_{\mathbf{FB}}$.

3.1.2 Extended closure of a formula

In order to determine the truth of a $\mathrm{CTL}_{\mathbf{FB}}$ formula η one has to consider a set of 'simpler' formulae which appear in the process of the truth evaluation of η and are needed in the tableaux construction for it. As in the case of CTL, some of these auxiliary formulae are not really simpler, as they come from the unfoldings of the temporal operators and are generally not subformulae of η. Still, they can be identified quite simply and collected in the **extended closure** of the formula η, denoted $ecl(\eta)$, obtained by closing under taking respective components of already added formulae, which is defined generically as follows.

▶ **Definition 6.** The **extended closure** $ecl(\varphi)$ of formula φ is the least set of formulae such that $\varphi \in ecl(\varphi)$ and $ecl(\varphi)$ is closed under taking all conjunctive, disjunctive, successor components of formulae in $ecl(\varphi)$. For any set of formulae Γ we define $ecl(\Gamma) := \bigcup\{\, ecl(\varphi) \mid \varphi \in \Gamma\,\}$. A set of formulae Γ is **closed** if $\Gamma = ecl(\Gamma)$.

▶ **Example 7.** Let $\eta := \mathsf{EG}\,p \wedge \neg(p \wedge \mathsf{EX}\,\mathsf{EG}\,p)$. This formula will be used in the running examples later on. Here is the extended closure of η: $ecl(\eta) = \{\eta, \mathsf{EG}\,p, \neg(p \wedge \mathsf{EX}\,\mathsf{EG}\,p)\} \cup \{\mathbf{G}^n_{\mathsf{E}}(p)\}_{n\in\mathbb{N}} \cup \{\neg p, \neg\mathsf{EX}\,\mathsf{EG}\,p\} \cup \{\mathsf{EX}\,\mathbf{G}^n_{\mathsf{E}}(p)\}_{n\in\mathbb{N}} \cup \{\neg\mathsf{EG}\,p\} \cup \{\neg\mathsf{EX}\,\mathbf{G}^n_{\mathsf{E}}(p)\}_{n\in\mathbb{N}} \cup \{\neg\mathbf{G}^n_{\mathsf{E}}(p)\}_{n\in\mathbb{N}}$.

3.1.3 Full expansions

▶ **Definition 8.** A set of formulae is **patently inconsistent** if it contains $\bot$, or $\neg\top$, or a contradictory pair of formulae φ and $\neg\varphi$.

▶ **Definition 9.** A set of formulae Γ is **fully expanded** iff:

1. it is not patently inconsistent,
2. for every conjunctive formula in Γ, all of its conjunctive components are in Γ,
3. for every disjunctive formula in Γ, at least one of its disjunctive components is in Γ.

▶ **Definition 10.** A fully expanded set of formulae Ψ is a **full expansion** of a set of formulae Γ if Ψ can be obtained from Γ by the following procedure FullExpansion, consisting in repeated application of the following rules, where initially no formula is marked as 'used':

(C-comp) for every conjunctive formula φ in the current set Γ', not yet marked as 'used', add all of its conjunctive components to Γ' and mark φ as 'used'.

(D-comp) for every disjunctive formula φ in the current set Γ', not yet marked as 'used', add to Γ' one of its disjunctive components that is not already in Γ' and mark φ as 'used'.

Since the procedure FullExpansion is non-deterministic, it can produce (possibly infinitely) many full expansions of Γ (or none). This procedure can be readily replaced with a deterministic process that constructs all fully expanded sets allowed by FullExpansion.

Moreover, in the finitary tableau (see Section 4), this process runs in EXPTIME. We denote the set of all full expansions of Γ (obtained by the procedure FullExpansion) by $\mathrm{FE}(\Gamma)$. Intuitively, a full expansion of Γ consists of all formulae appearing on some open branch in the saturated local (propositional) tableau for input set Γ.

▶ **Proposition 11.** *For any set of* $\mathrm{CTL}_{\boldsymbol{FB}}$*-formulae* Γ*, ITS* $\mathcal{M}$ *and state* $s \in \mathcal{M}$*:*

$$\mathcal{M}, s \models_{\mathrm{fb}} \Gamma \textit{ iff } \mathcal{M}, s \models_{\mathrm{fb}} \Delta \textit{ for some } \Delta \in \mathrm{FE}(\Gamma)\,.$$

The proof for this proposition is routine, using Lemma 5.

The purpose of the tableaux method outlined further is to determine whether at least one full expansion of the input formula set is satisfiable.

▶ **Example 12.** Let $\Gamma_1 := \{\eta\}$, where $\eta := \mathsf{EG}\,p \wedge \neg(p \wedge \mathsf{EX}\,\mathsf{EG}\,p)$ (recall Example 7) and let $\Gamma_2 := \{\mathbf{G}^k_{\mathsf{E}}(p), \neg\mathsf{EG}\,p\}$, where $k \in \mathbb{N}$. These sets of formulae will be used later in Example 17.

Γ_1 has the following full expansion: $\{\eta, \mathsf{EG}\,p, \neg(p\wedge\mathsf{EX}\,\mathsf{EG}\,p)\}\cup\{\mathbf{G}^n_{\mathsf{E}}(p)\}_{n\in\mathbb{N}}\cup\{\neg\mathsf{EX}\,\mathsf{EG}\,p\}\cup\{\mathsf{EX}\,\mathbf{G}^n_{\mathsf{E}}(p)\}_{n\in\mathbb{N}}$. This is the only full expansion of Γ_1, because choosing the disjunctive component $\neg p$ of $\neg(p \wedge \mathsf{EX}\,\mathsf{EG}\,p)$ creates a patently inconsistent set (since $\mathbf{G}^0_{\mathsf{E}}(p) = p$).

On the other hand, the set $\mathrm{FE}(\Gamma_2)$ is infinite, because for each $m \in \mathbb{N}$ such that $m \notin \{0, k\}$ the set $\Psi^m = \{\mathbf{G}^k_{\mathsf{E}}(p), \neg\mathsf{EG}\,p,\ p,\ \mathsf{EX}\,\mathbf{G}^{k-1}_{\mathsf{E}}(p), \neg\mathbf{G}^m_{\mathsf{E}}(p), \neg\mathsf{EX}\,\mathbf{G}^{m-1}_{\mathsf{E}}(p)\}$ is a full expansion of Γ_2. The disjunctive components $\neg\mathbf{G}^k_{\mathsf{E}}(p)$ and $\neg\mathbf{G}^0_{\mathsf{E}}(p) = \neg p$ of $\neg\mathsf{EG}\,p$ are the only ones that create patently inconsistent sets.

3.2 Hintikka structures

Intuitively, a Hintikka structure represents a partly defined rooted ITS satisfying the input formula. It is a graph, every node of which is labeled by a set of formulae. These labels are fully expanded subsets of the extended closure of a designated input formula, the satisfiability of which is tested by the tableau. All desired properties of the transition relations in a Hintikka structure are encoded by means of the labels of the states. Membership to the label of the state of a Hintikka structure simulates the notion of truth of a formula at a state of an interpreted transition system and the labelling of states must ensure that the Hintikka structure can generate a model of the input formula. The purpose of the tableau construction is to check for existence of a Hintikka structure 'satisfying' the input formula, in the sense described above. Here is the formal definition of a Hintikka structure for $\mathrm{CTL}_{\mathbf{FB}}$.

▶ **Definition 13.** Given a closed set of $\mathrm{CTL}_{\mathbf{FB}}$-formulae Γ, a **Hintikka structure (HS) for** Γ is a tuple $\mathcal{H} = (\mathrm{S}, \mathrm{R}, \mathrm{H})$ s.t. (S, R) is a transition system and $\mathrm{H} : \mathrm{S} \to \mathcal{P}(\Gamma)$ is a labelling function satisfying the following conditions for every $s \in \mathrm{S}$:

(H1) $\mathrm{H}(s)$ is fully expanded (in the sense of Def. 9).

(H2) If $\varphi \in \mathrm{H}(s)$ is an existential successor formula, then $\mathit{scomp}(\varphi) \in \mathrm{H}(s')$ for some s' such that $s\,\mathrm{R}\,s'$. (Recall that $\mathit{scomp}(\varphi)$ denotes the successor component of the formula φ.)

(H3) If $\varphi \in \mathrm{H}(s)$ is a universal successor formula, then $\mathit{scomp}(\varphi) \in \mathrm{H}(s')$ for every s' such that $s\,\mathrm{R}\,s'$.

▶ **Definition 14.** A formula $\varphi \in \mathrm{CTL}_{\mathbf{FB}}$ is **satisfiable in a Hintikka structure** $\mathcal{H} = (\mathrm{S}, \mathrm{R}, \mathrm{H})$ for a set Γ if $\varphi \in \mathrm{H}(s)$ for some $s \in \mathrm{S}$. A set of formulae $\Psi \subseteq \Gamma$ is **satisfiable in** $\mathcal{H}$ if $\Psi \subseteq \mathrm{H}(s)$ for some state s in $\mathcal{H}$.

Note that every rooted ITS uniformly generates a rooted Hintikka structure for any closed set of formulae Γ by labelling each state of the ITS with the set of all formulae from Γ that are true at that state. Formally:

▶ **Lemma 15.** *For any closed set of* $\mathrm{CTL}_{\boldsymbol{FB}}$*-formulae* Γ *(in the sense of Def. 6) over as set of atomic propositions* Φ *and for every interpreted transition system* $\mathcal{M} = (\mathrm{S}, \mathrm{R}, \Phi, \ell)$ *the structure* $\mathcal{H}(\mathcal{M}) = (\mathrm{S}, \mathrm{R}, \mathrm{H})$*, where* $\mathrm{H}(s) = \{\varphi \in \Gamma \mid \mathcal{M}, s \models \varphi\}$ *for every* $s \in \mathrm{S}$*, is a Hintikka structure for* Γ*.*

An essential difference between interpreted transition systems and Hintikka structures is that, while an ITS determines the truth value of every formula at every state, a Hintikka structure only contains just enough information to determine the truth values of those formulae that are directly involved in the evaluation of the input formula η at the root state.

Given a formula φ for which we look for a model, we will be interested in Hintikka structures for the set $ecl(\varphi)$. For that class of Hintikka structures to be suitable for our purpose, every formula satisfiable in a Hintikka structure must also be satisfiable in an ITS, so the two notions of satisfiability are equivalent. Thus, the following result is needed. (See the Appendix for a sketch of proof).

▶ **Theorem 16.** *A* $\mathrm{CTL}_{\boldsymbol{FB}}$*-formula* η *is satisfiable iff it is satisfiable in some Hintikka structure for* $ecl(\eta)$*.*

3.3 Construction of the tableaux

The tableau procedure presented here attempts to construct for a given input formula η a non-empty graph $\mathcal{T}^\eta$, called a **tableau**, representing (in a way) sufficiently many possible Hintikka structures for η. The procedure consists of three major phases:

1. **Construction phase.** In that phase a finite directed graph $\mathcal{P}^\eta$ with labeled vertices, called the **pretableau** for η, is produced, following prescribed **construction rules**. The set of nodes of the pretableau properly contains the set of nodes of the tableau $\mathcal{T}^\eta$ that is to be ultimately built. The pretableau has two types of nodes: **states** and **prestates**. The states are labeled with fully expanded subsets of $ecl(\eta)$ and represent states of a Hintikka structure, while the prestates can be labeled with any subsets of $ecl(\eta)$ and they play only an auxiliary and temporary role. In this tableau construction states and prestates with an already existing label are not created again, but reused. So, when there is no danger of confusion, we will identify prestates or states with their labels.
2. **Prestate elimination phase.** Here the prestates are removed using the **prestate elimination rule**. The result is a smaller graph $\mathcal{T}_0^\eta$, called the **initial tableau** for η.
3. **State elimination phase.** In this phase we remove, using **state elimination rules**, all (if any) states from $\mathcal{T}_0^\eta$ that cannot be satisfied in any Hintikka structure. In the case of $\mathrm{CTL}_{\mathbf{FB}}$, where there are no explicit eventualities to be satisfied, an elimination of a state can happen for only one reason: some of the successor states that the state needs for the satisfaction of its successor formulae, have already turned out unsatisfiable and have been removed in the elimination process so far. The state elimination phase results in a (possibly empty) subgraph $\mathcal{T}^\eta$ of $\mathcal{T}_0^\eta$, called the **final tableau** for η.

If there is a state in the final tableau $\mathcal{T}^\eta$ containing η in its label, the tableau is declared **open** and the input formula η is pronounced satisfiable; otherwise, the tableau is declared **closed** and η is pronounced unsatisfiable.

3.3.1 The pretableau construction phase

The pretableau construction phase consists of two rules: $\textsc{PrExp}^{\mathrm{CTL}_{\mathbf{FB}}}$ producing all **offspring states** of a given prestate; and $\textsc{Next}^{\mathrm{CTL}_{\mathbf{FB}}}$ producing the **successor prestates** of a given state. The rule PrExp$^{\mathrm{CTL}_{\mathbf{FB}}}$ involves the procedure FullExpansion, described in Section 3.1.3, for computing the family $\mathrm{FE}(\Gamma)$ of full expansions of a given set $\Gamma \subseteq ecl(\eta)$.

PrExp$^{\mathrm{CTL}_{\mathbf{FB}}}$: Given a prestate Γ to which the rule has not yet been applied, do the following:

1. Compute the family $\mathrm{FE}(\Gamma)$ of full expansions of Γ and add these as (labels of) new states in the pretableau, called the **offspring states** of Γ.
2. For each newly introduced state Δ, create an edge $\Gamma \dashrightarrow \Delta$.
3. If, however, the pretableau already contains a state with label Δ then do not create a new state with that label, but create an edge to Δ instead.

We denote the set $\{ \Delta \mid \Gamma \dashrightarrow \Delta \}$ of offspring states of Γ by $\mathsf{states}(\Gamma)$. Hereafter we write $\mathsf{X}(\Delta) := \{ \psi \mid \mathsf{AX}\,\psi \in \Delta \} \cup \{ \neg\psi \mid \neg\mathsf{EX}\,\psi \in \Delta \}$ for any set of $\mathrm{CTL}_{\mathbf{FB}}$-formulae Δ.

Next$^{\mathrm{CTL}_{\mathbf{FB}}}$: Given a state Δ to which the rule has not yet been applied, do the following:

1. For each existential successor formula $\varphi \in \Delta$ (i.e., $\varphi = \mathsf{EX}\,\chi$ or $\varphi = \neg\mathsf{AX}\,\chi$) add a successor prestate Γ of Δ with label $\mathsf{X}(\Delta) \cup \{scomp(\varphi)\}$ and create an edge $\Delta \xrightarrow{\varphi} \Gamma$.
2. Consider the case where Δ has no existential successor formulae. If $\mathsf{X}(\Delta) \neq \emptyset$, add one prestate Γ with label $\mathsf{X}(\Delta)$ and an edge $\Delta \to \Gamma$. If $\mathsf{X}(\Delta) = \emptyset$, simply create a loop $\Delta \to \Delta$.
3. If the pretableau already contains a prestate with the label of Γ, then do not create a new prestate with that label, but create an edge to Γ instead.

The construction phase of building a pretableau for η begins with creating a single prestate $\{\eta\}$, followed by alternating applications of the rules PrExp$^{\mathrm{CTL}_{\mathbf{FB}}}$ and Next$^{\mathrm{CTL}_{\mathbf{FB}}}$, respectively to the prestates and the states created at the previous stages of the construction. The construction phase is completed if/when none of these rules can add any new states or prestates to the current graph. The resulting graph is the pretableau $\mathcal{P}^{\eta}$.

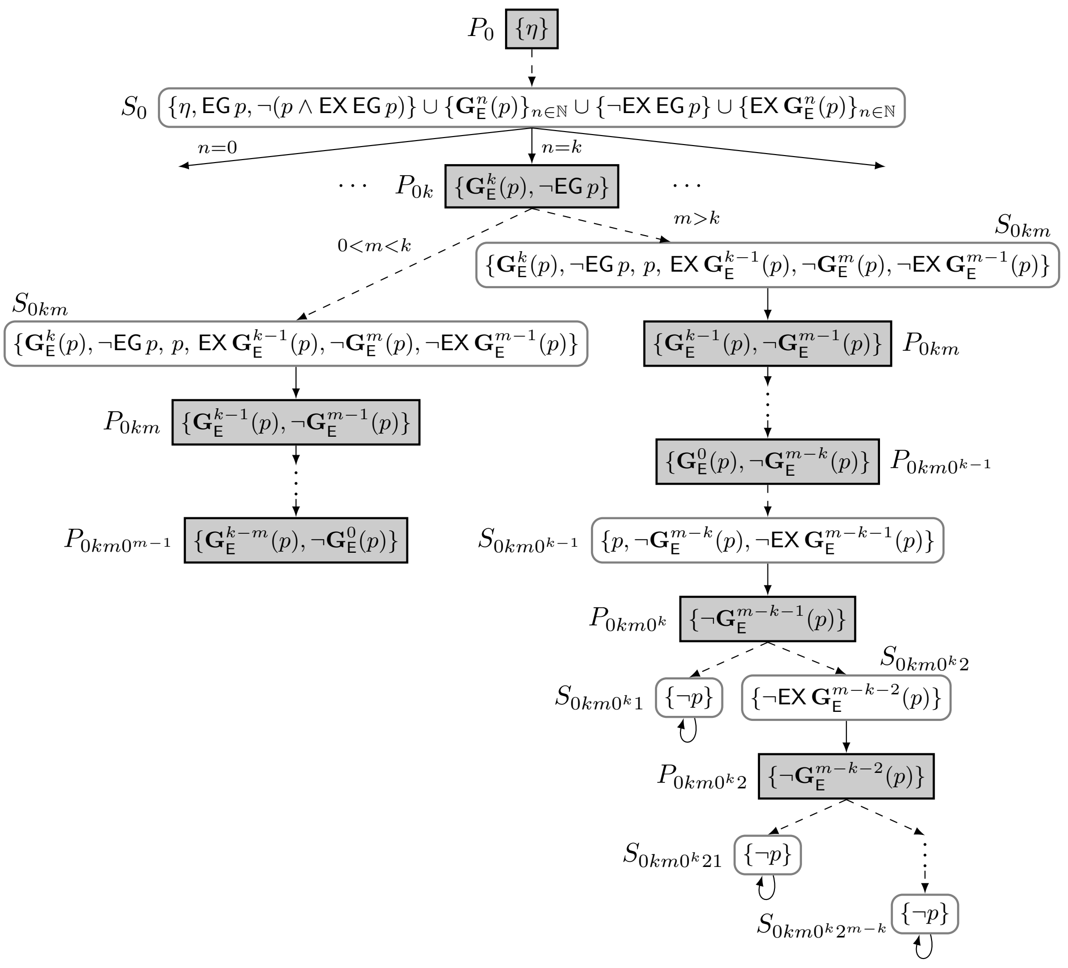
Note that there are two types of branching in the pretableau: **search branching**, from a prestate to its offspring states, indicated by $\dashrightarrow$, and **structural branching**, from a state to its successor prestates, indicated by $\xrightarrow{\chi}$ for an EX-formula χ. Search branching is branching of the search tree, and thus it is **disjunctive**, or **existential**: only one offspring state of every prestate is eventually needed to build a satisfying structure. Structural branching is **conjunctive**, or **universal**. Since it represents branching in the structure to be built, *all* successor prestates of every state are potentially needed in the construction.

In the subsequent figures illustrating tableau examples, prestates are indicated with shaded square boxes and labeled by P with indices, while states are indicated with transparent boxes with rounded corners and labeled by S with indices.

▶ **Example 17** (Pretableau). (Recall Example 12.) The sentence $\eta = \mathsf{EG}\,p \wedge \neg(p \wedge \mathsf{EX}\,\mathsf{EG}\,p)$ has the countably infinite pretableau given in Figure 2. From state S_0 there is an infinite structural branching to prestates $\{P_{0n} \mid n \in \mathbb{N}\}$. Since the pretableau construction is analogous for all of these prestates, we only present a single one of them here, namely P_{0k}. From P_{0k} there is an infinite search branching to states $\{S_{0km} \mid m \in \mathbb{N} \setminus \{0, k\}\}$. But all cases where $0 < m < k$ are analogous to each other. Similarly, all cases where $m > k$ are analogous to each other. Note that when $m < k$, the prestates $P_{0km0^{m-1}}$ do not have any offspring states since then $\mathrm{FE}(\{\mathbf{G}^{k-m}_{\mathsf{E}}(p), \neg\mathbf{G}^{0}_{\mathsf{E}}(p)\}) = \emptyset$.

3.3.2 Prestate elimination phase and initial tableaux

In this phase, all prestates are removed from $\mathcal{P}^{\eta}$, together with their incoming and outgoing arrows, by applying the following generic **prestate elimination rule**:



P_0 $\{\eta\}$
S_0 $\{\eta, \mathsf{EG}\,p, \neg(p \wedge \mathsf{EX}\,\mathsf{EG}\,p)\} \cup \{\mathbf{G}^n_{\mathsf{E}}(p)\}_{n\in\mathbb{N}} \cup \{\neg\mathsf{EX}\,\mathsf{EG}\,p\} \cup \{\mathsf{EX}\,\mathbf{G}^n_{\mathsf{E}}(p)\}_{n\in\mathbb{N}}$
$n{=}0$
$n{=}k$
$\cdots$ P_{0k} $\{\mathbf{G}^k_{\mathsf{E}}(p), \neg\mathsf{EG}\,p\}$ $\cdots$
$0{<}m{<}k$
$m{>}k$
S_{0km} $\{\mathbf{G}^k_{\mathsf{E}}(p), \neg\mathsf{EG}\,p,\ p,\ \mathsf{EX}\,\mathbf{G}^{k-1}_{\mathsf{E}}(p), \neg\mathbf{G}^m_{\mathsf{E}}(p), \neg\mathsf{EX}\,\mathbf{G}^{m-1}_{\mathsf{E}}(p)\}$
S_{0km} $\{\mathbf{G}^k_{\mathsf{E}}(p), \neg\mathsf{EG}\,p,\ p,\ \mathsf{EX}\,\mathbf{G}^{k-1}_{\mathsf{E}}(p), \neg\mathbf{G}^m_{\mathsf{E}}(p), \neg\mathsf{EX}\,\mathbf{G}^{m-1}_{\mathsf{E}}(p)\}$
$\{\mathbf{G}^{k-1}_{\mathsf{E}}(p), \neg\mathbf{G}^{m-1}_{\mathsf{E}}(p)\}$ P_{0km}
P_{0km} $\{\mathbf{G}^{k-1}_{\mathsf{E}}(p), \neg\mathbf{G}^{m-1}_{\mathsf{E}}(p)\}$
$\{\mathbf{G}^0_{\mathsf{E}}(p), \neg\mathbf{G}^{m-k}_{\mathsf{E}}(p)\}$ $P_{0km0^{k-1}}$
$P_{0km0^{m-1}}$ $\{\mathbf{G}^{k-m}_{\mathsf{E}}(p), \neg\mathbf{G}^0_{\mathsf{E}}(p)\}$
$S_{0km0^{k-1}}$ $\{p, \neg\mathbf{G}^{m-k}_{\mathsf{E}}(p), \neg\mathsf{EX}\,\mathbf{G}^{m-k-1}_{\mathsf{E}}(p)\}$
P_{0km0^k} $\{\neg\mathbf{G}^{m-k-1}_{\mathsf{E}}(p)\}$
S_{0km0^k2}
S_{0km0^k1} $\{\neg p\}$ $\{\neg\mathsf{EX}\,\mathbf{G}^{m-k-2}_{\mathsf{E}}(p)\}$
P_{0km0^k2} $\{\neg\mathbf{G}^{m-k-2}_{\mathsf{E}}(p)\}$
S_{0km0^k21} $\{\neg p\}$
$S_{0km0^k2^{m-k}}$ $\{\neg p\}$

Figure 2 Infinitary pretableau for $\eta = \mathsf{EG}\,p \wedge \neg(p \wedge \mathsf{EX}\,\mathsf{EG}\,p)$.

PrestateElim$^{\mathbf{CTL_{FB}}}$. For every prestate Γ in $\mathcal{P}^\eta$, do the following:

1. If there is a $\Delta \in \mathcal{P}^\eta$ with $\Delta \to \Gamma$, then for every $\Delta' \in \mathsf{states}(\Gamma)$, create an edge $\Delta \to \Delta'$;
2. Remove Γ from $\mathcal{P}^\eta$ together with its outgoing arrows.

The resulting graph is called the **initial tableau** for η, denoted $\mathcal{T}^\eta_0$. The offspring states of the input prestate $\{\eta\}$ are called **input states** of $\mathcal{T}^\eta_0$.

3.3.3 State elimination phase and the final tableau

This phase is carried out in a sequence of stages, starting with the initial tableau $\mathcal{T}^\eta_0$, and eliminating at each stage n at least one state for the current tableau $\mathcal{T}^\eta_n$, by applying the state elimination rule stated below, to produce the new current tableau $\mathcal{T}^\eta_{n+1}$, until stabilisation.

StateElim$^{\mathrm{CTL_{FB}}}$: If a state Δ, containing an existential successor formula $\mathsf{EX}\,\psi$ (respectively, $\neg\mathsf{AX}\,\psi$), has no successor states containing ψ (respectively, $\neg\psi$) in the current tableau, then remove Δ from the tableau. Remove Δ from the tableau also if Δ has no successor states.

The rule StateElim$^{\mathrm{CTL_{FB}}}$ is applied repeatedly until reaching a stage when no further elimination of states is possible. Such a stage may not be reached at any finite stage, so the elimination phase may have to go on for transfinitely many steps, in which case every limit-stage tableau is the limit of the decreasing chain (by inclusion) of current tableaux.

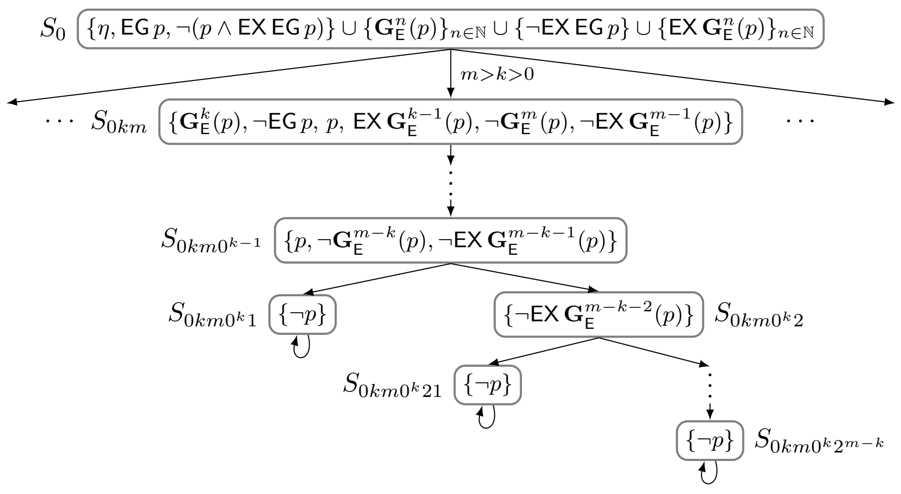

S_0 $\{\eta, \mathsf{EG}\, p, \neg(p \wedge \mathsf{EX}\,\mathsf{EG}\, p)\} \cup \{\mathbf{G}^n_{\mathsf{E}}(p)\}_{n\in\mathbb{N}} \cup \{\neg\mathsf{EX}\,\mathsf{EG}\, p\} \cup \{\mathsf{EX}\,\mathbf{G}^n_{\mathsf{E}}(p)\}_{n\in\mathbb{N}}$
$m{>}k{>}0$
$\cdots$ S_{0km} $\{\mathbf{G}^k_{\mathsf{E}}(p), \neg\mathsf{EG}\, p,\ p,\ \mathsf{EX}\,\mathbf{G}^{k-1}_{\mathsf{E}}(p), \neg\mathbf{G}^m_{\mathsf{E}}(p), \neg\mathsf{EX}\,\mathbf{G}^{m-1}_{\mathsf{E}}(p)\}$ $\cdots$
$S_{0km0^{k-1}}$ $\{p, \neg\mathbf{G}^{m-k}_{\mathsf{E}}(p), \neg\mathsf{EX}\,\mathbf{G}^{m-k-1}_{\mathsf{E}}(p)\}$
S_{0km0^k1} $\{\neg p\}$
$\{\neg\mathsf{EX}\,\mathbf{G}^{m-k-2}_{\mathsf{E}}(p)\}$ S_{0km0^k2}
S_{0km0^k21} $\{\neg p\}$
$\{\neg p\}$ $S_{0km0^k2^{m-k}}$

Figure 3 Final tableau for $\eta = \mathsf{EG}\, p \wedge \neg(p \wedge \mathsf{EX}\,\mathsf{EG}\, p)$.

When the state elimination phase reaches a (finite or limit) stabilization stage, the resulting tableau then is the **final tableau for** η, denoted by $\mathcal{T}^\eta$, with a set of states denoted by S^η.

▶ **Definition 18.** The final tableau $\mathcal{T}^\eta$ is **open** if $\eta \in \Delta$ for some $\Delta \in S^\eta$; else, $\mathcal{T}^\eta$ is **closed**.

The tableau procedure returns "not satisfiable" if the final tableau is closed; otherwise, it returns "satisfiable" and, moreover, provides sufficient information for producing a countable Hintikka structure satisfying η, as described in the completeness proof below.

▶ **Example 19** (Final tableau)**.** Recall the pretableau from Example 17. The initial tableau is obtained removing the prestates by applying $\textsc{PrestateElim}^{\mathrm{CTL}_{\mathbf{FB}}}$. The branches starting from states S_{0km} where $0 < m < k$ end at a prestate that has no offspring state. Therefore all of states on these branches are eliminated when the final tableau is formed. All the other states remain in the final tableau and therefore it is open. See Figure 3 for the final tableau.

3.4 Soundness and completeness of the infinitary tableau for $\mathsf{CTL}_{\mathsf{FB}}$

Note that in general none of the three phases of the construction terminates at any finite stage if performed consecutively, because the set of full expansions of a prestate may be infinite, or because infinitely many successor prestates may have to be introduced. However, each of these stages does reach saturation at some transfinite stage.

Soundness of the tableau procedure means that if the input formula η is satisfiable, then the final tableau $\mathcal{T}^\eta$ is open, so the tableau procedure is guaranteed to establish the satisfiability. A generic proof of soundness consists of showing that at least one tableau state containing η will survive forever, i.e. will never be eliminated. Note first that if η is satisfiable, and hence propositionally consistent, there will be at least one offspring state of the initial prestate containing η, due to Proposition 11. Moreover, for every rooted ITS $(\mathcal{M}, s)$ satisfying η, the set $\{\psi \in ecl(\eta) \mid \mathcal{M}, s \models \psi\}$ contains at least one such state. Thereafter, it suffices to show that *only unsatisfiable states* get eliminated in the state elimination phase. Thus, we will establish the soundness of the tableau for $\mathrm{CTL}_{\mathbf{FB}}$ by proving that the state elimination rule is 'sound' in a sense that it never eliminates a state with a satisfiable label. The soundness of the overall procedure is then an immediate consequence. The proof of that

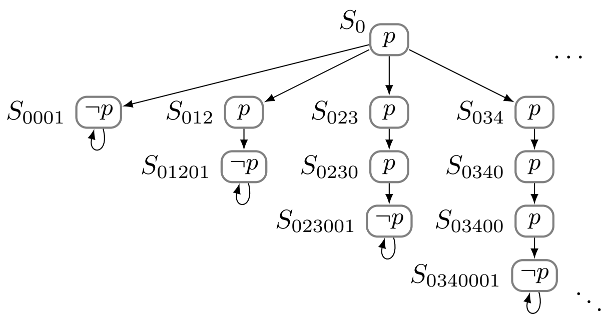


Figure 4 A model satisfying $\mathsf{EG}\,p \wedge \neg(p \wedge \mathsf{EX}\,\mathsf{EG}\,p)$.

claim follows a fairly routine argument (see [5, Chapter 13]) by induction on the applications of the state elimination rule. However, since the elimination phase for $\mathrm{CTL}_{\mathbf{FB}}$ may take a transfinite sequence of steps, the soundless argument now uses transfinite induction. For a proof sketch, see the Appendix.

▶ **Lemma 20.** *Let Γ be a prestate of $\mathcal{P}^\eta$ such that $\mathcal{M}, s \models \Gamma$ for some rooted interpreted transition system $(\mathcal{M}, s)$. Then $\mathcal{M}, s \models \Psi$ for at least one $\Psi \in \mathsf{states}(\Gamma)$.*

▶ **Lemma 21.** *No satisfiable state $\Delta \in \mathcal{T}_0^\eta$ is removed by any application of* $\textsc{StateElim}^{\mathrm{CTL}_{\mathbf{FB}}}$ *during the state elimination phase.*

▶ **Theorem 22** (Soundness). *If $\eta \in \mathrm{CTL}_{\mathbf{FB}}$ is satisfiable then $\mathcal{T}^\eta$ is open.*

Completeness of the tableau procedure means that if the input formula η is not satisfiable, then the final tableau $\mathcal{T}^\eta$ is closed, so the tableau procedure is guaranteed to establish the unsatisfiability. By contraposition, completeness means that if the final tableau is open, the input formula η is satisfiable. The proof of the completeness is sketched in the Appendix.

▶ **Theorem 23** (Completeness). *For any formula $\eta \in \mathrm{CTL}_{\mathbf{FB}}$, if the final tableau $\mathcal{T}^\eta$ is open, then η is satisfiable.*

▶ **Example 24.** Since the final tableau of $\eta = \mathsf{EG}\,p \wedge \neg(p \wedge \mathsf{EX}\,\mathsf{EG}\,p)$ is open, η is satisfiable by Theorem 23. Thus $\mathsf{EG}\,p \to (p \wedge \mathsf{EX}\,\mathsf{EG}\,p)$ is not a valid $\mathrm{CTL}_{\mathbf{FB}}$ sentence. Consider the final tableau in Example 19 as a transition system $\mathcal{T}$. By using any state description function ℓ that satisfies the labels of the states in $\mathcal{T}$, we obtain an ITS which satisfies η.

However, by analysing the pretableau of η (Example 17) we can construct a simpler ITS that satisfies η. It suffices that, for every prestate Γ, a single state in $\mathsf{states}(\Gamma)$ survives the state elimination process. Thus, for a prestate P_{0k} ($k \in \mathbb{N}$), we can choose any offspring state S_{0km} for which $m > k$; e.g. we can choose $m := k + 1$. Furthermore, it suffices that, for a prestate P_{0km0^k}, we choose only the state S_{0km0^k1} (and remove the state S_{0km0^k2}).

By removing states from $\mathcal{T}^\eta$ as described above, we obtain the ITS in Figure 4 which can be seen as the simplest model satisfying η. Note that since $S_{0001}, S_{01201}, S_{023001}, S_{0340001}, \ldots$ have the same label $\{\neg p\}$, they are actually merged into a single state by the tableau rules.

▶ **Example 25.** Consider $\eta' := (p \wedge \mathsf{EX}\,\mathsf{EG}\,p) \wedge \neg\mathsf{EG}\,p$. Since $\neg\mathbf{G}^0_{\mathsf{E}}(p) = \neg p$ is a conjunctive component of $\neg\mathsf{EG}\,p$ and p is a conjunctive component of $p \wedge \mathsf{EX}\,\mathsf{EG}\,p$, we have $\mathrm{FE}(\{\eta'\}) = \emptyset$. Therefore the pretableau of η' will only have a single prestate, namely $\{\eta'\}$, and not any states. Thus the final tableau for η' is empty and thus it is trivially closed. Hence by Theorem 22 η' is not satisfiable, i.e. $(p \wedge \mathsf{EX}\,\mathsf{EG}\,p) \to \mathsf{EG}\,p$ is a valid $\mathrm{CTL}_{\mathbf{FB}}$ formula.

It is easy to show that the structural branchings in the tableau are always countable. Thus we can prove the following claim for $\mathrm{CTL}_{\mathbf{FB}}$. For more details, see the the Appendix.

▶ **Proposition 26.** $\mathrm{CTL}_{\mathbf{FB}}$ *has the* ***countable model property****, i.e. if* $\eta \in \mathrm{CTL}_{\mathbf{FB}}$ *is satisfiable, then* η *is satisfiable in a countable model.*

4 Finitary tableaux for $\mathrm{CTL}_{\mathrm{FB}}$

4.1 Construction of finitary tableaux

Hereafter we will use a set of new symbols $\{\mathbf{n}_i \mid i \in \mathbb{N}^+\}$, called **iteration parameters**, which will be used instead of natural numbers n_i in formulae of type $\mathbf{G}^{n_i}_{\mathsf{Q}}(\varphi)$ and $\mathbf{U}^{n_i}_{\mathsf{Q}}(\varphi,\psi)$. We denote the resulting extended language $\mathrm{CTL}^{par}_{\mathbf{FB}}$. Here $\mathbf{G}^{\mathbf{n}_i}_{\mathsf{Q}}(\varphi), \mathbf{U}^{\mathbf{n}_i}_{\mathsf{Q}}(\varphi,\psi) \in \mathrm{CTL}^{par}_{\mathbf{FB}}$ $(i \in \mathbb{N}^+)$ *are not abbreviations*; they are treated as actual formulae of $\mathrm{CTL}^{par}_{\mathbf{FB}}$. We will not give any semantics for $\mathbf{G}^{\mathbf{n}_i}_{\mathsf{Q}}(\varphi)$ or $\mathbf{U}^{\mathbf{n}_i}_{\mathsf{Q}}(\varphi,\psi)$ as they will only be used "symbolically" in the construction of the finite tableau. In particular, the iteration parameter $\mathbf{n}_i$ is just a *symbol* which does not have any actual value. However, *intuitively*, each parameter $\mathbf{n}_i$ takes an "arbitrarily large" but finite and fixed value which represents the number of iterations of the given recursive operator. The index i (recall $i \in \mathbb{N}^+$) of the iteration parameter $\mathbf{n}_i$ indicates *when* the "value" of $\mathbf{n}_i$ has been fixed (with respect to the other iteration parameters). The "value" of $\mathbf{n}_i$ may depend on the "values" of all iteration parameters with smaller subindices, as they have been set earlier. Thus, we may assume that each iteration parameter is "arbitrarily larger" than all the parameters with smaller indices.

We now re-define the conjunctive and disjunctive components of formulae in $\mathrm{CTL}^{par}_{\mathbf{FB}}$ as in the table below (with no changes for the formulae that are not listed there). Note here that, since we give no semantics for the formulae $\mathbf{G}^{\mathbf{n}_i}_{\mathsf{Q}}(\varphi)$ and $\mathbf{U}^{\mathbf{n}_i}_{\mathsf{Q}}(\varphi,\psi)$, their components are "symbolical" and are only defined in order to be used with the procedure FullExpansion.

formulae	conjunctive component	formulae	disjunctive component
$\mathsf{AG}\,\varphi$	$\{\varphi, \mathsf{AX}\,\mathsf{AG}\,\varphi\}$	$\neg\mathsf{AG}\,\varphi,\ \neg\mathbf{G}^{\mathbf{n}_i}_{\mathsf{A}}(\varphi)$	$\{\neg\varphi, \neg\mathsf{AX}\,\mathbf{G}^{\mathbf{n}_i}_{\mathsf{A}}(\varphi)\}$
$\mathsf{EG}\,\varphi,\ \mathbf{G}^{\mathbf{n}_i}_{\mathsf{E}}(\varphi)$	$\{\varphi, \mathsf{EX}\,\mathbf{G}^{\mathbf{n}_i}_{\mathsf{E}}(\varphi)\}$	$\neg\mathsf{EG}\,\varphi,\ \neg\mathbf{G}^{\mathbf{n}_i}_{\mathsf{E}}(\varphi)$	$\{\neg\varphi, \neg\mathsf{EX}\,\mathbf{G}^{\mathbf{n}_i}_{\mathsf{E}}(\varphi)\}$
$\neg\mathsf{E}(\varphi\,\mathsf{U}\,\psi)$	$\{\neg\psi, \neg\varphi \vee \neg\mathsf{EX}\,\mathsf{E}(\varphi\,\mathsf{U}\,\psi)\}$	$\mathsf{E}(\varphi\,\mathsf{U}\,\psi),\ \mathbf{U}^{\mathbf{n}_i}_{\mathsf{E}}(\varphi,\psi)$	$\{\psi, \varphi \wedge \mathsf{EX}\,\mathbf{U}^{\mathbf{n}_i}_{\mathsf{E}}(\varphi,\psi)\}$
$\neg\mathsf{A}(\varphi\,\mathsf{U}\,\psi), \neg\mathbf{U}^{\mathbf{n}_i}_{\mathsf{A}}(\varphi,\psi)$	$\{\neg\psi, \neg\varphi \vee \neg\mathsf{AX}\,\mathbf{U}^{\mathbf{n}_i}_{\mathsf{A}}(\varphi,\psi)\}$	$\mathsf{A}(\varphi\,\mathsf{U}\,\psi),\ \mathbf{U}^{\mathbf{n}_i}_{\mathsf{A}}(\varphi,\psi)$	$\{\psi, \varphi \wedge \mathsf{AX}\,\mathbf{U}^{\mathbf{n}_i}_{\mathsf{A}}(\varphi,\psi)\}$

The cases with two formulae (e.g. $\mathsf{EG}\,\varphi$ and $\mathbf{G}^{\mathbf{n}_i}_{\mathsf{E}}(\varphi)$), in the table above, have the same component for both formulae. However, the *index* i chosen for the iteration parameter $\mathbf{n}_i$ with the second component (e.g. $\mathsf{EX}\,\mathbf{G}^{\mathbf{n}_i}_{\mathsf{E}}(\varphi)$) is chosen differently for the formula on the left (e.g. $\mathsf{EG}\,\varphi$) and the formula on the right (e.g. $\mathbf{G}^{\mathbf{n}_i}_{\mathsf{E}}(\varphi)$); see below.

- (Left) Here we need to introduce a *new* parameter $\mathbf{n}_i$ for the component. The value of the index $i \in \mathbb{N}^+$ depends on the context; see the construction of the set $\mathrm{FE}(\Delta)$ below.
- (Right) Here the *same* iteration parameter $\mathbf{n}_i$ (with the same index i) that occurs in the original formula is used for the second component as well.

The notions of extended closure and full expansions for any set of formulae Δ in $\mathrm{CTL}^{par}_{\mathbf{FB}}$ are defined as for $\mathrm{CTL}_{\mathbf{FB}}$. When creating the set $\mathrm{FE}(\Delta)$ we may need to introduce new iteration parameters $\mathbf{n}_i$ for the components of formulae in Δ (e.g. in the case of $\mathsf{EG}\,\varphi$ in the table above). When this happens, the index $i \in \mathbb{N}^+$ of $\mathbf{n}_i$ is chosen to be *the smallest integer* i *that is greater than* all other indexes of iteration parameters that currently occur in Δ. Note that the same index i can be used for all formulae for which we need to introduce a new parameter $\mathbf{n}_i$ (at the same time). Therefore in $\mathrm{FE}(\Delta)$ there is at most one parameter $\mathbf{n}_i$ that does not occur in Δ. Also note that for a finite set Δ, the set $\mathrm{FE}(\Delta)$ is also finite.

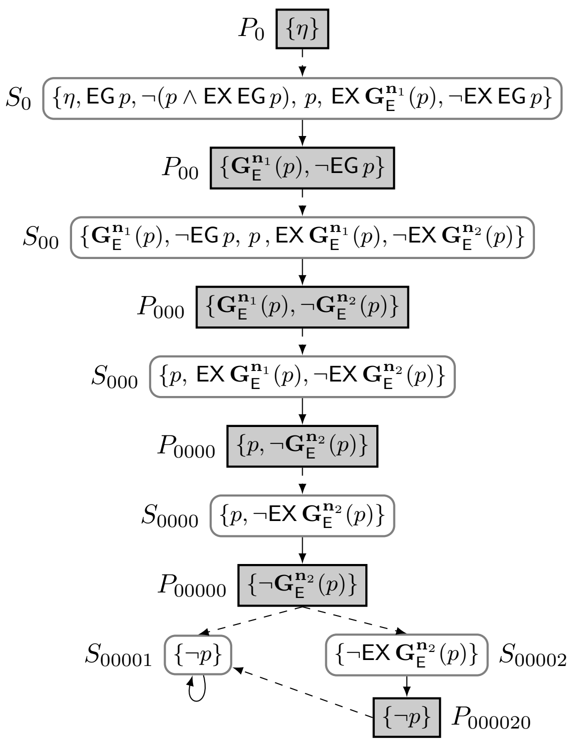

P_0 $\{\eta\}$
S_0 $\{\eta, \mathsf{EG}\,p, \neg(p \wedge \mathsf{EX\,EG}\,p),\ p,\ \mathsf{EX}\,\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_1}(p), \neg\mathsf{EX\,EG}\,p\}$
P_{00} $\{\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_1}(p), \neg\mathsf{EG}\,p\}$
S_{00} $\{\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_1}(p), \neg\mathsf{EG}\,p,\ p\,, \mathsf{EX}\,\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_1}(p), \neg\mathsf{EX}\,\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_2}(p)\}$
P_{000} $\{\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_1}(p), \neg\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_2}(p)\}$
S_{000} $\{p,\ \mathsf{EX}\,\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_1}(p), \neg\mathsf{EX}\,\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_2}(p)\}$
P_{0000} $\{p, \neg\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_2}(p)\}$
S_{0000} $\{p, \neg\mathsf{EX}\,\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_2}(p)\}$
P_{00000} $\{\neg\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_2}(p)\}$
S_{00001} $\{\neg p\}$
$\{\neg\mathsf{EX}\,\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_2}(p)\}$ S_{00002}
$\{\neg p\}$ P_{000020}

Figure 5 Finitary pretableau for $\eta = \mathsf{EG}\,p \wedge \neg(p \wedge \mathsf{EX\,EG}\,p)$.

Finally we consider the case when a successor prestate Γ would be created for a state Δ but there already exists a prestate Γ' with the same label as Γ. If no iteration parameter $\mathbf{n}_i$ occurs in Γ, then we proceed as normally by adding an arrow $\Delta \to \Gamma'$. Else, we apply the following "parameter elimination" procedure:

ParElim: Let i be the *smallest* integer for which $\mathbf{n}_i$ occurs (possibly many times) in Γ. Now we first create a set $\Gamma_{/i}$ by replacing all formulae in Γ that contain $\mathbf{n}_i$ with the formulae in the table below. Then we add a successor prestate of Δ with the label $\Gamma_{/i}$ and then continue the finitary pretableau construction as normally.

formula	replacement	formula	replacement
$\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_i}(\varphi)$	φ	$\neg\mathbf{G}_{\mathsf{A}}^{\mathbf{n}_i}(\varphi)$, $\neg\mathbf{G}_{\mathsf{E}}^{\mathbf{n}_i}(\varphi)$	$\neg\varphi$
$\mathbf{U}_{\mathsf{E}}^{\mathbf{n}_i}(\varphi,\psi)$, $\mathbf{U}_{\mathsf{A}}^{\mathbf{n}_i}(\varphi,\psi)$	ψ	$\neg\mathbf{U}_{\mathsf{A}}^{\mathbf{n}_i}(\varphi,\psi)$	$\neg\psi$

Note that here $\mathbf{n}_i$ is intuitively "lowered to zero", as e.g. $\neg\mathbf{G}_{\mathsf{A}}^{0}(\varphi) = \neg\mathbf{G}_{\mathsf{E}}^{0}(\varphi) = \neg\varphi$.

Apart from the changes described above, the finitary pretableau for $\mathrm{CTL}_{\mathbf{FB}}^{par}$ is constructed in the same way as the infinitary pretableau for $\mathrm{CTL}_{\mathbf{FB}}$.

▶ **Example 27.** The formula $\eta = \mathsf{EG}\,p \wedge \neg(p \wedge \mathsf{EX\,EG}\,p)$ has the finitary pretableau given in Figure 5. Here the procedure ParElim is applied when the prestates P_{0000} and P_{000020} are created. Note that this pretableau is finite and rather simple, but it still (in some sense) encodes all the relevant information of the corresponding infinitary pretableau in Example 17.

The prestate and state eliminations are done exactly as in the infinitary tableau for $\mathrm{CTL}_{\mathbf{FB}}$; they always terminate in finite number of steps and produce a finite final tableau. We will show that the final finitary tableau for a formula φ is open if and only if φ is satisfiable.

4.2 Equivalence of the two tableaux and their complexity

▶ **Theorem 28.** *The infinitary tableau for a formula $\eta \in \mathrm{CTL}_{\boldsymbol{FB}}$ is open if and only if the finitary tableau for η is open.*

A detailed proof sketch for this equivalence is given in the appendix, but we explain here the main ideas. The infinite branchings in the infinitary tableau are replaced with only a single branch in the finitary tableau. For example, for $\neg\mathsf{EG}\,\varphi$, (typically) in the infinitary tableau there is an infinite branching such that for each $n \in \mathbb{N}$, there is an offspring state in which $\neg\mathsf{EX}\,\mathbf{G}^{n}_{\mathsf{E}}(\varphi)$ occurs, while in the finitary tableau, the single $\mathrm{CTL}^{par}_{\mathbf{FB}}$-formula $\neg\mathsf{EX}\,\mathbf{G}^{\mathbf{n}_i}_{\mathsf{E}}(\varphi)$, for some $i \in \mathbb{N}^{+}$, is used in the place of all these formulae. We can show that for sufficiently large values of n, all the corresponding branches in the infinitary tableau have essentially the same structure. Furthermore, it suffices to only consider these high values of n when checking whether a state gets eliminated from the infinitary tableau.

Note that in the rules of the finitary tableau, the iteration parameter $\mathbf{n}_i$ is treated as an (actual) iteration counter n except that its value is not lowered normally in transitions. But if the same state is to be repeated in the finitary tableau, then one of the iteration parameters is replaced with the value zero (by the procedure ParElim). In the corresponding situation in the infinitary tableau, the similar states could be repeated until some of the iteration counters goes to zero. When considering only sufficiently large values of n, we may suppose that n is larger than all the iteration counters that have been set before the value of n has been fixed. Therefore n goes to zero only after all iteration counters set earlier have gone to zero. The same thing happens with the finitary tableau rules, as the iteration parameters $\mathbf{n}_i$ are lowered to zero in the order in which they have been introduced.

As seen above, the iteration parameter $\mathbf{n}_i$ is treated as if it had "as large a value as possible." Therefore we can show that the branches in the finitary tableau where $\mathbf{n}_i$ is used in the place of n, have essentially the same structure as the branches in the infinitary tableau where sufficiently large values of n are used. As only these branches are relevant for checking whether the infinitary tableau closes, it follows that the two tableaux are equivalent.

▶ **Corollary 29.** *The finitary tableau for $\mathrm{CTL}_{\boldsymbol{FB}}$ is sound and complete.*

▶ **Theorem 30.** *The complexity of the finitary tableau for $\mathrm{CTL}_{\boldsymbol{FB}}$ is* EXPTIME*-complete.*

The upper bound is easy to see by first noticing that the set of different labels of states in the finitary tableau is exponential with respect to the size of the input formula. The lower bound is obtained by a simple translation from the bi-modal logic with an additional reflexive-transitive closure operator. See more details in the Appendix.

▶ **Corollary 31.** *The satisfiability problem of $\mathrm{CTL}_{\boldsymbol{FB}}$ is decidable and* EXPTIME*-complete.*

5 Concluding remarks

The motivation for the logic $\mathrm{CTL}_{\mathbf{FB}}$ introduced here was two-fold: natural game-theoretic semantics and uniform boundedness of the time limit for satisfaction of eventualities across all branches. Both apply well beyond CTL, e.g. also to produce respective versions of the logics CTL* and ATL, to be studied further. While $\mathrm{CTL}_{\mathbf{FB}}$ is a simple semantic variation of CTL, it turns out to display some essentially different semantic features, the most striking being the lack of finite model property. That, in particular, makes capturing its validities more difficult and requiring an infinitary Hilbert-style axiomatization which will be presented in a subsequent work. An implementation of the finitary tableau method developed here will also be considered in future. Lastly, symbolic model checking of $\mathrm{CTL}_{\mathbf{FB}}$ on some classes of finitely presented infinite models is another natural direction for future work.

Acknowledgements We thank the reviewers for useful remarks.

References

1 R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.
2 Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *Proc. of TACAS'99*, pages 193–207, 1999.
3 P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. CUP, 2001.
4 E. M. Clarke and E. A. Emerson. Design and synthesis of synchronisation skeletons using branching time temporal logic. In *Logics of Programs*, pages 52–71. Springer, 1981.
5 Stéphane Demri, Valentin Goranko, and Martin Lange. *Temporal Logics in Computer Science*. Cambridge University Press, 2016.
6 Valentin Goranko, Antti Kuusisto, and Raine Rönnholm. Game-theoretic semantics for alternating-time temporal logic. In *Proc. of AAMAS 2016*, pages 671–679, 2016.
7 Valentin Goranko, Antti Kuusisto, and Raine Rönnholm. Game-theoretic semantics for alternating-time temporal logic. arXiv:1602.07667 [math.LO], submitted, 2017.
8 Wojciech Penczek, Bozena Wozna, and Andrzej Zbrzezny. Bounded model checking for the universal fragment of CTL. *Fundam. Inform.*, 51(1-2):135–156, 2002.
9 Bozena Wozna. ACTLS properties and bounded model checking. *Fundam. Inform.*, 63(1):65–87, 2004.
10 Wenhui Zhang. Bounded semantics. *Theor. Comput. Sci.*, 564:1–29, 2015.

A Appendix

Proof sketch of Theorem 16

One direction is immediate by Lemma 15, for $\Gamma = ecl(\eta)$. For the converse, suppose $\eta \in \mathrm{H}(s_0)$ for some state s_0 in some Hintikka structure $\mathcal{H} = (\mathrm{S}, \mathrm{R}, \mathrm{H})$. We define the interpreted transition system $\mathcal{M} = (\mathrm{S}, \mathrm{R}, \Phi, \ell)$ where ℓ is a state description in S defined as follows: $\ell(s) := \mathrm{PROP} \cap \mathrm{H}(s)$.

We show by induction on the main components of $\varphi \in ecl(\eta)$ that for every $s \in \mathrm{S}$:

if $\varphi \in \mathrm{H}(s)$ then $\mathcal{M}, s \models \varphi$.

The details are fairly routine. The cases of the temporal operators follow directly from Lemma 4.

Now, since $\eta \in \mathrm{H}(s_0)$ we have that $\mathcal{M}, s_0 \models \eta$.

Proof sketch of Lemma 21

Suppose the elimination phase terminates at some ordinal β, i.e. no states are eliminated from $\mathcal{T}^{\eta}_{\beta}$. It suffices to show, by transfinite induction on the ordinal number of elimination steps $\alpha \leq \beta$ taken in the elimination phase, that no satisfiable state $\Delta \in \mathcal{T}^{\eta}_{\alpha}$ is removed by an application of $\textsc{StateElim}^{\mathrm{CTL}_{\mathbf{FB}}}$ to $\mathcal{T}^{\eta}_{\alpha}$. For that purpose we will prove a somewhat stronger inductive claim, viz., that for every ordinal $\alpha \leq \beta$:

1. If $\Delta \in \mathcal{T}^{\eta}_{\alpha}$ is satisfiable, then for any $\mathsf{EX}\,\varphi \in \Delta$ there is a satisfiable state $\Psi_{\varphi} \in \mathcal{T}^{\eta}_{\alpha}$ such that $\Delta \xrightarrow{\mathsf{EX}\,\varphi} \Psi_{\varphi}$ in $\mathcal{T}^{\eta}_{\alpha}$.

2. If $\Delta \in \mathcal{T}^\eta_\alpha$ is satisfiable, then for any $\neg \mathsf{AX}\, \varphi \in \Delta$ there is a satisfiable state $\Psi_{\neg\varphi} \in \mathcal{T}^\eta_\alpha$ such that $\Delta \xrightarrow{\neg \mathsf{AX}\, \varphi} \Psi_{\neg\varphi}$ in $\mathcal{T}^\eta_\alpha$.
3. All satisfiable states in $\mathcal{T}^\eta_0$ are still present in $\mathcal{T}^\eta_\alpha$.

Note that the inductive hypothesis refers simultaneously to all satisfiable $\Delta \in \mathcal{T}^\eta_\alpha$ and all successor formulae $\mathsf{EX}\, \varphi \in \Delta$ and $\neg \mathsf{AX}\, \varphi \in \Delta$. We will only give the proof of claim 1; claim 2 is completely analogous, and then claim 3 follows immediately from these.

Assume the claim holds for all $\gamma < \alpha$ and consider 3 cases for α.

1. Let $\alpha = 0$. Take a satisfiable $\Delta \in \mathcal{T}^\eta_\alpha$ and $\mathsf{EX}\, \varphi \in \Delta$, Then, $\mathcal{M}, s \models \Delta$ for some rooted ITS $(\mathcal{M}, s)$. Recall, that all states $\Delta' \in \mathcal{T}^\eta_0$ such that $\Delta \xrightarrow{\mathsf{EX}\, \varphi} \Delta'$ in $\mathcal{T}^\eta_0$ are obtained as full expansions of the prestate $\Gamma = \{\varphi\} \cup \{\, \psi \mid \mathsf{AX}\, \psi \in \Delta \,\} \cup \{\, \neg\psi \mid \neg \mathsf{EX}\, \psi \in \Delta \,\}$. Since $\mathsf{EX}\, \varphi \in \Delta$, we have that $\mathcal{M}, s \models \mathsf{EX}\, \varphi$, hence there is an R-successor r of s in $\mathcal{M}$ such that $\mathcal{M}, r \models \varphi$. By the truth definition for successor formulae, it follows that $\mathcal{M}, r \models \Gamma$ because $\{\, \psi \mid \mathsf{AX}\, \psi \in \Delta \,\} \cup \{\, \neg\psi \mid \neg \mathsf{EX}\, \psi \in \Delta \,\}$ is satisfied at every R-successor of s in $\mathcal{M}$. Then, by Lemma 20, at least one full expansion Ψ_φ of Γ is satisfied by $(\mathcal{M}, r)$, and, by construction of the initial tableau, there is a state (with label) Ψ_φ in $\mathcal{T}^\eta_0$ such that $\Delta \xrightarrow{\mathsf{EX}\, \varphi} \Psi_\varphi$. Therefore, the state Δ cannot be removed from $\mathcal{T}^\eta_0$ by an application of the rule $\textsc{StateElim}^{\mathrm{CTL}_{\mathbf{FB}}}$.
2. Let α be a successor ordinal. Assuming the claim holds for all $\gamma < \alpha$, take a satisfiable $\Delta \in \mathcal{T}^\eta_\alpha$. For any $\mathsf{EX}\, \varphi \in \Delta$, by the argument for $\alpha = 0$ there is a satisfiable state Ψ_φ in $\mathcal{T}^\eta_0$ such that $\Delta \xrightarrow{\mathsf{EX}\, \varphi} \Psi_\varphi$ in $\mathcal{T}^\eta_0$, and hence, by the inductive hypothesis, Ψ_φ has remained intact in $\mathcal{T}^\eta_\alpha$. Therefore, Δ cannot be removed from $\mathcal{T}^\eta_\alpha$ by an application of the rule $\textsc{StateElim}^{\mathrm{CTL}_{\mathbf{FB}}}$.
3. In the case when α is a limit ordinal, $\mathcal{T}^\eta_\alpha$ is the intersection of all $\mathcal{T}^\eta_\gamma$ for $\gamma < \alpha$ and the argument is essentially the same as in the previous case, but now applied to each $\mathcal{T}^\eta_\gamma$ for $\gamma < \alpha$.

Proof of Lemma 20. Follows easily from Proposition 11.

Proof of Theorem 22. Follows immediately from Lemmas 20 and 21.

Proof sketch of Theorem 23

It suffices to show how from the open final tableau $\mathcal{T}^\eta$ one can construct a Hintikka structure satisfying η. That construction for $\mathrm{CTL}_{\mathbf{FB}}$ is similar to the construction for CTL described in [5, Chapter 13], to which the reader is referred for further details. It consists in extracting a subgraph of the final tableau $\mathcal{T}^\eta$ which contains an initial state with the input formula and represents any 'logical branch' of the search procedure encoded by $\mathcal{T}^\eta$. Such a 'logical branch' can be extracted by selecting just one alternative state from every set of states in $\mathcal{T}^\eta$ generated in the pretableau construction phase as full expansions of the same prestate. The resulting subgraph itself can be taken as a Hintikka structure, with a labelling function assigning to each state its own label. The proof that it is a Hintikka structure for $ecl(\eta)$ is straightforward by the construction of the tableau.

Proof sketch of Proposition 26

Since there are only countably many formulae, each state in a tableau is a finite or countably infinite set of formulae. Thus each state has at most countably many EX-formulae that require a successor prestate, one prestate for each EX-formula. In the process of selecting 'logical branches' (cf. the proof of Theorem 23), we eliminate all except for one of the states generated from a single prestate by the full expansion procedure. Thus, we observe that the out-degree of each state in the ultimate model is at most countably infinite and each such state is constructed after finitely many steps from the initial state. Therefore, the whole model is a countable union of countable sets of states.

Proof sketch of Theorem 28

For simplicity we suppose here that η does not have subformulae of the form $\mathsf{E}(\varphi \,\mathsf{U}\, \psi)$ or $\mathsf{A}(\varphi \,\mathsf{U}\, \psi)$, since these can be treated in a very similar way to the formulae $\mathsf{EG}\,\varphi$ and $\mathsf{AG}\,\varphi$. As the rules of the infinitary and the finitary tableau differ now only in those rules related to $\mathsf{EG}\,\varphi$, $\neg\mathsf{AG}\,\varphi$ and $\neg\mathsf{EG}\,\varphi$, we compare how these formulae are treated in the two tableaux. In the infinitary tableau, the rules for these formulae typically create infinite branchings, with a branch for every iteration counter $n \in \mathbb{N}$, while in the finitary tableau an iteration parameter $\mathbf{n}_i$ is used in the place of every n. We aim to show that it suffices to consider only certain 'sufficiently large' values of n in order to determine whether the infinitary tableau closes or not. Furthermore, we will argue that a branch with such a large value of n has (essentially) the same structure as a branch in the finitary tableau with the iteration parameter $\mathbf{n}_i$. The equivalence of the two tableaux will then be easy to see from these observations.

We will first define explicitly what we mean above by 'sufficiently large' value of n. For this, consider a prestate Γ whose label contains some formulae of the form $\mathsf{EG}\,\varphi$, $\neg\mathsf{AG}\,\varphi$ or $\neg\mathsf{EG}\,\varphi$. Let $m \in \mathbb{N}$ be the *smallest integer that is greater than all iteration counters* which currently occur in the label of Γ (in the infinitary tableau). We now define $n := m + k$, where k is the size of the formula η (i.e. the number of symbols in η). We will show that this (or any greater) value of n is the only one that needs to be considered as iteration counter when we create offspring states of Γ.

We first consider the case of the formula $\neg\mathsf{EG}\,\varphi$. If $\neg\mathsf{EG}\,\varphi$ occurs in some prestate Γ of the infinitary tableau, then for each value of $n \in \mathbb{N}$ the formula $\neg\mathsf{EX}\,\mathbf{G}_{\mathsf{E}}^{n-1}(\varphi)$ occurs in some offspring state of Γ; excluding those values of n that create patently inconsistent sets. The only difference between the different values of n here is that n is lowered to zero at a different stage in the tableau construction process. We first observe that if a state with certain value of n survives the state elimination process, then all the larger values will survive as well. Therefore if a state with some lower than $m + k$ value is not eliminated, then $m + k$ is not eliminated either. But on the other hand, if a state with some larger than $m + k$ value of n is not eliminated, then the state with $n = m + k$ cannot be eliminated either. In order to see this, we first observe the following facts:

1. $n = m + k$ cannot go down to zero before all iteration counters that have been set before n have gone down to zero.
2. After all other other iteration counters set before $n = m + k$ have gone to zero, all the nested temporal operators EX or AX will be removed before n goes to zero.

So after $n = m + k$ has gone to zero, the elimination of the reached state depends only on the *values set after* n. If this state will be eliminated, then it is easy to see that it would be eliminated even if some larger than $m + k$ value of n was selected, since all the values, set after fixing n, can always be arbitrarily larger than n.

The iteration parameter $\mathbf{n}_i$ in the finitary tableau is treated essentially the same way as the value $n = m + k$ in the infinitary tableau. This is simply because $\mathbf{n}_i$ cannot be 'lowered to zero' unless a state with the same label is to be repeated in the tableau, and all the iteration parameters set before $\mathbf{n}_i$ are first eliminated (recall the procedure $\mathsf{ParElim}$). Hence, it is easy to see that a state with $n = m + k$ in the infinitary tableau is eliminated if and only if the state with $\mathbf{n}_i$ in the finitary tableau is eliminated.

We then consider the case of $\neg\mathsf{AG}\,\varphi$. If $\neg\mathsf{AG}\,\varphi$ occurs in some prestate Γ of the infinitary tableau, then for each value of $n \in \mathbb{N}$ the formula $\neg\mathsf{AX}\,\mathbf{G}^n_{\mathsf{A}}(\varphi)$ occurs in some offspring state of Γ (except those values of n that create patently inconsistent sets). Hereafter the reasoning can be done similarly as for the formula $\neg\mathsf{EG}\,\varphi$.

Finally we consider the case of $\mathsf{EG}\,\varphi$. If $\mathsf{EG}\,\varphi$ occurs in some prestate Γ of the infinitary tableau, then all the formulae $\mathsf{EX}\,\mathbf{G}^{n-1}_{\mathsf{E}}(\varphi)$ $(n \in \mathbb{N})$ occur in *every* offspring state of Γ. Consider some offspring state Δ of Γ. When creating successor prestates for Δ, we must create a successor prestate Γ_n for each value $n \in \mathbb{N}$ and include $\mathbf{G}^{n-1}_{\mathsf{E}}(\varphi)$ in the label Γ_n. In order for Δ to survive the state elimination process, the following must hold for *every* $n \in \mathbb{N}$:

$$\Gamma_n \text{ has an offspring state } \Delta_n \text{ which survives the state elimination process.} \qquad (C_n)$$

We first observe that if the condition (C_n) does not hold for some $n < m + k$, then it cannot hold for $n = m + k$, either. But, with similar observations as in the case for $\neg\mathsf{EG}\,\varphi$, we can also see that if (C_n) does not hold for some $n > m + k$, then it cannot hold for $n = m + k$ either. Hence it suffices that we only consider the case $n = m + k$ to see whether (C_n) holds for every $n \in \mathbb{N}$. With similar reasoning as above we see that this case is essentially the same as the use of iteration parameter $\mathbf{n}_i$ in the finitary tableau.

Proof sketch of Theorem 30

(A) Upper bound. The number of formulae (possibly containing iteration parameters) that are used in a tableau for an input formula η, is linear in $|\eta|$. Thus, the number of different labels of states (prestates) is exponential in $|\eta|$. Therefore, the number of nodes appearing in the tableau construction is, likewise, exponential in $|\eta|$. The different construction maneuvers are computationally straightforward and can easily be done deterministically. Therefore, only exponentially many (in $|\eta|$) steps are required for carrying out the tableau procedure.

(B) Lower bound. Consider standard modal logic with the reflexive and transitive closure operator, i.e., the logic generated by the grammar

$$\varphi ::= p \,|\, \neg\varphi \,|\, (\varphi \wedge \varphi) \,|\Diamond\varphi \,|\, \Diamond^*\varphi$$

where p is a proposition symbol. The semantics is the standard one, with $\Diamond^*$ denoting the diamond interpreted by the reflexive-transitive closure of the binary relation interpreting $\Diamond$. It is well known that the satisfiability problem of this logic is EXPTIME-complete; see, e.g., [3]. Consider the simple translation T from this logic into $\mathrm{CTL}_{\mathbf{FB}}$, preserving atomic propositions and Boolean connectives, and mapping $T(\Diamond\varphi)$ to $\mathsf{EX}\,T(\varphi)$ and $T(\Diamond^*\varphi)$ to $\mathsf{EF}\,T(\varphi)$. It easy to see that this translation preserves equivalence, i.e., precisely the same points of any model satisfy φ and $T(\varphi)$.

A Relational Algebra for Streaming Tables Living in a Temporal Database World

Fabio Grandi[1], Federica Mandreoli[2], Riccardo Martoglia[3], and Wilma Penzo[4]

1 **DISI – University of Bologna, Bologna, Italy**
`fabio.grandi@unibo.it`
2 **FIM – University of Modena e Reggio Emilia, Modena, Italy**
`federica.mandreoli@unimo.it`
3 **FIM – University of Modena e Reggio Emilia, Modena, Italy**
`riccardo.martoglia@unimo.it`
4 **DISI – University of Bologna, Bologna, Italy**
`wilma.penzo@unibo.it`

Abstract

The recently introduced streaming table concept, a fully native representation of streaming data inside a DBMS, enabled modern data-intensive applications with one-time queries (OTQs) and continuous queries (CQs) capabilities on both streaming and standard relational tables. In this paper, we fully acknowledge the temporal nature of streaming tables and we propose to go one step further and integrate them in a temporal DBMS context, where time management is native. Our aim is to break the traditional barrier between the streaming and the temporal worlds, offering complete interoperability between streams and temporal data. To this end, we present a continuous temporal algebra supporting both OTQs and CQs seamlessly on streaming, standard and temporal relational tables. We further show how the transition from continuous to one-time semantics can be managed by defining suitable translation rules, which can also be used as a basis for the implementation of the proposed continuous algebra in a temporal DBMS.

1998 ACM Subject Classification H.2.1 Data Models, H.2.3 Query Languages

Keywords and phrases Continuous queries, Data streams, Relational algebra, Temporal DB

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.15

1 Introduction

Modern data-intensive applications, including for instance advanced surveillance (e.g., financial market enforcement), monitoring applications (e.g., air quality monitoring, Intelligent Transportation Systems - ITSs), military applications (e.g., platoon tracking), network applications (e.g., intrusion detection), more and more often require an increasingly wider range of data management capabilities in order to be fully supported. First of all, they need to manage very large quantities of continuously streaming data over which complex continuous queries (CQs) have to be efficiently executed. Unlike typical applications relying on Data Stream Management Systems (DSMSs) [23], such as sensor-data analysis and geospatial services, not only needs the most recent data to be retained and managed, but the whole incoming streams have to be stored as historical data in order to make them fully persistent and available to future analysis. Moreover, there is the need to be able to easily perform everything we are accustomed to do in a traditional Data Base Management Systems (DBMSs) context, including one-time queries (OTQs) also involving standard relational data, and possibly in a temporal DBMS context, allowing for complex temporal analytics and management of data versioning.

Being able to query recent, historical and non-temporal relational data by means of both CQs and OTQs in a uniform and powerful way, and with the expressive power of standard RDBMS's languages and algebras, is certainly a very strong requirement. The very diversified and lively panorama of data management systems, including streaming extensions to traditional DBMSs (e.g., [9, 10, 21]), big data processing engines, big data stores and stream processing frameworks (e.g., [7, 22]), is still trying to fully address it. One first step towards this objective has been performed in [6], where we proposed a fully native representation of streaming data into a DBMS. The introduction of a new kind of table, called the *streaming table*, allowed us to offer a persistent structure managing both historical and streaming data in a way completely transparent to users. Streaming tables can also be straightforwardly involved in OTQs and CQs, possibly together with standard relational tables, thus successfully merging the two worlds of RDBMSs and streaming data management.

Heading further in this direction, in this paper we aim at breaking another barrier: the one currently present between streaming and temporal data management. Even if streaming data is inherently temporal in nature, current systems and research proposals seem unable to build on this: on the one hand, temporal RDBMSs are not able to deal with streams, on the other hand current DSMSs and stream processing frameworks only provide limited temporal querying possibilities or do not deal with versioned data at all [7, 22]. By fully acknowledging the temporal nature of streaming data, we propose to integrate streaming tables in a temporal context where time management is native. Our final aim is to offer temporal RDBMSs' querying capabilities and data versioning even on streaming data, without overturning the common understanding of streaming data and CQs. In this way, data can be easily managed and queried by benefitting from the best of the relational, streaming and temporal worlds.

In this paper, we lay the semantic foundation for our objective by introducing a continuous temporal algebra for streaming, temporal, and standard tables. The proposed algebra, denoted as $\mathcal{CTA}$, extends with windowing operators a temporal algebra $\mathcal{TA}$ with well-defined semantics. Window operators take streaming tables as input and produce data snapshots at subsequent validity time points. Continuous and one-time queries are specified as algebraic expressions over the three different kinds of tables. In particular, OTQs can be formulated as $\mathcal{TA}$ expressions and their semantics relies on the traditional $\mathcal{TA}$ definition. CQs can be formulated as $\mathcal{CTA}$ expressions over standard and temporal tables and windowing expressions on streaming tables, and their semantics relies on a sampling operator that evaluates continuous temporal expressions at the required time points. In this way, we introduce a sort of *on-demand semantics*, where the CQs are executed when query results are needed and required data are available. This approach is highly flexible in that it allows to combine in a unified framework several continuous query features proposed in the literature (e.g., real time and historical analytics, backward and forward sliding windows, tumbling and hopping windows) but for which a well-founded semantics and full implementation agenda is still lacking, and to interoperate streaming data with non-temporal data and archival data stored in temporal tables in a consistent way.

As an application example in the context of financial market surveillance, we consider tracking of *insider trading* activities, that is the buying or selling of a security (e.g., stock options) while in possession of nonpublic material information (e.g., up/downgrade by a rating agency, unexpected revisions to earning results or projections, mergers and acquisitions news) about the security [17]. Since insider trading undermines investor confidence in the fairness and integrity of the financial markets, control bodies like the SEC have the detection and prosecution of insider trading violations as one of their market surveillance and enforcement priorities. Whereas early (i.e., when the material information is not yet publicly available) detection of insider trading patterns could allow to prevent frauds [13], their *a posteriori* (i.e., when the material information is of public domain) verification is most important for triggering and pursuing prosecution of illegal activities. Assume that option trading data are available through the streaming table `OPTION_TRADES`, with schema

`OPTION_TRADES(OPTION,STOCK,CLASS,STRIKE,EXPIR,CONTRACTS`$|T$`)`

automatically fed by the stock market information system, containing data concerning the negotiated option, the underlying stock, the trade timestamp (i.e., the implicit attribute T defined with a granularity of one second), the option class (call or put), strike price and expiration, and the number of contracts traded. Further, assume that relevant news are selected by several sources (e.g., press releases and financial news stories) and stored in a temporal table `NEWS`, with schema

`NEWS(STOCK,TYPE,SOURCE`$|T$`)`

containing information about the publication date (i.e., the implicit attribute T defined with a granularity of one day), the mentioned stock, the news type and source. In order to trigger an investigation procedure, an anomalous trading volume of an option, preceding by at most one week the public release of some relevant news concerning the underlying stock, needs to be identified. As anomalous volume, we consider a daily trading volume ten times higher than the average daily volume over the past month. Following our proposal, the suspect stock-day pairs deserving further investigations could be easily retrieved via a continuous query running at the beginning of each trading day and performing the temporal join between the relevant time windows of the streaming table `OPTION_TRADES` and the standard table `NEWS`. Notice that such a detection query could not be executed in a "classical" stream management system, because, when the windows used for computing volumes are evaluated, the relevant news have not been published yet. Hence, stream data must be stored in a streaming table for our purpose, and time windows evaluated in a delayed mode (i.e., when all relevant news will be available) as actually supported by the CQ on-demand semantics. The temporal join involved in the query will produce temporally consistent solutions over the past time window outputs of the streaming table `OPTION_TRADES` and the temporal table `NEWS`.

After revising the notion of streaming table (Sec. 2), this paper provides the following contributions:

- it introduces the continuous temporal algebra $\mathcal{CTA}$ supporting both OTQs and CQs seamlessly on streaming, standard and temporal relational tables (Sec. 3 and Sec. 4);
- it proposes a correct translation of the continuous temporal model presented so far into the temporal model where continuous queries are transformed into standard temporal queries, making it possible to instantiate the framework in the context of a standard temporal RDBMS (Sec. 6).

The paper is complemented by Sec. 5, expressing the reference example in the $\mathcal{CTA}$ algebra and Sec. 7, comparing our contribution with the state of the art and drawing conclusions.

2 Preliminaries

The continuous temporal data model we propose relies on a multi-temporal relational model [16], where temporal and non-temporal standard tables coexist, extended with streaming tables. In this Section we provide some preliminaries for its specification by reviewing the notion of streaming table, first introduced in [6].

First of all, we assume a discrete, ordered and unbounded time domain $\mathcal{T} = \{0, 1, 2, \ldots, \infty\}$ composed of *chronons* [18], where 0 stands for the earliest time. A chronon is a non-decomposable time interval of fixed unit duration used to represent time instants in the discrete model. We further assume that $\mathcal{T}$ has the semantics of *valid time* [18]. In order to represent a duration of time, we assume *time spans* [18] belong to a domain $\mathcal{I}$ composed of all possible multiples of a chronon duration (including the unbounded value ∞).

As far as the temporal model is concerned, a temporal relation R with explicit schema $R(A_1, \ldots, A_n)$, with $A_i \in \mathcal{A}$ $(1 \leq i \leq n)$ where $\mathcal{A}$ is the set of attribute names, is represented as $R(A_1, \ldots, A_n|T)$ where T is the implicit timestamp attribute with domain $\mathcal{T}$. If r is a tuple from R then $r.A_i$ denotes the value of A_i in r and $T(r)$ denotes the tuple timestamp. Moreover, given any time instant $t \in \mathcal{T}$, we denote with R^t the content of R at time t. Notice that we assume here an *abstract temporal database*, according to the terminology introduced in [11], to be used as a representation-independent data model. As far as non-temporal tables are concerned, we assume they are virtually converted to temporal tables to be interoperated with temporal tables and streaming tables by using a suitable temporal conversion map [8]. In particular, we assume each non-temporal table R to be virtually converted to a temporal table $R' = \{(r, \tau) \,|\, r \in R, \tau \in \mathcal{T}\}$.

As far as streams are concerned, we adopt the definition of continuous data stream (or simply stream) provided in [3], that is a potentially infinite stream of timestamped relational tuples having a fixed schema. A *streaming table* [6] is a relational table where streaming data enter and turn historical by remaining stored for a user-defined long period, ideally forever. Any streaming table inherits the temporal nature of the data it stores. Specifically, it is an event table [26, Ch. 16], that is a special kind of temporal table that stores event data and their occurrence time, for a limited time span named *historical period*. As time goes by, we assume the oldest data exiting the historical period are subject to *vacuuming* [26, Ch. 23].

▶ **Definition 1** (Streaming table). A streaming table S with explicit schema $S(A_1, \ldots, A_n)$ and historical period $hp \in \mathcal{I}$ is an event table, denoted as S_{hp}, with schema $S(A_1, \ldots, A_n|T)$, where T is the implicit timestamp attribute. The content of S_{hp} at the time instant $t \leq now$ (where now represents current time), i.e. S^t_{hp}, is the set of tuples such that the timestamp of each tuple s satisfies $T(s) \geq \max(t - hp, 0)$. If positive, $t - hp$ represents the chronon preceding t by a time span of hp chronons. A streaming table is subject to *continuous writes* in timestamp order, that is for any s_1 and s_2 in S^t_{hp}, $T(s_1) < T(s_2)$ iff s_1 arrived before s_2.

For ease of notation, in the following, whenever possible, we will use S in place of S_{hp}. In practice, in order to implement this insertion semantics, systems cope with out-of-order and skewed inputs. Interested readers can refer to [27] for an in-depth discussion of this aspect. In this paper we assume an input manager that guarantees in-order tuple arrival.

In the following, let $\mathcal{R}$ be the set of all temporal tables and $\mathcal{S}$ be the set of all streaming tables. Notice that $\mathcal{S} \subseteq \mathcal{R}$, as streaming tables are a special kind of temporal tables (one effect of the insertion semantics is that timestamps in a streaming table are always bounded by the current time now, whereas temporal tables may also contain timestamps greater than now to represent proactively inserted future data).

3 Querying streaming tables with OTQs: the temporal algebra $\mathcal{TA}$

In OTQs a streaming table is dealt with as a standard (event) table and it undergoes queries expressed in a relational algebra $\mathcal{TA}$ with the following temporal operators: selection σ^T, projection π^T, Cartesian product $\times^T$, union $\cup^T$, difference $-^T$, and grouping ϑ^T. Each of

Table 1 $\mathcal{TA}$ operator semantics.

Operator	Signature	Operator semantics
Selection	$\sigma^T : \mathcal{R} \times \mathcal{P} \to \mathcal{R}$	$\sigma_p^T(R) := \{(r,\tau) \mid (r,\tau) \in R \wedge p(r)\}$
Projection	$\pi^T : \mathcal{R} \times 2^{\mathcal{A}} \to \mathcal{R}$	$\pi_B^T(R) := \{(r.B,\tau) \mid (r,\tau) \in R\}$
Cart. prod.	$\times^T : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$	$R_1 \times^T R_2 := \{(r_1 \circ r_2,\tau) \mid (r_1,\tau) \in R_1 \wedge (r_2,\tau) \in R_2\}$
Union	$\cup^T : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$	$R_1 \cup^T R_2 := \{(r,\tau) \mid (r,\tau) \in R_1 \vee (r,\tau) \in R_2\}$
Difference	$-^T : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$	$R_1 -^T R_2 := \{(r,\tau) \mid (r,\tau) \in R_1 \wedge (r,\tau) \notin R_2\}$
Grouping	$\vartheta^T : \mathcal{R} \times 2^{\mathcal{A}} \times 2^{\mathcal{F}} \to \mathcal{R}$	${}_B\vartheta_F^T(R) := \{(r.B \circ Z,\tau) \mid (r,\tau) \in R \wedge$ $r_g = \{(r',\tau) \in R \mid r'.B = r.B\} \wedge Z = (f_1(r_g),\ldots,f_h(r_g))\}$
Extend	$\varepsilon : \mathcal{R} \times \mathcal{A} \to \mathcal{R}$	$\varepsilon_U(R) := \{(r \circ U,\tau) \mid (r,\tau) \in R \wedge U = \tau\}$
Timeslice	$\tau : \mathcal{R} \times \mathcal{T} \to \mathcal{R}$	$\tau_t(R) := \{(r,\tau) \mid (r,\tau) \in R \wedge \tau = t\}$

these temporal operators is a generalization of a standard relational operator where the T-superscript does not appear (derived operators, like the join $\bowtie^T$, can also be considered as usual). The definition of these operators is borrowed from [12] and, thus, $\mathcal{TA}$ supports a sequenced semantics in terms of extended snapshot reducibility. To this purpose, in order to support operators with predicates and functions that reference the timestamp, the additional extend operator ε_U copies the timestamp T to the additional attribute U. Notice that τ_t is an "extended" timeslice operator that maintains the timestamps, such that its result is still a temporal relation representing the *snapshot* valid at time t. Non temporal operators may be needed to express non-sequenced parts of queries (e.g., to join data belonging to different snapshots).

The semantics of the temporal algebra operators is shown in Table 1, where if (r,τ) is a tuple of a temporal relation with explicit schema $R(X)$ we consider r (with schema X) its explicit part and τ the tuple timestamp, $\mathcal{P}$ is the set of all well-defined predicates p over the explicit attributes X of R, $B \subset X$ is a subset of schema attributes, $\mathcal{F} = \{f_1,\ldots,f_k\}$ is a set of aggregation functions, and $\circ$ is a tuple concatenation operator. Notice that the Cartesian product applied to at least one streaming table returns a streaming table, the union operator returns a streaming table when both operands are streaming tables and the difference operator returns a streaming table when the first operand is a streaming table.

We assume readers are familiar with the syntax and semantics of relational queries[1] and we only provide a concise yet informal semantics of OTQs.

▶ **Definition 2** (Semantics of one-time queries over streaming and temporal tables). Given a $\mathcal{TA}$ expression $Q = E_{\mathcal{TA}}\,(S_1,\ldots,S_n,R_1,\ldots,R_m)$, over n streaming tables $S_1,\ldots,S_n$, with $n \geq 1$, and m temporal tables $R_1,\ldots,R_m$, with $m \geq 0$, its semantics is the result of the semantics of the involved temporal operators for which snapshot reducibility holds, that is for each $t \in \mathcal{T}$: $\tau_t(Q) = E_{\mathcal{TA}}\,(\tau_t(S_1),\ldots,\tau_t(S_n),\tau_t(R_1),\ldots,\tau_t(R_m))$.

4 Querying streaming tables with CQs: the continuous temporal algebra $\mathcal{CTA}$

As we have mentioned, streaming tables can also be involved in CQs. A continuous query Q^c is a query that is issued once, and then logically runs continuously until terminated by the user. Any streaming table S referenced in a continuous query must be accessed through

[1] Interested readers can refer to [2] for an in-depth study.

a sliding window w that specifies the boundaries of the range of tuples in S to be used for query evaluation. The continuous temporal algebra $\mathcal{CTA}$ we propose extends the set of windowing operators usually adopted in the streaming context [24], by generalizing their semantics to generate and operate on, possibly through aggregation, *sequences* of windows instead of single windows. Thus, $\mathcal{CTA}$ extends the temporal algebra $\mathcal{TA}$ ($\mathcal{TA} \subseteq \mathcal{CTA}$) with the introduction of such generalized windowing operators.

For ease of notation, we start introducing some utility operators. The definitions of $\mathcal{CTA}$ operators follow.

4.1 Utility operators

The following operators provide useful transformations of streaming tables.

▶ **Definition 3** (Substreaming)**.** The substreaming operator $\mathrm{Sub} : \mathcal{S} \times \mathcal{T}^2 \to \mathcal{S}$ restricts a streaming table $S \in \mathcal{S}$ to only tuples such that their timestamp belongs to an interval $[t_1, t_2]$:

$$\mathrm{Sub}_{[t_1,t_2]}(S) := \{(s,\tau) \mid (s,\tau) \in S \wedge t_1 \leq \tau \leq t_2\}.$$

For instance, the expression $\mathrm{Sub}_{[\text{2016-01-01 00:00:00, 2016-12-31 23:59:59}]}$(`OPTION_TRADES`) builds from `OPTION_TRADES` a streaming table containing the trades executed in 2016 only.

▶ **Definition 4** (Streaming Table Partition)**.** The partitioning operator $\zeta : \mathcal{S} \times 2^{\mathcal{A}} \to 2^{\mathcal{S}}$ partitions the streaming table $S \in \mathcal{S}$ into a set of streaming tables containing the tuples of $\mathcal{S}$ grouped by their attributes in B (as for the SQL group by mechanism, a partition is created for each combination of the values of the attributes $B_1, \ldots, B_k$ in B):

$$\zeta_B(S) := \{S' \mid (s,\tau) \in S \wedge S' = \{(s',\tau') \mid (s',\tau') \in S \wedge s'.B = s.B\}\}.$$

For instance, the expression $\zeta_{\texttt{CLASS}}$(`OPTION_TRADES`), partitioning `OPTION_TRADES` with respect to the values of `CLASS`, evaluates to a set containing two streaming tables, one containing all the trades on call-type options and the other containing all the trades on put-type options present in `OPTION_TRADES`.

4.2 $\mathcal{CTA}$ operators

In order to support continuous queries over streaming tables, $\mathcal{CTA}$ introduces two classes of operators: the former includes sliding window operators, the latter includes window flattening and aggregation operators. For the sake of clarity, we give an intuition of these two classes of operators that work in a complementary fashion: sliding window operators generate a sequence of portions of a given streaming table S according to a sliding window specification, thus resulting into a sequence of windows over S; window flattening operators and window aggregation operators reduce the result of a sliding window operator to a streaming table. Both classes of operators include one *standard* and one *partitioned* version of each operator.

4.3 Sliding window operators

In accordance with commonly adopted definitions of sliding windows [24], sliding window operators in $\mathcal{CTA}$ are time-based and count-based. Standard sliding window operators apply to a streaming table S and generate a temporal relation of streaming tables, each made of portions of S. More precisely, such a temporal relation is a streaming table, thus resulting in a streaming table of streaming tables (we denote by $\mathcal{S}^*$ the set of all possible streaming tables of streaming tables). Definitions of (backward and/or forward) standard sliding window operators are provided below.

▶ **Definition 5** (Time-based Sliding Window). The time-based sliding window operator $w^{\text{time}} : \mathcal{S} \times \mathcal{I}^2 \to \mathcal{S}^*$ over a streaming table $S \in \mathcal{S}$ creates a streaming table of streaming tables with a window size of duration $\omega_1 \geq 0$ before and $\omega_2 \geq 0$ after the timestamps around which it is computed:

$$w^{\text{time}}_{[\omega_1,\omega_2]}(S) := \{(S', \tau) \mid S' = \text{Sub}_{[\tau-\omega_1,\tau+\omega_2]}(S)\}.$$

For instance, the expression $w^{\text{time}}_{[\text{1week},0]}(\texttt{OPTION\_TRADES})$ defines a streaming table whose tuples, for each timestamp τ, are in turn streaming tables extracted from `OPTION_TRADES` by restricting it to the 1-week wide time window preceding τ.

▶ **Definition 6** (Count-based Sliding Window). The count-based sliding window operator $w^{\text{count}} : \mathcal{S} \times \mathbb{N}^2 \to \mathcal{S}^*$ over a streaming table $S \in \mathcal{S}$ creates a streaming table of streaming tables with a window containing the closest $n_1 \geq 0$ tuples valid before and the closest $n_2 \geq 0$ tuples valid after the timestamps around which it is computed:

$$w^{\text{count}}_{[n_1,n_2]}(S) := \{(S', \tau) \mid S' = \text{Sub}_{[t_1,t_2]}(S), |\text{Sub}_{[t_1,\tau-1]}(S')| = n_1, |\text{Sub}_{[\tau+1,t_2]}(S')| = n_2\}.$$

In the definition of (S', τ) above, in case $|\text{Sub}_{[t_1+1,\tau-1]}(S')| < n_1$ but $|\text{Sub}_{[t_1,\tau-1]}(S')| = n'_1 > n_1$, we assume $n'_1 - n_1$ tuples all valid at t_1 are, as customary in the streaming context, non-deterministically chosen and removed from S'. Similarly, in case $|\text{Sub}_{[\tau+1,t_2-1]}(S')| < n_2$ but $|\text{Sub}_{[\tau+1,t_2]}(S')| = n'_2 > n_2$, we assume $n'_2 - n_2$ tuples all valid at t_2 are non-deterministically chosen and removed from S'. For instance, the expression $w^{\text{count}}_{[1,1]}(\texttt{OPTION\_TRADES})$ defines a streaming table whose tuples, for each timestamp τ, are in turn streaming tables extracted from `OPTION_TRADES` and containing one tuple immediately preceding τ, the tuples valid at τ (if they exist), and one tuple immediately following τ.

The partitioned version of each sliding window operator applies to a streaming table S and generates a set of streaming tables of streaming tables, resulting from the application of the corresponding standard sliding window operator to each streaming table obtained by the partition of S according to a given set of attributes B. The operators' definitions follow.

▶ **Definition 7** (Time-based Partitioned Window). The time-based partitioned sliding window operator $W^{\text{time}} : \mathcal{S} \times 2^{\mathcal{A}} \times \mathcal{I}^2 \to 2^{\mathcal{S}^*}$ over a streaming table $S \in \mathcal{S}$ creates a set (of streaming tables of streaming tables) composed of the time-based sliding windows (with a window size of duration $\omega_1 \geq 0$ before and $\omega_2 \geq 0$ after the timestamps around which it is computed) computed over the streaming tables into which S is partitioned according to the attributes in B:

$$W^{\text{time},B}_{[\omega_1,\omega_2]}(S) := \{w^{\text{time}}_{[\omega_1,\omega_2]}(S') \mid S' \in \zeta_B(S)\}.$$

For instance, the expression $W^{\text{time},\texttt{OPTION}}_{[0,\text{1hour}]}(\texttt{OPTION\_TRADES})$, involving a time-based partitioned window, defines a set of streaming tables, each one corresponding to a different option, composed of the streaming tables whose tuples timestamped with τ are the streaming tables containing the trades involving that option negotiated in the hour that follows τ.

▶ **Definition 8** (Count-based Partitioned Window). The count-based partitioned sliding window operator $W^{\text{count}} : \mathcal{S} \times 2^{\mathcal{A}} \times \mathbb{N}^2 \to 2^{\mathcal{S}^*}$ over a streaming table $S \in \mathcal{S}$ creates a set (of streaming tables of streaming tables) composed of the count-based sliding window (with a window containing the closest $n_1 \geq 0$ tuples valid before and the closest $n_2 \geq 0$ tuples valid after the timestamps around which it is computed) computed over the streaming tables into which S is partitioned according to the attributes in B:

$$W^{\text{count},B}_{[n_1,n_2]}(S) := \{w^{\text{count}}_{[n_1,n_2]}(S') \mid S' \in \zeta_B(S)\}.$$

For instance, the expression $W^{\text{count},\texttt{STOCK}}_{[10,0]}(\texttt{OPTION\_TRADES})$, involving a count-based partitioned window definition, denotes a set of streaming tables, each for a different stock, composed of the streaming tables whose tuples timestamped with τ are the streaming tables containing the 10 most recent option trades preceding τ concerning that stock.

4.4 Window flattening operators

Window flattening operators allow for *normalizing* a streaming table (or a set of streaming tables) of streaming tables resulting from a time-based or a count-based sliding window operator to a flat streaming table. As for sliding window operators, the window flattening operator is introduced both in its standard and in its partitioned version.

▶ **Definition 9** (Window Flattening)**.** The window flattening operator $\varphi : \mathcal{S}^* \to \mathcal{S}$ over a streaming table of streaming tables w creates a streaming table composed of the tuples belonging to the streaming tables in w valid at the time at which the flattening is computed:

$$\varphi(w) := \{(\varepsilon_U(s), \tau) \mid (S, \tau) \in w \land s \in S\}.$$

For instance, the expression $\varphi(w^{\text{time}}_{[\text{1hour},0]}(\texttt{OPTION\_TRADES}))$, involving a window flattening operator, builds a streaming table whose tuples with timestamp τ are all the tuples belonging to the streaming table $w^{\text{time}}_{[\text{1hour},0]}(\texttt{OPTION\_TRADES})$ valid at τ, that is belonging to the 1-hour wide time window of OPTION_TRADES preceding τ. Such tuples come out all timestamped with τ in the result but preserve the value of the original timestamp they had in OPTION_TRADES converted into an explicit attribute U.

▶ **Definition 10** (Partitioned Window Flattening)**.** The partitioned window flattening operator $\Phi : 2^{\mathcal{S}^*} \to \mathcal{S}$ over a set of streaming tables of streaming tables W creates a streaming table composed of the tuples belonging to the streaming tables in $w \in W$ valid at the time at which the flattening is computed:

$$\Phi(w) := \{(\varepsilon_U(s), \tau) \mid w \in W \land (S, \tau) \in w \land s \in S\}.$$

For instance, the expression $\Phi(W^{\text{count},\texttt{OPTION,CLASS}}_{[4,0]}(\texttt{OPTION\_TRADES}))$, involving a partitioned flattening operator, retrieves the data necessary to display, for each time point and for each stock option, a book with the five latest put trades and the five latest call trades.

4.5 Window aggregation operators

Window aggregation operators are defined to compute aggregate data over time-based or count-based sliding windows, according to a set of aggregation functions $\mathcal{F}$. As for operators above, both standard and partitioned versions of the window aggregation operator are provided.

▶ **Definition 11** (Window Aggregation)**.** The sliding window aggregation operator $\vartheta : \mathcal{S}^* \times 2^{\mathcal{F}} \to \mathcal{S}$ over a streaming table of streaming tables w creates a streaming table having as attributes the values of the aggregates in $F = \{f_1, \ldots, f_h\}$ calculated over the streaming table in w valid at the time at which the aggregation is computed:

$$\vartheta_F(w) := \{(Z, \tau) \mid (S, \tau) \in w \land Z = (f_1(S), \ldots, f_h(S))\}.$$

The window aggregation operator ϑ can be used in queries for computing aggregate data over time-based or count-based sliding windows. For each time point τ, aggregates can

Table 2 $\mathcal{CTA}$ operators combinations.

α	ω	legal $\mathcal{CTA}$ expressions involving windows
$\varphi \mid \vartheta$	$w^{\text{time}} \mid w^{\text{count}}$	$\varphi(w^{\text{time}}_{[\omega_1,\omega_2]}(S))$, $\varphi(w^{\text{count}}_{[n_1,n_2]}(S))$, $\vartheta_F(w^{\text{time}}_{[\omega_1,\omega_2]}(S))$, $\vartheta_F(w^{\text{count}}_{[n_1,n_2]}(S))$
$\Phi \mid \Theta$	$W^{\text{time}} \mid W^{\text{count}}$	$\Phi(W^{\text{time},B}_{[\omega_1,\omega_2]}(S))$, $\Phi(W^{\text{count},B}_{[n_1,n_2]}(S))$, ${}_B\Theta_F(W^{\text{time},B}_{[\omega_1,\omega_2]}(S))$, ${}_B\Theta_F(W^{\text{count},B}_{[n_1,n_2]}(S))$

be computed over the timestamped tuples belonging to the streaming table S in w valid at time τ; aggregate functions `MIN`, `MAX`, `COUNT` (and `SUM`, `AVG` if numeric), `FIRST-VALUE`, `LAST-VALUE`, `NTH-VALUE(n)`, can be used on the explicit attributes of S, whereas aggregate functions `FIRST`, `LAST`, `DURATION` can be used on the timestamps of S. For instance, the expression $\vartheta_{\texttt{DURATION}}(w^{\text{count}}_{[9,0]}(\texttt{OPTION\_TRADES}))$ returns, for each time point, the width of the time window containing the 10 most recent option trades.

▶ **Definition 12** (Partitioned Window Aggregation)**.** The partitioned sliding window aggregation operator $\Theta : 2^{\mathcal{S}^*} \times 2^{\mathcal{A}} \times 2^{\mathcal{F}} \rightarrow \mathcal{S}$ over a set of streaming tables of streaming tables W creates a streaming table having as attributes the grouping attributes in B and the values of the aggregates in $F = \{f_1, \dots, f_h\}$ calculated over the streaming tables w belonging to W valid at the time at which the aggregation is computed:

$$ {}_B\Theta_F(W) := \{(S.B \circ Z, \tau) \mid w \in W \wedge (S, \tau) \in w \wedge Z = (f_1(S), \dots, f_h(S))\} \, . $$

The partitioned window aggregation operator Θ can be used in queries for computing aggregate data over partitioned time-based or count based sliding windows. Also in this case, aggregate functions acting on explicit attributes or timestamps can be used. For instance, the expression ${}_{\texttt{EXPIR}}\Theta_{\texttt{COUNT(OPTION)}}({}_{\texttt{EXPIR}}W^{\text{time}}_{[0.5\text{hour},0.5\text{hour}]}(\texttt{OPTION\_TRADES}))$, at each timepoint and for each expiration date, returns the number of options with that expiration date traded in a 1-hour wide time window centered around the timepoint.

Notice that the four types of sliding window operators (w^{time}, w^{count}, W^{time}, W^{count}) can be freely declared in querying streaming tables but they can be used in an algebraic expression only in combination with either a window flattening operator (φ, Φ) or a window aggregation operator (ϑ, Θ), which always produce a streaming table. Formally, a windowing expression applied to a streaming table S is of the form $\alpha(\omega(S))$ where the possible combinations are shown in Table 2. Notice that, standard (resp., partitioned) sliding window operators can only be combined with their stardard (resp., partitioned) window flattening or window aggregation counterparts. These constraints ensure that the value of continuous expressions augmenting $\mathcal{TA}$ is always a streaming table, so that the resulting continuous algebra $\mathcal{CTA}$ is closed with respect to (streaming and) temporal tables.

4.6 Supporting continuous queries

In order to support continuous queries, a sampling operator is formally introduced to evaluate an algebraic expression expressed in the continuous temporal algebra $\mathcal{CTA}$ at the required time points. In line with many CQ specification syntaxes (e.g. [3]), we assume a continuous query is always equipped with a *slide* parameter sl representing the query evaluation period, and with a further optional *alignment* parameter a specifying the position of the evaluation point within the evaluation period. Moreover, we also consider a *delay* parameter δ specifying that the evaluation of the query at time t has actually to be executed at time $t + \delta$. The slide parameter can be either a user-supplied time span or the special parameter `REALTIME`, that means that the query is re-evaluated as new tuples arrive. The alignment value is expressed as a period of time to be counted from the beginning of the time granules representing the evaluation periods (and is ignored in case sl=`REALTIME`). Parameters sl, a and δ used for sampling $\mathcal{CTA}$ expressions allow to generalize the usage of the so-called *tumbling windows* (and *hopping windows*) for producing continuous query results.

▶ **Definition 13** (Sampling Operator)**.** At execution time t, the sampling operator $\xi : \mathcal{CTA} \times \mathcal{T} \times \mathcal{I}^4 \to \mathcal{S}$, with an historical period parameter hp, a sliding parameter sl, an alignment parameter a, causes the evaluation of the continuous algebra expression $E \in \mathcal{CTA}$ at time points $t_0, t_1, \ldots, t_k \leq t$ only, where $t_i = (\lceil \frac{t-hp-a}{sl} \rceil + i) \cdot sl + a$. If a delay parameter δ is specified, it forces the evaluation of the expression E to be actually executed at time $t + \delta$:

$$\xi^{t,\delta}_{hp,sl,a}(E) := \bigcup_{i=0}^{k:t_k \leq t} \tau_{t_i}(E^{t+\delta}) .$$

For example, if sl="1 day", the continuous execution must produce one result per day: if the alignment parameter is a="30 minutes", the results are produced each day at "00:30" in the morning, whereas if the alignment parameter is a="16 hours", the results are produced each day at 4 p.m.. Notice that different results are produced with respect to the desired alignment, since time windows are defined with reference to the execution times, which depend on the alignment. For instance, assuming daily trading hours range from 9 a.m. to 4 p.m., the sliding window $w^{\text{time}}_{[\texttt{1day},0]}$(`OPTION_TRADES`) executed via a sampling with sl="1 day" and a="6 hours" includes all the trades executed the day before (from 9 a.m. to 4 p.m.) to contribute to a result produced daily at 6 a.m., but if it were executed with a="12 hours" it would include all the trades executed in the afternoon of the day before (from noon to 4 p.m.) and in the morning of the current day (from 9 a.m. to noon) to contribute to a result produced daily at noon.

The delay parameter, when specified, ensures that the evaluation of the expression E valid at time t is actually computed at time $t + \delta$: in general, the results (both valid at time t) computed at time t and at time $t + \delta$ may differ as some required contents of the temporal relations may not be available at time t yet, or even because their contents may have been retroactively changed after t (and also tuples in the streaming tables might be inserted with a little delay with respect to their validity, e.g., to enforce the right timestamp order). Consider, for instance, our example of insider trading detection that, in order to produce one result per trading day, needs to compare the trading volumes evaluated using the streaming data valid on a 1-month window preceding the execution time and on a 1-day window following the execution time (to this purpose, it would be sufficient to delay the execution at the end of the day). However, it also needs to join such volumes with the news concerning the same stocks published within one week. Since we can assume relevant news are selected and inserted by human analysts, we should also consider that they are likely inserted into the `NEWS` table retroactively, with the delay of some days with respect to their publication date. Hence, we should reasonably allow for a delay of, say, 10-15 days in the execution of the CQ in order to have all the relevant news available, otherwise the result of the join would always be empty and the insider trading cases could not be detected.

Notice that, when the streaming tables involved are defined at a finer time *granularity* than sl (e.g., sl="1 hour" but new tuples can be inserted into the streaming tables at every second), the values of E are usually defined for many more time points than required by the query. Hence, the sampling operator can be used to exactly specify the query execution timepoints of interest. This is also the reason for which we said in Sec. 2 that non-temporal tables (appearing in the expression E) are considered *virtually* converted into temporal tables that contain an infinite number of tuples: only the tuples timestamped with one of the timepoints of interest have actually to be generated.

According to the generally accepted definition of continuous query semantics [15], we define the semantics of a continuous query $Q^{c,\delta}_{hp,sl,a}$ denoted by an algebraic expression E in the continuous temporal algebra $\mathcal{CTA}$ to be equal to the sampling of E at the time points specified by the slide parameter sl and the alignment parameter a.

▶ **Definition 14** (Semantics of continuous queries over streaming and standard tables)**.** Let $E_{\mathcal{CTA}} = E_{\mathcal{TA}}(\alpha_1(\omega_1(S_1)), \ldots, \alpha_n(\omega_n(S_n)), R_1, \ldots, R_m)$ be an algebraic expression in $\mathcal{CTA}$ over n streaming tables $S_1, \ldots, S_n$, with $n \geq 1$, and m temporal tables $R_1, \ldots, R_m$, with $m \geq 0$, where $E_{\mathcal{TA}}$ is an expression in $\mathcal{TA}$, and α_i and ω_i, $i = 1, \ldots, n$, are window aggregation/flattening and sliding window operators, respectively.

The result at time t of the continuous query $Q^{c,\delta}_{hp,sl,a}$ with historical period parameter hp, slide parameter sl, alignment parameter a and delay parameter δ, expressed by $E_{\mathcal{CTA}}$ is the streaming table with historical period hp given by the sampling $\xi^{t,\delta}_{hp,sl,a}(E_{\mathcal{CTA}})$ of $E_{\mathcal{CTA}}$ at the time points specified by sl with alignment a, and evaluation delayed by δ, until t.

It is worth noting that, as the CQ semantics is founded on the sampling operator, we actually implement a CQ on-demand semantics that produce at the execution query time successive query evaluations at past query points, thus in a delayed mode (also when δ=0). Moreover, the semantics of joining streaming tables and temporal and standard tables as specified in the algebraic expression $E_{\mathcal{TA}}$ refers to the standard temporal semantics. In this way, we implement different kinds of joining semantics according to the involved tables. For instance, when temporal tables are involved, the joining results will be temporally consistent according to the required time points in the past. When, instead, standard tables are involved, the joining semantics allow users to "interoperate" current data with past streamed data.

5 Example reprise

In order to express the query solving our insider trading detection example problem presented in Sec. 1, the following partitioned sliding windows need to be defined:

$$W_D = W^{\text{time},\mathsf{STOCK}}_{[\mathtt{0,1day}]}(\mathtt{OPTION\_TRADES}), \quad W_M = W^{\text{time},\mathsf{STOCK}}_{[\mathtt{1month,0}]}(\mathtt{OPTION\_TRADES}).$$

The former is a 1-day wide sliding windows following the evaluation time, partitioned according to the STOCK values. The latter is a 1-month wide sliding windows preceding the evaluation time, partitioned according to the STOCK values. Then, for our convenience, we can define the following streaming table expressions involving partitioned window aggregation:

$$S_D = {}_{\mathtt{STOCK}}\Theta_{\mathtt{SUM(CONTRACTS)}}(W_D), \quad S_M = {}_{\mathtt{STOCK}}\Theta_{\mathtt{SUM(CONTRACTS)}}(W_M)$$

(implying the computation of the total number of contracts executed, for each timepoint, in the time windows W_D and W_M, respectively) to be used in the definition of the algebraic expression that follows:

$$\begin{aligned} E \;=\; & [\varepsilon_U(S_D) \bowtie_{S_D.\mathtt{STOCK}=\mathtt{NEWS.STOCK} \wedge (\mathtt{NEWS}.U - S_D.U) < \mathtt{1week}} \pi_{\mathtt{STOCK},U}(\varepsilon_U(\mathtt{NEWS}))] \\ & \bowtie^T_{S_D.\mathtt{STOCK}=S_M.\mathtt{STOCK} \wedge S_D.\mathtt{SUM(CONTRACTS)} > 10 \cdot S_M.\mathtt{SUM(CONTRACTS)}/30} S_M\,. \end{aligned}$$

The square brackets enclose a non-temporal join that embodies the non-sequenced part of the query, which is necessary to interoperate the tuples valid at time t of the streaming table S_D with the tuples of the NEWS relation valid at a time no later than 1 week with respect to t. To this purpose, the timestamping attributes of S_D and NEWS have to be made explicit via the extension operator ε to be referenced in the join predicate. Such join predicate also contains a conjunct imposing the equality of the STOCK value. The result is a streaming table whose timestamps are derived from S_D (the timestamp of NEWS has been projected out before executing the non-temporal join) and correspond to the execution time of the sliding window W_D. Once the expression in square brackets has been computed, a temporal join (on the streaming table timestamps) can be executed between the non-temporal join outcome and the streaming table S_M. The join predicate involves the equality of the STOCK value and the relationship between the results of the SUM(CONTRACTS) partitioned aggregates computed with Θ over S_M and S_D. In particular, the former (representing a daily volume) has to be greater than 10 times the value of the latter (representing the total volume computed over the preceding month) divided by 30 (yielding the average daily volume computed over the preceding month), in order to trigger an insider trading investigation. The continuous query result is finally given by the expression

$$\xi^{\text{now},15\texttt{day}}_{\texttt{1day},0}(\pi^{T}_{\texttt{STOCK}}(E)),$$

where the required sampling operator has been added, which causes the expression E to be evaluated (with a delay of 15 days to allow the NEWS table to be populated with relevant data) to produce a result each day at midnight containing the stocks suspect of illegal insider trading activity occurred on that day. The result is a temporal table with schema $(\texttt{STOCK}|T)$.

6 Translating CQs into OTQs (with Implementation on the Horizon)

In this section we propose a translation of the continuous temporal model presented so far into a new temporal model where continuous queries are transformed into temporal one-time queries. Furthermore, whereas the continuous model has been defined as an *abstract temporal model* (point-based), the new model is intended to be a *concrete temporal model* (interval-based) [11] amenable to implementation. In particular, the new temporal model can be implemented on a traditional relational DBMS following similar directions as presented in [12]. In fact, our final aim is to build comprehensive support for the continuous temporal model through a mixed stratum/built-in approach that relies on the full potentialities of an industrial-strength relational engine, extended with novel functionalities.

For the intended translation, the source algebra is therefore $\mathcal{CTA}$ and the target algebra is the standard temporal algebra $\mathcal{TA}$ that includes the operators shown in Tab. 1 but made to work on relations employing interval-timestamping, according to an extended sequenced semantics [12], in order to enforce *snapshot equivalence*. Although working on an interval-based concrete temporal model, the target algebra represents indeed a point-based query language (in the sense of [28]) and, thus, its implementation on a traditional DBMS does not require enforcement of *change preservation* (e.g., via adjustment, alignment and scaling techniques as proposed in [12]). For ease of presentation, hereinafter, with a little abuse of notation, when we need to distinguish the same concept at the two different levels, we will use the superscript $\mathcal{CTA}$ to denote tables and algebraic operators in the continuous temporal model and the superscript $\mathcal{TA}$ to denote the corresponding concepts in the target model.

The main issue for our goal is to mimic in a static context the behavior of $\mathcal{CTA}$ windowing operators, which are evaluated on user-specified time intervals and operate on the contents of the involved streaming tables at the evaluation instants. To this end, we first translate each streaming table $S^{\mathcal{CTA}}$ with schema $S(X|T)$ at the continuous level into an interval-based streaming table $S^{\mathcal{TA}}$ with schema $S(X, T|T')$, where the event occurrence time T associated to tuples is made explicit and T' is an implicit interval attribute that records tuple validity, that is $[st, \infty]$, where st is the time when the tuple s enters the system (without transaction-time support, it is worth noting that st can be approximated with $s(T)$). Each

Table 3 Semantics of the windowing operators at the target level, $\mathcal{TA}$.

Substreaming operator
$\text{Sub}_{[t_1,t_2]}(S)^{\mathcal{TA}} := \sigma^T_{t_1 \leq S.T \leq t_2}(S)$
Sliding window operators
${}^{tset}w^{\text{time}}_{[\omega_1,\omega_2]}(S)^{\mathcal{TA}} := \bigcup_{t \in tset} \tau_t(\text{Sub}_{[t-\omega_1,t+\omega_2]}(S)^{\mathcal{TA}})$ ${}^{tset}w^{\text{count}}_{[n_1,n_2]}(S)^{\mathcal{TA}} := \bigcup_{t \in tset} \tau_t(\{s \,\vert\, s \in S' = \text{Sub}_{[t_1,t_2]}(S),$ $\vert\text{Sub}_{[t_1,t]}(S')^{\mathcal{TA}}\vert = n_1, \vert\text{Sub}_{[t,t_2]}(S')^{\mathcal{TA}}\vert = n_2\})$
Window flattening operators
$\varphi(S)^{\mathcal{TA}} := S$ $\Phi({}^{tset}W^{\text{time},B}_{[\omega_1,\omega_2]}(S))^{\mathcal{TA}} := {}^{tset}w^{\text{time}}_{[\omega_1,\omega_2]}(S)^{\mathcal{TA}}$ $\Phi({}^{tset}W^{\text{count},B}_{[n_1,n_2]}(S))^{\mathcal{TA}} := {}^{tset}w^{\text{count}}_{[n_1,n_2]}(S)^{\mathcal{TA}}$
Window aggregation operators
$\vartheta_F(S)^{\mathcal{TA}} := ({}_{\emptyset}\vartheta^T_F(S))^{\mathcal{TA}}$ ${}_B\Theta_F({}^{tset}W^{\text{time},B}_{[\omega_1,\omega_2]}(S))^{\mathcal{TA}} := ({}_B\vartheta^T_F({}^{tset}w^{\text{time}}_{[\omega_1,\omega_2]}(S)))^{\mathcal{TA}}$ ${}_B\Theta_F({}^{tset}W^{\text{count},B}_{[n_1,n_2]}(S))^{\mathcal{TA}} := ({}_B\vartheta^T_F({}^{tset}w^{\text{count}}_{[n_1,n_2]}(S)))^{\mathcal{TA}}$

temporal relation $R(X|T)$ in $\mathcal{CTA}$ is translated into a relation $R(X|T')$ in $\mathcal{TA}$, by *coalescing* the timestamps of *value-equivalent* tuples in $R^{\mathcal{CTA}}$ into maximal intervals to be used as timestamps in $R^{\mathcal{TA}}$. Non temporal relations are converted into temporal relations whose tuples are timestamped with a $[0, \infty]$ validity interval in $\mathcal{TA}$.

Then, in Tab. 3 we define the semantics of the continuous operators introduced in Subsection 4.2 defined through $\mathcal{TA}$ operators[2]. Notice that, unlike their counterpart at the $\mathcal{CTA}$ level, sliding window operators at the $\mathcal{TA}$ level require a set of time instants *tset* to be evaluated and the flattening operator $\Phi^{\mathcal{TA}}$ simply undoes the effects of partitioning. It is worth stressing that, thanks to the translation rules of Tab. 3, any legal $\mathcal{CTA}$ expression can be evaluated via $\mathcal{TA}$ operators working on streaming tables only. In particular, there is no need for implementing (sets of) streaming tables of streaming tables as formally introduced in the definitions of $\mathcal{CTA}$ operators in Sec. 4.

Finally, we define the sampling operator at the $\mathcal{TA}$ level and show that the results of the two sampling operators, the one defined at the $\mathcal{CTA}$ level and the other one defined at the $\mathcal{TA}$ level, are equivalent (i.e., they provide the same results for the same continuous query).

▶ **Definition 15** (Sampling Operator$^{\mathcal{TA}}$). At execution time t, the evaluation delayed by δ of a continuous query $E = E_{\mathcal{TA}}(\alpha_1(\omega_1(S_1)), \ldots, \alpha_n(\omega_n(S_n)), R_1, \ldots, R_m) \in \mathcal{CTA}$, with an historical parameter *hp*, slide parameter *sl* and alignment parameter *a*, at the $\mathcal{TA}$ level is defined by the sampling operator $\xi^{\mathcal{TA}} : \mathcal{TA} \times \mathcal{T} \times \mathcal{I}^4 \to \mathcal{S}^{\mathcal{TA}}$ as follows:

$$\xi^{t,\delta}_{hp,sl,a}(E)^{\mathcal{TA}} := E_{\mathcal{TA}}((\alpha_1({}^{tset}\omega_1(S_1^{t+\delta})))^{\mathcal{TA}}, \ldots, (\alpha_n({}^{tset}\omega_n(S_n^{t+\delta})))^{\mathcal{TA}}, R_1^{t+\delta}, \ldots, R_m^{t+\delta})$$

where *tset* is the evaluation time instant set: $tset = \{t' \mid t' \leq t \wedge t' = (\lceil \frac{t-hp-a}{sl} \rceil + i) \cdot sl + a \text{ for some } i \in \mathbb{N}\}$.

▶ **Theorem 16.** *Given the continuous query $E = E_{\mathcal{TA}}(\alpha_1(\omega_1(S_1)), \ldots, \alpha_n(\omega_n(S_n)), R_1, \ldots, R_m) \in \mathcal{CTA}$, with slide parameter sl and alignment parameter a, then, for each execution*

[2] To be rigorous, the definition of $w^{\text{count}}_{[n_1,n_2]}$, involving constraints on the cardinality of the results of Sub, is not expressible with $\mathcal{TA}$ operators only. However, we can augment $\mathcal{TA}$ with an operator directly evaluating $w^{\text{count}}_{[n_1,n_2]}$, which can be easily and efficiently implemented by exploiting the ordering of timestamps in a streaming table.

time t with delay δ:

$$\xi^{t,\delta}_{sl,a}(E)^{\mathcal{CTA}} = \xi^{t,\delta}_{sl,a}(E)^{\mathcal{TA}}$$

Proof. For the sake of simplicity, we assume $\delta = 0$ and replace each R^t and S^t in the operator semantics with R and S, respectively (the proof can be straightforwardly adapted to the case when $\delta > 0$). First, notice that:

$$\begin{aligned}\xi^{t,\delta}_{sl,a}(E)^{\mathcal{CTA}} &= (\bigcup_{i=0}^{k:t_k\le t} \tau_{t_i}(E_{\mathcal{TA}}(\alpha_1(\omega_1(S_1)),\dots,\alpha_n(\omega_n(S_n)),R_1,\dots,R_m)))^{\mathcal{CTA}} \\ &= (E_{\mathcal{TA}}(\bigcup_{i=0}^{k:t_k\le t} \tau_{t_i}(\alpha_1(\omega_1(S_1))),\dots,\bigcup_{i=0}^{k:t_k\le t} \tau_{t_i}(\alpha_n(\omega_n(S_n))),R_1,\dots,R_m))^{\mathcal{CTA}}\end{aligned}$$

Therefore, if we show that $\bigcup_{i=0}^{k:t_k\le t} \tau_{t_i}(\alpha_j(\omega_j(S_j))^{\mathcal{CTA}})) = (\alpha_j({}^{tset}\omega_j(S_j)))^{\mathcal{TA}}$, then

$$\begin{aligned}&E_{\mathcal{TA}}(\bigcup_{i=0}^{k:t_k\le t} \tau_{t_i}(\alpha_1(\omega_1(S_1))),\dots,\bigcup_{i=0}^{k:t_k\le t} \tau_{t_i}(\alpha_n(\omega_n(S_n))),R_1,\dots,R_m))^{\mathcal{CTA}} \\ &\quad = E_{\mathcal{TA}}((\alpha_1({}^{tset}\omega_1(S_1)))^{\mathcal{TA}},\dots,(\alpha_n({}^{tset}\omega_n(S_n)))^{\mathcal{TA}},R_1,\dots,R_m)\end{aligned}$$

and $\xi^{t,\delta}_{sl,a}(E)^{\mathcal{CTA}} = \xi^{t,\delta}_{sl,a}(E)^{\mathcal{TA}}$.

To this end, as far as $\alpha_j(\omega_j(S_j))$ is concerned, all possible operator combinations should be considered. For the sake of brevity, in the following we will consider only the case when $\alpha_j = \vartheta_F$ and $\omega_j = w^{\text{time}}_{[\omega_1,\omega_2]}$, but all the other combinations can be managed in a similar way. Given $\vartheta_F(w^{\text{time}}_{[\omega_1,\omega_2]}(S_j))$, in the following we will show that $s \in \bigcup_{i=0}^{k:t_k\le t} \tau_{t_i}(\vartheta_F(w^{\text{time}}_{[\omega_1,\omega_2]}(S_j))^{\mathcal{CTA}})$ iff $s \in \xi^{t,\delta}_{sl,a}(E)^{\mathcal{TA}}$. Let $s \in \bigcup_{i=0}^{k:t_k\le t} \tau_{t_i}(\vartheta_F(w^{\text{time}}_{[\omega_1,\omega_2]}(S_j))^{\mathcal{CTA}})$, then $s = (X, t_i) \in \tau_{t_i}(\vartheta_F(w^{\text{time}}_{[\omega_1,\omega_2]}(S_j))^{\mathcal{CTA}})$ for some i. Being $s \in \tau_{t_i}(\vartheta_F(w)^{\mathcal{CTA}})$, where $w = w^{\text{time}}_{[\omega_1,\omega_2]}(S_j))$, then $s \in \tau_{t_i}(\vartheta_F(w)^{\mathcal{CTA}})$ iff $t = (X, t_i)$, where $X = (f_1(S), \dots, f_h(S))$ and $(S, \tau) \in w$. This means that $S = \text{Sub}_{[t_i-\omega_1,t_i+\omega_2]}(S_j)^{\mathcal{CTA}} = \{(s,\tau) \mid (s,\tau) \in S_j, (t_i - \omega_1) \le \tau \le (t_i + \omega_2)\}$. Hence, $(s,\tau) \in \text{Sub}_{[t_i-\omega_1,t_i+\omega_2]}(S_j)^{\mathcal{CTA}}$ iff $(s,\tau,t_i) \in \tau_{t_i}(\text{Sub}_{[t_i-\omega_1,t_i+\omega_2]}(S_j)^{\mathcal{TA}})$. As $t_i \in tset$, it follows that:

$$(s,\tau,t_i) \in \bigcup_{t\in tset} \tau_t(\text{Sub}_{[t-\omega_1,t+\omega_2]}(S_j)^{\mathcal{TA}}) =^{tset} w^{\text{time}}_{[\omega_1,\omega_2]}(S_j)^{\mathcal{TA}}$$

From $S = \tau_{t_i}(\text{Sub}_{[t_i-\omega_1,t_i+\omega_2]}(S_j)^{\mathcal{TA}})$ and $X = (f_1(S), \dots, f_h(S))$, it follows that $(X, t_i) \in {}_{\emptyset}\vartheta^{T}_{F}(S_j)^{\mathcal{TA}}$, which is equivalent to say that $(X, t_i) \in (\vartheta_F(w^{\text{time}}_{[\omega_1,\omega_2]}(S_j))^{\mathcal{TA}}$. ◀

Therefore, thanks to the above theorem, we can safely translate each continuous query in $\mathcal{CTA}$ into an equivalent expression in $\mathcal{TA}$ and execute it on a static relational engine. In fact, the above theorem ensures that the semantics of execution is preserved.

7 Related works and concluding remarks

DSMSs [1, 3] natively support CQs over continuous unbounded streams of data according to windows where only the most recent data is retained. In CQL [3] and SyncSQL [14] streams are transformed into instantaneous/syncronized relations that are manipulated through relational operators, and then transformed back to streams. In this paper we proved that it is possible to exploit the full potential of a native representation of temporal data to query streaming data seamlessly, thus overcoming the transformation overhead of stream-relation-stream approaches like [3, 14]. In line with this approach, recent research proposals extend traditional DBMSs' query model and language towards streaming query capabilities [10, 20]. However, these works present extensions to SQL through query examples and do not offer a formal algebraic framework for a clear specification of query operators.

With regard to algebras for querying streaming data, in [19] snapshot-reducible algebraic operators are implemented on ad-hoc data structures for state maintenance under a time-interval approach. This approach does not integrate with the theoretical and practical solutions proposed for the development of a robust temporal database technology, including [12] which presents a proposal for the implementation of a standard temporal algebra in an off-the-shelf DBMS, supporting a sequenced semantics that guarantees extended snapshot reducibility. As to $\mathcal{CTA}$ operators, we proved this fundamental requirement by showing how $\mathcal{CTA}$ expressions can be translated into equivalent expressions in the temporal algebra $\mathcal{TA}$.

An additional note concerns the semantics of ad-hoc proposals of temporal operators that often proves to be ambiguous as to timestamp management. A consensus is not shared among existing approaches. For instance, the timestamps of tuples resulting from a windowed join can be either the minimum of the two original timestamp values [1, 24], or the most recent one [4], or the time instant at which the join is executed [5]. Operators in $\mathcal{TA}$ undergo a precise and commonly adopted temporal semantics [12]. Further, in [3, 19] windowing operators overwrite the original timestamp of tuples with a new timestamp corresponding to the window evaluation time instant. $\mathcal{CTA}$ operators maintain instead both this information and the original tuple timestamp, thus not losing relevant information.

A further property featured by $\mathcal{CTA}$ is the capability of defining “forward” windows, thus opening to the possibility of evaluating queries (possibly in an approximated way) by referring tuples that will be observed after a given time instance, as proposed for SQL:2011 [29] (e.g., SQLStream Blaze[3] considers them in its query syntax specifications but does not provide an implementation). Moreover, both in SQL:2011 and proposed stream query languages, sliding windows can only be used, via aggregation operators, to produce results in the target list of a query, whereas $\mathcal{CTA}$ allows us to use them everywhere (e.g., in a selection predicate as in our running example). To the authors' knowledge, these features are not covered by any existing approach dealing with streaming data.

From a system perspective, existing stream processing frameworks (e.g., Apache Flink [7] and Samza [22]) provide SQL extensions to deal with streaming data but they do not expose a clear and unambiguous semantics of query operators through an algebra definition. In general, DSMSs and stream processing frameworks [7, 22] do not support queries involving both streaming and relational data changing over time, since their data and query models do not include temporal semantics and versioning. On the other hand, much work has been devoted to extending DBMSs towards a flexible and efficient management of temporal data [25]. However, quite surprisingly considering the temporal nature of streaming data, no built-in streaming functionalities are provided in these systems.

Following the first step provided by the streaming table concept introduced in [6], in this paper we presented a temporal algebra extended with windowing and aggregation operators supporting both OTQs and CQs on streaming, standard and temporal relational data. In our future work, we plan to explore algebraic optimization issues and indexing techniques to efficiently support the implementation of $\mathcal{CTA}$ operators in a temporal DBMS.

[3] `http://sqlstream.com`

References

1 Daniel J. Abadi, Donald Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stanley B. Zdonik. Aurora: a new model and architecture for data stream management. *VLDB J.*, 12(2):120–139, 2003.

2 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

3 Arvind Arasu, Shivnath Babu, and Jennifer Widom. The CQL continuous query language: semantic foundations and query execution. *VLDB J.*, 15(2):121–142, 2006.

4 Ahmed M. Ayad and Jeffrey F. Naughton. Static Optimization of Conjunctive Queries with Sliding Windows over Infinite Streams. In *Proc of ACM SIGMOD*, pages 419–430, 2004.

5 Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and Issues in Data Stream Systems. In *Proc. of ACM PODS*, pages 1–16, 2002.

6 Luca Carafoli, Federica Mandreoli, Riccardo Martoglia, and Wilma Penzo. Streaming Tables: Native Support to Streaming Data in DBMSs. *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, pages 1–15, 2017.

7 Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache Flink™: Stream and Batch Processing in a Single Engine. *IEEE Data Eng. Bull.*, 38(4):28–38, 2015.

8 Cristina De Castro, Fabio Grandi, and Maria Rita Scalas. Semantic interoperability of multitemporal relational databases. In *Proceedings of the 12th International Conference on the Entity-Relationship Approach*, pages 463–474. Springer-Verlag, 1993.

9 Ugur Cetintemel, Jiang Du, Tim Kraska, Samuel Madden, David Maier, John Meehan, Andrew Pavlo, Michael Stonebraker, Erik Sutherland, Nesime Tatbul, Kristin Tufte, Hao Wang, and Stanley Zdonik. S-Store: A Streaming NewSQL System for Big Velocity Applications. *Proc. VLDB*, 7(13):1633–1636, August 2014.

10 Qiming Chen and Meichun Hsu. Cut-and-Rewind: Extending Query Engine for Continuous Stream Analytics. In A. Hameurlain, J. Kueng, R. Wagner, A. Cuzzocrea, and U. Dayal, editors, *TLDKS XXI*, volume 9260 of *LNCS*, pages 94–114. Springer, 2015.

11 Jan Chomicki. Temporal query languages: A survey. In *Proceedings of the First International Conference on Temporal Logic*, pages 506–534. Springer-Verlag, 1994.

12 Anton Dignös, Michael H. Böhlen, Johann Gamper, and Christian S. Jensen. Extending the Kernel of a Relational DBMS with Comprehensive Support for Sequenced Temporal Queries. *ACM Trans. Database Syst.*, 41(4):26:1–26:46, 2016.

13 Steve Donoho. Early detection of insider trading in option markets. In *Proc. of the ACM KDD*, pages 420–429. ACM, 2004.

14 Thanaa M. Ghanem, Ahmed K. Elmagarmid, Per-ake Larson, and Walid G. Aref. Supporting Views in Data Stream Management Systems. *ACM Trans. Database Syst.*, 35(1):1, 2010.

15 Lukasz Golab and M. Tamer Özsu. Update-Pattern-Aware Modeling and Processing of Continuous Queries. In *Proc. of ACM SIGMOD*, pages 658–669, 2005.

16 Fabio Grandi. Temporal interoperability in Multi+Temporal databases. *J. Database Manag.*, 9(1):14–23, 1998.

17 Insider Trading. `https://www.sec.gov/fast-answers/answersinsiderhtm.html`.

18 Christian S. Jensen, Curtis E. Dyreson, Michael H. Böhlen, James Clifford, Ramez Elmasri, Shashi K. Gadia, Fabio Grandi, Patrick J. Hayes, Sushil Jajodia, Wolfgang Käfer, Nick Kline, Nikos A. Lorentzos, Yannis G. Mitsopoulos, Angelo Montanari, Daniel A. Nonen, Elisa Peressi, Barbara Pernici, John F. Roddick, Nandlal L. Sarda, Maria Rita Scalas, Arie Segev, Richard T. Snodgrass, Michael D. Soo, Abdullah Uz Tansel, Paolo Tiberio, and Gio Wiederhold. The consensus glossary of temporal database concepts – february 1998

version. In O. Etzion, S Jajodia, and S. Sripada, editors, *Temporal Databases: Research and Practice*, pages 367–405. Springer-Verlag, 1998.

19 Jürgen Krämer and Bernhard Seeger. Semantics and Implementation of Continuous Sliding Window Queries over Data Streams. *ACM Trans. Database Syst.*, 34(1):4, 2009.

20 Nikolay Laptev, Barzan Mozafari, Hani Mousavi, Hetal Thakkar, Haixun Wang, Kai Zeng, and Carlo Zaniolo. Extending relational query languages for data streams. In Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi, editors, *Data Stream Management: Processing High-Speed Data Streams*, pages 361–386. Springer Berlin Heidelberg, 2016.

21 Erietta Liarou, Stratos Idreos, Stefan Manegold, and Martin Kersten. Enhanced stream processing in a dbms kernel. In *Proc. of EDBT*, pages 501–512, 2013.

22 Yi Pan Milinda Pathirage, Julian Hyde and Beth Plale. SamzaSQL: Scalable fast data management with streaming SQL. In *Proc. of IEEE IPDP Symposium Workshops*, pages 1627–1636. IEEE Computer Society, 2016.

23 Emanuele Panigati, Fabio A. Schreiber, and Carlo Zaniolo. Data streams and data stream management systems and languages. In Francesco Colace, Massimo De Santo, Vincenzo Moscato, Antonio Picariello, Fabio A. Schreiber, and Letizia Tanca, editors, *Data Management in Pervasive Systems*, pages 93–111. Springer, 2015.

24 Kostas Patroumpas and Timos Sellis. Window specification over data streams. In *Current Trends in Database Technology – EDBT 2006 Workshops*, pages 445–464, 2006.

25 Dušan Petkovic. Temporal data in relational database systems: A comparison. In Álvaro Rocha, Ana Maria Correia, Hojjat Adeli, Luis Paulo Reis, and Marcelo Mendonça Teixeira, editors, *New Advances in Information Systems and Technologies*, pages 13–23. Springer, 2016.

26 Richard T. Snodgrass, editor. *The TSQL2 Temporal Query Language*. Kluwer, 1995.

27 Utkarsh Srivastava and Jennifer Widom. Flexible Time Management in Data Stream Systems. In *Proc. of ACM PODS*, pages 263–274, 2004.

28 David Toman. Point vs. interval-based query languages for temporal databases. In *Proc. of ACM PODS*, pages 58–67. ACM Press, 1996.

29 Fred Zemke. What's new in SQL:2011. *SIGMOD Rec.*, 41(1):67–73, April 2012.

The Time Ontology of Allen's Interval Algebra

Michael Grüninger[1] **and Zhuojun Li**[2]

1 **Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, ON, Canada**
`gruninger@mie.utoronto.ca`

2 **Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, ON, Canada**
`zhuojun.li@mail.utoronto.ca`

Abstract

Allen's interval algebra is a set of thirteen jointly exhaustive and pairwise disjoint binary relations representing temporal relationships between pairs of time intervals. Despite widespread use, there is still the question of which time ontology actually underlies Allen's algebra. Early work specified a first-order ontology that can interpret Allen's interval algebra; in this paper, we identify the first-order ontology that is logically synonymous with Allen's interval algebra, so that there is a one-to-one correspondence between models of the ontology and solutions to temporal constraints that are specified using the temporal relations. We further prove a representation theorem for the ontology, thus characterizing its models up to isomorphism.

1998 ACM Subject Classification I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases time ontology, intervals, composition table, first-order logic, synonymy

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.16

1 Introduction

Temporal reasoning has long been studied in artificial intelligence, particularly since the seminal work of Allen, in which time is represented using binary relations over intervals. The composition of these relations leads to an algebra, which have been widely used for constraint satisfaction problems. Today, virtually every presentation of a time ontology includes at least the diagram of the temporal relations in Allen's algebra and the composition table for the relations. This work was later extended by Hayes and Allen, who proposed a first-order ontology corresponding to Allen's algebra; additional extensions were proposed by Ladkin and Maddox.

Given this long story, it might be surprising that there is anything left to say; yet a closer inspection of the ontologies involved leads to some interesting observations. First, nobody has shown which ontology of time intervals is equivalent to Allen's Interval Algebra; previous work has only shown that a first-order axiomatization of the algebra is interpreted by a particular axiomatization of an ontology of time intervals.

Second, there has been no characterization up to isomorphism of the models of the first-order axiomatization of Allen's Interval Algebra. The closest work along these lines has been a discussion of the models of time interval ontologies, but this is far short of a full characterization. The models are often informally specified; the more formal specifications of the models refer to intervals of integers or rational numbers, rather than an explicit formal specification in the signature of the ontologies. Furthermore, there have been no proofs of representation theorems for these classes of models that do not refer to an underlying set of points. Finally, the relationship between different ontologies of time intervals has not been fully explicated. The relationship between these other interval ontologies and Allen's Interval Algebra is thus also not clear.

In this paper we investigate Allen's Interval Algebra based on the metalogical relationship between the first-order theory of the composition table and the different axiomatizations of time ontologies of intervals. After reviewing the basic axiomatizations of the time ontologies in Section 2, we discuss the relationship between the theory of the interval algebra T_{allen} and Hayes' axiomatization ($T_{interval\_meeting}$) in Section 3. After showing that $T_{interval\_meeting}$ cannot be interpreted by T_{allen}, we propose a new ontology $T_{bounded\_meeting}$, which is weaker than $T_{interval\_meeting}$. Our key result is that a nonconservative extension of the interval algebra, which we call T^*_{allen}, is logically synonymous with the $T_{bounded\_meeting}$, meaning $T_{bounded\_meeting}$ and T^*_{allen} axiomatize the same class of structures. In other words,the two theories are semantically equivalent, and only differ in signature (i.e., the non-logical symbols). Finally, in Section 4, we present a characterization of models of $T_{bounded\_meeting}$ up to isomorphism, and explain how such a characterization can be used in characterizing algebraic properties of models of T_{allen}.

2 Preliminaries

2.1 Allen's Interval Algebra

Allen's introduction of thirteen relations over temporal intervals [2] laid the foundations for qualitative temporal reasoning and representation. The interval relations are *meets*, *before*, *starts*, *ends*, *overlaps*, *during*, their inverses *met_by*,*after*, *started_by*, *ended_by*, *overlapped_by*, *contains*, and equality. These relations are pairwise disjoint and exhaustive (that is, any two time intervals must be related by one of these relations). The notion of an algebra over these relations arises from considering the intersection, union, and composition of a pair of temporal relations. This leads to the composition table CT, which is a 13×13 matrix such that for each ordered pair of interval relations R_i,R_j, the cell $CT(R_i, R_j)$ indicates the possible temporal relations between two intervals a and c assuming that $R_i(a, b)$ and $R_j(b, c)$ holds. For example, $CT(\mathit{starts}, \mathit{overlaps}) = \{\mathit{before}, \mathit{overlaps}, \mathit{meets}\}$, meaning that if $\mathit{starts}(a, b)$ and $\mathit{overlaps}(b, c)$, then the interval a is before, overlaps, or meets the interval c.

2.2 Ontologies for Time Intervals

Although the application of Allen's interval algebra was widespread in the specification and solution of temporal constraint satisfaction problems, it was several years before people considered its relationship to the time ontologies being developed within the knowledge representation community. In this section, we review the primary time ontologies that are relevant to the axiomatization of the interval algebra in first-order logic.

Some of the earliest time ontologies [4] treated timepoints as the primitive entities in the domain. However, the entities for the interval algebra are time *intervals* – points do not exist. The first proposal for the axiomatization of an ontology[1] of time intervals as related to Allen's interval algebra was the work of Hayes [8], [1], in which there is one primitive binary relation *meets* over intervals. This axiomatization, which we will refer to as $T_{interval\_meeting}$ is shown in Figure 1.

[1] A theory is set of first-order sentences closed under logical entailment. In this paper, we use the terms ontology and theory interchangably.

$$(\forall i,j,k,m)\ meets(i,k) \wedge meets(j,k) \wedge meets(i,m) \supset meets(j,m) \tag{1}$$

$$(\forall i)(\exists j,k)\ meets(j,i) \wedge meets(i,k) \tag{2}$$

$$(\forall i,j,k,l)\ (meets(i,j) \wedge meets(k,l)) \supset$$
$$(meets(i,l) \vee ((\exists n)\ ((meets(i,n) \wedge meets(n,l)) \vee (meets(k,n) \wedge meets(n,j))))) \tag{3}$$

$$(\forall i,j)\ meets(i,j) \supset \neg meets(j,i) \tag{4}$$

$$(\forall i,j,k,m)\ meets(i,j) \wedge meets(j,k) \wedge meets(k,m) \supset (\exists n)\ meets(i,n) \wedge meets(n,m) \tag{5}$$

Figure 1 The axioms of $T_{interval\_meeting}$.

$$(\forall i,j)\ before(i,j) \equiv (\exists k)\ meets(i,k) \wedge meets(k,j) \tag{6}$$

$$(\forall i,j)\ starts(i,j) \equiv (\exists k,m,n)\ meets(k,i) \wedge meets(i,m)$$
$$\wedge meets(m,n) \wedge meets(k,j) \wedge meets(j,n) \tag{7}$$

$$(\forall i,j)\ ends(i,j) \equiv (\exists k,m,n)\ meets(k,m) \wedge meets(m,i)$$
$$\wedge meets(i,n) \wedge meets(k,j) \wedge meets(j,n) \tag{8}$$

$$(\forall i,j)\ overlaps(i,j) \equiv (\exists k,m,n,o,p)\ meets(k,m) \wedge meets(m,n)$$
$$\wedge meets(n,o) \wedge meets(o,p) \wedge meets(m,j) \wedge meets(j,p) \wedge meets(k,i) \wedge meets(i,o) \tag{9}$$

$$(\forall i,j)\ during(i,j) \equiv (\exists k,m,n.o)\ meets(k,m) \wedge meets(m,i)$$
$$\wedge meets(i,n) \wedge meets(n,o) \wedge meets(k,j) \wedge meets(j,o) \tag{10}$$

Figure 2 $T_{interval\_rel}$: the definitions for Allen's Interval Algebra relations.

By Axiom (1) if two intervals meet a common interval, then the sets of intervals that each meet is equivalent to each other. For each time interval, Axiom (2) guarantees the existence of an interval that it *meets*, and an interval that is met by it. Since this leads to infinite models, we will refer to this as the Infinity Axiom. Axiom (3) captures the intuition that the *meets* relation leads to an ordering over time intervals. By Axiom (4), the *meets* relation is asymmetric. Axiom (5) is often referred to as the Sum Axiom, since it entails the existence of an interval that is formed by the union of two intervals that meet.

The axiomatization of $T_{interval\_meeting}$ is sufficient to enable the definition of the relations in Allen's interval algebra (see Figure 2). $T_{interval\_meeting} \cup T_{interval\_rel}$ is therefore a definitional extension of $T_{interval\_meeting}$. This extension will play a key role in determining the relationship between $T_{interval\_meeting}$ and the interval algebra.

Hayes describes the models of $T_{interval\_meeting}$ and its extensions in terms of the set of intervals on $\mathbb{Q}$ (rational numbers) and $\mathbb{Z}$ (integers). For example, he describes one model which interprets intervals as open connected subsets of $\mathbb{Q}$, such that (a,b) meets (c,d) when $a = c$ and the intersection of two meeting intervals is empty. Alternative models exist, which interpret intervals as closed connected subsets. Analogous models are also described as sets of closed intervals on $\mathbb{Z}$.

This treatment is inadequate for several reasons. Strictly speaking, a structure on the sets of intervals on $\mathbb{Q}$ is not a model of $T_{interval\_meeting}$ because it does not have the same signature; at best, this is saying that models of $T_{interval\_meeting}$ can be interpreted in such a structure. Yet even this falls short because it does not determine whether *all* models can be interpreted in this way, or whether there exist other models which must be constructed in a different fashion (i.e. do there exist models of the ontology that are not isomorphic to intervals over $\mathbb{Q}$ or $\mathbb{Z}$?). Furthermore, the description is informal, without a formal proof of equivalence; there is no explicit definition or characterization of the models of $T_{interval\_meeting}$ in and of themselves.

Several extensions to $T_{interval\_meeting}$ have been proposed. The work of Ladkin ([10], in particular explored an extension which is categorical. Ladkin provides a more formal characterization for the models of his axiomatization which also specifies intervals as pairs of points in an underlying linear ordering. As with Hayes, this is essentially a representation theorem for models of the ontology, rather than a direct characterization of the models themselves.

3 Relationship between Allen's Interval Algebra and Time Ontologies

Although [8] stated that Allen's Interval Algebra can be derived from the ontology $T_{interval\_meeting}$, the two approaches display different properties. In particular, all models of $T_{interval\_meeting}$ are infinite, whereas Allen's Interval Algebra allows finite models. These differences raise the question of which ontology actually underlies the interval algebra.

We begin this section by describing the logical theory that captures the composition table for Allen's Interval Algebra, and then search for the time ontology that is equivalent to it.

3.1 First-Order Theory of Allen's Interval Algebra

To specify the first-order theory T_{allen} of Allen's Interval Algebra, we follow [3] and we assume that for each cell in the composition table, we have a first-order sentence of the form

$$R_i(x, y) \wedge R_j(y, z) \supset T_1(x, z) \vee \ldots \vee T_n(x, z)$$

where $CT(R_i, R_j) = \{T_1, \ldots, T_n\}$. For example, the following sentence is the axiom in which corresponds with $CT(meets, ends)$:

$$meets(x, y) \wedge ends(y, z) \supset (overlaps(x, z) \vee during(x, z) \vee starts(x, z)).$$

Since the composition table consists of 13×13 cells, T_{allen} must contain 169 axioms corresponding with the table. We will denote this set of axioms as $T_{allen\_compose}$.

In addition to these axioms, we assume that for each interval algebra relation R_1, T_{allen} contains a sentence of the following form stating that the relations are pairwise disjoint (PD):

$$R_1(x, y) \supset \neg(R_2(x, y) \vee \ldots \vee R_{13}(x, y))$$

where $R_2, \ldots, R_{13}$ are interval algebra relations other than R_1. The following sentence, for example, is the PD axiom corresponding with *meets*:

$$\begin{aligned} meets(x, y) \supset \neg[&before(x, y) \vee starts(x, y) \vee ends(x, y) \vee overlaps(x, y) \vee during(x, y) \\ &\vee met\_by(x, y) \vee after(x, y) \vee started\_by(x, y) \vee ended\_by(x, y) \\ &\vee overlapped\_by(x, y) \vee contains(x, y) \vee (x = y)]. \end{aligned}$$

As there are 13 interval algebra relations, T_{allen} contains 13 PD axioms; we will denote the disjointness axioms by $T_{allen\_disjoint}$.

Finally, T_{allen} contains an axiom that specifies that the interval algebra relations are jointly exhaustive:

$$meets(x,y) \lor before(x,y) \lor starts(x,y) \lor ends(x,y) \lor ends(x,y) \lor overlaps(x,y)$$
$$\lor\, during(x,y) \lor \lor met\_by(x,y) \lor after(x,y) \land started\_by(x,y)$$
$$\lor\, ended\_by(x,y) \land overlapped\_by(x,y) \lor contains(x,y) \lor (x=y).$$

We will refer to this axiom as $T_{exhaustive}$.

All other sentences in T_{allen} are those which are entailed by the $169+13+1$ above-mentioned axioms. Thus,

$$T_{allen} = T_{allen\_compose} \cup T_{allen\_disjoint} \cup T_{exhaustive}\,.$$

Models of T_{allen} are equivalent to solutions of temporal constraints that are expressed using the interval relations, but the question of a characterization of models of T_{allen} remains unresolved. In the following subsections, we identify the time ontology that is equivalent to T_{allen}, and in the latter part of the paper, we characterize the models of this time ontology up to isomorphism.

3.2 T_{allen} and $T_{interval\_meeting}$

Hayes and Allen ([1]) state that the interval algebra composition table can be derived from the ontology of time intervals. More precisely, the first-order theory of the interval algebra composition table can be entailed from a definitional extension of $T_{interval\_meeting}$:

▶ **Definition 1** (adopted from [9]). Let T be a first-order theory and Π be a set containing sentences of the following form [2]

$$R(x_1,\ldots,x_n) \equiv \Phi(x_1,\ldots,x_n)$$

where R is a predicate which is not in $\Sigma(T)$ and Φ is a formula in $\mathcal{L}(T)$ in which at most variables $x_1,\ldots,x_n$ occur free. $T \cup \Pi$ is called a *definitional extension* of T.

▶ **Theorem 2.** *T_{allen} is entailed by $T_{interval\_meeting} \cup T_{interval\_rel}$.*

Proof. Using the automated theorem prover Prover9 [11], we have shown[3] that $T_{interval\_meeting} \cup T_{interval\_rel}$ entails for each axiom Φ of T_{allen},

$$T_{interval\_meeting} \cup T_{interval\_rel} \models T_{allen}\,. \qquad ◀$$

This Theorem is equivalent to saying that T_{allen} has an interpretation in $T_{interval\_meeting}$. A theory T_1 has a relative interpretation [6] in another theory T_2 if every sentence in T_1 can be translated into a sentence in T_2. In other words, for all sentences $\Phi \in \mathcal{L}(T_1)$, if T_1 entails Φ, then T_2 entails a translation of Φ into the language of T_2. The work of [7] shows that if a definitional extension of T_2 entails T_1, translations for sentences of T_1 is obtained based on the formulas which define predicates of T_1 in the definitional extension. For instance,

[2] For a theory T, $\Sigma(T)$ denotes the *signature* of T, i.e., the set of non-logical symbols used in sentences of T; $\mathcal{L}(T)$ denotes the *language* of T, i.e., the set of all first-order formulae generated by symbols in $\Sigma(T)$; $Mod(T)$ denotes the class of all models of T.

[3] The input files and proofs can be found at `http://colore.oor.net/allen_interval_algebra/mappings/theorems/intervalmeeting2allen/`.

a translation of Axiom (5) of $T_{interval\_meeting}$ into the language of T_{allen} can be obtained by replacing formulas with *before* literals (whose definition can be found in Figure 2). The result is the following sentence, which provably is a sentence in T_{allen}:

$$(\forall i, j, m) before(i, j) \wedge meets(j, m) \supset before(i, m)$$

Theories T_1 and T_2 are mutually interpretable iff they are relatively interpretable in each other. In our case, $T_{interval\_meeting}$ is not relatively interpretable in T_{allen}, since T_{allen} cannot entail all axioms of $T_{interval\_meeting}$:

▶ **Proposition 3.** $T_{allen} \not\models (\forall i)(\exists j)\ meets(i, j)$.

Proof. The model generated by Mace4 can be found at
`http://colore.oor.net/allen_interval_algebra/mappings/theorems/allen2boundedmeeting/finite.model/`. ◀

By Proposition 3, T_{allen} does not interpret $T_{interval\_meeting}$ because it allows finite models, whereas all models of $T_{interval\_meeting}$ are infinite. To achieve mutual interpretability, we need to weaken $T_{interval\_meeting}$, However, simply removing the infinity axiom from $T_{interval\_meeting}$ doesn't work:

▶ **Proposition 4.** *Let T_{finite} be the set of all axioms in $T_{interval\_meeting}$ except Axiom* (2).

$$T_{finite} \cup T_{interval\_rel} \models T_{allen\_compose} \cup T_{allen\_disjoint}\,,$$
$$T_{finite} \cup T_{interval\_rel} \not\models T_{exhaustive}\,.$$

Proof. In the proofs of Theorem 2, Axiom (2) is not used to entail any sentence in $T_{allen\_compose}$ or $T_{allen\_disjoint}$. A model of $T_{finite} \cup T_{interval\_rel}$ that falsifies $T_{exhaustive}$ can be found at
`http://colore.oor.net/allen_interval_algebra/mappings/theorems/allen2boundedmeeting/exhaustive.model`. ◀

We therefore need a theory that is stronger than T_{finite} but weaker than $T_{interval\_meeting}$. The natural question is therefore: what is the theory in the $\mathbb{H}^{interval\_meeting}$ Hierarchy that is entailed by T_{allen}?

3.3 Bounded Meeting

In this section, we search for a theory that is weaker than $T_{interval\_meeting}$, yet which is still able to interpret T_{allen}. We begin by taking a closer look at the role that the Infinity Axiom plays in the proofs for Theorem 2. This axiom guarantees that each interval is bounded by an earlier and later interval. If we also look at the definitions of the interval relations *starts*, *ends*, *overlaps*, and *during*, we see that each definition entails the existence of two intervals – one that is earlier than the others and one that is later than the others.

Inspired by this observation, we propose the definition of a new relation, *prec*, which specifies an ordering over intervals. The new axioms guarantee the existence of lower and upper bounds for each pair of intervals with respect to this ordering.

▶ **Proposition 5.**

$$T_{interval\_meeting} \models (\forall x, y)(\exists z)\ prec(x, z) \wedge prec(y, z)\,,$$
$$T_{interval\_meeting} \models (\forall x, y)(\exists z)\ prec(z, x) \wedge prec(z, y)\,.$$

$$(\forall i, j)\ meets(i, j) \supset timeinterval(i) \wedge timeinterval(j) \tag{11}$$
$$(\forall i, j, k, m)\ meets(i, k) \wedge meets(j, k) \wedge meets(i, m) \supset meets(j, m) \tag{12}$$
$$(\forall i, j, k, l)\ (meets(i, j) \wedge meets(k, l)) \supset$$
$$(meets(i, l) \vee ((\exists n)\ ((meets(i, n) \wedge meets(n, l)) \vee (meets(k, n) \wedge meets(n, j))))) \tag{13}$$
$$(\forall i, j)\ meets(i, j) \supset \neg meets(j, i) \tag{14}$$
$$(\forall i, j, k, m)\ meets(i, j) \wedge meets(j, k) \wedge meets(k, m) \supset$$
$$(\exists n)\ meets(i, n) \wedge meets(n, m) \tag{15}$$
$$(\forall x, y)(\exists z)\ prec(x, z) \wedge prec(y, z) \tag{16}$$
$$(\forall x, y)(\exists z)\ prec(z, x) \wedge prec(z, y) \tag{17}$$
$$(\forall x, y)\ prec(x, y) \equiv (meets(x, y) \vee ((\exists z)\ meets(x, z) \wedge meets(z, y)) \vee (x = y)) \tag{18}$$

Figure 3 The axioms of $T_{bounded\_meeting}$.

Proof. The proof generated by Prover9 can be found at
`http://colore.oor.net/allen_interval_algebra/theorems/interval-bounded/`. ◀

Thus, $T_{bounded\_meeting}$ is entailed by $T_{interval\_meeting}$, and it entails T_{finite}.

3.4 T_{allen} and $T_{bounded\_meeting}$

Although $T_{bounded\_meeting}$ is weaker than $T_{interval\_meeting}$, it is strong enough to relatively interpret T_{allen}:

▶ **Theorem 6.** *T_{allen} is entailed by $T_{bounded\_meeting} \cup T_{interval\_rel}$.*

Proof. Using Prover9, we have shown[4] that

$$T_{bounded\_meeting} \cup T_{interval\_rel} \models \Phi$$

for each axiom Φ of T_{allen}. ◀

Unlike $T_{interval\_meeting}$, the theory $T_{bounded\_meeting}$ allows finite models, but is it weak enough to be interpreted by T_{allen}?

▶ **Proposition 7.**

$$T_{allen} \not\models (\forall i, j)(\exists k)(meets(i, k) \vee before(i, k) \vee (i = k)) \wedge (meets(j, k) \vee before(j, k) \vee (k = j))\,.$$

Proof. The model generated by Mace4 that falsifies the sentence can be found at
`http://colore.oor.net/allen_interval_algebra/mappings/theorems/allen2boundedmeeting/bounded.model/`. ◀

[4] The input files and proofs can be found at
`http://colore.oor.net/allen_interval_algebra/mappings/theorems/boundedmeeting2allen/`.

$$(\forall x, y)\ before(x, y) \supset (\exists z)\ meets(x, z) \wedge meets(z, y) \tag{19}$$
$$(\forall x, y)\ overlaps(x, y) \supset (\exists z)\ ends(z, x) \wedge starts(z, y) \tag{20}$$
$$(\forall x, y)\ during(x, y) \supset (\exists z)\ ends(x, z) \wedge starts(z, y) \tag{21}$$
$$(\forall x, y)\ starts(x, y) \supset (\exists z)\ meets(z, x) \wedge meets(z, y) \tag{22}$$
$$(\forall x, y)\ ends(x, y) \supset (\exists z)\ meets(x, z) \wedge meets(y, z) \tag{23}$$
$$(\forall x, z)\ starts(x, z) \supset (\exists y)\ before(x, y) \wedge meets(z, y) \tag{24}$$
$$(\forall x, z)\ ends(x, z) \supset (\exists y)\ before(y, x) \wedge meets(y, z) \tag{25}$$

Figure 4 T_{allen_exist}: Additional axioms to extend T_{allen}.

Thus, T_{allen} cannot interpret $T_{bounded\_meeting}$ either, yet a cursory glance at the definitions of the temporal relations in $T_{interval\_rel}$ seems to indicate that it should. In the preceding section, we used the consistency-based definition [3] to axiomatize Allen's Interval Algebra. An alternative approach to the axiomatization of the composition table is known as the extensional definition approach [3] – the composition of R_1 with R_2 is the set of ordered pairs $\langle \mathbf{x}, \mathbf{y} \rangle$ such that for some $\mathbf{z}$, we have $\langle \mathbf{x}, \mathbf{z} \rangle \in R_1$ and $\langle \mathbf{z}, \mathbf{y} \rangle \in R_2$. For example, the extensional definition axiom for the cell $CT(meets, meets)$ is

$$(\forall x, y)\ before(x, y) \equiv (\exists z)\ meets(x, z) \wedge meets(z, y)\,.$$

If we look closely, we notice that there are sentences from the extensional definition of the composition table that are entailed by the definitions of the relations in Figure 2. In particular, each sentence in Figure 4 is entailed by $T_{interval\_meeting} \cup T_{interval\_rel}$, and each of these sentences corresponds to the converse of an axiom from the consistency-based definition of the composition table. Since we are ultimately interested in understanding the relationship between $T_{interval\_meeting}$ and the composition table, we will extend the theory T_{allen} with the sentences of Figure 4, and refer to the resulting theory as T^*_{allen}.

▶ **Lemma 8.** $T_{bounded\_meeting} \cup T_{interval\_rel}$ *entails* T^*_{allen}.

Proof. Using Prover9, we have shown[5] that

$$T_{bounded\_meeting} \cup T_{interval\_rel} \models \Phi$$

for each axiom Φ of T_{allen_exist} (and hence T^*_{allen}). ◀

▶ **Lemma 9.** T^*_{allen} *entails* $T_{bounded\_meeting}$.

Proof. Using Prover9, we have shown[6] that $T^*_{allen} \models \Phi$ for each axiom Φ of $T_{bounded\_meeting}$. ◀

These two Lemmas, it can be shown that T_{allen} and $T_{bounded\_meeting}$ are *mutually interpretable* [6] in each other. Relative interpretation alone does not guarantee a one-to-one

[5] The input files and proofs can be found at `colore.oor.net/allen_interval_algebra/mappings/theorems/boundedmeeting2allen/`.

[6] The input files and proofs can be found at `colore.oor.net/allen_interval_algebra/mappings/theorems/allen2boundedmeeting/`.

correspondence between models of the theories. When T_1 is interpretable in T_2, we can only show every model of T_2 defines a model of T_1 using the translation definitions between T_1 and T_2. Thus, establishing mutual relative interpretation between $T_{bounded\_meeting}$ and T_{allen} does not help with characterizing all models of T_{allen}.

To study models of T_{allen} based on models of $T_{bounded\_meeting}$, we need a notion stronger than relative interpretation:

▶ **Definition 10** ([9]). Two theories T_1 and T_2 are logically synonymous iff they have a common definitional extension.

Considering Definition 10, it is easy to see that T_1 and T_2 are synonymous iff there exist two sets of translation definitions, Δ and Π, such that $T_1 \cup \Pi$ is a definitional extension of T_1, $T_2 \cup \Delta$ is a definitional extension of T_2, and $T_1 \cup \Pi$ and $T_2 \cup \Delta$ are logically equivalent.

When two theories are synonymous, there is a one-to-one correspondence between their models such that the corresponding models can be defined based on each other [12].

▶ **Theorem 11.** *$T_{bounded\_meeting}$ is logically synonymous with T^*_{allen}.*

Proof. By Lemma 8 and Lemma 9, $T_{bounded\_meeting}$ and T^*_{allen} are mutually interpretable. Using Prover9, we have shown[7] that $T^*_{allen} \models \Phi$ for each axiom Φ of $T_{interval\_rel}$. Thus, $T_{bounded\_meeting} \cup T_{interval\_rel}$ and T^*_{allen} are logically equivalent. ◀

According to [12], synonymous theories axiomatize the same class of structures. Thus, $T_{bounded\_meeting}$ and T^*_{allen} are semantically equivalent and only differ in signature.

In what sense has this achieved our objective, since we have we have shown that the time ontology $T_{bounded\_meeting}$ is synonymous with an extension of first-order axiomatization of Allen's interval algebra, rather than the original axiomatization? First, the additional axioms in T_{allen_exist} that we need to prove Theorem 11 are all entailed from the definitions of the interval relations; any attempt to use a weaker axiomatization of the composition table would require us to change these defintions. However, these definitions capture the intended semantics of the interval relations, so any weaker definition would lead to unintended models. The underlying ontology of time intervals therefore plays no role in entailment of the axioms in T_{allen_exist}.

All relations in the composition table can be deduced from T^*_{allen} as it is an extension of T_{allen}. In addition, for every entry $CT(R_i, R_j)$ of the composition table and every interval algebra relation $S \notin CT(R_i, R_j)$ we proved a sentence of the form: $R_i(x, y) \wedge R_j(y, z) \supset \neg S(x, z)$

Thus, the additional axioms of T^*_{allen} does not change the composition table, but only eliminate those models of T_{allen} that do not satisfy the axiomatic definitions of the interval relations.

4 Model-Theoretic Characterization of $T_{bounded\_meeting}$

It is tempting to see the equivalence between T_{allen} and $T_{bounded\_meeting}$ as an intellectual curiosity that does not give us any new insights into the interval algebra. Nevertheless, if we recall that Allen's Interval Algebra is primarily used in constraint satisfaction problems, in which one constructs a satisfying interpretation of a set of expressions in the signature

[7] The input files and proofs can be found at `colore.oor.net/allen_interval_algebra/mappings/theorems/allen2boundedmeeting/`.

of T_{allen}, then the set of all possible solutions of interval algebra problems, excluding those eliminated by T_{allen}, is equivalent to the set of all possible models of $T_{bounded\_meeting}$. In this section, we provide a characterization of the models[8] of $T_{bounded\_meeting}$ up to isomorphism, by first specifying a class of mathematical structures, and then showing that $T_{bounded\_meeting}$ axiomatizes this class of structures.

4.1 Representation Theorem for Models of $T_{bounded\_meeting}$

Verification is concerned with the relationship between the models of the axiomatization of the ontology and a class of mathematical structures. In particular, we want to characterize the models of an ontology up to isomorphism and determine whether or not these models are equivalent to the intended models of the ontology. In this section, we characterize the models of $T_{bounded\_meeting}$.

Since **meets** is a asymmetric binary relation, we turn to directed graphs for the underlying structures:

▶ **Definition 12.** A directed graph is a pair $\langle V, A\rangle$ such that $A \subseteq V \times V$.

Before stating the representation theorem, we need some notation.

▶ **Definition 13.** Let $\mathcal{M} = (V, A)$ be a directed graph. For each $\mathbf{x} \in V$,

$$N(\mathbf{x}) = \{\mathbf{y} \ : \ (\mathbf{x},\mathbf{y}) \in A\} \qquad N^{-1}(\mathbf{x}) = \{\mathbf{y} \ : \ (\mathbf{y},\mathbf{x}) \in A\}\,,$$
$$N^k(\mathbf{x}) = N(N^{k-1}(\mathbf{x})) \qquad N^{-k}(\mathbf{x}) = N^{-1}(N^{-(k-1)}(\mathbf{x}))\,,$$
$$D^k(\mathbf{x}) = \bigcup_{i=1}^{i=k} N^i(\mathbf{x}) \qquad D^{-k}(\mathbf{x}) = \bigcup_{i=1}^{i=k} N^{-i}(\mathbf{x})\,.$$

▶ **Definition 14.** $\mathfrak{M}^{bounded\_meeting}$ is the following class of structures:
$\mathcal{M} \in \mathfrak{M}^{bounded\_meeting}$ iff $\mathcal{M} = (V, A)$ is a directed graph such that

1. $N(\mathbf{x}) \cap N^2(\mathbf{x}) = \emptyset$ for any $\mathbf{x} \in V$;
2. $N^3(\mathbf{x}) \subseteq N^2(\mathbf{x})$ for any $\mathbf{x} \in V$;
3. for any $\mathbf{x}, \mathbf{y} \in V$,

$$D^2(\mathbf{x}) \cap D^2(\mathbf{y}) \neq \emptyset\,,$$
$$D^{-2}(\mathbf{x}) \cap D^{-2}(\mathbf{y}) \neq \emptyset\,;$$

4. $N(\mathbf{x}) \subseteq N^2(\mathbf{y})$ or $N(\mathbf{y}) \subseteq D^2(\mathbf{x})$, for any $\mathbf{x}, \mathbf{y} \in V$,

▶ **Theorem 15.** *There exists a bijection $\varphi : Mod(T_{bounded\_meeting}) \to \mathfrak{M}^{bounded\_meeting}$ such that*

1. $\langle \mathbf{x}\rangle \in \mathbf{timeinterval}$ *iff* $\mathbf{x} \in V^{\varphi(\mathcal{M})}$*;*
2. $\langle \mathbf{x}, \mathbf{y}\rangle \in \mathbf{meets}^{\mathcal{M}}$ *iff* $(\mathbf{x}, \mathbf{y}) \in A^{\varphi(\mathcal{M})}$.

Proof (Sketch). Condition 1 in Definition 14 is equivalent to the sentence

$$(\forall x, y, z)\ meets(x, y) \wedge meets(y, z) \supset \neg meets(x, z)$$

[8] We denote *structures* by calligraphic uppercase letters, e.g. $\mathcal{M}, \mathcal{N}$; elements of a structure by **boldface** font, e.g., $\mathbf{a}, \mathbf{b}$; and the *extension of predicate* R in a structure $\mathcal{M}$ by $\mathbf{R}^{\mathcal{M}}$.

and using Prover9[9], we can show that this sentence is logically equivalent to Axiom (12). Condition 2 in Definition 14 is equivalent to Axiom (15). Condition 3 in Definition 14 is equivalent to Axioms (16) and (16). Condition 4 in Definition 14 is equivalent to Axiom (13). Axiom (14) is equivalent to the property that $\mathcal{M}$ is a directed graph. ◀

We can therefore refer interchangably to the models of $T_{bounded\_meeting}$ and structures in $\mathfrak{M}^{bounded\_meeting}$.

4.2 Representation Theorem for $\mathfrak{M}^{bounded\_meeting}$

Although Theorem 15 characterizes the models of $T_{bounded\_meeting}$, it leaves unresolved the explicit characterization of $\mathfrak{M}^{bounded\_meeting}$, and a deeper understanding of how to construct models of $T_{bounded\_meeting}$. In this section, we characterize the class of directed graphs that satisfy the conditions in Definition 14.

Although structures in $\mathfrak{M}^{bounded\_meeting}$ are directed graphs, it will be easier to construct them from undirected graphs, allowing us to exploit a wider range of existing work in graph theory.

▶ **Definition 16.** An undirected graph is a pair $G = \langle V, E\rangle$ of sets such that $E \subseteq \{V\}^2$.

Directed and undirected graphs are related to each other through the notion of orientation.

▶ **Definition 17.** $G_1 = \langle V, A\rangle$ is an orientation of an undirected graph $G_2 = \langle V, E\rangle$ iff for each $(\mathbf{x}, \mathbf{y}) \in E$, either $(\mathbf{x}, \mathbf{y}) \in A$ or $(\mathbf{y}, \mathbf{x}) \in A$.

If G_1 is a directed graph, then G_2 is the undirected graph for G_1 iff G_1 is an orientation of G_2.

We therefore need to identify the class of graphs that correspond to structures in $\mathfrak{M}^{bounded\_meeting}$.

4.2.1 Twin-free Graphs

We first notice that

$$T_{bounded\_meeting} \not\models (\forall x, y, z, u)\, meets(x, y) \wedge meets(x, z) \wedge meets(y, u) \wedge meets(z, u) \supset (y = z)\,.$$

Thus there exist models in which there exist multiple intervals that meet and are met by the same intervals.

▶ **Definition 18.** Let $G = (V, E)$ be a graph. Two vertices $\mathbf{x}, \mathbf{y} \in V$ are twins iff for all other vertices $\mathbf{w}$, we have $(\mathbf{x}, \mathbf{w}) \in E$ iff $(\mathbf{y}, \mathbf{w}) \in E$.

G is twin-free iff it contains no twins.

▶ **Definition 19.** Suppose $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are graphs such that $V_1 \subset V_2$ and $E_1 \subset E_2$. G_2 is a twinned extension of G_1 iff each $\mathbf{x} \in V_2 \setminus V_1$ is a twin of some $\mathbf{y} \in V_1$.

▶ **Lemma 20.** *If $\mathcal{M} \in \mathfrak{M}^{bounded\_meeting}$, and $\mathcal{N}$ is a twinned extension of $\mathcal{M}$, then $\mathcal{N} \in \mathfrak{M}^{bounded\_meeting}$.*

[9] http://color.oor.net/allen_interval_algebra/theorems/triangle_free/

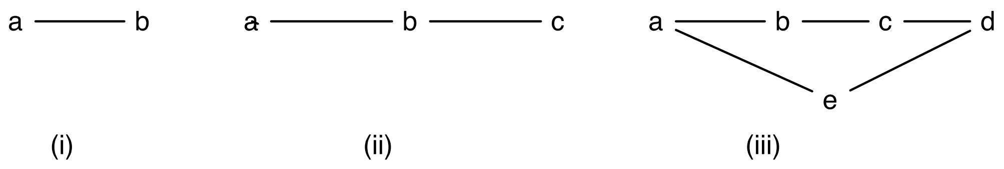


Figure 5 Short models.

Proof. If $\mathbf{x}, \mathbf{y} \in V$ are twins, then

$$N^k(\mathbf{x}) = N^k(\mathbf{y}) \qquad N^{-k}(\mathbf{x}) = N^{-k}(\mathbf{y})$$

so that $\mathcal{N}$ satisfies all conditions in Definition 14 and hence $\mathcal{N} \in \mathfrak{M}^{bounded\_meeting}$. ◀

▶ **Lemma 21.** *Each $\mathcal{M} \in \mathfrak{M}^{bounded\_meeting}$ contains a unique twin-free subgraph.*

Proof. If $\mathbf{x}, \mathbf{y} \in V$ are twins, then removing $\mathbf{y}$ does not change $N^k(\mathbf{x})$ or $N^{-k}(\mathbf{x})$, so that the conditions in Definition 14 will be satisfied by the subgraph of $\mathcal{M}$ that is induced by $V \setminus \{\mathbf{y}\}$. ◀

Thus, we can characterize $\mathfrak{M}^{bounded\_meeting}$ by characterizing the twin-free graphs in $\mathfrak{M}^{bounded\_meeting}$.

4.2.2 Building Blocks

We now consider some special graphs that serve as the basic structures from which models of $T_{bounded\_neeting}$ are constructed.

▶ **Definition 22.** A short model is an undirected graph that is isomorphic to either P_2 (path graph with two vertices), P_3 (path graph with three vertices), or C_5 (cyclic graph with five vertices).

The three short models are depicted in Figure 5, and they each correspond to a different set of temporal relations. The orientation of P_2 is the smallest model of $T_{bounded\_meeting}$ with a nontrivial extension of the **meets** relation. The orientation of P_3 is the smallest model of $T_{bounded\_meeting}$ with a nontrivial extension of the **before** relation. The orientation of C_5 is the smallest model of $T_{bounded\_meeting}$ with a nontrivial extension of the **starts** and **ends** relations.

Note that the orientation of a cyclic undirected graph can be a directed acyclic graph.

▶ **Lemma 23.** *If $\mathcal{M}$ is an orientation of a short model G, then $\mathcal{M} \in \mathfrak{M}^{bounded\_meeting}$.*

▶ **Definition 24.** The $\mathbb{H}$ graph is the connected graph on six vertices such that exactly two vertices have degree 3 and the remaining vertices have degree 1.

Figure 6(ii) shows two examples of $\mathbb{H}$. Now $\mathbb{H}$ by itself is not the undirected graph for any structure in $\mathfrak{M}^{bounded\_meeting}$; however, we can use $\mathbb{H}$ to construct a graph that will play a critical role in the characterization of $\mathfrak{M}^{bounded\_meeting}$.

We first need to specify a way in which new graphs can be constructed from existing ones.

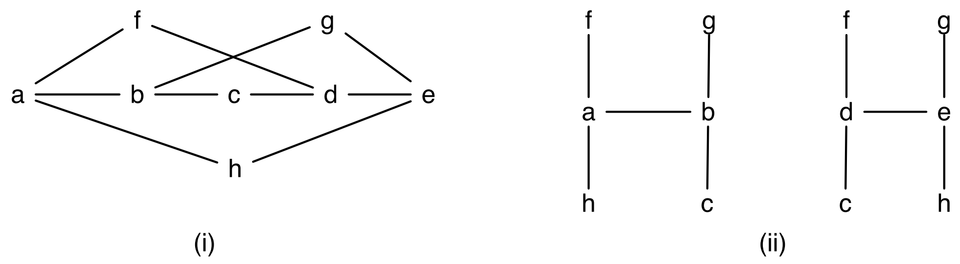


Figure 6 The model $\mathbb{H} \uplus \mathbb{H}$, and its edge-decomposition into two $\mathbb{H}$ graphs.

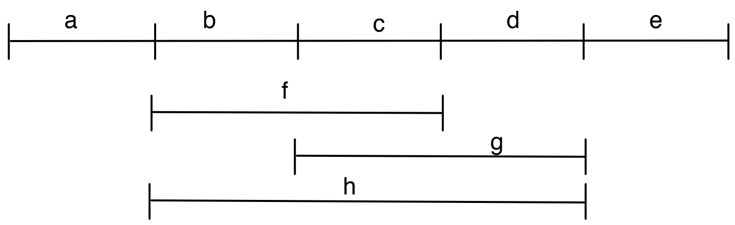


Figure 7 A depiction of the interval relations corresponding to an $\mathbb{H} \uplus \mathbb{H}$ graph.

▶ **Definition 25.** A graph $\mathcal{G} = \langle V, E \rangle$ is edge-decomposable into a set of graphs $\mathcal{H}$ iff

1. $\mathcal{H}_i \subset \mathcal{G}$, for each $\mathcal{H}_i \in H$;
2. $E_i \cap E_j = \emptyset$, for each $\mathcal{H}_i = \langle V_i, E_i \rangle$ and $\mathcal{H}_j = \langle V_j, E_j \rangle$;
3. $E = \bigcup_i E_i$.

Thus, a graph $\mathcal{G}$ is edge-decomposable into a set of subgraphs iff the set of edges in $\mathcal{G}$ can be partitioned. Figure 6(i) depicts the graph $\mathbb{H} \uplus \mathbb{H}$; it is edge-decomposable into the two graphs in Figure 6(ii). We will use the notation $\mathcal{G} = \mathcal{H}_1 \uplus \ldots \uplus \mathcal{H}_n$ to indicate that $\mathcal{G}$ is edge-decomposable into $\mathcal{H}_1, \ldots, \mathcal{H}_n$.

Elsewhere in graph theory, $\mathbb{H} \uplus \mathbb{H}$ is the unique extremal graph of order 8, that is, it contains the maximal number of edges for a graph of girth[10] 5 on 8 vertices.

▶ **Lemma 26.** *If $\mathcal{M}$ is an orientation of $\mathbb{H} \uplus \mathbb{H}$, then $\mathcal{M} \in \mathfrak{M}^{bounded\_meeting}$.*

Figure 7 illustrates how $\mathbb{H} \uplus \mathbb{H}$ is related to the interval relations.

4.3 Characterization of $\mathfrak{M}^{bounded\_meeting}$

In the preceding section, we explicitly identified some graphs that are structures in $\mathfrak{M}^{bounded\_meeting}$, but we are ultimately interested in characterizing *all* such graphs.

▶ **Definition 27.** A graph G is triangle-free iff it does not contain any induced cycles of length 3 (i.e. G contains no induced K_3 subgraphs).

▶ **Definition 28.** A graph G is 2-connected iff there exist two vertex-disjoint paths between any two vertices in G.

[10] The girth of a graph G is the length of the shortest cycle in G.

A well-known result in graph theory [5] shows that a graph is 2-connected iff each pair of vertices are elements of the same cycle.

▶ **Definition 29.** The distance between two vertices in a graph G is the number of edges in a shortest path that connects the two vertices.

The diameter of a graph G (denoted by $diam(G)$) is the greatest distance between any two vertices of G.

▶ **Lemma 30.** *Suppose $\mathcal{M} \in \mathfrak{M}^{bounded\_meeting}$ and let G be the undirected graph for $\mathcal{M}$ such that G is not a short model. G is a 2-connected triangle-free graph such that $diam(G) = 3$.*

Proof. G is triangle-free iff for any $\mathbf{x} \in V$, $N(\mathbf{x}) \cap N^2(\mathbf{x}) = \emptyset$, which is equivalent to condition (1) in Definition 14. By condition (3) in Definition 14, there are at most two vertices between any $\mathbf{x}, \mathbf{y}$, so that $diam(G) = 3$ (and hence G is connected). Suppose G is not 2-connected; since G is not a short model, there must exist $\mathbf{x}, \mathbf{y} \in V$ such that there is a unique P_4 subgraph with $\mathbf{x}$ and $\mathbf{y}$ as endpoints. However, this violates condition (2) in Definition 14. ◀

Note that the converse of this Lemma does not hold; we need to determine which 2-connected triangle-free graphs are structures in $\mathfrak{M}^{bounded\_meeting}$.

▶ **Theorem 31.** *Suppose $\mathcal{M}$ is the orientation of a twin-free graph. $\mathcal{M} \in \mathfrak{M}^{bounded\_meeting}$ iff $\mathcal{M}$ is the orientation of a short model or of a graph in which any two elements $\mathbf{x}, \mathbf{y}$ are elements of a subgraph $G \subseteq \mathcal{M}$ such that*

$$G \cong \mathbb{H} \cup \mathbb{H}.$$

Proof (Sketch). By Lemma 30, any two elements $\mathbf{x}, \mathbf{y}$ are elements of an induced P_2, P_3, C_5, or C_6 subgraph. By condition (2) in Definition 14, there exist additional elements $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3} \in V$ that create new C_5 subgraphs, so that the C_6 subgraph generates an $\mathbb{H} \cup \mathbb{H}$ subgraph that contains $\mathbf{x}, \mathbf{y}$. ◀

Finally, we can give a constructive characterization for finite structures in $\mathfrak{M}^{bounded\_meeting}$.

▶ **Definition 32.** Let G_1,G_2 be two graphs that each contain an induced P_k subgraph. A graph obtained from G_1 and G_2 by identifying the two subgraphs is a P_k-gluing of G_1 and G_2.

▶ **Definition 33.** A full graph is a graph in which each P_4 subgraph is an edge cover of an induced C_5 subgraph.

▶ **Theorem 34.** *Suppose $\mathcal{M} \in \mathfrak{M}^{bounded\_meeting}$ such that $\mathcal{M}$ is twin-free and finite, and that G is the undirected graph for $\mathcal{M}$.*

- *G is not 2-connected iff $G \cong P_2$ or $G \cong P_3$.*
- *G is 2-connected iff there exists a sequence $G_1, \ldots, G$ such that*
  1. *$G_1 \cong P_n$;*
  2. *G is a full graph;*
  3. *G_{i+1} is the result of P_4-gluing G_i and C_5.*

Proof (Sketch). By Lemma 23 and Lemma 30, G is connected but not cyclic iff $G \cong P_2$ or $G \cong P_3$. Suppose G is 2-connected. By Theorem 31, each pair of vertices are elements of an $\mathbb{H} \cup \mathbb{H}$ subgraph, so that G is a full graph. Let P_k be the longest path in G. By condition (2) in Definition 14, every P_4 subgraph is an edge cover of an induced C_5 subgraph, which is equivalent to the P_4-gluing of a subgraph to C_5. ◀

Recall the original motivation for this work – since all applications of Allen's Interval Algebra consists in the specification of temporal constraints and the use of constraint satisfaction techniques to find solutions. Such solutions correspond to models of T^*_{allen}. Given the synonymy of T^*_{allen} and $T_{bounded\_meeting}$ (Theorem 11), these last two results not only give us a complete characterization of the finite models of $T_{bounded\_meeting}$ up to isomorphism, but they also give us a complete characterization of the solutions of a set of temporal constraints.

5 Summary

Constraint satisfaction with relational calculi such as Allen's Interval Algebra has been the predominant application of temporal concepts within commonsense reasoning. Yet in some way, this has diminished the role played by the different time ontologies that provide their foundations. It has long been known that the first-order theory of Allen's Interval Algebra is interpretable by certain ontologies of time intervals, in particular, the ontology $T_{interval\_meeting}$. This perspective has been considered sufficient for showing that Allen's Interval Algebra was in some sense sound with respect to its ontological foundations. In this paper, we have specified an ontology $T_{bounded\_meeting}$ that is weaker than $T_{interval\_meeting}$ and which is logically synonymous with Allen's Interval Algebra. Finally, we have provided a characterization of the models of $T_{bounded\_meeting}$ up to isomorphism, by first specifying a class of mathematical structures, and then showing that $T_{bounded\_meeting}$ axiomatizes this class of structures. This characterization gives us insights into the set of all possible solutions for a set of temporal constraints that can be specified by Allen's Interval Algebra.

The next step in this direction is a full characterization of the infinite models of $T_{bounded\_meeting}$, which would provide an alternative characterization of the models of $T_{interval\_meeting}$. It would also enable us to explore extensions of $T_{bounded\_meeting}$ that axiomatize dense orderings. This is equivalent to revisiting Hayes' and Ladkins description of models with respect to intervals on $\mathbb{Q}$ and $\mathbb{Z}$. Formalize these results as representation theorems.

Another major question is the relationship of $T_{bounded\_meeting}$ to other ontologies of time intervals, such as periods [4], in which the primitive relations are precedence and inclusion rather than meets. In particular, this would require a characterization of the mereology of intervals that is definable within models of $T_{interval\_meeting}$.

References

1 J. Allen and P. Hayes. Moments and points in an interval-based temporal logic. *Computational Intelligence*, 5:225–238, 1989.

2 James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983. doi:10.1145/182.358434.

3 B. Bennett, A. Isli, and A. G. Cohn. When does a composition table provide a complete and tractable proof procedure for a relational constraint language? In *Proceedings of the IJCAI-97 workshop on Spatial and Temporal Reasoning*, 1997.

4 J. van Benthem. *Logic of Time*. Springer Verlag, 1983.

5 R. Diestel. *Graph Theory*. Springer Verlag, 1997.

6 H. Enderton. *Mathematical Introduction to Logic*. Academic Press, 1972.

7 Michael Grüninger, Torsten Hahmann, Ali Hashemi, Darren Ong, and Atalay Özgövde. Modular first-order ontologies via repositories. *Applied Ontology*, 7(2):169–209, 2012.

8 P. Hayes. Catalog of temporal theories. Technical Report Technical Report UIUC-BI-AI-96-01, University of Illinois Urbana-Champagne, 1996.

9 W. Hodges. *Model theory*. Cambridge University Press Cambridge, 1993.

10 P. B. Ladkin. Models for axioms of time intervals. In *Proceedings of AAAI-87*, 1987.

11 W. McCune. Prover9 and Mace4. 2010. URL: `http://www.cs.unm.edu/~mccune/prover9/`.

12 Charles Pinter. Properties preserved under definitional equivalence and interpretations. *Zeitschrift fur Mathematik Logik und Grundlagen der Mathematik*, 24:481–488, 1978.

The Fully Hybrid μ-Calculus

Daniel Kernberger[1] **and Martin Lange**[2]

1 **School of Electrical Engineering and Computer Science, University of Kassel, Kassel, Germany**

2 **School of Electrical Engineering and Computer Science, University of Kassel, Kassel, Germany**

Abstract

We consider the hybridisation of the μ-calculus through the addition of nominals, binder and jump. Especially the use of the binder differentiates our approach from earlier hybridisations of the μ-calculus and also results in a more involved formal semantics. We then investigate the model checking problem and obtain EXPTIME-completeness for the full logic and the same complexity as the modal μ-calculus for a fixed number of variables. We also show that this logic is invariant under hybrid bisimulation and use this result to show that – contrary to the non-hybrid case – the hybrid extension of the full branching time logic CTL* is not a fragment of the fully hybrid μ-calculus.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases μ-calculus, hybrid logics, model checking, bisimulation invariance

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.17

1 Introduction

Hybrid Extensions of Modal Logic

Hybrid logic [19, 4] has emerged from modal logic as an attempt to extend a well-behaved but relatively weak (in terms of expressive power) fragment of first-order logic with additional features whilst retaining good properties like decidability etc. This is achieved by extending the syntax of modal logic with first-order variables and some very restricted form of first-order quantification over these variables.

The availability of first-order variables in the language gives the logic the power to express properties that are inherently non-modal. For instance, it is possible to express that a state of a Kripke structure has an edge to itself; it is – even without the definition of a formal semantics – not hard to guess that the formula $x \wedge \Diamond x$ should be true at exactly the states of that kind. The two other typical operators are the *binder* and the *jump*. Intuitively, $\downarrow x.\varphi$ binds x to the current state for the evaluation of φ. It is sometimes also known as the *freeze* modality [1]. The jump operator – also called the *satisfaction* operator [4] – is written $@_x\,\varphi$ and, intuitively, continues the evaluation of φ at the state that is bound to x. For both operators it is important to remember that modal formulas, as opposed to (e.g. closed) first-order formulas are interpreted at states of a Kripke structure, not the structure as a whole.

Sometimes, when defining hybrid logics, one distinguishes two kinds of variables, depending on whether they can be bound or not, and calls those that do not get bound *nominals*.

The additional power induced by these hybrid features and operators comes at a cost when compared to modal logic. Clearly, hybrid logics do not possess the tree model property anymore, as the little example $x \wedge \Diamond x$ above shows. Strongly related to that is the loss of bisimulation-invariance, an inherent feature of modal logic [5]. The notion of bisimulation has been refined accordingly to *hybrid bisimulation* which relates two states when they can mimick each others transitions locally in the presence of a fixed number of named states. It has been shown that the hybrd extension of modal logic with binder and jump is invariant under this equivalence relation [3].

The term *hybrid logic* suggests speaking about one particular logic but in fact denotes a family of logics; its members are obtained by extending *some* modal logic with *some* of the features mentioned above. This is of course not restricted to modal logics alone, any logic that is not subsumed by first-order logic is a natural candidate for the basis of a hybrid logic. For instance, temporal and dynamic logics have been extended in this way, namely the EF-fragment of CTL with past operators [9], CTL and CTL$^+$ [11], as well as CTL* [12]. In [11] the hybrid extensions of CTL and CTL$^+$ are only interpreted over computation trees to retain decidability of the satisfiability problem. However, the semantics naturally extends to Kripke structures. Hybrid CTL and CTL$^+$ are then – as in the non-hybrid case – subsumed by the hybrid extensions of CTL* in [12].

Hybrid Extensions of the μ-Calculus

In this paper we consider the extension of the well-known modal μ-calculus L_μ [13] with hybrid operators. This is not the first attempt at doing so; Sattler and Vardi [16] have considered a hybrid μ-calculus which extends L_μ with nominals only, i.e. with additional first-order variables but no mechanism to change them during the course of the evaluation of a formula. This is in some sense a smallest hybrid extension of L_μ, even though they use the term *hybrid full μ-calculus* for this logic. "Full" in that context seems to refer to the addition of converse modalities. They are, however, thrown away then in favour of a universal modality with which one can jump to any state in an underlying Kripke structure. It is easy to see that this subsumes the specialised jumps $@_x$ conventionally used in hybrid logics.

Here we consider a different logic, namely the extension of the modal μ-calculus with *all* hybrid features, in particular including binders. To subtly distinguish these logics, we refer to the one used here as the *fully hybrid μ-calculus*. This then clearly subsumes full hybrid modal logic for which satisfiability is undecidable [4]. The context of Sattler and Vardi's hybrid μ-calculus of course forbids this, as their primary interest is description logics for which decidability of satisfiability is a must. The motivation for the extension of L_μ with all hybrid features here is not driven by concrete applications; we study this logic in order to understand the effect that extending temporal logics in various ways has on their logical and computational properties.

Contribution and Organisation

The paper is organised as follows. Section 2 defines the fully hybrid μ-calculus formally. We discuss that the semantics of hybrid temporal logics is inadequate for a logic with fixpoint quantifiers as under this semantics the implicit recursion mechanism does not obey the meaning one would intuitively expect the binder to have. This is fixed by letting second-order variables stand for sets of pairs of states and first-order variable bindings, rather than sets of states only.

Section 3 examines the complexity of model checking the fully hybrid μ-calculus. Using a reduction to L_μ model checking we obtain (1) an EXPTIME upper bound in general, (2) that for formulas with a fixed number of variables the complexity is only polynomially worse than that for L_μ model checking, and (3) a game-theoretic characterisation similar to the one for L_μ [17] which can be used to understand the properties expressed by formulas of the fully hybrid μ-calculus. We also show that the EXPTIME upper bound is tight by giving a matching lower bound.

In Section 4 we investigate questions of the logic's expressiveness. We prove that, not surprisingly, invariance under hybrid bisimulations carries over from hybrid modal logic to the fully hybrid μ-calculus. To ease argumentation in this context, we also develop a game-theoretic characterisation of hybrid bisimilarity similar to the well-known bisimulation games [18]. We then use these games to show indistinguishability between two different Kripke structures and deduce that, perhaps surprisingly, the fully hybrid μ-calculus does not subsume the hybrid extensions of CTL*, namely that it cannot express the property "there is a path on which no state occurs twice", even though this is easily possible in hybrid CTL*.

We conclude the paper with a discussion on further work in this area.

2 Preliminaries

2.1 Syntax

Let $k \in \mathbb{N}$, $\mathcal{V} = \{x_1, x_2, \dots, x_k\}$ be a finite set of first-order variables, $\mathit{Prop} = \{p, q, \dots\}$ be a countable set of atomic propositions, $\mathcal{V}_2 = \{X, Y, \dots\}$ be a countable set of second-order variables and $\mathit{Nom} = \{m, n, \dots\}$ be a countable set of first-order constants referred to as nominals. All sets are assumed to be pairwise disjoint. Formulas of the k-variable fragment of the fully hybrid μ-calculus H_μ^k are given by the grammar

$$\varphi := p \mid x \mid X \mid \neg\varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid @_x\,\varphi \mid {\downarrow} x.\varphi \mid \mu X.\varphi(X)$$

where $p \in \mathit{Prop}$, $x \in \mathcal{V} \cup \mathit{Nom}$ and $X \in \mathcal{V}_2$. The fully hybrid μ-calculus H_μ is the union of all H_μ^k for $k \geq 1$. The modal μ-calculus L_μ is obtained as H_μ in the special case when $\mathcal{V} = \mathit{Nom} = \emptyset$.

We are making use of $\mathtt{tt}$, $\mathtt{ff}$, $\wedge$, $\Diamond$, $\nu X.\varphi$ as abbreviations in the usual way. By $\mathsf{Sub}(\varphi)$ we denote the set of all subformulas of φ. Further, we say that a formula $\varphi \in H_\mu$ is in *negation normal form* if and only if negation only occurs directly in front of atomic formulas.

We assume the following standard sanity condition on formulas: every $X \in \mathcal{V}_2$ is bound at most once by a fixpoint quantifier μ or ν and can only occur under an even number of negations. The function mapping each $X \in \mathcal{V}_2$ to its unique binding formula is called fp_φ. We say that a second-order variable X is of type μ or ν if its defining fixpoint formula $\mathsf{fp}_\varphi(X)$ is a least, resp. greatest fixpoint formula. Formulas with no free first-order variables will be referred to as sentences.

All results easily extend to a multi-modal version of H_μ; for the sake of simplicity we only work with a uni-modal version here.

2.2 Considerations on the Semantics in the Presence of Fixpoints

A Kripke structure is a tuple $K = \langle S, \to, L\rangle$ where S is a set of states, $\to \subseteq S \times S$ is a transition relation and $L : \mathit{Prop} \to 2^S$ labels the states with the sets of propositions that hold true in them.

Formulas of L_μ are usually interpreted over Kripke structures via a mapping $[\![\cdot]\!]_\rho^K$, which maps a formula together with a Kripke structure K as above and an assignment $\rho : \mathcal{V}_2 \to 2^S$ to the states that satisfy this formula. A formula $\varphi(X)$ with a free second-order variable X

thus induces a monotonic operator $V \mapsto [\![\varphi(X)]\!]^K_{\rho[X\mapsto V]}$, mapping a set V of states to the set of states that satisfy $\varphi(X)$ under the assumption that X holds on the states in V.

The hybrid μ-calculus considered by Sattler and Vardi [16] can be given a semantics in the same way, in particular with an interpretation of type $\mathcal{V}_2 \to 2^S$ of the second-order variables. First-order variables are nominals in the absence of a binder; hence, their interpretation can be fixed in the Kripke structure by extending the labelling function to the type $L : \mathit{Prop} \cup \mathit{Nom} \to 2^S$ with the requirement that $L(m)$ is a singleton set for all $m \in \mathit{Nom}$.

This, however, is not enough in the presence of the binder modality as it should change the mapping of first-order variables to states dynamically during the evaluation of a formula. The naïve approach is to extend the assignment ρ of all second-order variables to a function $\mathcal{V} \cup \mathcal{V}_2 \to 2^S$ such that $\rho(x)$ is a singleton set for each $x \in \mathcal{V}$. This is essentially incorporating their treatment in hybrid temporal logics, c.f. [12]. We could then just extend the usual semantics for L_μ to H_μ via

$$[\![x]\!]^K_\rho = \rho(x)$$
$$[\![@_x\, \varphi]\!]^K_\rho = \begin{cases} S & \text{if } \rho(x) \in [\![\varphi]\!]^K_\rho, \\ \emptyset & \text{otherwise} \end{cases}$$

as it was done in [16] and

$$[\![\downarrow x.\varphi]\!]^K_\rho = \{s \in S \mid s \in [\![\varphi]\!]^K_{\rho[x\to s]}\}.$$

for the binder modality. However, this does not capture the intuition one would have about the interaction between binders and fixpoint recursion; namely that bindings made in one iteration have an effect on the following iterations.

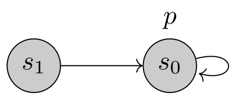
▶ **Example 1.** Consider the formula $(p \wedge \neg x) \vee \downarrow x.\Diamond X$. Obviously the value of x is supposed to change throughout the evaluation of this formula: the second disjunct is satisfied by a tuple (K, s, ρ) if $(K, s, \rho[x \to s])$ satisfies $\Diamond X$. However, the update on ρ does not have any impact on the valuation of $\Diamond X$ because under the standard μ-calculus semantics extended as stated above, $\Diamond X$ is evaluated without involving x at all and thus $(p\wedge\neg x)\vee\downarrow x.\Diamond X \equiv (p\wedge\neg x)\vee\Diamond X$.

Now consider the least fixpoint of the transformation defined by this formula, $\psi := \mu X.(p \wedge \neg x) \vee \downarrow x.\Diamond X$. This change in $\rho(x)$ *should* have some impact on the fixpoint in the sense that $\Diamond X$ should be calculated relative to the new valuation of x. Moreover, the unfolding principle for fixpoints postulates that X should just be a placeholder for ψ, but the evaluation of ψ surely depends on the value of x. Nonetheless, we have

$$\begin{aligned} \mu X.(p \wedge \neg x) \vee \downarrow x.\Diamond X &\equiv \mu X.(p \wedge \neg x) \vee \Diamond X \\ &\not\equiv (p \wedge \neg x) \vee \downarrow x.\Diamond(\mu X.(p \wedge \neg x) \vee \downarrow x.\Diamond X)\ . \end{aligned} \tag{1}$$

The equivalence is a simple consequence of the fact that – under the semantics proposed above – we have $\downarrow x.\varphi \equiv \varphi$ whenever x is not free in φ. The inequivalence is also easy to grasp: the left-hand side is evaluated independently of the update of x. The right-hand side, however, updates x once before the evaluation of the fixpoint formula is started, then with the new value of x.

To illustrate this, we evaluate both formulas on the following simple Kripke structure:



It is quite obvious that $s_1, \{x \mapsto s_0\} \not\models \mu X.(p \wedge \neg x) \vee \Diamond X$ because under this assignment for the variable x no state satisfies $p \wedge \neg x$ and thus the fixpoint is just the empty set.

However, $s_1, \{x \mapsto s_0\} \models {\downarrow} x.\Diamond(\mu X.(p \wedge \neg x) \vee {\downarrow} x.\Diamond X)$ because changing the assignment of x to s_1 and then calculating the fixpoint results in $\{s_1, s_0\}$ and thus $s_1, \{x \mapsto s_0\} \models (p \wedge \neg x) \vee {\downarrow} x.\Diamond(\mu X.(p \wedge \neg x) \vee {\downarrow} x.\Diamond X)$. This is because after unfolding the fixpoint once, it is calculated relative to the updated valuation of x rather than the old.

On the other hand, the last formula in (1) should be equivalent to the first one in there because of the desirable equivalence $\mu X.\varphi(X) \equiv \varphi(\mu X.\varphi)$ – the aforementioned unfolding principle.

This example shows that the proposed semantics is inadequate for the fully hybrid μ-calculus including the binder modality. Interestingly, the fixpoint principle still holds semantically but not syntactically, i.e. in general we have

$$[\![\mu X.\varphi(X)]\!]^K_\rho \quad = \quad [\![\varphi(X)]\!]^K_{\rho[X \mapsto [\![\mu X.\varphi(X)]\!]^K_\rho]}$$

for any K, X, φ and ρ, but

$$[\![\mu X.\varphi(X)]\!]^K_\rho \quad \neq \quad [\![\varphi(\mu X.\varphi(X))]\!]^K_\rho$$

as the evaluation of the outer φ may change the variable assignment ρ that is used for the evaluation of the inner fixpoint formula. In other words, this semantics is not compositional, i.e. in general we have

$$[\![\varphi[\psi/X]]\!]^K_\rho \quad \neq \quad [\![\varphi]\!]^K_{\rho[X \mapsto [\![\psi]\!]^K_\rho]} .$$

2.3 A Compositional Semantics

To account for such phenomena we propose a new semantics for the fully hybrid μ-calculus. Formulas are still interpreted over Kripke structures $K = \langle S, \to, L \rangle$. However, the meaning of a formula is now a set pairs consisting of a state and an assignment for the first-order variables. Consequently the variable assignment has to map second-order variables to the same type; it becomes an assignment $\rho : \mathcal{V}_2 \to 2^{S \times (\mathcal{V} \to S)}$.

Formally the semantics for H^k_μ for all k with respect to a Kripke structure $K = \langle S, \to, L \rangle$ over *Prop* and *Nom* and an assignment $\rho : \mathcal{V}_2 \to 2^{S \times (\mathcal{V} \to S)}$ is the following:

$$\begin{aligned}
[\![p]\!]^K_\rho &= \{(s, \sigma) \in S \times (\mathcal{V} \to S) \mid s \in L(p)\}, \\
[\![x]\!]^K_\rho &= \{(s, \sigma) \in S \times (\mathcal{V} \to S) \mid s = \sigma(x)\}, \\
[\![X]\!]^K_\rho &= \rho(X), \\
[\![\neg\varphi]\!]^K_\rho &= \{(s, \sigma) \in S \times (\mathcal{V} \to S) \mid (s, \sigma) \notin [\![\varphi]\!]^K_\rho\}, \\
[\![\varphi_1 \vee \varphi_2]\!]^K_\rho &= [\![\varphi_1]\!]^K_\rho \cup [\![\varphi_2]\!]^K_\rho, \\
[\![\Box\varphi]\!]^K_\rho &= \{(s, \sigma) \in S \times (\mathcal{V} \to S) \mid \forall t \in S : \text{ if } s \to t, \text{ then } (t, \sigma) \in [\![\varphi]\!]^K_\rho\}, \\
[\![@_x\, \varphi]\!]^K_\rho &= \{(s, \sigma) \in S \times (\mathcal{V} \to S) \mid (\sigma(x), \sigma) \in [\![\varphi]\!]^K_\rho\}, \\
[\![{\downarrow} x.\varphi]\!]^K_\rho &= \{(s, \sigma) \in S \times (\mathcal{V} \to S) \mid (s, \sigma[x \mapsto s]) \in [\![\varphi]\!]^K_\rho\}, \\
[\![\mu X.\varphi(X)]\!]^K_\rho &= \bigcap\{T \subseteq S \times (\mathcal{V} \to S) \mid [\![\varphi]\!]^K_{\rho[X \to T]} \subseteq T\}
\end{aligned}$$

with $p \in \mathit{Prop} \cup \mathit{Nom}$, $x \in \mathcal{V}$ and $X \in \mathcal{V}_2$.

We will write $K, s, \sigma, \rho \models \varphi$ if $(s, \sigma) \in [\![\varphi]\!]^K_\rho$. If there are no free second-order variables we also may drop ρ. Furthermore, we will also sometimes write $(s_1, \ldots, s_k)$ to indicate the function $\sigma : \mathcal{V} \to S$ with $\sigma(x_i) = s_i$ when an order on $\mathcal{V}$ is implictly given, for instance when $\mathcal{V} = \{x_1, \ldots, x_k\}$. To shorten notation even further we sometimes write $K, s \models \varphi$ to express that $K, s, (s, \ldots, s) \models \varphi$.

▶ **Example 2.** Reconsider the formula given in Example 1, now with the new semantics proposed above. We claim

$$\mu X.(p \wedge \neg x) \vee \downarrow x.\Diamond X \quad \equiv \quad (p \wedge \neg x) \vee \downarrow x.\Diamond(\mu X.(p \wedge \neg x) \vee \downarrow x.\Diamond X)$$

holds. We do not prove this formally here; see Proposition 3 below for a general statement. Instead we just give a hint that now this equivalence holds by re-evaluating both formulas on the Kripke structure given in Example 1.

Having two states and one variable, the domain for the semantics is the set of subsets of

$$\{(s_0, x \mapsto s_0), (s_0, x \mapsto s_1), (s_1, x \mapsto s_0), (s_1, x \mapsto s_1)\} \ .$$

Clearly, $p \wedge \neg x$ holds only at $(s_0, x \mapsto s_1)$. Moreover, the least fixpoint $\mu X.(p \wedge \neg x) \vee \downarrow x.\Diamond X$ evaluates to the set $M := \{(s_0, x \mapsto s_1), (s_1, x \mapsto s_0), (s_1, x \mapsto s_1)\}$. The first element is included because it satisfies $(p \wedge \neg x)$ so every prefixpoint must contain it. The other two elements then also have to be elements of all prefixpoints because for every prefixpoint T with $T \supseteq \{(s_0, x \mapsto s_1)\}$ we have $[\![\downarrow x.\Diamond X]\!]^K_{\rho[X \to T]} \supseteq \{(s_1, x \mapsto s_0), (s_1, x \mapsto s_1)\}$. Finally one can easily check that M a fixpoint.

On the other hand, $M \subseteq [\![(p\wedge\neg x)\vee\downarrow x.\Diamond(\mu X.(p\wedge\neg x)\vee\downarrow x.\Diamond X)$ because the first element of M satisfies $p\wedge\neg x$ and the other two elements satisfy this formula because clearly when placing x at s_1 we get to $(s_1, x \mapsto s_1)$ and then we can make a transition to $(s_0, x \mapsto s_1)$ which is already part of the least fixpoint. Lastly, $(s_0, x \mapsto s_0) \not\models (p\wedge\neg x)\vee\downarrow x.\Diamond(\mu X.(p\wedge\neg x)\vee\downarrow x.\Diamond X)$ because it does not satisfy the first disjunct $p \wedge \neg x$ as seen and placing the x at s_0 still leaves us with $(s_0, x \mapsto s_0)$ from where we can only get back to itself with any transition available, and $(s_0, x \mapsto s_0)$ is not an element of the least fixpoint M as seen. So it does not satisfy the second disjunct either.

The semantics proposed here is indeed compositional, as one can routinely check by induction over the formula structure.

▶ **Proposition 3.** *Let $\varphi(X), \psi \in H_\mu$, K be any Kripke structure and ρ assign values of the variables in φ, ψ w.r.t. K. Let $\varphi[\psi/X]$ denote the formula that is obtained from φ by replacing every free occurrence of X with ψ. We have $[\![\varphi[\psi/X]]\!]^K_\rho = [\![\varphi(X)]\!]^K_{\rho[X \mapsto V]}$ where $V = [\![\psi]\!]^K_\rho$.*

This means in particular, that we can use the fixpoint unfolding principle syntactically in H_μ.

Our analysis will mostly focus on formulas in negation normal form. This is not a restriction as the following Lemma shows.

▶ **Lemma 4.** *For every formula $\varphi \in H^k_\mu$ there is an equivalent formula $\varphi' \in H^k_\mu$ in negation normal form and φ' is only polynomially larger.*

Proof. The proof is fairly standard. We simply push negation inwards with de Morgan's laws, the usual equivalences for fixpoints, like $\mu X.\varphi \equiv \neg\nu X.\neg\varphi[\neg X/X]$ and the following equivalences for hybrid operators: $\neg \downarrow x.\varphi \equiv \downarrow x.\neg\varphi$, $\neg @_x \varphi \equiv @_x \neg\varphi$. ◀

H_μ clearly subsumes hybrid modal logic which is known to be undecidable [4]. Thus, we immediately get the following result concerning H_μ's satisfiability problem.

▶ **Theorem 5.** *Satisfiability for H_μ is undecidable.*

3 Model Checking

In this section we investigate the model checking problem for H_μ. We provide a reduction to L_μ model checking and derive upper complexity bounds for this, prove a matching lower bound for the general case, and finally define model checking games for H_μ based on this reduction and the well-known games for L_μ [17]. They can then be used to aid the understanding of properties expressed by formula of H_μ.

3.1 A Reduction to L_μ Model Checking

Let $\varphi \in H^k_\mu$ for some $k \in \mathbb{N}$ and $K = \langle S, \to, L\rangle$ be a Kripke structure over *Prop*. From these, we construct a Kripke structure $\widehat{K}$ and a formula $\widehat{\varphi}$ of the (multi-modal) μ-calculus over the set of actions $\mathcal{A} = \{\bullet\} \cup \{@_x \mid x \in \mathcal{V}\} \cup \{\downarrow x \mid x \in \mathcal{V}\}$ and atomic propositions from $Prop \cup Nom \cup \mathcal{V}$ as follows.

Let $\varphi \mapsto \widehat{\varphi}$ be the homomorphism such that $\widehat{\Diamond\psi} = \langle\bullet\rangle\widehat{\psi}$, $\widehat{@_x\,\psi} = \langle @_x\rangle\widehat{\psi}$ and $\widehat{\downarrow x.\psi} = \langle\downarrow x\rangle\widehat{\psi}$. Moreover, $\widehat{K} = \langle S \times (\mathcal{V} \to S), \Delta, \widehat{L}\rangle$ where the labeling is defined as $\widehat{L}(p) = \{(s,\sigma) \mid s \in L(p)\}$ for every $p \in Prop \cup Nom$ and $\widehat{L}(x) = \{(s,\sigma) \mid s = \sigma(x)\}$ for every $x \in \mathcal{V}$.

The transition relation Δ is defined as follows.

- $(s,\sigma) \xrightarrow{a} (t,\sigma)$ iff $s \to t$ in K,
- $(s,\sigma) \xrightarrow{@_x} (\sigma(x),\sigma)$ for every $x \in \mathcal{V}$, and
- $(s,\sigma) \xrightarrow{\downarrow x} (s,\sigma[x \mapsto s])$ for every $x \in \mathcal{V}$.

The following can be proved by a straightforward induction over φ.

▶ **Lemma 6.** *For all Kripke structures $K = \langle S, \to, L\rangle$, $s \in S$ and $\sigma : \mathcal{V} \to S$ we have $K, s, \sigma \models \varphi$ iff $\widehat{K}, (s,\sigma) \models \widehat{\varphi}$.*

This realises a reduction from H_μ model checking to L_μ model checking which is polynomial for every fixed k. From this we can derive the following upper complexity bound on the former in the general case.

▶ **Theorem 7.** *The model checking problem for H_μ is in* EXPTIME.

Proof. It is known that L_μ model checking on a Kripke structure K' and a formula ψ can be done in time $\mathcal{O}((|K'| \cdot |\psi|)^{\mathsf{ad}(\psi)})$ [8] where $\mathsf{ad}(\psi)$ denotes the depth of fixpoint alternation in φ. Moreover, $|\psi|$ denotes the size of ψ as measured by the number of its distinct subformulas, and $|K'|$ is the sum of the number of states and edges in K'.

Now take an H^k_μ formula φ and a Kripke structure K and consider $\widehat{K}$ and $\widehat{\varphi}$ as defined above. It is not hard to see that $|\widehat{\varphi}| = \mathcal{O}(|\varphi|)$ and $|\widehat{K}| = \mathcal{O}(|K|^{k+1})$. Hence, Lemma 6 facilitates an exponential reduction to L_μ model checking. Since this is not known to be solvable in polynomial time, the EXPTIME upper bound does not follow directly but requires a slightly more detailed analysis: the reduction produces a Kripke structure $\widehat{K}$ and a formula $\widehat{\varphi}$ such that $\mathsf{ad}(\widehat{\varphi}) = \mathsf{ad}(\varphi)$ and, hence, model checking on these can be performed in time $\mathcal{O}((|K|^{k+1} \cdot |\varphi|)^{\mathsf{ad}(\varphi)})$, i.e. in exponential time. ◀

Clearly, the number of first-order variables is the only source of exponentiation in this reduction. Hence, if this number is fixed, we obtain a better bound.

▶ **Corollary 8.** *For any fixed $k \in \mathbb{N}$ we have that the model checking problem for H^k_μ is at most polynomially worse than that of L_μ.*

This implies membership in NP∩coNP [8], UP∩coUP [10], PLS [20], etc. for model checking each H^k_μ.

$$\frac{s,\sigma \vdash \psi_1 \land \psi_2}{s,\sigma \vdash \psi_1 \quad s,\sigma \vdash \psi_2}\ (1) \qquad \frac{s,\sigma \vdash @_x\, \varphi}{\sigma(x),\sigma \vdash \varphi} \qquad (0)\ \frac{s,\sigma \vdash \psi_1 \lor \psi_2}{s,\sigma \vdash \psi_1 \quad s,\sigma \vdash \psi_2}$$

$$\frac{s,\sigma \vdash \Box\psi}{t,\sigma \vdash \psi}\ (1 : s \to t) \qquad \frac{s,\sigma \vdash \downarrow x.\varphi}{s,\sigma[x \mapsto s] \vdash \varphi} \qquad (0 : s \to t)\ \frac{s,\sigma \vdash \Diamond\psi}{t,\sigma \vdash \psi}$$

$$\frac{s,\sigma \vdash \nu X.\psi(X)}{s,\sigma \vdash \psi(X)}\ (1) \qquad \frac{s,\sigma \vdash X}{s,\sigma \vdash \mathsf{fp}_\varphi(X)} \qquad (0)\ \frac{s,\sigma \vdash \mu X.\psi(X)}{s,\sigma \vdash \psi(X)}$$

Figure 1 The game rules for H_μ model checking.

3.2 Model Checking Games

Next we give a game-theoretic characterisation of H_μ's model checking problem. Such games are particularly useful for reasoning about the (un-)satisfaction of a formula and therefore to understand the properties expressed by H_μ formulas, for instance in the proof of the lower bound in the next section.

▶ **Definition 9.** Let $\varphi \in H_\mu$ be in negation normal form, and $K = \langle S, \to, L\rangle$ be a Kripke structure. The model checking game $\mathcal{G}(K,\varphi)$ is played by 2 players – called 0 and 1. It is Player 0's task to show that the formula holds while Player 1 tries to refute this. The game's positions are $S \times (\mathcal{V} \to S) \times \mathsf{Sub}(\varphi)$. We usually write such a position as $s,\sigma \vdash \psi$.

The game can evolve using the rules in Figure 1. Those that are annotated with player i induce a choice for this player. For example in a configuration $(s,\sigma) \vdash \Diamond\psi$ it is player 0's task to choose a successor t of s in K and then the play continues in the position $t,\sigma \vdash \psi$.

A player wins a play if their opponent is stuck, i.e. cannot perform a prescribed choice anymore. Furthermore, player 0 wins if she can reach a position $s,\sigma \vdash p$ with $s \in L(p)$ for some $p \in \mathit{Prop} \cup \mathit{Nom}$ or $s,\sigma \vdash x$ with $\sigma(x) = s$ for some $x \in \mathcal{V}$. On the other hand, if $s \notin L(p)$ resp. $\sigma(x) \neq s$ player 1 wins. Likewise, player 0 wins in a position $s,\sigma \vdash \neg p$ if $s \notin L(p)$, and a position $s,\sigma \vdash \neg x$ if $\sigma(x) \neq s$.

Finally, let $>_\varphi$ be the smallest relation such that $X >_\varphi Y$ if X has a free occurrence in $\mathsf{fp}_\varphi(Y)$ that is closed under transitivity. The winner of an infinite play is determined by the type of the unique largest (with respect to $>_\varphi$) fixpoint variable that occurs infinitely often. Player 0 wins if its type is ν and player 1 wins if its type is μ.

Next we need to show that these games characterise the model checking problem for H_μ. This is particularly easy with Lemma 6 at hand, which lets us lift the correctness property of the L_μ model checking games – player 0 wins iff the formula holds – to the H_μ games.

Let $K = \langle S, \to, L\rangle$ and $\varphi \in H^k_\mu$ for some k be given, and let $\widehat{\mathcal{G}}(\widehat{K},\widehat{\varphi})$ be the model checking game in the multi-modal L_μ for $\widehat{K}$ and $\widehat{\varphi}$. Remember that $L_\mu = H^0_\mu$ and note that model checking games for H^0_μ can ignore the variable assignment in their positions.

▶ **Lemma 10.** *Player* 0 *wins a position* $s,\sigma \vdash \varphi$ *in* $\mathcal{G}(K,\varphi)$ *if and only if Player* 0 *wins a position* $(s,\sigma) \vdash \widehat{\varphi}$ *in* $\widehat{\mathcal{G}}(\widehat{K},\widehat{\varphi})$.

Proof. "$\Leftarrow$" Suppose player 0 has a winning strategy χ for $\widehat{\mathcal{G}}(\widehat{K},\widehat{\varphi})$. Because of positional determinacy for L_μ model checking games [17] we can assume χ to prescribe a choice to player 0 in each configuration that contains a disjunction or a diamond formula (regardless of the play's history). This strategy can easily be transferred into a positional strategy χ' for player 0 in $\mathcal{G}(K,\varphi)$ via $\chi'((s,\sigma) \vdash \psi) = \chi(s,\sigma \vdash \widehat{\psi})$. Note that states in K are of the form (s,σ) and, as said above, positions in the L_μ, resp. H^0_μ model checking games are pairs of states and subformulas only. Nominally, player 0 has more choices with χ than with χ'

because binder und jump modalities in φ have become diamond modalities in $\widehat{\varphi}$. However, the underlying edge relations in $\widehat{K}$ are deterministic which means that player 0's only choice in such positions is to do what the semantics of binders and jumps require.

It is not hard to see that χ' is winning if χ is because $\widehat{\varphi}$ has essentially the same structure as φ. Hence, if player 0 can use χ to enforce a play in which the outermost fixpoint variable occurring infinitely often is of type ν then so can she using χ'.

"$\Rightarrow$" This can be proved in the same way by transforming a winning strategy in $\mathcal{G}(K,\varphi)$ into one in $\widehat{\mathcal{G}}(\widehat{K},\widehat{\varphi})$ now adding the deterministic choices for player 0 at additional diamond subformulas. ◀

Putting Lemmas 6 and 10 together we obtain correctness of the H_μ model checking games.

▶ **Theorem 11.** *Player 0 has a winning strategy in a position $s,\sigma \vdash \varphi$ in the model checking game $\mathcal{G}(K,\varphi)$ if and only if $K,s,\sigma \models \varphi$.*

3.3 A Lower Bound

It remains to be seen that the exponential time upper bound for H_μ is tight. For this we reduce the n-corridor tiling game problem [6] to the model checking problem of H_μ.

A tiling system is a tuple $\mathcal{T} = \langle T, H, V, t_1\rangle$ consisting of a set of tiles $T = \{t_1,\dots,t_m\}$, a horizontal matching relation $H \subseteq T \times T$, a vertical matching relation $V \subseteq T \times T$ and an initial tile t_1.

Let $n \geq 1$. The n-corridor tiling game is played between two players Adam and Eve on such a $\mathcal{T}$ and the $(\mathbb{N} \times \{0,\dots,n-1\})$-corridor as follows. At the beginning, the initial tile t_1 is being placed at position $(0,0)$. Whenever the first tile of a row has been placed, Eve needs to complete this row with tiles respecting the vertical and horizontal matching relations. Whenever a row is finished, Adam's places a tile onto the first position of the next row such that this tile matches the one below w.r.t. V.

A play is won by a player if their opponent is unable to place a tile without violating the matching relations. Additionally, Eve wins any play that goes on forever.

The n-corridor tiling game problem is the following: given a tiling system $\mathcal{T}$ and an $n \in \mathbb{N}$ in unary encoding, decide whether Eve has a winning strategy for the n-corridor tiling game on $\mathcal{T}$. This problem is known to be ExpTime-hard [6].

▶ **Theorem 12.** *The model checking problem for H_μ is* ExpTime*-hard.*

Proof. Let $\mathcal{T} = \langle T, H, V, t_1\rangle$ with $T = \{t_1,\dots,t_m\}$ and $n \in \mathbb{N}$ be given. To help with notation define $H_{t_i} := \{t \mid (t_i,t) \in H\}$ as the possible horizontal successors of t_i and $V_{t_i} := \{t \mid (t_i,t) \in V\}$ as the possible vertical successors of t_i for each $i \in \{1,\dots,m\}$.

We build a Kripke structure $K^n_{\mathcal{T}}$ over $\mathit{Prop} = T$ with a designated state s_0 and an H^n_μ formula $\varphi^n_{\mathcal{T}}$ such that $K^n_{\mathcal{T}}, s_0, \sigma \models \varphi_{\mathcal{T}}$ for any σ if and only if Eve wins the n-corridor tiling game on $\mathcal{T}$.

Intuitively, a path through $K^n_{\mathcal{T}}$ corresponds to a particular play in the n-corridor tiling game on $\mathcal{T}$. Each state is labeled with exactly one atomic proposition from T representing the tile placed at a particular position in the n-corridor. It is encoded row-wise as an infinite path, i.e. the i-th state on this path represents the position $(\lfloor i \div n \rfloor, i \mod n)$ in the n-corridor. It is possible to let $K^n_{\mathcal{T}}$ consist of the full clique of m states only – one for each tile. However, it is more convenient to encode the horizontal matching relation into the structure such that an edge from (a state labeled) t in some column to t' in the next column only exists if $(t,t') \in H$ and both represent positions in a common row in the n-corridor.

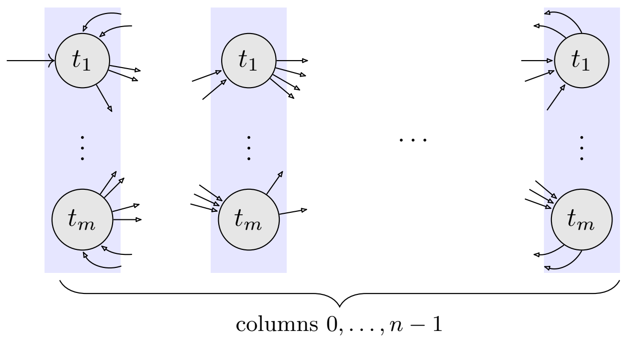


Figure 2 The Kripke structure $K^n_{\mathcal{T}}$ used in the reduction from the n-corridor tiling game problem.

$K^n_{\mathcal{T}}$ is depicted in Figure 2. The initial state is the one labeled with the initial tile t_1 in column 0. There is an edge from a state (identified by its unique label) t in column i to state t' in column $i+1$ iff $(t,t') \in H$ and $i < n-1$. Additionally, there are edges from every state in column $n-1$ to every state in column 0.

The formula $\varphi^n_{\mathcal{T}}$ then needs to describe the evolution of the n-corridor tiling game. The fact that it potentially goes on forever is modeled by a greatest fixpoint recursion. Each iteration corresponds to the construction of a row. For technical convenience, since the initial tile in the n-corridor tiling game is fixed, it actually corresponds to the construction of the $n-1$ last tiles of a row plus the first tile of the next row. In order to check whether the players only choose tiles that match vertically, we use n first-order variables $x_0, \dots, x_{n-1}$ which are placed during the construction of a row, and then can be used to remember those tiles for the construction of the next row.

As a shorthand we use the formula

$$vm_i(Z) := \big(\bigvee_{(t,t')\in V} t' \wedge @_{x_i} t\big) \wedge {\downarrow} x_i.Z$$

for $i = 0, \dots, n-1$ which compares the tile at the current position with the tile at the position stored in x_i, and additionally binds the variable x_i to the state that it is currently evaluated in. Then let

$$\varphi^n_{\mathcal{T}} := {\downarrow} x_0.\Diamond{\downarrow} x_1.\Diamond{\downarrow} x_2. \dots \Diamond{\downarrow} x_{n-1}.\Box\big(\nu Y.\neg vm_0(\neg\Diamond vm_1(\Diamond vm_2(\dots\Diamond vm_{n-1}(\Box Y)\dots)))\big).$$

Using the previously introduced model checking games for H_μ one can check that a winning strategy for Eve in the n-corridor tiling game induces a winning strategy for player 0 in the model checking game on $K^n_{\mathcal{T}}$ and $\varphi^n_{\mathcal{T}}$: her choices of tiles in the tiling game correspond directly to choices she can do at $\Diamond$-formulas. With Theorem 11 we then get correctness of this reduction. Moreover, it is easy to see that both $K^n_{\mathcal{T}}$ and $\varphi^n_{\mathcal{T}}$ can be constructed in time polynomial in $|\mathcal{T}|$ and the value of n. ◀

4 Expressiveness

This section studies the expressive power of H_μ with a particular focus on principle bounds imposed in the sense of bisimulation invariance. It is well known that L_μ is bisimulation-invariant, i.e. formulas of L_μ cannot distinguish bisimilar models, but that hybrid operators break this invariance. For example with just one variable one can distinguish a single self-loop from its tree-unraveling using the formula $x \wedge \Diamond x$.

4.1 A Game-Theoretic Characterisation of k-Bisimulation

In [3] it is shown that hybrid modal logic is invariant under a refined form of bisimulation, called k-bisimulation; its formal definition is recalled below. To better suit our framework of games we present the definition in terms of k-bisimulation games extending the well known (ordinary) bisimulation games [18].

▶ **Definition 13** (k-Bisimulation Game). Given two Kripke structures $K_0 = \langle S_0, \rightarrow_0, L_0 \rangle$ and $K_1 = \langle S_1, \rightarrow_1, L_1 \rangle$ over a set of atomic propositions *Prop* and a set of nominals *Nom*, the k-bisimulation game $\mathcal{G}^k(K_0, K_1)$ is played between two players – Spoiler and Duplicator – on the configuration space $S_1^{k+1} \times S_2^{k+1}$.

We can imagine that on each structure we have one *active* pebble that gets moved across the structure and k *inactive* pebbles that just mark certain states as well as some *fixed* pebbles that mark the positions of the nominals.

The game is strictly turn-based. First, in a configuration $(s, s_1, \dots, s_k, t, t_1, \dots, t_k)$ Spoiler starts by choosing one of the structures K_i for some $i \in \{0, 1\}$ and then chooses to either

- take a transition $s \rightarrow_0 s'$, resulting in a configuration $(s', s_1, \dots, s_k, t, t_1, \dots, t_k)$, or
- move pebble i from s_i to the current state s, resulting in a configuration $(s, s_1, \dots, s_{i-1}, s, s_{i+1}, \dots, s_k, t, t_1, \dots, t_k)$, or
- jump from the current state s to some pebble s_i, resulting in a configuration $(s_i, s_1, \dots, s_k, t, t_1, \dots, t_k)$ or to some nominal n resulting in a configuration $(n, s_1, \dots, s_k, t, t_1, \dots, t_k)$.

After that Duplicator makes the same kind of move on the other structure K_{1-i}.

Spoiler wins the game if after Duplicator's move the game is in a configuration $(s, s_1, \dots, s_k, t, t_1, \dots, t_k)$ such that the atomic propositions or nominals on s and t do not match or for some $i = 1, \dots, k$, $s = s_i$ but not $t = t_i$ or vice versa. On the other hand Duplicator wins the game if he can always successfully mimic Spoiler's move which means that on s and t the atomic propositions and nominals match and $s = s_i$ if and only if $t = t_i$ for all $i = 1, \dots, k$ and thus the game goes on forever.

We say that $(s, s_1, \dots, s_k) \sim^k (t, t_1, \dots, t_k)$ if Duplicator wins $\mathcal{G}^k(K_0, K_1)$ from the configuration $(s, s_1, \dots, s_k, t, t_1, \dots, t_k)$. We say that $s \sim^k t$ if Duplicator wins $\mathcal{G}^k(K_0, K_1)$ from configuration $(s, s, \dots, s, t, t, \dots, t)$.

It is easy to see that for $k = 0$ and no nominals we get the well-known bisimulation games. Furthermore, as also remarked in [3], these games can be restricted to the hybrid operators in use. For example, the restricted games with $k = 0$ variables and only nominals characterise the expressiveness of the hybrid μ-calculus with only nominals and jumps investigated in [16].

The following lemma states some easy observations that will be of use later on.

▶ **Lemma 14.** *Let K_0, K_1 be as in Definition 13. If $(s, s_1, \dots, s_k) \sim^k (t, t_1, \dots, t_k)$, then*

(a) *for every $s \rightarrow_0 s'$ there is a transition $t \rightarrow_1 t'$ such that $(s', s_1, \dots, s_k) \sim^k (t', t_1, \dots, t_k)$,*
(b) *$(s, s_1, \dots, s_{i-1}, s, s_{i+1}, \dots, s_k) \sim^k (t, t_1, \dots, t_{i-1}, t, t_{i+1}, \dots, t_k)$ for every $i = 1, \dots, k$, and*
(c) *$(s_i, s_1, \dots, s_k) \sim^k (t_i, t_1, \dots, t_k)$ for every $i = 1, \dots, k$.*

4.2 The Hybrid μ-Calculus and k-Bisimulation

We will now show that the expressive power of H_μ^k is limited by k-bisimulations.

▶ **Theorem 15.** *Let $\varphi \in H_\mu^k$ be closed and $K_0 = \langle S_0, \to_0, L_0\rangle, K_1 = \langle S_1, \to_1, L_1\rangle$ be two Kripke structures. If $(s, s_1, \ldots, s_k) \in S_0^{k+1}$ and $(t, t_1, \ldots, t_k) \in S_1^{k+1}$ such that $(s, s_1, \ldots, s_k) \sim^k (t, t_1, \ldots, t_k)$, then $K_0, s, (s_1, \ldots, s_k) \models \varphi$ if and only if $K_1, t, (t_1, \ldots, t_k) \models \varphi$.*

Proof. To prove this by induction we have to strengthen the hypothesis in order to account for formulas $\varphi(X_1, \ldots, X_m)$ with free second-order variables $X_1, \ldots, X_m$. Let ρ and ρ' be interpretations for these that respect k-bisimilarity in the sense that for all $(s, s_1, \ldots, s_k) \in S_0^{k+1}$ and $(t, t_1, \ldots, t_k) \in S_1^{k+1}$ it holds that if $(s, s_1, \ldots, s_k) \sim^k (t, t_1, \ldots, t_k)$ then $(s, s_1, \ldots, s_k) \in \rho(X_i)$ if and only if $(t, t_1, \ldots, t_k) \in \rho'(X_i)$ for any i.

We will prove the following by induction on the structure of φ: $(s, s_1, \ldots, s_k) \sim^k (t, t_1, \ldots, t_k)$ implies that $K_0, s, (s_1, \ldots, s_k), \rho \models \varphi$ if and only if $K_1, t, (t_1, \ldots, t_k), \rho' \models \varphi$. For technical convenience, the variable assignments have been split into the parts interpreting the first- resp. second-order variables.

So, assume that $(s, s_1, \ldots, s_k) \sim^k (t, t_1, \ldots, t_k)$ and ρ, ρ' are as stated above. For the base case let $\varphi = p$ for some $p \in \mathit{Prop}$. Then from $(s, s_1, \ldots, s_k) \sim^k (t, t_1, \ldots, t_k)$ we immediately get that $s \in L_0(p) \Leftrightarrow t \in L_1(p)$. The case $\varphi = x$ is similar. For the case $\varphi = X_i$ we can use the assumption for free second-order variables as stated above.

The cases for Boolean operators follow immediately from the hypothesis and the case $\varphi = \Diamond\psi$ follows immediately with Lemma 14 a).

For the hybrid operators suppose that $\varphi = \downarrow x_i.\psi$ for some $i \in \{1, \ldots, k\}$. We get that

$$\begin{aligned} K_0, s, (s_1, \ldots, s_k), \rho \models \varphi \quad &\Leftrightarrow \quad K_0, s, (s_1, \ldots, s_{i-1}, s, s_{i+1}, \ldots, s_k), \rho \models \psi \\ &\Leftrightarrow \quad K_1, t, (t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_k), \rho' \models \psi \\ &\Leftrightarrow \quad K_1, t, (t_1, \ldots, t_k), \rho' \models \varphi \end{aligned}$$

where the first and last equivalence are simply the semantics of H_μ^k and the second equivalence is Lemma 14 b) and the induction hypothesis. The case for $\varphi = @_{x_i}\psi$ then follows with Lemma 14 c).

For the last case suppose that $\varphi = \mu X.\psi(X, X_1, \ldots, X_m)$ with free variables in ψ as depicted.

Let $\psi^0 := \psi[\mathtt{ff}/X]$ and $\psi^{\alpha+1} := \psi[\psi^\alpha/X]$. With Proposition 3 and the characterisation of least fixpoints via approximations we have that $[\![\mu X.\psi(X)]\!]_\varrho^K = \bigcup_{\alpha<\omega}[\![\psi^\alpha]\!]_\varrho^K$ for some Kripke structure K and assignment ϱ.

We show by a separate induction over α that $K_0, s, (s_1, \ldots, s_k), \rho \models \psi^\alpha$ if and only if $K_1, t, (t_1, \ldots, t_k), \rho' \models \psi^\alpha$. The case for least fixpoints then follows immediately.

For the base case $\alpha = 0$ observe that $[\![\psi[\mathtt{ff}/X]]\!]_\varrho^K = [\![\psi(X)]\!]_{\varrho[X\mapsto\emptyset]}^K$. Thus,

$$\begin{aligned} K_0, s, (s_1, \ldots, s_k), \rho \models \psi^0 \quad &\Leftrightarrow K_0, s, (s_1, \ldots, s_k), \rho[X \mapsto \emptyset] \models \psi(X) \\ &\Leftrightarrow K_1, t, (t_1, \ldots, t_k), \rho'[X \mapsto \emptyset] \models \psi(X) \\ &\Leftrightarrow K_1, t, (t_1, \ldots, t_k), \rho' \models \psi^0 \end{aligned}$$

where the second equivalence is by the fact that for all $(s, s_1, \ldots, s_k) \sim^k (t, t_1, \ldots, t_k)$ it holds that $(s, s_1, \ldots, s_k) \in \emptyset \Leftrightarrow (t, t_1, \ldots, t_k) \in \emptyset$ so we can use the induction hypothesis of the outer induction for ψ. For the induction step we then get

$$\begin{aligned} K_0, s, (s_1, \ldots, s_k), \rho \models \psi^{\alpha+1} \quad &\Leftrightarrow K_0, s, (s_1, \ldots, s_k), \rho[X \mapsto [\![\psi^\alpha]\!]_\rho^{K_0}] \models \psi(X) \\ &\Leftrightarrow K_1, t, (t_1, \ldots, t_k), \rho'[X \mapsto [\![\psi^\alpha]\!]_{\rho'}^{K_1}] \models \psi(X) \\ &\Leftrightarrow K_1, t, (t_1, \ldots, t_k), \rho' \models \psi^{\alpha+1}. \end{aligned}$$

Here, the induction hypothesis for ψ^α makes sure that the conditions for the induction hypothesis for ψ are met which is why the second equivalence holds. This finishes both inductions. ◀

▶ **Corollary 16.** *Let $\varphi \in H_\mu^k$ be a sentence and $K_0 = \langle S_0, \to_0, L_0\rangle, K_1 = \langle S_1, \to_1, L_1\rangle$ be two Kripke structures. If $s \in S_0$ and $t \in S_1$ such that $s \sim^k t$, then $K_0, s \models \varphi$ if and only if $K_1, t \models \varphi$.*

There is another interesting connection between H_μ and bisimulations: H_μ^2 can *express* bisimilarity in the sense that there is a fixed formula $\varphi_\sim$ (relative to a fixed *Prop*) which is true in a Kripke structure with a valuation of two variables if and only if these two variables point at bisimilar states. For technical convenience we assume $\mathit{Nom} = \emptyset$ here, since the usual and simple reduction from bisimilarity between two Kripke structures to bisimilarity within a single structure does not work in the presence of nominals. Instead, one has to rename them uniquely in one of them to allow the disjoint union of two Kripke structures with nominals to be seen as one Kripke structure with nominals again.

▶ **Example 17.** The formula

$$\varphi_\sim := \nu X.\Big(\big(\bigwedge_{p\in \mathit{Prop}} @_x\, p \leftrightarrow @_y\, p\big) \wedge (@_x \Box {\downarrow} x.\, @_y \Diamond {\downarrow} y.X) \wedge (@_y \Box {\downarrow} y.\, @_x \Diamond {\downarrow} x.X)\Big)$$

states that x and y are bisimilar. This can be seen using the model checking games of Section 3.2 for instance.

It is even possible to express k-bisimilarity; however this requires $2k+2$ variables.

▶ **Example 18.** The formula

$$\begin{aligned}\varphi_\sim^k := \nu X.\Big(& \bigwedge_{p\in \mathit{Prop}} (@_x\, p \leftrightarrow @_y\, p) \ \wedge\ (@_x \Box {\downarrow} x.\, @_y \Diamond {\downarrow} y.X) \ \wedge\ (@_y \Box {\downarrow} y.\, @_x \Diamond {\downarrow} x.X) \\ & \bigwedge_{i=1}^{k} \big((@_x\, x_i \leftrightarrow @_y\, y_i) \ \wedge\ (@_x {\downarrow} x_i.\, @_y {\downarrow} y_i.X) \ \wedge\ (@_{x_i} {\downarrow} x.\, @_{y_i} {\downarrow} y.X)\big)\Big)\end{aligned}$$

over the variables $\{x, x_1, \dots, x_k, y, y_1, \dots, y_k\}$ ordered in this way is true on $(s, s_1, \dots, s_k, t, t_1, \dots, t_k)$ in K if and only if $(s, s_1, \dots, s_k) \sim^k (t, t_1, \dots, t_k)$.

▶ **Theorem 19.** *H_μ^{2k+2} can express k-bisimilarity for any $k \geq 0$.*

4.3 Comparing H_μ with Hybrid Branching Time Logics

A natural question that arises when studying the expressiveness of H_μ is its relationship to other hybrid logics. It is easy to see that H_μ is more expressive than hybrid modal logic using the same example which shows that L_μ is more expressive than modal logic. Candidates for an interesting comparison are hybrid extensions of branching time logics. There are three hybrid extensions of CTL* [12] defined by constraints on how hybrid operators can be used on path formulas which are not state formulas, giving rise to the syntactical hierarchy HCTL$^*_{\mathsf{ss}}$ $\leq$ HCTL$^*_{\mathsf{ps}}$ $\leq$ HCTL$^*_{\mathsf{pp}}$. The smallest already subsumes the previously studied hybrid extensions of CTL and CTL$^+$ [11]. We will show that HCTL$^*_{\mathsf{ps}}$ already is not subsumed by H_μ. This is somewhat surprising given that L_μ is known to subsume CTL* [7].

Let $C_k = \langle S_k, \to, L\rangle$ be the complete clique over k states and $C_\infty = \langle S_\infty, \to, L\rangle$ be the complete clique over $\mathbb{N}$, as depicted for $k = 8$ in Figure 3, as Kripke structures over $\mathit{Prop} = \mathit{Nom} = \emptyset$.

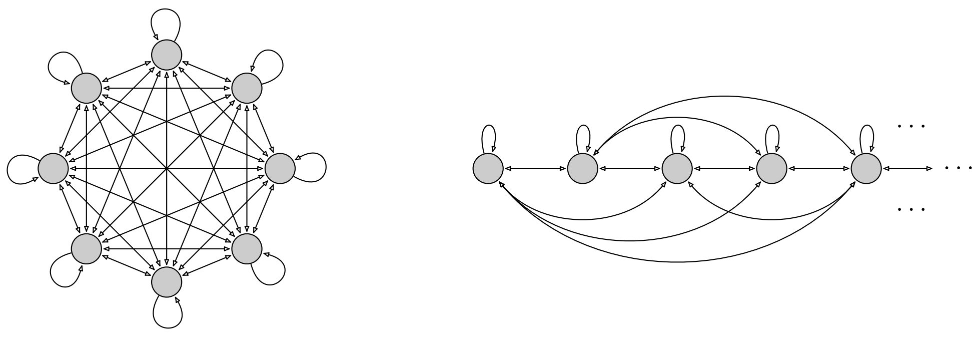


Figure 3 C_8 and C_∞.

▶ **Lemma 20.** *Let $(s, s_1, \dots, s_k, t, t_1, \dots, t_k)$ be a configuration in the k-bisimulation game $\mathcal{G}^k(C_{k+1}, C_\infty)$ such that $s = s_i$ if and only if $t = t_i$ for all $i = 1, \dots, k$. Then Duplicator has a winning strategy for this game starting in this configuration.*

Proof. We call a configuration $(s, s_1, \dots, s_k, t, t_1, \dots, t_k)$ *consistent* if for all i we have $s = s_i$ if and only if $t = t_i$. By assumption, the game starts in a consistent configuration. There are two observations to be made:

1. Regardless of Spoiler's choices in a consistent configuration, Duplicator can always answer such that the next configuration is also consistent. This is particularly easy for moves in C_∞, and in C_{k+1} it is possible because every state is reachable from every other in one step, and there is always at least one state which is not inhabited by an inactive pebble.
2. Spoiler wins the k-bisimulation game only when an inconsistent configuration has been reached.

Hence, the simple strategy of preserving consistency is a winning strategy for Duplicator in $\mathcal{G}^k(C_{k+1}, C_\infty)$. ◀

Duplicator especially wins the game starting in configurations of the form $(s, s, \dots, s, t, t, \dots, t)$, since they are obviously consistent, which means that $s \sim^k t$ for any $s \in C_{k+1}$ and $t \in C_\infty$.

▶ **Theorem 21.** *There is a formula $\varphi \in HCTL^*_{\mathsf{ps}}$ that cannot be expressed by any formula in H_μ.*

Proof. The formula $\varphi := \mathsf{EG}({\downarrow} x.\mathsf{XG}\neg x) \in \mathrm{HCTL}^*_{\mathsf{ps}}$ states that there is an infinite path such that no state on this path is seen twice. Clearly, we have $C_\infty, t \models \varphi$ for any state t and $C_k, s \not\models \varphi$ for any state s and any $k \in \mathbb{N}$. Now suppose there was a formula $\psi \in H_\mu$ expressing this property. Then we would have $\psi \in H^k_\mu$ for some k. By equivalence we would get $C_\infty, t \models \psi$ and $C_{k+1}, s \not\models \psi$ for any states s, t. On the other hand, according to Lemma 20 we have $C_{k+1}, s \sim^k C_\infty, t$, and by Theorem 15, no H_μ formula – in particular not ψ – can distinguish these two structures. Hence, no such ψ can exist. ◀

5 Conclusion and Further Work

We have introduced a hybrid extension of the μ-calculus with nominals, binders and jumps and have shown that the model checking problem for this logic is ExpTime-complete for the general case and only polynomially worse than model checking the modal μ-calculus for a fixed number of variables. We have investigated the expressiveness of the fully hybrid μ-calculus and have shown that it is invariant under hybrid k-bisimulations introduced in [3] when restricted to k variables. We used this result to show that – contrary to the pure modal case – the hybrid extension of the full branching time logic CTL* is not a fragment of H_μ.

Future work will investigate the relationship between hybrid branching time logics and the μ-calculus in more detail. We have shown here that the second level of the syntactic hierarchy of hybrid branching time logics introduced in [12] cannot be translated to the fully hybrid μ-calculus. We believe that the variant which allows binders and jumps over state formulas only could be a fragment of H_μ and will investigate this further.

Another interesting connection that needs to be explored is that between H_μ and the polyadic μ-calculus [2, 15], another extension of L_μ which can express bisimilarity in the sense of Example 17. Its formulas are interpreted in tuples of states of fixed arity, rather than single states as in the case of L_μ. This is reminiscent of the mechanisms in H_μ, especially under the semantics developed here, where a formula with k first-order variables is essentially interpreted by a $(k+1)$-tuple of states. We believe that the polyadic μ-calculus can be embedded into H_μ. The opposite direction cannot hold because the polyadic μ-calculus is known to be bisimulation-invariant. However, this obviously breaks when it is equipped with an equality predicate [14], and we believe that then it becomes strong enough to embed H_μ.

References

1 R. Alur and T. A. Henzinger. A really temporal logic. *J. of the ACM*, 41(1):181–204, 1994.

2 H. R. Andersen. A polyadic modal μ-calculus. Technical Report ID-TR: 1994-195, Dept. of Computer Science, Technical University of Denmark, Copenhagen, 1994.

3 C. Areces, P. Blackburn, and M. Marx. Hybrid logics: Characterization, interpolation and complexity. *J. of Symb. Log.*, 66:2001, 1999.

4 C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logics*, pages 821–868. Elsevier, 2006.

5 J. van Benthem. Correspondence theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic*, pages 167–247. D. Reidel, 1984.

6 B. S. Chlebus. Domino-tiling games. *J. of Comp. and Sys. Sc.*, 32:374–392, 1986.

7 M. Dam. CTL* and ECTL* as fragments of the modal μ-calculus. *TCS*, 126(1):77–96, 1994.

8 E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In *Proc. 5th Conf. on Computer Aided Verification, CAV'93*, volume 697 of *LNCS*, pages 385–396. Springer, 1993.

9 M. Franceschet and M. de Rijke. Model checking hybrid logics (with an application to semistructured data). *J. of Applied Logic*, 4(3):279–304, 2006.

10 M. Jurdziński. Deciding the winner in parity games is in $UP \cap$co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998.

11 A. Kara, V. Weber, M. Lange, and T. Schwentick. On the hybrid extension of CTL and CTL$^+$. In *Proc. 34th Int. Symp. on Mathematical Foundations of Computer Science, MFCS'09*, volume 5734 of *LNCS*, pages 427–438. Springer, 2009.

12 D. Kernberger and M. Lange. Model checking for the full hybrid computation tree logic. In *Proc. 23rd Int. Symp. on Temporal Representation and Reasoning, TIME'16*, pages 31–40. IEEE Computer Society, 2016.

13 D. Kozen. Results on the propositional μ-calculus. *TCS*, 27:333–354, 1983.

14 M. Lange and E. Lozes. Model checking the higher-dimensional modal μ-calculus. In *Proc. 8th Workshop on Fixpoints in Computer Science, FICS'12*, volume 77 of *Electr. Proc. in Theor. Comp. Sc.*, pages 39–46, 2012.

15 M. Otto. Bisimulation-invariant PTIME and higher-dimensional μ-calculus. *Theor. Comput. Sci.*, 224(1–2):237–265, 1999.

16 U. Sattler and M. Y. Vardi. The hybrid μ-calculus. In *Proc. 1st Int. Joint Conf. on Automated Reasoning, IJCAR'01*, volume 2083 of *LNCS*, pages 76–91. Springer, 2001.

17 C. Stirling. Local model checking games. In *Proc. 6th Conf. on Concurrency Theory, CONCUR'95*, volume 962 of *LNCS*, pages 1–11. Springer, 1995.

18 C. Stirling. Bisimulation, modal logic and model checking games. *Logic J. of the IGPL*, 7(1):103–124, 1999.

19 B. ten Cate. *Model theory for extended modal languages.* PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands, 2005.

20 J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *Proc. 12th Int. Conf. on Computer Aided Verification, CAV'00*, volume 1855 of *LNCS*, pages 202–215. Springer, 2000.

Similarity Search for Spatial Trajectories Using Online Lower Bounding DTW and Presorting Strategies

Marie Kiermeier[1] and Martin Werner[2]

1 Mobile and Distributed Systems Group, Ludwig-Maximilians-Universität, Munich, Germany
marie.kiermeier@ifi.lmu.de

2 Institute of Cartography and Geoinformatics, Leibniz-University, Hanover, Germany
martin.werner@ikg.uni-hannover.de

Abstract

Similarity search with respect to time series has received much attention from research and industry in the last decade. Dynamic time warping is one of the most widely used distance measures in this context. This is due to the simplicity of its definition and the surprising quality of dynamic time warping for time series classification. However, dynamic time warping is not well-behaving with respect to many dimensionality reduction techniques as it does not fulfill the triangle inequality. Additionally, most research on dynamic time warping has been performed with one-dimensional time series or in multivariate cases of varying dimensions. With this paper, we propose three extensions to $\mathrm{LB}_{\mathrm{Rotation}}$ for two-dimensional time series (trajectories). We simplify $\mathrm{LB}_{\mathrm{Rotation}}$ and adapt it to the online and data streaming case and show how to tune the pruning ratio in similarity search by using presorting strategies based on simple summaries of trajectories. Finally, we provide a thorough evaluation of these aspects on a large variety of datasets of spatial trajectories.

1998 ACM Subject Classification H.2.8 Database Applications

Keywords and phrases Trajectory Computing, Similarity Search, Dynamic Time Warping, Lower Bounds, k Nearest Neighbor Search, Spatial Presorting

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.18

1 Introduction

Dynamic time warping (DTW) is a family of algorithms designed to compare time series with each other. Though dynamic time warping lacks several important properties such as the triangle inequality, it has seen a wide adoption in many fields. This might be due to the very intuitive definition, relatively efficient calculations, and, of course, successful applications. In the domain of time series classification, each sequence in a set of time series has an associated class label and the task is to predict class labels on unknown instances given a labeled training dataset. In this setting, one nearest neighbor using dynamic time warping with warping constraints calibrated from cross-validation (DTWCV) on the training set is surprisingly hard to beat [7]. A recent paper of Bagnall et al. confirms this observation [2]. They evaluate on a large set of datasets and are able to beat DTWCV with one nearest neighbor, but only with a large ensemble of data transformations. This paper impressively confirms the importance, effectiveness, and efficiency of dynamic time warping similarity search in time series classification.

Nowadays, large datasets of time series are available to evaluate and discuss results on. However, one-dimensional time series predominate these datasets. With this paper, however, we want to concentrate on the special area of spatial time series, often called trajectories. In this area, trajectories are considered as time series in a two-dimensional space. The two-dimensional space has important characteristics allowing for a novel type of lower bound, $\text{LB}_{\text{Rotation}}$, and are very important for practical applications such as tracking, navigation, and location-based services. With this paper, we evaluate and improve the lower bound $\text{LB}_{\text{Rotation}}$ proposed by Gong et. al [10] and extend it with efficient presorting strategies for scalable nearest neighbor search.

More concretely, the main contributions of this paper are as follows:

- We show that $\text{LB}_{\text{Rotation}}$ can be calculated more efficiently without least square fitting and without loss of tightness.
- We show that $\text{LB}_{\text{Rotation}}$ can be calculated in a data stream setting.
- We define three presorting strategies leading to considerable increase in pruning power.
- We provide a thorough evaluation of $\text{LB}_{\text{Rotation}}$ variants and presorting strategies on datasets of very different characteristics.

The remainder of the paper is structured as follows: Section 2 reviews related work on dynamic time warping including speedup techniques. In Section 3, we explain how to adapt the lower bound $\text{LB}_{\text{Rotation}}$ for an online algorithm. Additionally, we introduce presorting strategies in Section 3.3.

In Section 4, we provide a detailed analysis of four variants of $\text{LB}_{\text{Rotation}}$ on several datasets and discuss the impact and caveats with presorting strategies for similarity search. Finally, Section 5 concludes the paper with some hints on possible future work.

2 Related Work

Dynamic time warping is a distance definition which has found wide adoption in various domains. Historically, dynamic time warping originates in the area of speech recognition [23], but has soon been used in many other domains including geometric recognition tasks such as handwriting recognition and signature verification [6], in computer vision [1], in shape retrieval [18], in biology and medicine [14], pattern recognition [4], and recently similarity search for spatial trajectories [24].

The simplest way of calculating dynamic time warping by an algorithm is given by dynamic programming [29]:

$$\delta_{DTW}(a_{1..n}, b_{1..m}) = \begin{cases} 0, \text{ if } a \text{ and } b \text{ are both empty,} \\ \infty, \text{ if only one of } a \text{ and } b \text{ is empty,} \\ \delta(a_n, b_m) + \min \begin{cases} \delta_{DTW}(a_{1..n-1}, b_{1..m-1}) \\ \delta_{DTW}(a_{1..n-1}, b_{1..m}) \\ \delta_{DTW}(a_{1..n}, b_{1..m-1}) \end{cases} \end{cases} \tag{1}$$

This distance δ_{DTW} can be computed in $\mathcal{O}(mn)$ time [4] where m and n are the lengths of the trajectories.

One way to achieve this is by first evaluating a complete distance matrix

$$D_{i,j}(a_{1..n}, b_{1..m}) = (\delta(a_i, b_j))_{i=1..n, j=1..m}$$

between all points of the trajectories. Then, the problem of calculating dynamic time warping is reduced to finding the shortest path through this matrix from the lower left corner to the upper right corner using only movements to the right, up or diagonally up and summing up all the entries in the distance matrix. The sequence of coordinates in this distance matrix gives the warping path.

Due to its quadratic complexity in time and space for equal-length trajectories this measure is only applicable to short time series. There are two different approaches to speeding up dynamic time warping calculation [22]: either one creates additional constraints on the warping path that allow to only reduce the number of evaluated cells in the distance matrix [21, 11] or one carefully reduces the size of the input trajectories beforehand. However, neither of these approaches will be strictly correct as some warping paths cannot be found in both approaches.

2.1 Lower Bounds

For speeding up similarity search performance under DTW, lower bounds can be used for pruning candidate trajectories. A lower bound in this context is a function taking two sequences and calculating a lower bound to their true DTW distance, that is $\mathrm{LB}(Q, R) \leq \delta_{DTW}(Q, R)$.

The most popular lower bounding measures for dynamic time warping are $\mathrm{LB}_{\mathrm{Kim}}$ [13], $\mathrm{LB}_{\mathrm{Yi}}$ [29], $\mathrm{LB}_{\mathrm{Keogh}}$ [12], and $\mathrm{LB}_{\mathrm{Improved}}$ [15].

Even if there exist multidimensional versions of them like $\mathrm{LB}_{\mathrm{MV}}$ [20], they have actually been designed for one dimensional time series. In contrast, Gong et al. propose $\mathrm{LB}_{\mathrm{Rotation}}$ as a first approach which also takes into account special characteristics of trajectories, i.e. 2-dimensional time series [10].

The idea of $\mathrm{LB}_{\mathrm{Rotation}}$ is to first divide the query trajectory $Q = q_1, q_2, ..., q_n$ in consecutive and non-overlapping segments s_i which are as straight as possible. Then, each segment is rotated such that it is parallel to the X-axis. By doing so, the area of the bounding envelopes which are defined as $U_i = \max(q_{i-c} \dots q_{i+c})$ and $L_i = \min(q_{i-c} \dots q_{i+c})$, where c is the global warping constraint, is reduced. With these small envelopes, we get a more accurate, i.e. greater, lower bound, since the second trajectory $R = r_1, r_2, ..., r_n$ pokes out of the envelopes more often. Note that the rotation of the points in this trajectory is defined from the rotation of the matching point candidates in the query. If the candidates hit several segments, we are free to use the minimum of the resulting numbers.

Then, similar to the construction of $\mathrm{LB}_{\mathrm{Keogh}}$, the lower bound $\mathrm{LB}_{\mathrm{Rotation}}$ is defined to be the sum of the distances d to the segments' envelopes:

$$\mathrm{LB}_{\mathrm{Rotation}}(Q, R) = \sqrt{\sum_{i=1}^{n} \min_{s_j \in S_i} d(r_i, \mathrm{Env}(s_j))}, \tag{2}$$

with S_i is the set of segments of Q which are hit by the candidates and $\mathrm{Env}(s_j) = (U_1, U_2, ..., U_m, L_1, L_2, ..., L_m)$ the corresponding envelopes.

Figure 1 illustrates this situation: a sinus wave is considered as a two-dimensional spatial trajectory (i.e., time is moving forward along the trajectory and not along the X-axis). First, this trajectory is split into linear segments (lower left of Figure 1), then these linear segments are rotated to become parallel to the X-axis resulting in quite small envelopes (upper right in Figure 1). Finally, this trajectory is compared to the graph of a cosinus wave. Note that the points of this graph have to be rotated with the rotations prescribed from the query rotations. Down right in Figure 1, one sees that the intersection of the query and the rotated example is quite small leading to efficient pruning. For reference, Figure 2 depicts the same situation for the multivariate version of $\mathrm{LB}_{\mathrm{Keogh}}$.

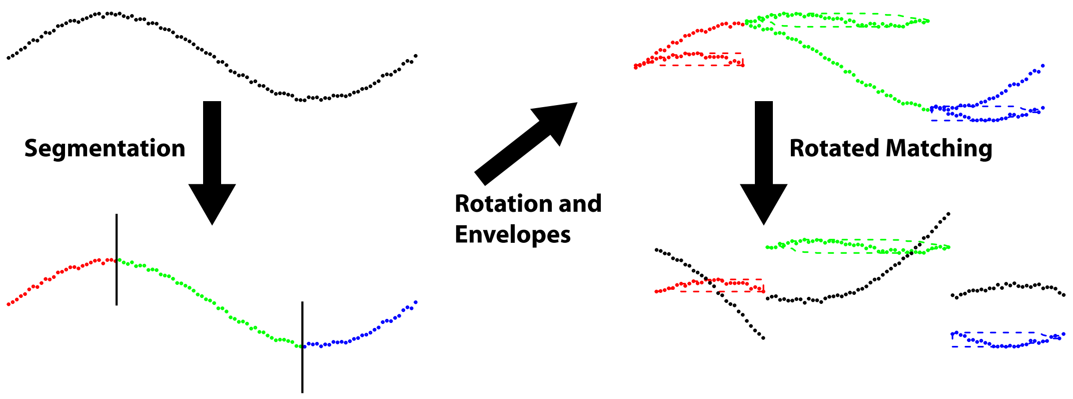


Figure 1 Principles of LB_{Rotation} illustrated with a sinus graph.

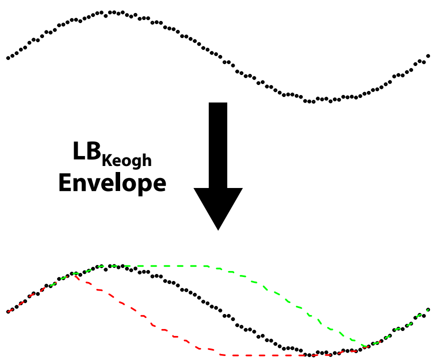


Figure 2 Envelope of LB_{Keogh} for a sinus graph.

Note, however, that the version of LB_{Rotation} as proposed by Gong et al. is an offline algorithm and cannot be applied in streaming settings. It is based on using the Douglas-Peucker algorithm [8] for segmenting the query trajectory.

3 Online Similarity Search

In this section, we first present our efficient online solution for LB_{Rotation} and show, how it can be used for k-nearest neighbor search. Then, four presorting strategies based on extremely simple trajectory summaries for increasing the pruning power when searching trajectory databases are explained in detail.

3.1 Improvements for Lower Bounds Based on Segment Rotation

Facing the challenge of *online* similarity search, the current version of LB_{Rotation} can not be used as a lower bound for DTW, since it is not applicable to data streams. Therefore, in the following, we propose an online version of LB_{Rotation} which can be applied for speeding up online similarity search. Since we do not have access to the complete trajectory when working with data streams, the segmentation step of LB_{Rotation} has to work online, too. Therefore, we replace Douglas Peucker with the Opening Window Algorithm [9, 33], which segments the incoming data stream online in subsequences as straight as possible. By doing this, every time the segmentation algorithm provides a segment, the corresponding rotation angle and envelope can be computed and recorded. Thanks to the sum in Eq. 2, the lower bound can then easily be updated online by computing for the incoming data points the matching points in the query and summing the minimal distances.

In addition, not only the method for the segmentation step is interchangeable, but also the rotation angles can be determined in different ways. For our implementation, we decided not to use *least square fitting* as suggested by Gong et al., but the arcus tangens of the segment's start and end point. This simplification is possible, since the segmentation algorithms are already supposed to find straight line segments and so the subsequence can be approximated by the line from the start to the end point. With this simplification, the rotation angle θ is:

$$\theta = -\arctan\left(\frac{\Delta y}{\Delta x}\right),$$

where $\Delta x = x_{last} - x_{start}$ and $\Delta y = y_{last} - y_{start}$ with x_{start} (y_{start}) are the first x-(y)-coordinates of the segment and x_{end} (y_{end}) the last x-(y)-coordinates, respectively.

3.2 k-Nearest Neighbor Search

For similarity search applications, especially for k-nearest neighbors, lower bounds can be used to speed up the process of searching. Therefore, a search by example strategy is employed: one proceeds with a linear search over the complete dataset remembering the best result (e.g., nearest neighbor) found so far. For each new element in the linear search, the lower bound is calculated. If the lower bound is higher than the current nearest neighbor distance, we can avoid calculating the true distance and prune the element. If, however, the lower bound is smaller than the current nearest neighbor distance, the true distance must be computed in order to decide, whether the current element is better than the nearest neighbor found so far.

In the context of this algorithm, the two metrics *pruning ratio* and *tightness* are widely used to assess the quality of lower bounds.

Tightness is a measure comparing the lower bound with the true distance and we define it, according to [12], as follows:

$$T = \frac{\text{Value of the Lower Bound}}{\text{True Value of DTW}}.$$

This creates a value between zero and one, the larger, the better. It summarizes how much the lower bound resembles the true distance. For a tightness of one, the lower bound coincides with the distance, for a tightness of zero, the lower bound is the trivial lower bound given by a the constant zero function.

Though tightness is a good measure for comparing different lower bounds, it is not directly related to search speed. Therefore, a more practicable metric called *pruning ratio* has been defined. This measure is based on counting how often the lower bound has successfully avoided the calculation of the true distance.

Formally, the *pruning ratio* can be defined as follows:

$$P = \frac{\text{Number of Omitted Elements}}{\text{Total Number of Elements in the Dataset}}.$$

This measure creates a value between zero and one with one representing the best case.

Note that tightness is a quite general measure and only influenced by the dataset and the lower bound algorithm. The pruning ratio also depends on the best known candidate in a linear search at a given iteration, hence, on the ordering of the dataset. As an extreme example, consider the nearest neighbor being the first element to be inspected. Then, the lower bound has a higher chance of pruning elements as if the best example is very far away from the beginning of the linear scan.

We will exploit this fact by proposing some presorting strategies for trajectories in order to increase the pruning ratio while the tightness keeps constant.

3.3 Presorting Strategies

A central idea of this paper is to increase the pruning ratio by presorting strategies for two-dimensional time series. Even though very simple summaries of trajectories such as the centroid of the set of points do not carry much information about trajectory similarity with respect to DTW distance, they can be used to reorder the dataset or to prepone the analysis of promising candidates in order to be able to more often use the lower bound for pruning. Note that we do not want to sort the complete database in order to have the majority of the linear scan access data in its ordering in memory or on disk. We just prepone a set of examples, before we enter the classical linear scan.

The main idea is to summarize a complete trajectory (e.g., the sequence of spatial points) into a single indexable object. For example, we can summarize the point set of a trajectory by the centroid of the point set. As the centroid maps a trajectory database into a database of points of the same size, existing efficient point indexing strategies such as R^*-trees can be used to efficiently retrieve k-nearest neighbors.

The following sections introduce three presorting strategies for trajectory data in the same framework: first, a single object summary is calculated and then similarity of these summaries can be used to prepone some examples to increase pruning power. Of course, many other variants can be easily defined and it depends on the application and datasets how to choose such a summary.

3.3.1 Presorting Using Centroids

For all datasets, we used the centroid of the points of the given trajectory (i.e., the mean of the coordinates) as a summary. For increasing pruning ratio, we first looked at a given number of nearest neighbors in the set of centroids (which is a simple point set to be indexed for example by an R^* tree). The idea is that similarity of centroids is a necessary condition for similarity with respect to DTW: if the centroids are far from each other, the average distance between points of two trajectories will be large, hence, DTW distance will be large.

3.3.2 Presorting Using the Last Point

Another point-based presorting strategy is given by bringing forward some trajectories whose end points are near each other. This is especially useful in online scenarios in which similarity is in any way searched with respect to the immediate past.

3.3.3 Presorting Using the Jaccard distance of Geohash sets

The third presorting strategy is given by encoding every point of every time series using the Geohash mechanism [30, 19] and considering the Jaccard distance of the involved sets as an indicator of nearness. This strategy is especially interesting as the Jaccard distance can be indexed by using Bloom filters in an extremely memory- and time-efficient way [26].

4 Evaluation

We implemented the lower bounds $\mathrm{LB}_{\mathrm{Keogh}}$ and four different variants of $\mathrm{LB}_{\mathrm{Rotation}}$ as a library for the statistical computing environment R using C++[1]. We appreciate that the authors of the initial paper on $\mathrm{LB}_{\mathrm{Rotation}}$ provided us with their implementation, which we used as a reference for reconstructing some details of their approach.

In a preparation step, we preprocess all trajectories of a dataset to contain the same number of points (see Equation 2). Therefore, we use piecewise linear interpolation (PLI) for trajectories with fewer points than desired as well as picewise aggregate approximation (PAA) for trajectories which are too long. Optionally, we independently normalize trajectories by subtracting the mean and dividing by the standard deviation.

For the segmentation of trajectories, we implement three approaches: the first approach is, of course, Douglas Peucker simplification with a given threshold. Note that the original paper prescribed the number of segments to generate. Due to the variations in our datasets, the approach using a threshold in terms of the distance in the dataset is better. However, we do not know the number of segments for a given trajectory beforehand. For being able to process $\mathrm{LB}_{\mathrm{Rotation}}$ in an online manner, we used the Opening Window Algorithm. Though the Opening Window Algorithm has quadratic complexity, it is often used as a replacement for Douglas Peucker for the case of data streams [33].

For assessing the orientation of segments, we used the linear models provided by the R environment with the `lm` function. As this function rejects to fit lines to sequences of equal points, we completed the linear model fitting by setting an angle of zero in any case, where `lm` rejects the creation of a linear model for a segment. This amounts to not rotating these segments. This is a minor difference to the original version, which tries to derive an angle also in numerically unstable situations. Additionally, we argue that the segmentation algorithm should have already taken care of creating linear segments and just assess the orientation based on the angle formed by the first and last point of the segments.

Presorting has been implemented by an appropriate index for the given presorting strategy: An R^* tree as provided by the `boost::geometry::index` library for the case of points and Bloom Aggregated Cell Representations for the case of Jaccard distance of Geohash sets [26].

4.1 Data Sets

We evaluate our approach against the original $\mathrm{LB}_{\mathrm{Rotation}}$ and $\mathrm{LB}_{\mathrm{Keogh}}$ and use several variants of segmentation and line fitting algorithms on datasets of varying type.

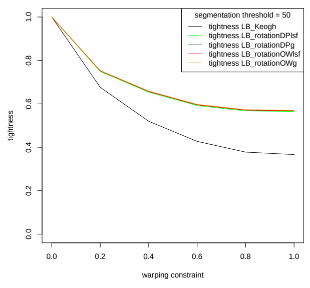
4.1.1 Geolife GPS Trajectories

The Geolife dataset is a large GPS dataset containing 18,670 GPS trajectories with 24,876,978 points. It contains the trajectories of 178 users in a period of over four years from April 2007 to October 2011. The data is given in WGS84 using latitude and longitude [31, 32].

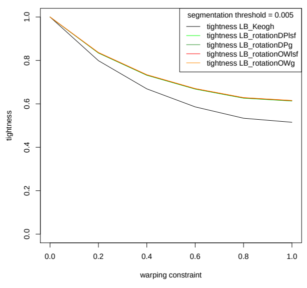
4.1.2 Character Trajectories

The Character dataset, available from the UCI Machine Learning Repository, contains 2858 trajectories of writing 20 different letters on a digital tablet [16]. Note that the samples in this dataset are given as the smoothed derivative of location. This makes the letters

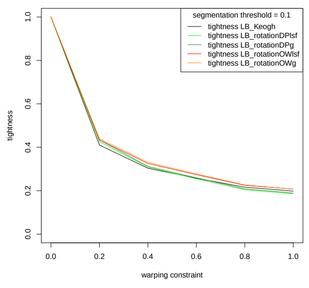
[1] We plan to publish all the source code via CRAN and on our web pages.



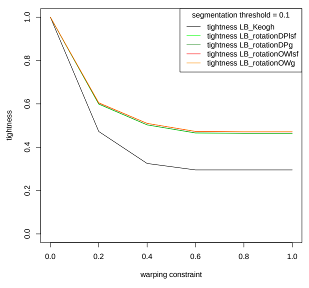
(a) Prague



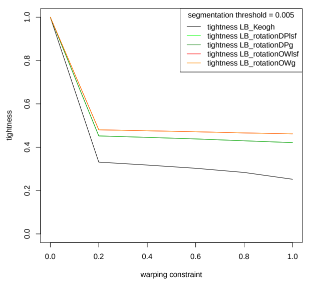
(b) San Francisco



(c) Roma



(d) Character



(e) Geolife

Figure 3 Tightness of $\mathrm{LB}_{\mathrm{Keogh}}$ and variants of $\mathrm{LB}_{\mathrm{Rotation}}$.

independet from the actual pen location on the tablet. For our experiments, we calculate the cumulative sums of these discrete derivatives and perform the matching of characters on their actual shape. Note that by this procedure, all characters start at the zero point.

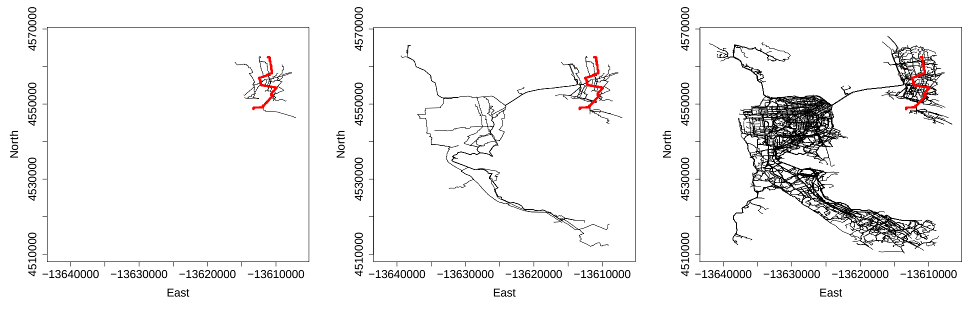
4.1.3 Prague Ego-Shooter Dataset

The Prague Ego-Shooter Dataset is a dataset containing 50 minutes of five players playing Urban Terror in Capture and Hold Mode on the map Prague (ut4_prague) [25]. In this game mode, players of two teams start in two different areas and the goal of the game is to "hold" the flag. The flag is positioned at a third fixed location on the map and the team who was the last walking over the flag is "holding" the flag. The team scores, who is holding the flag at specific time points. This map data has a lot of internal structure including the three locations and other tactically important spots around the map.

The dataset contains 275 trajectories for a total of 244,675 samples of player locations taken every 50ms. The coordinate system is similar to an orthogonal coordinate system with a unit of meters.

4.1.4 Roma Taxi Dataset

The Roma Taxi Dataset contains trajectories of 315 taxi drivers working in the center of Rome [5]. The traces contain the position of the taxis roughly every 7 seconds. The dataset was acquired with various Android tablets and is given in WGS84 coordinates. Note that the getAccuracy method of the Android location API has been used to reject positions for which this API method estimates a precision worse than 20m. In other words, only good signal situations have been kept. The dataset contains 21,743,005 points.



(a) First 10 trajectories **(b)** First 25 trajectories **(c)** All trajectories

Figure 4 A reordering of the San Francisco dataset.

4.1.5 Fastest Paths in San Francisco Dataset

The Fastest Paths in San Francisco dataset is a synthetic dataset we created from a street network of greater San Francisco area [28]. The street network was extracted from OpenStreetMap data and used in the ACM SIGSPATIAL GIS Cup 2015 [27]. The dataset provides the coordinates of nodes in the street network as modelled in the OpenStreetMap and calculates the fastest way between two random nodes using a weighting based on speed annotations in OpenStreetMap. It contains 539 trajectories with 136,978 points. Trajectory length ranges from 4 to 681 with a mean length of 254.1 points per trajectory.

4.2 Experiments

In this section, we present several experiments, we performed on the datasets. First, we evaluate tightness as a function of the warping constraint. Then, we look into pruning ratio and presorting.

4.2.1 Tightness

First, we calculate average tightness of five lower bounds: LB_{Keogh} and four version of LB_{Rotation}, namely for Douglas Peucker and Opening Window together with least square fitting and simple segment fitting. Figure 3 depicts the results for the five datasets Prague, Roma, San Francisco, Character and Geolife. We conclude that all variants of LB_{Rotation} perform similar, however, we also note that the gain of LB_{Rotation} is quite small for Roma as you can see in Figure 3c. This is to be expected as the trajectories of the Roma dataset are quite long (2,994 km on average; each trajectory is the complete trace of a Taxi driver during a month) and similar with each other. Interestingly, the GeoLife dataset shows nearly identical tightness for both Douglas Peucker variants and significantly better tightness for the Opening Window algorithm. This is particularly interesting as the Douglas Peucker algorithm is often used as a reference for segmentation algorithms due to its mathematical and perceptive quality and this can be seen as a warning that these two properties are not always deciding.

In summary, we conclude that LB_{Rotation} can be calculated more efficiently without loss of tightness by using the segment start and end points instead of fitting a linear model. Additionally, we conclude that an online segmentation such as Opening Window is applicable. We remark, however, that the quality of the segmentation has a severe impact on LB_{Rotation}

Table 1 Pruning Ratio and Speedup.

Dataset	Strategy	Average Pruning Ratio with Strategy	Average Pruning Ratio with Random Ordering	Speedup
San Fransisco A	Centroid	0.9925651	0.9849442	2.03
San Fransisco A	Geohash-Jaccard	0.8461538	0.8038462	1.27
San Fransisco A	Last Point	0.9925373	0.9826493	2.32
San Fransisco B	Centroid	0.9944238	0.9830855	3.03
San Fransisco B	Geohash-Jaccard	0.9868421	0.9236842	5.8
San Fransisco B	Last Point	0.9962687	0.9809701	5.1
Roma Taxi Dataset	Centroid	0.4493631	0.4407643	1.02
Roma Taxi Dataset	Geohash-Jaccard	0.5306122	0.4938776	1.08
Roma Taxi Dataset	Last Point	0.4423567	0.4417197	1.0011
Prague Dataset	Centroid	0.9817518	0.970073	1.64
Prague Dataset	Last Point	0.9817518	0.9689781	1.7
Character Trajectories	Centroid	0.8989899	0.8484848	1.5
Character Trajectories	Last Point	0.9292929	0.8565657	2.0
Geolife Dataset	Centroid	0.9985719	0.9985438	1.02
Geolife Dataset	Geohash-Jaccard	0.998602	0.9984555	1.11
Geolife Dataset	Last Point	0.9836735	0.98322	1.03

and that very long trajectories such as those of the Roma dataset degrade the usefulness of all variants of LB_{Rotation} altogether.

4.2.2 Pruning Ratio and Presorting

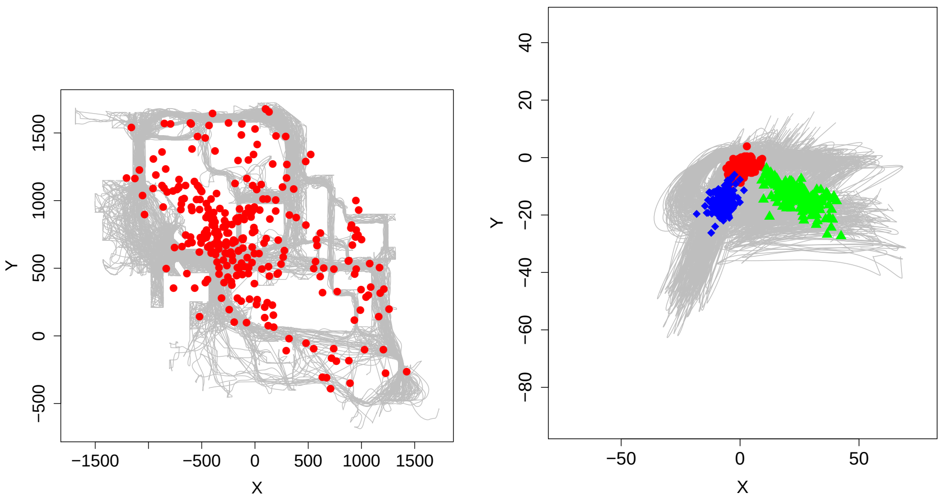
For evaluating the presorting strategies, we select a query object in each dataset and calculate ten reorderings of the dataset in which each time the ten nearest-neighbors have been brought forward.

Figure 4 illustrates this approach for the San Francisco dataset. In this Figure, the trajectory centroid was used as a presorting strategy and we select the ten nearest neighbors with respect to the centroids in order to bootstrap similarity search from them (see Figure 4a). After that, as you can see in Figure 4b, trajectories arrive in the ordering they are stored on disk in order to reduce the amount of random access to the dataset. In Figure 4c, one can see the full dataset.

Additionally, we calculate ten fully random orderings of the same dataset. In any case, the query was removed from the dataset for a more realistic setting.

In general, any of the presorting strategies showed a good increase in pruning power. Table 1 gives results on the datasets.

For the San Francisco dataset, two queries have been used. One on the continent (San Francisco A) and one on the island (San Francisco B). The probability of finding a good example by chance is larger for the island as the dataset contains only a small part of the continent (see Figure 4). This can be seen in the pruning ratio of random orderings. On average, it is easier to find a good candidate for the island case. This is to be expected, as the dataset has a bias towards more trajectories on the island due to the fact that more graph nodes have been modelled on the island. With respect to the pruning strategies, San Francisco shows very good centroid and last point performance, but only moderate Geohash set similarity performance. This is to be expected as the trajectories are shortest paths and the centroid as well as the last point of a shortest path is a good summary of the path. Furthermore, the trajectories are relatively short and, hence, the centroid is a good summary



(a) All Centroids **(b)** Letters "a", "p", "z"

Figure 5 Centroid Distribution of (a) Prague Ego-Shooter Dataset and (b) Character Trajectories Dataset.

of the trajectory. Anyways, the tightness of all variants is quite good (only few percents of the dataset are actually analyzed with full DTW calculations). The speedup illustrates the number of DTW calculations, that have to be used without presorting. It is calculated as follows:

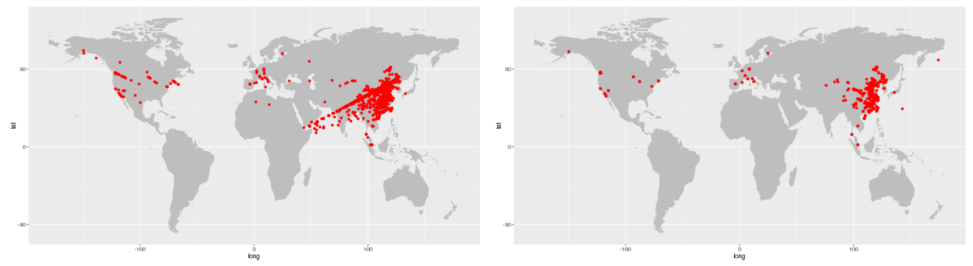
$$\text{Speedup} = \frac{1 - \text{Pruning Ratio with Presorting}}{1 - \text{Pruning Ratio without Presorting}}.$$

With an average speedup of 3.25, the classical approach needs 3.25 times as many DTW calculations as the presorting algorithms.

The Roma dataset contains long trajectories of taxis driving around the center area of Rome. Due to the taxi nature of the trajectories, the centroids are not very informative, as they are inside the center of Rome for all trajectories.

Therefore, the speedup of the point-based strategies is quite poor. The Jaccard distance of Geohash sets strategy, however, performs better. This is due to the fact that the center of Rome, which is part of all trajectories, does not contribute to the dissimilarity of two trajectories and they are indexed by their set of extreme locations (i.e., those Geohash cells that are not in other trajectories). This results in a considerable speedup. Remember that this dataset is a hard dataset for lower bounds (see Figure 3c) due to the very long trajectories traveling around the city. This is expressed by the low general pruning ratio. In this situation, the speedup of 8% is to be understood relative to the large fraction of roughly half of the dataset, which has not been pruned. In this way, our pruning strategy prunes a large absolute number of DTW calculations in comparison to random orderings.

For the Prague Ego-Shooter dataset, the centroid information was quite useful and well-distributed. We expected this behavior as the game rules let players of both teams start in two isolated regions and they aim to quickly reach and defend the flag. But as teams can only score on specific points in time, players tend to look for a safe place on the map and wait for the right moment for a team attack. Figure 5a depicts the distribution of centroids on the Prague Ego-Shooter Dataset.



(a) Centroids **(b)** Last Point

Figure 6 Geolife Point Summary Distributions.

The Character Trajectories dataset is another example of presorting as a powerful tool if the summary is carefully selected. Recall that we integrate the Character dataset such that the shapes of the characters are used (the dataset contains the first derivative of the location and is often used in this modality). By the integration process, all our character strokes start at coordinate $(0, 0)$. Hence, the centroid partly covers the shape of the letters. This leads to a considerable speedup of 1.5 for the centroid strategy. This means, that without presorting, we would have to calculate 1.5 times as many true DTW as with the simple presorting strategy. Figure 5b illustrates examples of the centroid behaviour on the character dataset for all samples of the letters “a”,“p”, and “z”. You can clearly see that the centroids of those characters form separated clusters.

The Geolife dataset behaves similar to the Roma dataset, however, the pruning power is much higher. This is due to the fact that the Geolife dataset is split into many smaller trajectories. Again, and similar to Roma, the centroid does not cover enough information while the Geohash Jaccard distance is able to ignore the common parts of trajectories and concentrate on significantly different locations. Figure 6a depicts the distribution of centroids for this dataset, which is clearly centered on Beijing. Similarly, the distribution of last points of trajectories is also not very informative as depicted in Figure 6b.

In order to illustrate the power of using Geohash Jaccard distance, we provide an embedding based on multidimensional scaling (MDS) for a subset of 400 trajectories of Geolife with respect to the Jaccard distance. In multidimensional scaling, a distance matrix is given and MDS tries to find a set of points such that the relative distance between points is preserved [17].

Figure 7 depicts the pairwise Geohash distances embedded into two-dimensional space as good as possible. Therefore, we employed the SMACOF algorithm minimizing the metric stress

$$\sigma(X) = \sum_{i<j\leq n} w_{ij}(d_{ij}(X) - \delta_{ij})^2$$

between the Geohash distances δ_{ij} and the Euclidean distances d_{ij}(X) in the two-dimensional embedding X. The weights for the individual terms w_{ij} have all been set to one as there is no uncertainty in the distances in our case. We added some jitter to the plot just to make the number of identical points visible. You can clearly see that the points are distributed around the model space without much clustering structure. So this summaries created from Jaccard distance of Geohash cells actually contain useful information for preponing candidates explaining the very good presorting power in San Francisco B, Roma, and Geolife.

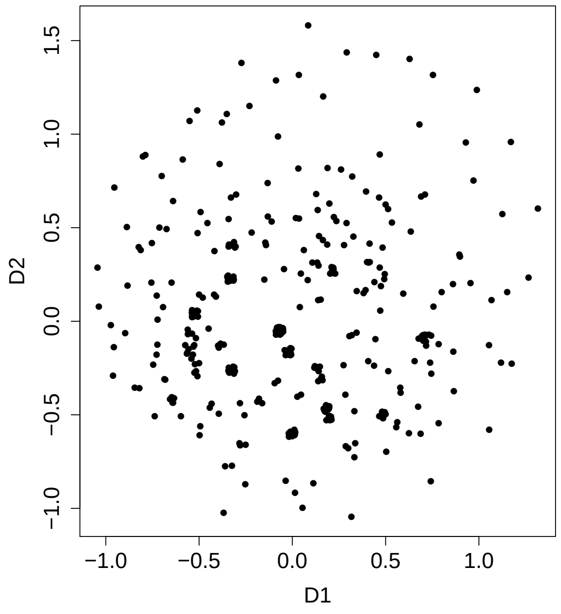


Figure 7 Metric Embedding of 400 Geolife trajectories with respect to Jaccard distance of Geohash sets.

5 Conclusion

With this paper, we have thoroughly evaluated the novel lower bound for dynamic time warping called $\mathrm{LB}_{\mathrm{Rotation}}$ on several datasets. Additionally, we show that it can be extended to an online algorithm without severe degradation in performance. Third, we showed that a presorting strategy with simple summaries of trajectories such as the last point, the centroid, or the set of visited spatial cells creates better pruning power, sometimes considerably, sometimes marginally.

The most important distinction in this context is the question, how a trajectory can be aggregated in a useful way as a single object. One way to do that is calculating some aggregate. The last point strategy and the centroid strategy are examples for that. The centroid is clearly an "average" as it can be calculated as the mean of the coordinates. The last point also describes the average movement, especially, for datasets where all trajectories start at the same point or are normalized to do so. Another way of presorting is describing the extreme behavior of a single trajectory compared to others.

We have done so – to some extent – by measuring the dissimilarity of the sets of Geohash cells each trajectory hits. Future work could include other ways of extracting the most extreme behavior from trajectories, such as extracting features from the convex hull or performing archetypal analysis [3]. Additionally, ensembles of summaries can be constructed and evaluated. However, the most interesting problem in this area would be, how to select and configure summaries in order to gain the highest pruning power from presorting for similarity search.

References

1 Assaf Almog, Avi Levi, and Alfred M. Bruckstein. Spatial de-interlacing using dynamic time warping. In *IEEE International Conference on Image Processing*, volume 2, pages 1006–1010. IEEE, 2005.

2 Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with COTE: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.

3 Christian Bauckhage and Christian Thurau. Making archetypal analysis practical. In *Pattern Recognition*, pages 272–281. Springer, 2009.

4 Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, pages 359–370. Seattle, WA, 1994.

5 Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from `http://crawdad.org/roma/taxi/20140717`, July 2014. `doi:10.15783/C7QC7M`.

6 Won-Du Chang and Jungpil Shin. Modified dynamic time warping for stroke-based on-line signature verification. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 724–728. IEEE, 2007.

7 Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

8 David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

9 Fosca Giannotti and Dino Pedreschi. *Mobility, data mining and privacy: Geographic knowledge discovery*. Springer Science & Business Media, 2008.

10 Xudong Gong, Yan Xiong, Wenchao Huang, Lei Chen, Qiwei Lu, and Yiqing Hu. Fast similarity search of multi-dimensional time series via segment rotation. In *Database Systems for Advanced Applications*, pages 108–124. Springer, 2015.

11 Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):67–72, 1975.

12 Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

13 Sang-Wook Kim, Sanghyun Park, and Wesley W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *Proceedings of the 17th International Conference on Data Engineering*, pages 607–614. IEEE, 2001.

14 Benoit Legrand, C. S. Chang, S. H. Ong, Soek-Ying Neo, and Nallasivam Palanisamy. Chromosome classification using dynamic time warping. *Pattern Recognition Letters*, 29(3):215–222, 2008.

15 Daniel Lemire. Faster retrieval with a two-pass dynamic-time-warping lower bound. *Pattern recognition*, 42(9):2169–2180, 2009.

16 M. Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013.

17 Patrick Mair, Jan de Leeuw, and Patrick J. F. Groenen. Multidimensional Scaling in R: SMACOF. URL: `https://cran.r-project.org/web/packages/smacof/vignettes/smacof.pdf`.

18 Andrés Marzal, Vicente Palazón, and Guillermo Peris. Contour-based shape retrieval using dynamic time warping. In *Current Topics in Artificial Intelligence*, pages 190–199. Springer, 2005.

19 Gustavo Niemeyer. Geohash. `http://geohash.org/`, 2008.

20 Toni M. Rath and R. Manmatha. Lower-bounding of dynamic time warping distances for multivariate time series. Available at `http://works.bepress.com/r_manmatha/19/`, 2003.

21 Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

22 Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

23 V. M. Velichko and N. G. Zagoruyko. Automatic recognition of 200 words. *International Journal of Man-Machine Studies*, 2(3):223–234, 1970.

24 Haozhou Wang, Han Su, Kai Zheng, Shazia Sadiq, and Xiaofang Zhou. An effectiveness study on trajectory similarity measures. In *Proceedings of the Twenty-Fourth Australasian Database Conference-Volume 137*, pages 13–22. Australian Computer Society, Inc., 2013.

25 Martin Werner. Dataset containing trajectories of four players playing one hour of Urban Terror Capture and Hold on the map Prague. Available at `http://trajectorycomputing.com/datasets/prague-ego-shooter-dataset/`, 2014.

26 Martin Werner. BACR: Set Similarities with Lower Bounds and Application to Spatial Trajectories. In *23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2015)*, 2015.

27 Martin Werner. GISCUP 2015: Notes on Routing with Polygonal Constraints. In *SIGSPATIAL GIS CUP 15, in conjunction with 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2015)*, 2015. URL: `http://www.martinwerner.de/preprint/GISCUP.pdf`.

28 Martin Werner. Dataset containing 500 random shortest paths in greater San Francisco Area. Available at `http://www.martinwerner.de/files/fastest_sanfrancisco.tgz`, 2016.

29 Byoung-Kee Yi, H. V. Jagadish, and Christos Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceedings of the 14th International Conference on Data Engineering*, pages 201–208. IEEE, 1998.

30 Chun-Ting Zhang, Ren Zhang, and Hong-Yu Ou. The Z curve database: a graphic representation of genome sequences. *Bioinformatics*, 19(5):593–599, 2003.

31 Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on GPS data. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 312–321. ACM, 2008.

32 Yu Zheng, Xing Xie, and Wei-Ying Ma. GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. *IEEE Data Engineering Bulletin*, 33(2):32–39, 2010.

33 Yu Zheng and Xiaofang Zhou. *Computing with spatial trajectories.* Springer Science & Business Media, 2011.

Collective Singleton-Based Consistency for Qualitative Constraint Networks

Michael Sioutis*[1], Anastasia Paparrizou[2], and Jean-François Condotta[3]

1 Örebro University, MPI@AASS, Örebro, Sweden
michael.sioutis@oru.se
2 Artois University, CRIL UMR 8188, Lens, France
paparrizou@cril.fr
3 Artois University, CRIL UMR 8188, Lens, France
condotta@cril.fr

Abstract

Partial singleton closure under weak composition, or partial $\blacklozenge$-consistency for short, is essential for approximating satisfiability of qualitative constraints networks. Briefly put, partial $\blacklozenge$-consistency ensures that each base relation of each of the constraints of a qualitative constraint network can define a singleton relation in the corresponding partial closure of that network under weak composition, or in its corresponding partially $\diamond$-consistent subnetwork for short. In particular, partial $\blacklozenge$-consistency has been shown to play a crucial role in tackling the minimal labeling problem of a qualitative constraint network, which is the problem of finding the strongest implied constraints of that network. In this paper, we propose a stronger local consistency that couples $\blacklozenge$-consistency with the idea of collectively deleting certain unfeasible base relations by exploiting singleton checks. We then propose an efficient algorithm for enforcing this new consistency that, given a qualitative constraint network, performs fewer constraint checks than the respective algorithm for enforcing partial $\blacklozenge$-consistency in that network. We formally prove certain properties of our new local consistency, and motivate its usefulness through demonstrative examples and a preliminary experimental evaluation with qualitative constraint networks of Interval Algebra.

1998 ACM Subject Classification I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases Qualitative constraint network, qualitative spatial and temporal reasoning, partial singleton path-consistency, local consistency, minimal labeling problem

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.19

1 Introduction

Qualitative Spatial and Temporal Reasoning (QSTR) is a major field of study in Artificial Intelligence, and in particular in Knowledge Representation & Reasoning. This field has received a lot of attention over the past decades, as it extends to a plethora of areas and domains that include ambient intelligence, dynamic GIS, cognitive robotics, and spatiotemporal design [6]. QSTR abstracts from numerical quantities of space and time by using qualitative descriptions instead (e.g., *precedes*, *contains*, *is left of*), thus providing a concise framework that allows for rather inexpensive reasoning about entities located in space and time.

* Michael Sioutis wishes to acknowledge support from the MoveCare project (ID: **732158**), which is funded by the European Commission under the H2020 framework programme for research and innovation. Finally, parts of this work were conducted during the postdoctoral research of Michael Sioutis in Artois University.

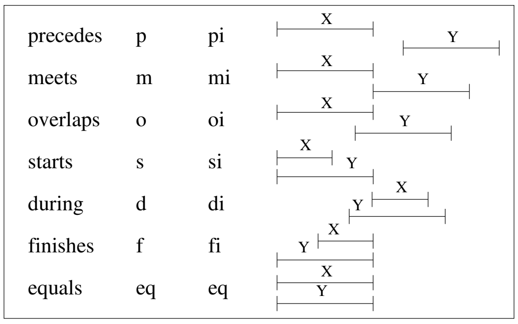
The problem of representing and reasoning about qualitative information can be modeled as a qualitative constraint network (QCN) using a qualitative constraint language. Specifically, a QCN is a network of constraints corresponding to qualitative spatial or temporal relations between spatial or temporal variables respectively, and a qualitative constraint language is used to define those constraints over a finite set of binary relations, called *base relations* (or *atoms*) [19]. An example of such a qualitative constraint language is Interval Algebra (IA), introduced by Allen [1]. IA considers time intervals (as its temporal entities) and each of its base relations represents an ordering of the four endpoints of two intervals in the timeline.

The fundamental reasoning problems associated with a given QCN $\mathcal{N}$ are the problems of *satisfiability checking*, *minimal labeling* (or *deductive closure*), and *redundancy* (or *entailment*) [28]. In particular, the satisfiability checking problem is the problem of deciding if there exists a spatial or temporal valuation of the variables of $\mathcal{N}$ that satisfies its constraints, such a valuation being called a *solution* of $\mathcal{N}$, the minimal labeling problem is the problem of finding the strongest implied constraints of $\mathcal{N}$, and the redundancy problem is the problem of determining if a given constraint is entailed by the rest of $\mathcal{N}$ (that constraint being called redundant, as its removal does not change the solution set of the QCN). In general, for most qualitative constraint languages the satisfiability checking problem is NP-complete. Further, the redundancy problem, the minimal labeling problem, and the satisfiability checking problem are equivalent under polynomial Turing reductions [13].

The vast amount, if not all, of the published works that study the aforementioned reasoning problems, use *partial $\diamond$-consistency* as a means to define practical algorithms for efficiently tackling them [2, 30, 29, 17, 27, 23, 15]. Given a QCN $\mathcal{N}$ and a graph G, partial $\diamond$-consistency with respect to G, denoted by $\diamond_G$-consistency, entails (weak) consistency for all triples of variables in $\mathcal{N}$ that correspond to three-vertex cycles (triangles) in G. We note that if G is complete, $\diamond_G$-consistency becomes identical to *$\diamond$-consistency* [26]. Hence, $\diamond$-consistency is a special case of $\diamond_G$-consistency. In fact, earlier works have relied solely on $\diamond$-consistency; it was not until the introduction of chordal (or triangulated) graphs in QSTR, due to some generalized theoretical results of [14], that researchers started restricting $\diamond$-consistency to a triangulation (or chordal completion) of the constraint graph of an input QCN and benefiting from better complexity properties in more recent works.

Adding to the previous paragraph, and with respect to the satisfiability checking problem in particular, the literature suggests that $\diamond_G$-consistency alone is sufficient in most cases to guarantee that a solution for a given QCN, should it exist, is efficiently obtained (see also [8]). However, for the more challenging problems of minimal labeling and redundancy, a stronger local consistency is typically employed that builds upon $\diamond_G$-consistency, called singleton $\diamond_G$-consistency and denoted by $\blacklozenge_G$-consistency [2, 30]. Given a QCN $\mathcal{N}$ and a graph G, $\blacklozenge_G$-consistency holds on $\mathcal{N}$ if and only if each base relation of each of the constraints of $\mathcal{N}$ is closed under $\diamond_G$-consistency, i.e., after instatiating any constraint of that network with one of its base relations b and closing the network under $\diamond_G$-consistency, the corresponding constraint in the $\diamond_G$-consistent subnetwork will continue being defined by b.

It is then natural to ask whether we can have an even stronger local consistency than $\blacklozenge_G$-consistency (and $\diamond_G$-consistency) for QCNs that can also be enforced more efficiently than $\blacklozenge_G$-consistency, as a positive answer to that question would suggest an immediate improvement for any algorithm that currently employs $\blacklozenge_G$-consistency. In this paper, we contribute towards obtaining such a positive answer. In particular, we enrich the family of consistencies for QCNs by proposing a new singleton style consistency inspired by k-partitioning consistency for constraint satisfaction problems (CSPs) [4]. This filtering technique is based on domain partitioning combined with a local consistency, typically *arc consistency* [5], and allows for



precedes	p	pi
meets	m	mi
overlaps	o	oi
starts	s	si
during	d	di
finishes	f	fi
equals	eq	eq

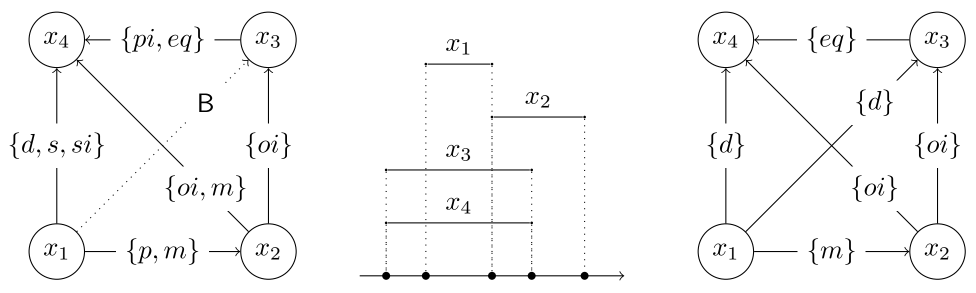
Figure 1 The base relations of IA.

improved pruning capability over *singleton arc consistency* [9]. Similarly to k-partitioning consistency, our new consistency, denoted by $\overset{\blacklozenge}{G}^{\cup}$-consistency, combines singleton checks and $\overset{\diamond}{G}$-consistency to present itself as a better alternative to $\overset{\blacklozenge}{G}$-consistency. With respect to our new consistency, we also propose an algorithm for applying it on a given QCN, which turns out being more efficient than the respective algorirthm for applying $\overset{\blacklozenge}{G}$-consistency on that same QCN. As a brief intuitive explanation of this, $\overset{\blacklozenge}{G}^{\cup}$-consistency allows for proactively eliminating base relations anywhere in a given QCN and not only in the set of base relations of the constraint at hand that is singleton checked. Further, we obtain several theoretical results and show, among other things, that $\overset{\blacklozenge}{G}^{\cup}$-consistency is *strictly stronger* than $\overset{\blacklozenge}{G}$-consistency and, hence, than $\overset{\diamond}{G}$-consistency. Finally, we present a preliminary experimental evaluation of $\overset{\blacklozenge}{G}^{\cup}$-consistency and $\overset{\blacklozenge}{G}$-consistency using QCNs of IA, in support of our argument that $\overset{\blacklozenge}{G}^{\cup}$-consistency can be enforced more efficiently than $\overset{\blacklozenge}{G}$-consistency for a given QCN.

The rest of the paper is organized as follows. In Section 2 we give some preliminaries on qualitative spatial and temporal reasoning, and in Section 3 we focus on $\overset{\diamond}{G}$-consistency and $\overset{\blacklozenge}{G}$-consistency and, in particular, recall some related result from the literature, but also provide some new results of our own. Then, in Section 4 we introduce, formally define, and thoroughly study our new local consistency, namely, $\overset{\blacklozenge}{G}^{\cup}$-consistency. In Section 5 we present an algorithm for efficiently applying $\overset{\blacklozenge}{G}^{\cup}$-consistency on a given QCN $\mathcal{N}$, and in Section 6 we evaluate this algorithm against the state-of-the-art algorithm for achieving $\overset{\blacklozenge}{G}$-consistency. Finally, in Section 7 we conclude the paper and give some directions for future work.

2 Preliminaries

A (binary) qualitative spatial or temporal constraint language, is based on a finite set B of *jointly exhaustive and pairwise disjoint* relations defined over an infinite domain D, which is called the set of *base relations* [19]. The base relations of a particular qualitative constraint language can be used to represent the definite knowledge between any two of its entities with respect to the level of granularity provided by the domain D. The set B contains the identity relation Id, and is closed under the *converse* operation ($^{-1}$). Indefinite knowledge can be specified by a union of possible base relations, and is represented by the set containing them. Hence, 2^{B} represents the total set of relations. The set 2^{B} is equipped with the usual set-theoretic operations of union and intersection, the converse operation, and the *weak composition* operation denoted by the symbol $\diamond$ [19]. For all $r \in 2^{\mathsf{B}}$, we have that $r^{-1} = \bigcup\{b^{-1} \mid b \in r\}$. The weak composition ($\diamond$) of two base relations $b, b' \in \mathsf{B}$ is defined as the smallest (i.e., strongest) relation $r \in 2^{\mathsf{B}}$ that includes $b \circ b'$, or, formally, $b \diamond b' = \{b'' \in \mathsf{B} \mid b'' \cap (b \circ b') \neq \emptyset\}$, where $b \circ b' = \{(x, y) \in \mathsf{D} \times \mathsf{D} \mid \exists z \in \mathsf{D}$ such that $(x, z) \in$



(a) A satisfiable QCN $\mathcal{N}$ **(b)** A solution σ of $\mathcal{N}$ **(c)** A scenario $\mathcal{S}$ of $\mathcal{N}$

Figure 2 Figurative examples of QCN terminology using IA.

$b \wedge (z, y) \in b'\}$ is the (true) composition of b and b'. For all $r, r' \in 2^{\mathsf{B}}$, we have that $r \diamond r' = \bigcup\{b \diamond b' \mid b \in r, b' \in r'\}$.

As an illustration, consider the well known qualitative temporal constraint language of Interval Algebra (IA) introduced by Allen [1]. IA considers time intervals (as its temporal entities) and the set of base relations $\mathsf{B} = \{eq, p, pi, m, mi, o, oi, s, si, d, di, f, fi\}$; each base relation of IA represents a particular ordering of the four endpoints of two intervals in the timeline, as demonstrated in Figure 1. The base relation eq is the identity relation Id of IA. As another illustration, the Region Connection Calculus (RCC) is a first-order theory for representing and reasoning about mereotopological information [24]. The domain D of RCC comprises all possible non-empty regular subsets of some topological space. These subsets do not have to be internally connected and do not have a particular dimension, but are commonly required to be regular *closed* [25]. Other notable and well known qualitative spatial and temporal constraint languages include Point Algebra [35], Cardinal Direction Calculus [18, 11], and Block Algebra [3].

The weak composition operation $\diamond$, the converse operation $^{-1}$, the union operation $\cup$, the complement operation $^{\complement}$, and the total set of relations 2^{B} along with the identity relation Id of a qualitative constraint language, form an algebraic structure $(2^{\mathsf{B}}, \mathsf{Id}, \diamond, ^{-1}, ^{\complement}, \cup)$ that can correspond to a *relation algebra* in the sense of Tarski [33].

▶ **Proposition 1** ([10]). *The languages of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, and* RCC*-8 are each a relation algebra with the algebraic structure* $(2^{\mathsf{B}}, \mathsf{Id}, \diamond, ^{-1}, ^{\complement}, \cup)$.

In what follows, for a qualitative constraint language that is a relation algebra with the algebraic structure $(2^{\mathsf{B}}, \mathsf{Id}, \diamond, ^{-1}, ^{\complement}, \cup)$, we will simply use the term *relation algebra*, as the algebraic structure will always be of the same format.

The problem of representing and reasoning about qualitative information can be modeled as a *qualitative constraint network* (QCN), defined in the following manner:

▶ **Definition 2.** A *qualitative constraint network* (QCN) is a tuple (V, C) where:

- $V = \{v_1, \ldots, v_n\}$ is a non-empty finite set of variables, each representing an entity;
- and C is a mapping $C : V \times V \to 2^{\mathsf{B}}$ such that $C(v, v) = \{\mathsf{Id}\}$ for all $v \in V$ and $C(v, v') = (C(v', v))^{-1}$ for all $v, v' \in V$.

An example of a QCN of IA is shown in Figure 2a; for simplicity, converse relations as well as Id loops are not mentioned or shown in the figure.

▶ **Definition 3.** Let $\mathcal{N} = (V, C)$ be a QCN, then:

- a *solution* of $\mathcal{N}$ is a mapping $\sigma : V \rightarrow \mathsf{D}$ such that $\forall (u, v) \in V \times V$, $\exists b \in C(u, v)$ such that $(\sigma(u), \sigma(v)) \in b$ (see Figure 2b);
- $\mathcal{N}$ is *satisfiable* iff it admits a solution;
- a QCN is *equivalent* to $\mathcal{N}$ iff it admits the same set of solutions as $\mathcal{N}$;
- a *sub*-QCN $\mathcal{N}'$ of $\mathcal{N}$, denoted by $\mathcal{N}' \subseteq \mathcal{N}$, is a QCN (V, C') such that $C'(u, v) \subseteq C(u, v)$ $\forall u, v \in V$; if in addition $\exists u, v \in V$ such that $C'(u, v) \subset C(u, v)$, then $\mathcal{N}' \subset \mathcal{N}$;
- $\mathcal{N}$ is *atomic* iff $\forall v, v' \in V$, $C(v, v')$ is a *singleton relation*, i.e., a relation $\{b\}$ with $b \in \mathsf{B}$;
- a *scenario* $\mathcal{S}$ of $\mathcal{N}$ is an atomic satisfiable sub-QCN of $\mathcal{N}$ (see Figure 2c);
- a base relation $b \in C(v, v')$ with $v, v' \in V$ is *feasible* (resp. *unfeasible*) iff there exists (resp. there does not exist) a scenario $\mathcal{S} = (V, C')$ of $\mathcal{N}$ such that $C'(v, v') = \{b\}$;
- $\mathcal{N}$ is *minimal* iff $\forall v, v' \in V$ and $\forall b \in C(v, v')$, b is a feasible base relation of $\mathcal{N}$;
- the *constraint graph* of $\mathcal{N}$, denoted by $\mathsf{G}(\mathcal{N})$, is the graph (V, E) where $\{u, v\} \in E$ iff $C(u, v) \neq \mathsf{B}$ and $u \neq v$;
- $\mathcal{N}$ is *trivially inconsistent* iff $\exists u, v \in V$ such that $C(u, v) = \emptyset$;
- $\mathcal{N}$ is the *empty* QCN on V, denoted by $\bot^V$, iff $C(u, v) = \emptyset$ for all $u, v \in V$.

Let us further introduce the following operations with respect to QCNs:

- given a QCN $\mathcal{N} = (V, C)$ and $v, v' \in V$, we have that $\mathcal{N}_{[v,v']/r}$ with $r \in 2^{\mathsf{B}}$ yields the QCN $\mathcal{N}' = (V, C')$ defined by $C'(v, v') = r$, $C'(v', v) = r^{-1}$ and $C'(y, w) = C(y, w)$ $\forall (y, w) \in (V \times V) \setminus \{(v, v'), (v', v)\}$;
- given two QCNs $\mathcal{N} = (V, C)$ and $\mathcal{N}' = (V, C')$ on the same set of variables V, we have that $\mathcal{N} \cup \mathcal{N}'$ yields the QCN $\mathcal{N}'' = (V, C'')$, where $C''(v, v') = C(v, v') \cup C'(v, v')$ for all $v, v' \in V$.

We recall the following definition of $\overset{\diamond}{_G}$-consistency, which, as noted in the introduction, is the basic local consistency used in the literature for solving fundamental reasoning problems of QCNs, such as the satisfiability checking problem.

▶ **Definition 4.** Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, $\mathcal{N}$ is said to be $\overset{\diamond}{_G}$-*consistent* iff $\forall \{v_i, v_j\}, \{v_i, v_k\}, \{v_k, v_j\} \in E$ we have that $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$.

We note again that if G is complete, $\overset{\diamond}{_G}$-consistency becomes identical to $\diamond$-consistency [26], and, hence, $\diamond$-consistency is a special case of $\overset{\diamond}{_G}$-consistency.

Given a graph $G = (V, E)$, a QCN $\mathcal{N} = (V, C)$ is $\overset{\blacklozenge}{_G}$-consistent iff for every pair of variables $\{v, v'\} \in E$ and every base relation $b \in C(v, v')$, after instantiating $C(v, v')$ with $\{b\}$ and computing the closure of $\mathcal{N}$ under $\overset{\diamond}{_G}$-consistency, the revised constraint $C(v, v')$ is always defined by $\{b\}$. Formally, $\overset{\blacklozenge}{_G}$-consistency of a QCN is defined as follows:

▶ **Definition 5.** Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, $\mathcal{N}$ is said to be $\overset{\blacklozenge}{_G}$-consistent iff $\forall \{v, v'\} \in E$ and $\forall b \in C(v, v')$ we have that $\{b\} = C'(v, v')$, where $(V, C') = \overset{\diamond}{_G}(\mathcal{N}_{[v,v']/\{b\}})$.

If G is a complete graph, i.e., $G = K_V$, we can easily verify that $\overset{\blacklozenge}{_G}$-consistency corresponds to $\overset{\diamond}{_{\mathsf{B}}}$-consistency of the family of $\overset{\diamond}{_f}$-consistencies studied in [8]. Interestingly, $\overset{\blacklozenge}{_G}$-consistency can also be seen as a counterpart of singleton arc consistency (SAC) [9] for QCNs. Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, for every $b \in \mathsf{B}$ and every $\{v, v'\} \in E$, we will say that b is $\overset{\blacklozenge}{_G}$-consistent for $C(v, v')$ iff $\{b\} = C'(v, v')$, where $(V, C') = \overset{\diamond}{_G}(\mathcal{N}_{[v,v']/\{b\}})$.

▶ **Definition 6.** A *subclass* of relations is a subset $\mathcal{A} \subseteq 2^{\mathsf{B}}$ that contains the singleton relations of 2^{B} and is closed under converse, intersection, and weak composition.

Given three relations r, r', $r'' \in 2^{\mathsf{B}}$, we say that weak composition distributes over intersection if we have that $r \diamond (r' \cap r'') = (r \diamond r') \cap (r \diamond r'')$ and $(r' \cap r'') \diamond r = (r' \diamond r) \cap (r'' \diamond r)$.

▶ **Definition 7.** A subclass $\mathcal{A}$ is *distributive* iff weak composition distributes over non-empty intersection $\forall r, r', r'' \in \mathcal{A}$.

Distributive subclasses of relations are defined for all of the qualitative constraint languages mentioned in Proposition 1 [20].

Finally, for the sake of simplicity in phrasing some results, in what follows we assume that all considered graphs are biconnected.

3 A closer look at $^{\diamond}_{G}$-consistency and $^{\blacklozenge}_{G}$-consistency

Let us come back to $^{\diamond}_{G}$-consistency and $^{\blacklozenge}_{G}$-consistency and recall in this section some results from the literature that will be relevant in the rest of the paper, but also provide some new results of our own.

In order to compare the pruning (or inference) capability of different consistencies, we introduce a preorder. Let $^{\phi}_{G}$ and $^{\psi}_{G}$ be two consistencies defined by some operations ϕ and ψ respectively and a graph G. Then, $^{\phi}_{G}$ is *stronger* than $^{\psi}_{G}$, denoted by $^{\phi}_{G} \trianglerighteq ^{\psi}_{G}$, iff whenever $^{\phi}_{G}$ holds on a QCN $\mathcal{N}$ with respect to a graph G, $^{\psi}_{G}$ also holds on $\mathcal{N}$ with respect to G, and $^{\phi}_{G}$ is *strictly stronger* than $^{\psi}_{G}$, denoted by $^{\phi}_{G} \triangleright ^{\psi}_{G}$, iff $^{\phi}_{G} \trianglerighteq ^{\psi}_{G}$ and there exists at least one QCN $\mathcal{N}$ and a graph G such that $^{\psi}_{G}$ holds on $\mathcal{N}$ with respect to G but $^{\phi}_{G}$ does not hold on $\mathcal{N}$ with respect to G. Finally, $^{\phi}_{G}$ and $^{\psi}_{G}$ are *equivalent*, denoted by $^{\phi}_{G} \equiv ^{\psi}_{G}$, iff we have both $^{\phi}_{G} \trianglerighteq ^{\psi}_{G}$ and $^{\psi}_{G} \trianglerighteq ^{\phi}_{G}$.

We now recall the definition of a *well-behaved* consistency [8].

▶ **Definition 8.** A consistency $^{\phi}_{G}$ is *well-behaved* iff for any QCN $\mathcal{N} = (V, E)$ and any graph $G = (V, E)$ the following properties hold:

- $^{\phi}_{G}(\mathcal{N}) \subseteq \mathcal{N}$ (viz., the $^{\phi}_{G}$-closure of $\mathcal{N}$ w.r.t. G) is the largest (w.r.t. $\subseteq$) $^{\phi}_{G}$-consistent sub-QCN of $\mathcal{N}$ *(Dominance)*;
- $^{\phi}_{G}(\mathcal{N})$ is equivalent to $\mathcal{N}$ *(Equivalence)*;
- $^{\phi}_{G}(^{\phi}_{G}(\mathcal{N})) = ^{\phi}_{G}(\mathcal{N})$ *(Idempotence)*;
- if $\mathcal{N}' \subseteq \mathcal{N}$ then $^{\phi}_{G}(\mathcal{N}') \subseteq ^{\phi}_{G}(\mathcal{N})$ *(Monotonicity)*.

It is routine to formally prove the following result for $^{\diamond}_{G}$-consistency:

▶ **Corollary 9** (cf. [8]). *We have that $^{\diamond}_{G}$-consistency is well-behaved.*

It is routine to formally prove the following result for $^{\blacklozenge}_{G}$-consistency as well:

▶ **Corollary 10** (cf. [8]). *We have that $^{\blacklozenge}_{G}$-consistency is well-behaved.*

The aforementioned two results are derived from respective results of [8] where complete graphs are used in all cases. The generalization to an arbitrary graph G is trivial.

We recall the following general result regarding the pruning capability of $^{\diamond}_{G}$-consistency in comparison with that of $^{\blacklozenge}_{G}$-consistency:

▶ **Proposition 11** ([8]). *We have that $^{\blacklozenge}_{G}$-consistency $\triangleright$ $^{\diamond}_{G}$-consistency.*

Before we proceed, we introduce the following lemma to be used in the next proposition:

▶ **Lemma 12** (cf. [20]). *Let $\mathcal{A}$ be a distributive subclass of relations of a relation algebra. Then, for any* QCN *$\mathcal{N} = (V, C)$ over $\mathcal{A}$ and any graph $G = (V, E)$, if $^{\diamond}_{G}(\mathcal{N}) = (V, C')$ is not trivially inconsistent, we have that $\forall u, v \in V$ and $\forall b \in C'(u, v)$ there exists an atomic $^{\diamond}_{G}$-consistent sub-*QCN *(V, C'') of $\mathcal{N}$ such that $\{b\} = C''(u, v)$.*

Proof (Sketch). The proof can be obtained by concatenating the proofs of Theorems 2 and 5 in [20] and applying that merged proof on each maximal chordal subgraph of G. The only major difference is that in those proofs the property that *any atomic* QCN *of a relation algebra that is* $\diamond$*-consistent is satisfiable* is used in addition to guarantee a stronger result, which is of no use to us for proving this particular lemma. ◀

Next, we introduce a result that identifies the case where $\overset{\diamond}{_G}$-consistency and $\overset{\blacklozenge}{_G}$-consistency are equivalent.

▶ **Proposition 13.** *Let $\mathcal{A}$ be a distributive subclass of relations of a relation algebra. Then, for any* QCN *$\mathcal{N} = (V, C)$ over $\mathcal{A}$ and any graph $G = (V, E)$, we have that $\overset{\diamond}{_G}$-consistency $\equiv$ $\overset{\blacklozenge}{_G}$-consistency.*

Proof. If $\overset{\diamond}{_G}(\mathcal{N}) = (V, C)$ is not trivially inconsistent, then by Lemma 12 we have that $\forall u, v \in V$ and $\forall b \in C(u, v)$ there exists an atomic $\overset{\diamond}{_G}$-consistent sub-QCN (V, C') of $\overset{\diamond}{_G}(\mathcal{N})$ such that $\{b\} = C'(u, v)$. This suggests that $\overset{\diamond}{_G}(\mathcal{N})$ is also $\overset{\blacklozenge}{_G}$-consistent and, hence, that $\overset{\diamond}{_G}$-consistency $\trianglerighteq$ $\overset{\blacklozenge}{_G}$-consistency in this case. If $\overset{\diamond}{_G}(\mathcal{N})$ is trivially inconsistent, then due to the closure under $\overset{\diamond}{_G}$-consistency there is no triangle in G containing both an edge $\{v, v'\}$ such that $C(v, v') = \emptyset$ and an edge $\{u, u'\}$ such that $C(u, u') \neq \emptyset$; we can prove that $\overset{\diamond}{_G}$-consistency $\trianglerighteq$ $\overset{\blacklozenge}{_G}$-consistency in this case as well, by isolating the trivial inconsistencies and using the first part of the proof. Finally, by Proposition 11 we have that $\overset{\blacklozenge}{_G}$-consistency $\trianglerighteq$ $\overset{\diamond}{_G}$-consistency in all cases. ◀

It is interesting to note that Proposition 13 is a more general result that the respective one of [30], namely, Proposition 7 in that work. In particular, Proposition 7 in [30] requires a chordal supergraph of the constraint graph of a QCN over a distributive subclass of relations of a relation algebra to be used, along with the property that any such QCN that is $\diamond$-consistent and not trivially inconsistent is minimal, in order to prove the equivalence between $\overset{\diamond}{_G}$-consistency and $\overset{\blacklozenge}{_G}$-consistency for distributive subclasses of relations.

The following result shows the connection between $\overset{\diamond}{_G}$-consistency and minimal QCNs:

▶ **Proposition 14** ([21])**.** *Let $\mathcal{A}$ be a distributive subclass of relations of a relation algebra with the property that any atomic* QCN *over $\mathcal{A}$ that is $\diamond$-consistent is satisfiable. Then, for any* QCN *$\mathcal{N} = (V, C)$ over $\mathcal{A}$ and any chordal graph $G = (V, E)$ such that* $\mathsf{G}(\mathcal{N}) \subseteq G$*, we have that $\forall \{u, v\} \in E$ and $\forall b \in C'(u, v)$, where $(V, C') = \overset{\diamond}{_G}(\mathcal{N})$, the base relation b is feasible.*

The property described in Proposition 14 is satisfied by all of the qualitative constraint languages mentioned in Proposition 1 [10].

Finally, the following result shows the connection between $\overset{\blacklozenge}{_G}$-consistency and minimal QCNs:

▶ **Proposition 15** ([2])**.** *Let $\mathcal{A}$ be a subclass of relations of a relation algebra with the property that for any* QCN *$\mathcal{N} = (V, C)$ over $\mathcal{A}$ there exists a graph $G = (V, E)$ such that, if $\overset{\diamond}{_G}(\mathcal{N})$ is not trivially inconsistent, then $\mathcal{N}$ is satisfiable. Then, for any such $\mathcal{N}$ and G, we have that $\forall \{u, v\} \in E$ and $\forall b \in C'(u, v)$, where $(V, C') = \overset{\blacklozenge}{_G}(\mathcal{N})$, the base relation b is feasible.*

As a note, an interesting case where the property described in Proposition 15 can be satisfied, is the case where the considered subclass of relations is obtained from a relation algebra that has *patchwork* [22] for $\overset{\diamond}{_G}$-consistent and not trivially inconsistent QCNs over that subclass, where $G = (V, E)$ is any chordal graph such that $\mathsf{G}(\mathcal{N}) \subseteq G$ for a given QCN $\mathcal{N} = (V, C)$. In that case, we will indeed have that $\mathcal{N}$ is satisfiable if $\overset{\diamond}{_G}(\mathcal{N})$ is not trivially inconsistent [2]. As a matter of fact, patchwork holds for all the qualitative constraint languages mentioned in Proposition 1 [14]. Of course, in general, the property may be satisfied in other cases as well; for instance, patchwork may not hold, but the overall property may hold for complete graphs (and, hence, when $\diamond$-consistency is used) or when constraints in the structure of the constraint graphs of the QCNs are imposed (a trivial case being restricting the constraint graphs of QCNs to being trees).

4 $\blacklozenge_G^{\cup}$-Consistency: a new local consistency for QCNs

We define a new local consistency for QCNs inspired by k-partitioning consistency for constraint satisfaction problems (CSPs), where arc consistency is used as the underlying local consistency of choice, or *k-Partition*-AC for short [4]. This technique divides a variable domain into disjoint domains, where each of them contains at most k elements. In the case of QCNs, these elements correspond to base relations. With respect to k-Partition-AC, the most common and preferred approach is dividing a domain into singleton sub-domains, which is the case where $k = 1$, otherwise many questions arise, such as what should the size of each sub-domain be, how should this size be fixed, and which elements should be considered for a given use case. Although having many questions to deal with is not necessarily bad in general, the most important aspect regarding 1-Partition-AC is that it offers the nice property that it is strictly stronger than singleton arc consistency (SAC) [9].

In this work, we adapt the aforementioned technique to QCNs using $\diamond_G$-consistency as our underlying local consistency of choice.[1] Given a QCN $\mathcal{N}$, enforcing this consistency for $k = 1$ will eliminate every base relation that is not $\blacklozenge_G$-consistent for some constraint in $\mathcal{N}$, but also every base relation that is not *supported* by some base relation in $\mathcal{N}$ through $\diamond_G$-consistency. We call this new local consistency $\blacklozenge_G^{\cup}$-consistency, and better explain it with a demonstrative example as follows. Consider the $\blacklozenge_G$-consistent QCN $\mathcal{N} = (V, C)$ of IA in Figure 3. We can see that the base relation d is $\blacklozenge_G$-consistent for $C(x_1, x_2)$, but it is not supported by any of the base relations that define constraint $C(x_1, x_3)$, namely, p and pi, through $\diamond_G$-consistency. In particular, by instatiating $C(x_1, x_3)$ with either p or pi and closing the respective QCN under $\diamond_G$-consistency, the base relation d is eliminated in $C(x_1, x_2)$. After eliminating the base relation d in $C(x_1, x_2)$, the revised QCN $\mathcal{N}$ becomes $\blacklozenge_G^{\cup}$-consistent.

Now we can formally define this consistency.

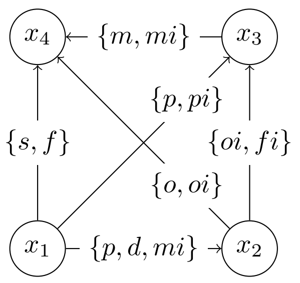
▶ **Definition 16.** Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, $\mathcal{N}$ is said to be $\blacklozenge_G^{\cup}$-consistent iff $\mathcal{N}$ is $\blacklozenge_G$-consistent and $\forall\{v, v'\} \in E$, $\forall b \in C(v, v')$, and $\forall\{u, u'\} \in E$ we have that $\exists b' \in C(u, u')$ such that $b \in C'(v, v')$, where $(V, C') = \diamond_G(\mathcal{N}_{[u,u']/\{b'\}})$.

We prove the following result to be used in the sequel, which suggests that $\blacklozenge_G^{\cup}$-consistency can only eliminate unfeasible base relations:

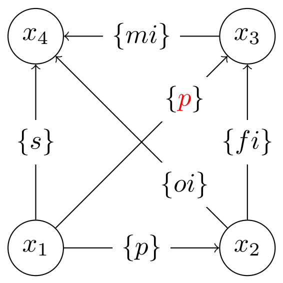
▶ **Proposition 17.** *Let $\mathcal{N} = (V, C)$ be a QCN, $G = (V, E)$ a graph, and $b \in C(u, u')$ with $u, u' \in V$ a base relation. Then, if $\exists\{v, v'\} \in E$ such that $b \notin C'(u, u')$, where $(V, C') = \bigcup\{\diamond_G(\mathcal{N}_{[v,v']/\{b'\}}) \mid b' \in C(v, v')\}$, we have that b is an unfeasible base relation.*

Proof. Let us assume that b is a feasible base relation. Then, by definition of feasible base relations there exists a scenario $\mathcal{S} = (V, C')$ of $\mathcal{N}$ such that $C'(u, u') = \{b\}$. Further, by the equivalence property of $\diamond_G$-consistency it holds that $\diamond_G(\mathcal{S}) = \mathcal{S}$ (as $\mathcal{S}$, being a scenario, is an

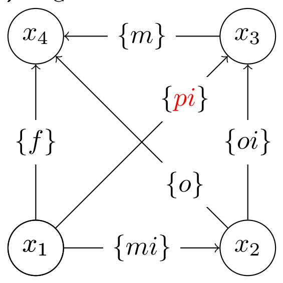
[1] The partitioning scheme can be combined with any local consistency or propagation technique. Here, the definition is restricted to $\diamond_G$-consistency as it is the most essential of local consistencies used for dealing with QCNs.



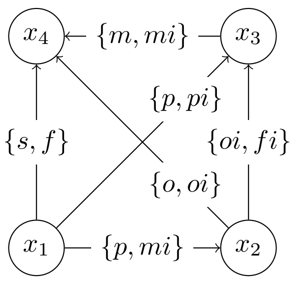
(a) A $\blacklozenge_G$-consistent QCN $\mathcal{N}$.



(b) $\mathcal{N}_1 = \lozenge_G(\mathcal{N}_{[x_1,x_3]/\{p\}})$



(c) $\mathcal{N}_2 = \lozenge_G(\mathcal{N}_{[x_1,x_3]/\{pi\}})$



(d) $\mathcal{N}_1 \cup \mathcal{N}_2$

Figure 3 A $\blacklozenge_G$-consistent QCN $\mathcal{N}$ of IA along with a demonstration of how enforcing $\blacklozenge^{\cup}_G$-consistency can further eliminate base relations; here G is the complete graph on the set of variables of $\mathcal{N}$.

atomic and satisfiable QCN and, hence, none of its base relations can be removed by application of $\lozenge_G$-consistency). Thus, it follows that $\forall\{v, v'\} \in E$ we have that $b \in C''(u, u')$, where $(V, C'') = \lozenge_G(\mathcal{N}_{[v,v']/C'(v,v')})$, as $\mathcal{S} \subseteq \mathcal{N}_{[v,v']/C'(v,v')}$ and, hence, $\lozenge_G(\mathcal{S}) \subseteq \lozenge_G(\mathcal{N}_{[v,v']/C'(v,v')})$ by the monotonicity property of $\lozenge_G$-consistency. As $\mathcal{S} \subseteq \mathcal{N}$, it follows that $\forall\{v, v'\} \in E$ we have that $C'(v, v') \subseteq C(v, v')$ and, hence, that $\exists b' \in C(v, v')$ such that $b \in C'''(u, u')$, where $(V, C''') = \lozenge_G(\mathcal{N}_{[v,v']/\{b'\}})$, by simply considering the base relation $b' \in C(v, v')$ to be the one of the singleton relation $C'(v, v')$ of $\mathcal{S}$. Therefore, by definition of operation $\cup$ with respect to QCNs we can derive that $\forall\{v, v'\} \in E$ it holds that $b \in C^*(u, u')$, where $(V, C^*) = \bigcup\{\lozenge_G(\mathcal{N}_{[v,v']/\{b'\}}) \mid b' \in C(v, v')\}$, which concludes our proof by contraposition. ◀

We recall the following result to be used in one of our proofs later on:

▶ **Proposition 18** ([2]). *For any* QCN*s $\mathcal{N}_1$ and $\mathcal{N}_2$ on a set of variables V and any graph $G = (V, E)$, if $\mathcal{N}_1$ and $\mathcal{N}_2$ are $\blacklozenge_G$-consistent, then $(\mathcal{N}_1 \cup \mathcal{N}_2)$ is $\blacklozenge_G$-consistent as well.*

We note that the aforementioned result describes a sufficient property for proving dominance for a new consistency, but that property might not be necessary in general and, hence, does not solely follow from the well-behaveness of the consistency at hand. We prove the same property for $\blacklozenge^{\cup}_G$-consistency, to be used in what follows.

▶ **Proposition 19.** *For any* QCN*s $\mathcal{N}_1$ and $\mathcal{N}_2$ on a set of variables V and any graph $G = (V, E)$, if $\mathcal{N}_1$ and $\mathcal{N}_2$ are $\blacklozenge^{\cup}_G$-consistent, then $(\mathcal{N}_1 \cup \mathcal{N}_2)$ is $\blacklozenge^{\cup}_G$-consistent as well.*

Proof. Let $\mathcal{N}_1 = (V, C_1)$, $\mathcal{N}_2 = (V, C_2)$, $(\mathcal{N}_1 \cup \mathcal{N}_2) = (V, C)$, $v, v' \in V$ be two variables, and $b \in C(v, v')$ a base relation. We only need to consider the case where $b \in C_1(v, v')$, as the case where $b \in C_2(v, v')$ is symmetric. Since $\mathcal{N}_1$ is $\blacklozenge^{\cup}_G$-consistent, we have that $\mathcal{N}_1$ is $\blacklozenge_G$-consistent and $\forall\{u, u'\} \in E$ there exists $b' \in C_1(u, u')$ such that $b \in C'_1(v, v')$, where $(V, C'_1) = \lozenge_G({\mathcal{N}_1}_{[u,u']/\{b'\}})$, by definition of $\blacklozenge^{\cup}_G$-consistency. In addition, we have that

$(\mathcal{N}_1 \cup \mathcal{N}_2)$ is $\underset{G}{\blacklozenge}$-consistent by Proposition 18. As $\mathcal{N}_1 \subseteq (\mathcal{N}_1 \cup \mathcal{N}_2)$, we have that $\mathcal{N}_{1[u,u']/\{b'\}} \subseteq (\mathcal{N}_1 \cup \mathcal{N}_2)_{[u,u']/\{b'\}}$ $\forall\{u,u'\} \in E$ and $\forall b' \in C_1(u,u')$. Thus, we have that $\underset{G}{\lozenge}(\mathcal{N}_{1[u,u']/\{b'\}}) \subseteq \underset{G}{\lozenge}((\mathcal{N}_1 \cup \mathcal{N}_2)_{[u,u']/\{b'\}})$ $\forall\{u,u'\} \in E$ and $\forall b' \in C_1(u,u')$ by the monotonicity property of $\underset{G}{\lozenge}$-consistency. From that we can deduce that $\forall\{u,u'\} \in E$ there exists $b' \in C(u,u')$ such that $b \in C'(v,v')$, where $(V,C') = \underset{G}{\lozenge}((\mathcal{N}_1 \cup \mathcal{N}_2)_{[u,u']/\{b'\}})$. Hence, by the assumption that $\mathcal{N}_1$ and $\mathcal{N}_2$ are $\underset{G}{\blacklozenge}^{\cup}$-consistent, we have proved that $(\mathcal{N}_1 \cup \mathcal{N}_2)$ is $\underset{G}{\blacklozenge}^{\cup}$-consistent as well. ◀

Next, we arrive to one of our main results in this work.

▶ **Theorem 20.** *We have that $\underset{G}{\blacklozenge}^{\cup}$-consistency is well-behaved.*

Proof. (Dominance) From Proposition 19 we can assert that, for any QCN $\mathcal{N} = (V,C)$ and any graph $G = (V,E)$, there exists a unique $\underset{G}{\blacklozenge}^{\cup}$-consistent QCN $\bigcup\{\mathcal{N}' \mid \mathcal{N}' \subseteq \mathcal{N}$ and $\mathcal{N}'$ is $\underset{G}{\blacklozenge}^{\cup}$-consistent$\}$, which by its definition is the largest (w.r.t. $\subseteq$) $\underset{G}{\blacklozenge}^{\cup}$-consistent sub-QCN of $\mathcal{N}$ and, hence, the closure of $\mathcal{N}$ under $\underset{G}{\blacklozenge}^{\cup}$-consistency. (Equivalence) Let $\mathcal{N} = (V,C)$ be a QCN, $G = (V,E)$ a graph, and $\mathcal{N}' = (V,C')$ the QCN where $\forall v,v' \in V$ and $\forall b \in \mathsf{B}$ we have that $b \in C'(v,v')$ iff there exists a solution σ of $\mathcal{N}$ such that $(\sigma(v),\sigma(v')) \in b$. Clearly, $\mathcal{N}'$ is a sub-QCN of $\mathcal{N}$ and it is necessarily $\underset{K_V}{\blacklozenge}^{\cup}$-consistent (where K_V denotes the complete graph on the set of variables V of $\mathcal{N}$), as by Proposition 17 we have that the application of $\underset{G}{\blacklozenge}^{\cup}$-consistency on any QCN (V,C) w.r.t. any graph $G = (V,E)$ can only remove unfeasible base relations, and not feasible ones. It follows that $\mathcal{N}' \subseteq \underset{G}{\blacklozenge}^{\cup}(\mathcal{N}) \subseteq \mathcal{N}$ and, as such, $\underset{G}{\blacklozenge}^{\cup}(\mathcal{N})$ and $\mathcal{N}$ share the same set of solutions. (Idempotence) Let $\mathcal{N} = (V,C)$ be a QCN, and $G = (V,E)$ a graph. Then, $\underset{G}{\blacklozenge}^{\cup}(\mathcal{N})$ is $\underset{G}{\blacklozenge}^{\cup}$-consistent. Now, by dominance of $\underset{G}{\blacklozenge}^{\cup}$-consistency the largest $\underset{G}{\blacklozenge}^{\cup}$-consistent sub-QCN of $\underset{G}{\blacklozenge}^{\cup}(\mathcal{N})$ is itself and, hence, $\underset{G}{\blacklozenge}^{\cup}(\underset{G}{\blacklozenge}^{\cup}(\mathcal{N})) = \underset{G}{\blacklozenge}^{\cup}(\mathcal{N})$. (Monotonicity) Let $\mathcal{N} = (V,C)$ and $\mathcal{N}' = (V,C')$ be two QCNs such that $\mathcal{N}' \subseteq \mathcal{N}$, and $G = (V,E)$ a graph. As $\mathcal{N}' \subseteq \mathcal{N}$, we have that $\underset{G}{\blacklozenge}^{\cup}(\mathcal{N}')$ is a $\underset{G}{\blacklozenge}^{\cup}$-consistent sub-QCN of $\mathcal{N}$. In addition, by dominance of $\underset{G}{\blacklozenge}^{\cup}$-consistency we can assert that $\underset{G}{\blacklozenge}^{\cup}(\mathcal{N})$ is the largest $\underset{G}{\blacklozenge}^{\cup}$-consistent sub-QCN of $\mathcal{N}$. Therefore, we have that $\underset{G}{\blacklozenge}^{\cup}(\mathcal{N}') \subseteq \underset{G}{\blacklozenge}^{\cup}(\mathcal{N})$. ◀

We prove the following general result regarding the pruning capability of $\underset{G}{\blacklozenge}$-consistency in comparison with that of $\underset{G}{\blacklozenge}^{\cup}$-consistency:

▶ **Proposition 21.** *We have that $\underset{G}{\blacklozenge}^{\cup}$-consistency $\triangleright$ $\underset{G}{\blacklozenge}$-consistency.*

Proof. By definition of $\underset{G}{\blacklozenge}^{\cup}$-consistency, we have that $\underset{G}{\blacklozenge}^{\cup}$-consistency $\trianglerighteq$ $\underset{G}{\blacklozenge}$-consistency, since, for any graph $G = (V,E)$, any QCN (V,C) that is $\underset{G}{\blacklozenge}^{\cup}$-consistent is already $\underset{G}{\blacklozenge}$-consistent. To prove strictness we use an example as follows. Consider the QCN $\mathcal{N} = (V,C)$ of Figure 3. The reader can verify that $\mathcal{N}$ is $\underset{G}{\blacklozenge}$-consistent, as we have that b is $\underset{G}{\blacklozenge}$-consistent for $C(v,v')$ $\forall\{v,v'\} \in E$ and $\forall b \in C(v,v')$. However, we have that $d \notin C'(x_1,x_2)$, where $(V,C') = \bigcup\{\underset{G}{\lozenge}(\mathcal{N}_{[x_1,x_3]/\{b'\}}) \mid b' \in C(x_1,x_3)\}$, as demonstrated in the figure. In detail, $\underset{G}{\lozenge}(\mathcal{N}_{[x_1,x_3]/\{p\}}) \cup \underset{G}{\lozenge}(\mathcal{N}_{[x_1,x_3]/\{pi\}})$ is a QCN such that d is not among the base relations that define the constraint on variables x_1 and x_2. Thus, $\underset{G}{\blacklozenge}^{\cup}$-consistency does not hold in $\mathcal{N}$. ◀

The next result follows trivially:

▶ **Proposition 22.** *We have that $\underset{G}{\blacklozenge}^{\cup}$-consistency $\triangleright$ $\underset{G}{\lozenge}$-consistency.*

Proof. A direct consequence of Propositions 11 and 21 and the transitivity of $\triangleright$. ◀

Finally, we introduce the following result that identifies the case where $\underset{G}{\blacklozenge}$-consistency and $\underset{G}{\blacklozenge}^{\cup}$-consistency are equivalent:

Algorithm 1: PSWC$^{\cup}(\mathcal{N}, G)$

```
in   : A QCN $\mathcal{N} = (V, C)$, and a graph $G = (V, E)$.
out  : A sub-QCN of $\mathcal{N}$.
1  begin
2      $\mathcal{N} \leftarrow$ PWC$(\mathcal{N}, G)$;
3      $Q \leftarrow E$;
4      while $Q \neq \emptyset$ do
5          $\{v, v'\} \leftarrow Q.pop()$;
6          $(V, C') \leftarrow \bot^V$;
7          foreach $b \in C(v, v')$ do
8              $(V, C') \leftarrow (V, C') \cup$ PWC$(\mathcal{N}_{[v,v']/\{b\}}, G, \{\{v, v'\}\})$;
9          if $(V, C') \subset \mathcal{N}$ then
10             foreach $\{u, u'\} \in E \mid C'(u, u') \subset C(u, u')$ do
11                 $C(u, u') \leftarrow C'(u, u')$;
12                 $C(u', u) \leftarrow C'(u', u)$;
13             $Q \leftarrow E$;
14     return $\mathcal{N}$;
```

▶ **Proposition 23.** *Let $\mathcal{A}$ be a subclass of relations of a relation algebra with the property that for any* QCN $\mathcal{N} = (V, C)$ *over $\mathcal{A}$ there exists a graph $G = (V, E)$ such that, if $\underset{G}{\lozenge}(\mathcal{N})$ is not trivially inconsistent, then $\mathcal{N}$ is satisfiable. Then, for any such $\mathcal{N}$ and G, we have that $\underset{G}{\blacklozenge}$-consistency $\equiv$ $\underset{G}{\blacklozenge}^{\cup}$-consistency.*

Proof. We first prove that, if $\mathcal{N}$ is $\underset{G}{\blacklozenge}$-consistent, then $\mathcal{N}$ is also $\underset{G}{\blacklozenge}^{\cup}$-consistent. By Proposition 15 we have that $\forall\{u, v\} \in E$ and $\forall b \in C(u, v)$ the base relation b is feasible. In addition, by the equivalence property of $\underset{G}{\blacklozenge}^{\cup}$-consistency we have that the application of $\underset{G}{\blacklozenge}^{\cup}$-consistency on $\mathcal{N}$ can only remove unfeasible base relations and, hence, that $\underset{G}{\blacklozenge}^{\cup}(\mathcal{N}) = \mathcal{N}$, as every base relation $b \in C(u, v)$ $\forall\{u, v\} \in E$ is feasible. The proof that, if $\mathcal{N}$ is $\underset{G}{\blacklozenge}^{\cup}$-consistent, then $\mathcal{N}$ is also $\underset{G}{\blacklozenge}$-consistent, follows directly from the definition of $\underset{G}{\blacklozenge}^{\cup}$-consistency. ◀

A hasty reading of Proposition 23 might give the impression that one should always opt to apply $\underset{G}{\blacklozenge}$-consistency for the cases where the considered QCN and the graph G satisfy the prerequisites detailed in that proposition, as $\underset{G}{\blacklozenge}$-consistency, being a weaker consistency than $\underset{G}{\blacklozenge}^{\cup}$-consistency in general, should be "easier" to apply. However, as we will demonstrate in our experimental section, $\underset{G}{\blacklozenge}^{\cup}$-consistency is faster to apply. To give an intuition, any well-structured algorithm that will try to enforce $\underset{G}{\blacklozenge}^{\cup}$-consistency in a given QCN for some graph G, will inescapably make better use of the singleton checks than the respective algorithm for enforcing $\underset{G}{\blacklozenge}$-consistency. This is because the former algorithm will exploit the singleton checks (by the very definition of $\underset{G}{\blacklozenge}^{\cup}$-consistency) to *proactively* eliminate certain base relations that are unfeasible and, hence, possibly not $\underset{G}{\blacklozenge}$-consistent for the corresponding constraints.

5 An algorithm for achieving $\underset{G}{\blacklozenge}^{\cup}$-consistency

In this section, we propose an algorithm for efficiently applying $\underset{G}{\blacklozenge}^{\cup}$-consistency on a given QCN $\mathcal{N}$, called PSWC$^{\cup}$ ($\cup$-*collective* partial singleton closure under weak composition) and presented in Algorithm 1. This algorithm builds on the algorithm for efficiently achieving $\underset{G}{\blacklozenge}$-consistency, called PSWC (partial singleton closure under weak composition) and presented in Algorithm 2, which in itself is an advancement of the respective algorithm for enforcing

Algorithm 2: PSWC($\mathcal{N}, G$)

```
in    : A QCN $\mathcal{N} = (V, C)$, and a graph $G = (V, E)$.
out   : A sub-QCN of $\mathcal{N}$.
1  begin
2      $\mathcal{N} \leftarrow$ PWC($\mathcal{N}, G$);
3      $Q \leftarrow E$;
4      while $Q \neq \emptyset$ do
5          $\{v, v'\} \leftarrow Q.pop()$;
6          $(V, C') \leftarrow \bot^V$;
7          foreach $b \in C(v, v')$ do
8              $(V, C') \leftarrow (V, C') \cup$ PWC($\mathcal{N}_{[v,v']/\{b\}}, G, \{\{v, v'\}\}$);
9          if $C'(v, v') \subset C(v, v')$ then
10             $C(v, v') \leftarrow C'(v, v')$;
11             $C(v', v) \leftarrow C'(v', v)$;
12             $Q \leftarrow E$;
13     return $\mathcal{N}$;
```

Algorithm 3: PWC($\mathcal{N}, G, e \leftarrow \emptyset$)

```
in    : A QCN $\mathcal{N} = (V, C)$, a graph $G = (V, E)$, and optionally a set $e$ such that $e \subseteq E$.
out   : A sub-QCN of $\mathcal{N}$.
1  begin
2      $Q \leftarrow$ ($e$ if $e \neq \emptyset$ else $E$);
3      while $Q \neq \emptyset$ do
4          $\{v, v'\} \leftarrow Q.pop()$;
5          foreach $v'' \in V \mid \{v, v''\}, \{v', v''\} \in E$ do
6              $r \leftarrow C(v, v'') \cap (C(v, v') \diamond C(v', v''))$;
7              if $r \subset C(v, v'')$ then
8                  $C(v, v'') \leftarrow r$;
9                  $C(v'', v) \leftarrow r^{-1}$;
10                 $Q \leftarrow Q \cup \{\{v, v''\}\}$;
11             $r \leftarrow C(v'', v') \cap (C(v'', v) \diamond C(v, v'))$;
12             if $r \subset C(v'', v')$ then
13                 $C(v'', v') \leftarrow r$;
14                 $C(v', v'') \leftarrow r^{-1}$;
15                 $Q \leftarrow Q \cup \{\{v'', v'\}\}$;
16     return $\mathcal{N}$;
```

$\blacklozenge_G$-consistency that is presented in [2]; we explain as follows. We use a queue in both of our algorithms that is initialized with all of the edges of a given graph G that correspond to constraints of a given QCN $\mathcal{N}$. In addition, this queue is filled with all of the aforementioned edges whenever any of the constraints of $\mathcal{N}$ is revised, i.e., whenever a base relation is removed. This operation is equivalent to introducing a break statement in the algorithm of [2] whenever a singleton check fails and, hence, a constraint is revised, forcing the inner loop in that algorithm to stop and using the outer loop to initiate singleton checks in a fresh QCN. We have found this tactic to perform much better in practice, cutting down on the number of constraint checks by around 20%. Further, the use of a queue allows for prioritizing certain edges, a strategy which is in line with similar techniques used in the algorithm for enforcing

$\diamond_G$-consistency [34, 27, 16], but this is something that we have not yet explored and retain for future work. As we will also remind the reader in the experimental evaluation to follow, we use a simple FIFO (first-in, first-out) queue for our algorithms. For the sake of completeness, we also present the state-of-the-art algorithm for applying $\diamond_G$-consistency on a given QCN, called PWC (partial closure under weak composition), which is utilized as a subroutine by both PSWC$^{\cup}$ and PSWC (see Algorithm 3).

The difference between algorithms PSWC$^{\cup}$ and PSWC lies solely in the way that they exploit singleton checks. In particular, note the difference between the conditions in line 9 of both algorithms; PSWC$^{\cup}$ will bring up all edges in the queue for revising the entire QCN even when the constraint at hand was not revised, but another constraint somewhere in the QCN was, whereas PSWC will keep its focus solely on the constraint at hand. This is due to the fact that algorithm PSWC$^{\cup}$ will use a single singleton check to eliminate base relations anywhere in the network, and not just in the constraint at hand as algorithm PSWC does. Before proving the correctness of algorithm PSWC$^{\cup}$, we recall the following result regarding the correctness of algorithm PSWC:

▶ **Proposition 24** (cf. [2, 8]). *Given a* QCN $\mathcal{N} = (V, C)$ *of a relation algebra and a graph* $G = (V, E)$*, we have that algorithm* PSWC *terminates and returns* $\blacklozenge_G(\mathcal{N})$.

Now, we prove that algorithm PSWC$^{\cup}$ is complete for applying $\blacklozenge_G^{\cup}$-consistency on a given $\mathcal{N} = (V, C)$ for a given graph $G = (V, E)$. Due to space limitations, an intuitive proof is provided, which however manages to explain the overall functionallity of algorithm PSWC$^{\cup}$ in sufficient detail.

▶ **Theorem 25.** *Given a* QCN $\mathcal{N} = (V, C)$ *of a relation algebra and a graph* $G = (V, E)$*, we have that algorithm* PSWC$^{\cup}$ *terminates and returns* $\blacklozenge_G^{\cup}(\mathcal{N})$.

Proof (Intuition). It is easy to see that lines 9–12 in Algorithm 1 perform a superset of the operations performed in lines 9–11 in Algorithm 2. Thus, by Proposition 24 we know that given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, algorithm PSWC$^{\cup}$ applies the set of operations required to make $\mathcal{N}$ $\blacklozenge_G$-consistent. We need to show that the rest of the operations maintain $\blacklozenge_G$-consistency and further achieve $\blacklozenge_G^{\cup}$-consistency. With respect to that, it is again easy to see that algorithm PSWC$^{\cup}$ enforces exactly the conditions specified in Proposition 17 and, hence, removes the (unfeasible) base relations required to make $\mathcal{N}$ $\blacklozenge_G^{\cup}$-consistent. Further, since the algorithm will only terminate when b is guaranteed to have become $\blacklozenge_G$-consistent for $C(u, v)$ $\forall\{u, v\} \in E$ and $\forall b \in C(u, v)$ and no constraint is further revised to additionally achieve $\blacklozenge_G^{\cup}$-consistency, we can conclude that algorithm PSWC$^{\cup}$ correctly applies $\blacklozenge_G^{\cup}$-consistency on $\mathcal{N}$. ◀

Time complexity analysis

Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, we have that algorithm PSWC$^{\cup}$ applies $\blacklozenge_G^{\cup}$-consistency on $\mathcal{N}$ in $O(\Delta \cdot |E|^3 \cdot \mathsf{B}^3)$ time, where Δ is the maximum vertex degree of graph G. In particular, algorithm PWC is executed $O(|E| \cdot |B|)$ times every time a constraint is revised, and such a constraint revision can occur $O(|E| \cdot |B|)$ times. Further, we note that the unification operations that take place in line 8 of the algorithm are handled in $O(|E| \cdot |B|)$ time in total, as we keep track of the constraints that are revised by algorithm PWC and we can have a total of $O(|E| \cdot |B|)$ constraint revisions. The same argument holds for the operations that take place in lines 9–12 of the alorithm. (These details are not included in the algorithm to allow for a more compact representation.) Now, by taking into account the worst-case time complexity of algorithm PWC, which is $O(\Delta \cdot |E| \cdot \mathsf{B})$ [7], a worst-case

Table 1 Evaluation with random IA networks of model $\mathsf{S}(\mathsf{n} = 70, \mathsf{l} = 6.5, \mathsf{d})$.

	min		μ		max		σ	
d	PSWC	PSWC$^{\cup}$	PSWC	PSWC$^{\cup}$	PSWC	PSWC$^{\cup}$	PSWC	PSWC$^{\cup}$
10	$\frac{3.19s}{9k}$	$\frac{3.11s}{9k}$	$\frac{4.25s}{14k}$	$\frac{4.02s}{13k}$	$\frac{5.63s}{22k}$	$\frac{5.43s}{21k}$	$\frac{0.50s}{2k}$	$\frac{0.52s}{2k}$
12	$\frac{5.79s}{9k}$	$\frac{5.65s}{9k}$	$\frac{7.82s}{14k}$	$\frac{7.51s}{14k}$	$\frac{11.85s}{22k}$	$\frac{10.88s}{22k}$	$\frac{1.38s}{2k}$	$\frac{1.18s}{2k}$
14	$\frac{9.92s}{12k}$	$\frac{9.63s}{12k}$	$\frac{13.81s}{19k}$	$\frac{13.08s}{18k}$	$\frac{21.26s}{38k}$	$\frac{19.24s}{28k}$	$\frac{2.68s}{5k}$	$\frac{2.40s}{4k}$
16	$\frac{14.14s}{18k}$	$\frac{13.91s}{18k}$	$\frac{30.19s}{31k}$	$\frac{27.74s}{29k}$	$\frac{96.24s}{61k}$	$\frac{99.56s}{62k}$	$\frac{14.20s}{8k}$	$\frac{14.95s}{8k}$
18	$\frac{21.39s}{26k}$	$\frac{20.94s}{26k}$	$\frac{56.16s}{45k}$	$\frac{53.17s}{44k}$	$\frac{154.14s}{88k}$	$\frac{149.98s}{88k}$	$\frac{22.92s}{13k}$	$\frac{21.22s}{12k}$
20	$\frac{54.42s}{39k}$	$\frac{52.82s}{36k}$	$\frac{100.22s}{58k}$	$\frac{89.32s}{55k}$	$\frac{192.68s}{108k}$	$\frac{188.54s}{100k}$	$\frac{30.68s}{14k}$	$\frac{26.08s}{12k}$
22	$\frac{19.66s}{48k}$	$\frac{17.09s}{46k}$	$\frac{42.51s}{67k}$	$\frac{39.83s}{63k}$	$\frac{86.09s}{108k}$	$\frac{75.55s}{108k}$	$\frac{16.67s}{14k}$	$\frac{15.19s}{14k}$

time complexity of $O(\Delta \cdot |E|^3 \cdot \mathsf{B}^3)$ can be obtained for algorithm PSWC$^{\cup}$; this is also the worst-case time complexity of algorithm PSWC [2]. It is important to note that we cannot utilize the incremental functionality of algorithm PWC (see Theorem 1 in [12, Section 3] and the surrounding text) to obtain a better bound for our algorithm, as the singleton checks are perfomed independently[2] of one another.

6 Experimental evaluation

We evaluated the performance of an implementation of algorithm PSWC$^{\cup}$, against an implementation of the algorithm for enforcing partial $\diamond_G$-consistency that was presented here, namely, PSWC, with a varied dataset of arbitrary QCNs of IA.

Technical specifications. The evaluation was carried out on a computer with an Intel Core i5-6200U processor (which has a max frequency of 2.7 GHz per CPU core under turbo mode[3]), 8 GB of RAM, and the Xenial Xerus x86_64 OS (Ubuntu Linux). All algorithms were coded in Python and run using the PyPy intepreter under version 5.1.2, which implements Python 2.7.10; the code is available upon request. Only one CPU core was used.

Datasets and measures. We considered random IA networks generated by the $\mathsf{S}(\mathsf{n}, \mathsf{l}, \mathsf{d})$ model [34]. This model can randomly generate satisfiable QCNs of n variables with an average number l of base relations per non-universal constraint and an average degree d for the corresponding constraint graphs. Further, this model is typically used in the evaluation of algorithms dealing with problems associated with QCNs, with an emphasis on the minimal labelling problem [34, 2, 23]. We generated 30 QCNs of IA of $\mathsf{n} = 70$ variables

[2] To be more precise, the unification operations that take place in line 8 of the algorithm do not provide the level of interdependency required to tap into the incrementality of PWC.

[3] Turbo mode was maintained throughout the experimental evaluation by staying well within thermal design power (TDP) limit.

with $\mathsf{l} = |\mathsf{B}|/2 = 6.5$ base relations per non-universal constraint on average for all values of d ranging from 10 to 22 with a step of 2 (a typical range for evaluating related algorithms [2]); hence, we considered a total of 210 QCNs of IA. Finally, the *maximum cardinality search* algorithm [32] was used to obtain a triangulation of the constraint graph of a given QCN. Notice that, with respect to our evaluation, any kind of graphs would have been adequate (even complete ones), as they would have affected all involved algorithms proportionally and would not have qualitatively distorted the obtained results; however, the choice of chordal graphs was more reasonable given their extensive use in the recent literature [31].

Our evaluation involved two measures, which we describe as follows. The first measure considers the number of *constraint checks per base relation removals* performed by an algorithm for enforcing the respective local consistency. Given a QCN $\mathcal{N} = (V, C)$ and three variables $v_i, v_k, v_j \in V$, a constraint check occurs when we compute the relation $r = C(v_i, v_j) \cap (C(v_i, v_k) \diamond C(v_k, v_j))$ and check if $r \subset C(v_i, v_j)$, so that we can propagate its constrainedness if that condition is satisfied. Weak compositions that yield relation B are disregarded. The second measure concerns the CPU time and is strongly correlated with the first one, as the run-time of any proper implementation of an algorithm for enforcing a local consistency should, in principle, rely mainly on the number of constraint checks performed.

Results. The results of our experimental evaluation are detailed in Table 1, where a fraction $\frac{x}{y}$ denotes that an approach required x seconds of CPU time and performed y constraint checks per base relation removals on average per dataset of networks during its operation. In short, the advantage of $\mathsf{PSWC}^{\cup}$ over PSWC is clear across all parameters and for all settings and corresponds to around 10%. This is a promising result in terms of achieving a new stronger local consistency faster than what was possible to date even when considering a weaker local consistency. Further, we recall to the reader that we used a simple FIFO queue for our implementation; it would be interesting to explore prioritizing edges corresponding to constraints that are revised at any given step of the execution. We retain a more thorough experimental evaluation, which will also include the effect of our new stronger local consistency on backtracking-based algorithms, for future work. Here, we opted to maintain a simple configuration for our algorithms in order to obtain a first pure comparison that will serve as a basis for further evaluation.

7 Conclusion and future work

Partial singleton closure under weak composition, or partial $\blacklozenge$-consistency for short, is a local consistency that ensures that each base relation of each of the constraints of a qualitative constraint network can define a singleton relation in the corresponding partial closure of that network under weak composition, or in its corresponding partial $\diamond$-consistent subnetwork for short. This local consistency is essential for approximating satisfiability of qualitative constraints networks, and has been shown to play a crucial role in tackling the minimal labeling problem of a qualitative constraint network in particular, which is the problem of finding the strongest implied constraints of that network. In this paper, we proposed a stronger local consistency that couples $\blacklozenge$-consistency with the idea of collectively deleting certain unfeasible base relations by exploiting singleton checks. Further, we proposed an efficient algorithm for enforcing this new consistency that, given a qualitative constraint network, performs fewer constraint checks than the respective algorithm for enforcing partial $\blacklozenge$-consistency in that network. We formally proved certain properties of our new local consistency, and motivated its usefulness through demonstrative examples and a preliminary experimental evaluation with qualitative constraint networks of Interval Algebra.

There are several directions for future work. Regarding the algorithm that enforces our new consistency, we would like to explore queuing strategies such that the singleton checks are applied in a more fruitful manner. In particular, it would make sense to prioritize certain singleton checks that are more likely to eliminate base relations anywhere in the network at hand, because this could unveil certain inconsistencies faster, but also lead to fewer constraint checks overall. Such strategies have been used in the case of partial $\diamond$-consistency [34, 27, 16]. Further, regarding the new local consistency itself, we would like to define a weaker variant of it that considers singleton checks in the neighborhood of the constraint in question, instead of the entire network. Early experiments in this direction have shown really promising results with respect to constraint satisfaction problems, which is due to the fact that constraint revisions tend to propagate themselves to just neighboring constraints [36].

Acknowledgements. We would like to thank the reviewers for their helpful comments.

References

1 James F. Allen. Maintaining Knowledge about Temporal Intervals. *Commun. ACM*, 26:832–843, 1983. doi:10.1145/182.358434.

2 Nouhad Amaneddine, Jean-François Condotta, and Michael Sioutis. Efficient Approach to Solve the Minimal Labeling Problem of Temporal and Spatial Qualitative Constraints. In *IJCAI*, 2013.

3 Philippe Balbiani, Jean-François Condotta, and Luis Fariñas del Cerro. Tractability Results in the Block Algebra. *J. Log. Comput.*, 12:885–909, 2002. doi:10.1093/logcom/12.5.885.

4 Hachemi Bennaceur and Mohamed-Salah Affane. Partition-k-AC: An Efficient Filtering Technique Combining Domain Partition and Arc Consistency. In *CP*, 2001.

5 Christian Bessière. Arc-Consistency and Arc-Consistency Again. *Artif. Intell.*, 65:179–190, 1994.

6 Mehul Bhatt, Hans W. Guesgen, Stefan Wölfl, and Shyamanta Moni Hazarika. Qualitative Spatial and Temporal Reasoning: Emerging Applications, Trends, and Directions. *Spatial Cognition & Computation*, 11:1–14, 2011. doi:10.1080/13875868.2010.548568.

7 Assef Chmeiss and Jean-François Condotta. Consistency of Triangulated Temporal Qualitative Constraint Networks. In *ICTAI*, 2011. doi:10.1109/ICTAI.2011.125.

8 Jean-François Condotta and Christophe Lecoutre. A Class of df-Consistencies for Qualitative Constraint Networks. In *KR*, 2010.

9 Romuald Debruyne and Christian Bessière. Some Practicable Filtering Techniques for the Constraint Satisfaction Problem. In *IJCAI*, 1997.

10 Frank Dylla, Till Mossakowski, Thomas Schneider, and Diedrich Wolter. Algebraic Properties of Qualitative Spatio-Temporal Calculi. In *COSIT*, 2013. doi:10.1007/978-3-319-01790-7_28.

11 Andrew U. Frank. Qualitative Spatial Reasoning with Cardinal Directions. In *ÖGAI*, 1991.

12 Alfonso Gerevini. Incremental qualitative temporal reasoning: Algorithms for the Point Algebra and the ORD-Horn class. *Artif. Intell.*, 166:37–80, 2005. doi:10.1016/j.artint.2005.04.005.

13 Martin Charles Golumbic and Ron Shamir. Complexity and Algorithms for Reasoning about Time: A Graph-Theoretic Approach. *J. ACM*, 40:1108–1133, 1993. doi:10.1145/174147.169675.

14 Jinbo Huang. Compactness and its implications for qualitative spatial and temporal reasoning. In *KR*, 2012.

15 Jinbo Huang, Jason Jingshi Li, and Jochen Renz. Decomposition and tractability in qualitative spatial and temporal reasoning. *Artif. Intell.*, 195:140–164, 2013. doi:10.1016/j.artint.2012.09.009.

16 Peter B. Ladkin and Alexander Reinefeld. Fast Algebraic Methods for Interval Constraint Problems. *Ann. Math. Artif. Intell.*, 19:383–411, 1997. doi:10.1023/A:1018968024833.
17 Sanjiang Li, Zhiguo Long, Weiming Liu, Matt Duckham, and Alan Both. On redundant topological constraints. *Artif. Intell.*, 225:51–76, 2015. doi:10.1016/j.artint.2015.03.010.
18 Gérard Ligozat. Reasoning about cardinal directions. *J. Vis. Lang. Comput.*, 9:23–44, 1998. doi:10.1006/jvlc.1997.9999.
19 Gérard Ligozat and Jochen Renz. What Is a Qualitative Calculus? A General Framework. In *PRICAI*, 2004. doi:10.1007/978-3-540-28633-2_8.
20 Zhiguo Long and Sanjiang Li. On Distributive Subalgebras of Qualitative Spatial and Temporal Calculi. In *COSIT*, 2015. doi:10.1007/978-3-319-23374-1_17.
21 Zhiguo Long, Michael Sioutis, and Sanjiang Li. Efficient Path Consistency Algorithm for Large Qualitative Constraint Networks. In *IJCAI*, 2016.
22 Carsten Lutz and Maja Milicic. A Tableau Algorithm for DLs with Concrete Domains and GCIs. *J. Autom. Reasoning*, 38:227–259, 2007.
23 Bernhard Nebel. Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class. *Constraints*, 1:175–190, 1997. doi:10.1007/BF00137869.
24 David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions & Connection. In *KR*, 1992.
25 Jochen Renz. A Canonical Model of the Region Connection Calculus. *J. Appl. Non-Classical Logics*, 12:469–494, 2002.
26 Jochen Renz and Gérard Ligozat. Weak Composition for Qualitative Spatial and Temporal Reasoning. In *CP*, 2005. doi:10.1007/11564751_40.
27 Jochen Renz and Bernhard Nebel. Efficient Methods for Qualitative Spatial Reasoning. *J. Artif. Intell. Res.*, 15:289–318, 2001. doi:10.1613/jair.872.
28 Jochen Renz and Bernhard Nebel. Qualitative Spatial Reasoning Using Constraint Calculi. In *Handbook of Spatial Logics*, pages 161–215. Springer, 2007. doi:10.1007/978-1-4020-5587-4_4.
29 Michael Sioutis, Jean-François Condotta, and Manolis Koubarakis. An Efficient Approach for Tackling Large Real World Qualitative Spatial Networks. *Int. J. Artif. Intell. Tools*, 25:1–33, 2016.
30 Michael Sioutis, Sanjiang Li, and Jean-François Condotta. Efficiently Characterizing Non-Redundant Constraints in Large Real World Qualitative Spatial Networks. In *IJCAI*, 2015.
31 Michael Sioutis, Yakoub Salhi, and Jean-François Condotta. Studying the use and effect of graph decomposition in qualitative spatial and temporal reasoning. *Knowl. Eng. Rev.*, 32:e4, 2016.
32 Robert E. Tarjan and Mihalis Yannakakis. Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. *SIAM J. Comput.*, 13:566–579, 1984. doi:10.1137/0213035.
33 Alfred Tarski. On the calculus of relations. *J. Symb. Log.*, 6:73–89, 1941. doi:10.2307/2268577.
34 Peter van Beek and Dennis W. Manchak. The design and experimental analysis of algorithms for temporal reasoning. *J. Artif. Intell. Res.*, 4:1–18, 1996.
35 Marc B. Vilain and Henry A. Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In *AAAI*, 1986.
36 Richard J. Wallace. Neighbourhood SAC: Extensions and new algorithms. *AI Commun.*, 29:249–268, 2016. doi:10.3233/AIC-150696.

Dynamic Purpose Decomposition of Mobility Flows Based on Geographical Data

Etienne Thuillier[1], Laurent Moalic[2], and Alexandre Caminada[3]

1 **UTBM, OPERA, Belfort, France**
etienne.thuillier@utbm.fr
2 **Université de Haute-Alsace, LMIA EA 3993, Mulhouse, France**
laurent.moalic@uha.fr
3 **UTBM, OPERA, Belfort, France**
alexandre.caminada@utbm.fr

Abstract

Spatial and temporal decomposition of aggregated mobility flows is nowadays a commonly addressed issue, but a trip-purpose decomposition of mobility flows is a more challenging topic, which requires more sensitive analysis such as heterogeneous data fusion. In this paper, we study the relation between land use and mobility purposes. We propose a model that dynamically decomposes mobility flows into six mobility purposes. To this end, we use a national transportation database that surveyed more than 35,000 individuals and a national ground description database that identifies six distinct ground types. Based on these two types of data, we dynamically solve several overdetermined systems of linear equations from a training set and we infer the travel purposes. Our experimental results demonstrate that our model effectively predicts the purposes of mobility from the land use. Furthermore, our model shows great results compared with a reference supervised learning decomposition.

1998 ACM Subject Classification G.1.3 Linear Systems

Keywords and phrases Human mobility, Purpose decomposition, Information extraction, Linear model

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.20

1 Introduction

Human mobility is a field of research that has significantly been studied during the last decades. We now understand that individuals' displacements are motivated by several factors such as jobs, occupations, social life, etc., but also by the nature of ground infrastructures. Hence, ground infrastructures are revealing items of human occupations over a territory. Moreover, humans develop tendencies to adopt regular mobility patterns, often linked to land use [9, 3]. With the apparition of pervasive devices over the last years, human mobility modeling has been significantly improved and allows us now a better understanding of such patterns. Call Detail Records (CDR) have rapidly been used as presence indicators in the literature [15, 2], but by nature CDR data represent dis-aggregated mobility flows, and often at a relatively small geographic scale (cells of the cellular network). Nowadays, many works propose interesting ways for mobility prediction, by using new technologies such as social networks [13, 1], or through heterogeneous data fusion and big data [14, 4].

From these studies emerges the idea of a link between land use and human mobility [3, 11, 17], and we understand that if the analysis of human mobility patterns leads to the characterization of land use, then, land infrastructures must be a catalyst for human displacements. However, for privacy reasons, mobility flows are often aggregated by data providers, it is the case when dealing with mobile network data, counting loops, or any large scale mobility data. Such data can be spatially or temporarily aggregated, which represents a consequent loss of information. However, spatial and temporal decomposition of aggregated flows is a common issue, and has been largely studied in the recent years [12, 5, 7].

Purpose decomposition of aggregated mobility flows is a difficult and delicate problem. Knowing the end-purpose of any mobility flow helps local actors to better understand the dynamics of individuals traveling over their territories. Many related fields benefit from this knowledge; urbanization, transportation planning, commercial activities, etc. Many works have tried to tackle trip-purpose reconstitution in the last five years. However, the proposed methods are always dependent on the provided data nature. Floating Car Data (FCD) are GPS traces for vehicles, and by definition are not aggregated, as for CDR data. We can cite [18, 10, 6] whom infer trip activities from CDR, and in [8] the authors use FCD to determine travel purposes.

Assigning purposes to aggregated mobility flows is a heterogeneous data fusion problem, and we propose to inject knowledge into raw data to tackle this issue. In this paper, we propose to study the relation between land use (or ground) and mobility flow purposes. The main objective is to propose a method that allows us to decompose mobility flows into several sub-flows, each carrying a distinct mobility purpose. In section 2 we describe the data sets used and we explain the methodology developed to collect land use indicators. Then we propose in section 3 a reference model inferring mobility purposes from land use indicators. We provide two major improvements to this reference model and we analyze the prediction rates of the three algorithms. Finally, we propose in section 4 an analysis of these results, and section 5 concludes our work.

2 Data sets

In this paper, we propose to study the relationship between mobility flows and land use. We focus our study on the Ile-de-France province, a 12 million inhabitants region around Paris, France. We base our study on two freely available data sets. Mobility flows are gathered from a national transportation survey while land use indicators are obtained through several national databases.

2.1 Mobility flows data set

In this proposed model, we use a national transportation survey (Enquete Globale Transport, EGT). This national survey contains the declarations about displacements and commuting habits of more than 35,000 individuals. It also contains the main purpose, time, duration, origin and destination zones of each displacement. Moreover, each surveyed displacement is given a weight corresponding to the number of individuals it actually represents, based on its social and professional category, commuting habits, etc. From the EGT, we build an Origin-Destination matrix (OD matrix) that we call M_{global} and which corresponds to the daily displacements occurring within the whole Ile-de-France province.

2.1.1 Territorial division

The EGT is based on a territorial division whose base unit is a 100x100 meter mesh. By using such meshing, all of the 1,300 Ile-de-France cities are divided into regular squares. According to the EGT, these cities (and thus meshes) are themselves grouped into 118 sectors, which

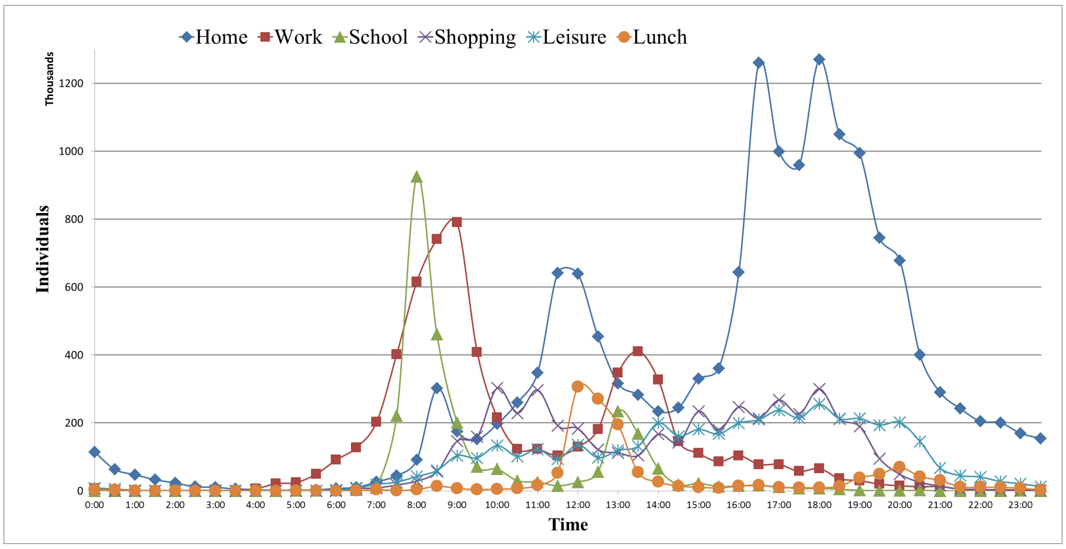


Figure 1 Number of individuals traveling with a specific purpose from the EGT.

means that a sector is composed of 11 cities on average. Origin and destination of each displacement are represented by two 100x100m meshes. This allows us to build OD matrices with any desired territorial division, from meshes to sectors. In this study we decide to use an OD matrix based on the sector division. Indeed, although the 100x100 meter meshing is interesting, it does not provide a statistically realistic information. We call the set of 118 sectors (also known as zones) Z_{global}. The origins and destinations of the M_{global} matrix belong to Z_{global}, which gives 13,924 possible OD pairs at any time.

2.1.2 Temporal division

In this study, we use temporal time slots of 30 minutes. It is a frequently used time gap which allows us to better display the mobility dynamics of the individuals. We note that in the EGT, the start and end times of each displacement are given to within a minute. To build M_{global}, we round down every time to the nearest 30 minute gap. To be more statistically correct, we also could use a Gaussian distribution model that would provide a more uniform time distribution. We refer to a timestamp of M_{global} by t.

2.1.3 Purpose division

Many works in the literature aim at inferring purposes (also called activities) of displacements from mobility data. The numbers of purposes vary greatly according to the studies. For example, in [11] the authors use nine purposes of mobility, in [18] they use five classes, and in [6] they use eight purposes. In the EGT, we have access to 38 purposes of mobility that are classified into eight main groups.

In this paper, we propose to study the six most significant distinct purposes of mobility: Home, Work, School, Shopping, Leisure and Lunch. We consider that these purposes hold the principal reasons of dynamics and movements of individuals on a territory. We notice that the proposed model can easily deal with another number of purposes. We propose to study the evolution of the EGT mobility flows, according to their purposes. Figure 1 shows the number of displacements at any time, grouped by activity.

2.2 Land use

Simultaneously, we collect land usage information from the 118 sectors of Z_{global}. In this paper, we consider that the land use of a sector is an indicator of a specific human activity done in this zone. We do not focus on the distribution of infrastructures on the ground, but we rather collect activity indicators that reflect an understanding of the usage of ground infrastructures. For example, contrarily to [17] that focuses on five land uses obtained through aerial analysis of ground infrastructures (massGIS), we propose to collect land use information from national databases. We focus on six land use indicators that are grouped into two main fields:

1. **Presence indicators**
   - Number of residents
   - Number of employees
   - Number of students (from elementary to postgraduate education)
2. **Economical activity indicators**
   - Number of megastores ($> 2500m^2$)
   - Number of supermarkets ($> 400m^2$)
   - Number of stores ($< 400m^2$)

These indicators are collected from several INSEE free-access databases. INSEE is the official French national institute for statistics and economic studies, in charge of statistics and censuses (national census, surveys, economic indicators, etc.). We collect the number of residents from the national census, which contains information about the 1,300 cities of the Ile-de-France province. The number of employees is obtained from a dedicated database[1] that nationally identifies every company, its number of employees and its location. We do not make assumptions between the different types of workers (commuting, teleworking, transporters, etc.). A version considering these special features will be checked further. We obtain the number of students from another commonly used database[2]. Finally, the number of megastores, supermarkets and stores is collected from a third database[3] that censuses every community facilities with their location.

These three economic activity indicators appear as particularly relevant since they are strong catalysts of mobility, in the sense that they do attract individuals, for identified reasons, and in different quantities. Megastores ($> 2500m^2$) mainly attract individuals for leisure, shopping or errands, in great quantities. They are strategically located over territories and are great mobility hubs. Supermarkets generate lower displacements. Individuals generally go to supermarkets to buy groceries, and more rarely for leisure. Finally, city stores ($< 400m^2$) are representative of the attractiveness and dynamics of a city center. The more city stores there are, the more individuals are present for leisure, lunch, shopping, etc. at specific times of the day. A version considering the sales volumes of these infrastructures will be checked further. As a matter of fact, some stores may be more attractive than others (services, leisure facilities, etc.) and thus may present different attraction behaviors.

We present in Table 1 some statistics about the number of indicators for the 118 zones of Z_{global}.

[1] http://www.sirene.fr/sirene/public/accueil?sirene_locale=en
[2] https://www.insee.fr/fr/statistiques/1913211
[3] https://www.insee.fr/fr/metadonnees/source/s1161

Table 1 Ground indicators details over the zones.

Indicator	min	median	max
Residents	1,177	58,732	626,676
Employees	825	23,499	365,446
Students	168	13,864	179,063
Megastores	0	1	7
Supermarkets	0	9	110
Stores	1	109	2,802

3 Model

To study the relation between land use and mobility purposes we propose to use a supervised learning model. We split our data into two parts, one part will be used for training and learning process while the second part will be used for testing and validation.

3.1 Creation of a training set

In supervised learning we have to split our database in two. The number of zones in the study being relatively small, we propose to use a 50% ratio for separating training and validation zones. With this 50% ratio we limit the risks of having too much outliers in the validation set. A version with different ratios and statistical inference models will be checked further. We propose then separate 59 EGT of the 118 EGT sectors that we put in a Z_{tr} set (tr is for training). We put the 59 other zones in a set called Z_{val} (val is for validation). All the OD flows from M_{global} with a destination zone d within Z_{tr} are added to a M_{tr} matrix, and all flows with a destination zone within Z_{val} are added to a M_{val} matrix. Additionally we create two other OD matrices called M^*_{tr} and M^*_{val}. These matrices correspond to the M_{tr} and M_{val} matrices respectively, but aggregated by destination and time slot. This means that there are no purposes information in these last two matrices. The choice of sectors is random.

3.2 Purpose Flow Decomposition algorithm (PFD)

In this paper we study the relationship between land use and displacements purposes. From one side we collect purposes decomposed flows in M_{train}, and from the other side we collect information about six land use indicators from all the zones of M_{global}. From now, we refer to a zone as a destination zone d with d in Z_{global}, and to a land use indicator at d by $ground_i(d)$. Hence, $ground_1(d)$ is the number of *Residents* and $ground_6(d)$ the number of *Stores* at zone d.

3.2.1 Normalization

In the next parts, we use the notation $n\_ < variable >$. This notation corresponds to the normalized value of a variable instance relatively to the maximum known value of this variable. For example, we use the notation $n\_ground_i(d)$ which corresponds to the normalized value of the land use indicator $ground_i$ at destination zone d relatively to the maximum known value of this ground indicator in all zones. This allows us to compare indicators between them, without scaling effect problems. We compute this normalized value as follows:

$$n\_ground_i(d) = \frac{ground_i(d)}{\max\limits_{d \in Z_{global}}(ground_i(d))} \quad . \tag{1}$$

3.2.2 Main equation

We want to mathematically write the relationship between the land uses and a purpose of mobility. For such a linear relation between the ground and mobility flows, we write for any time t, and any destination d a linear equation linking the land usage $ground_i$ and a specific purpose p:

$$\forall d, \forall t, \forall p, \; \sum_{i=1}^{n} (\alpha_i(t,p) \, . \, n\_ground_i(d)) = M_{tr}(d,t,p) \quad . \tag{2}$$

where for every timestamp t, every purpose p, and every destination d,

- n is the number of ground indicators, fixed to 6
- $\alpha_i(t,p)$ is a coefficient to determine
- $n\_ground_i(d)$ is the normalized value of the ground indicator $ground_i(d)$
- $M_{tr}(d,t,p)$ is the sum of flows with purpose p from any origin zone of Z_{tr} to the destination zone d at timestamp t

3.2.3 Overdetermined system

The training matrix M_{tr} uses 59 zones, thus we can write for any couple of timestamp t of 30 minutes and purpose p, 59 equations with n unknown. As an example, we represent the specific system used for purpose *Home* and timestamp 13:30. For readability reasons we use for *n_Residents* the abbreviation *Res*, for *n_Employees*, *Emp*, etc. With same concerns, the couple $(t,p) =$ (13:30, *Home*) is removed but implicitly considered. Therefore, the values of $M_{tr}(d)$ correspond actually to M_{tr}(d,13:30,*Home*), and α_i to α_i(13:30,*Home*) coefficients. We call that system S.

$$\begin{cases} \alpha_1.Res_1 + \alpha_2.Emp_1 + \alpha_3.Stu_1 + \alpha_4.Meg_1 + \alpha_5.Sup_1 + \alpha_6.Sto_1 = M_{tr}(1) \\ \alpha_1.Res_2 + \alpha_2.Emp_2 + \alpha_3.Stu_2 + \alpha_4.Meg_2 + \alpha_5.Sup_2 + \alpha_6.Sto_2 = M_{tr}(2) \\ \vdots \\ \alpha_1.Res_{59} + \alpha_2.Emp_{59} + \alpha_3.Stu_{59} + \alpha_4.Meg_{59} + \alpha_5.Sup_{59} + \alpha_6.Sto_{59} = M_{tr}(59) \end{cases}$$

As the number of ground indicators (*Res, Emp,* etc.) is equal to six, we are faced to an overdetermined linear equations system (6 unknown and 59 equations). We propose to solve these overdetermined linear systems based on a least squares approach for every timestamp t and purpose p. Since the number of equations is not large, we use a SVD decomposition as a first step for this study despite the computational cost. A version with other supervised learning models will be checked further. In the end we obtain 288 systems, where we compute n coefficients (α_1 to α_6) for any (t,p) pair.

3.2.4 Application

Now, to predict the displacements purposes, we apply to any aggregated OD flow $M^*_{val}(d,t)$, the dedicated $\alpha_i(t,p)$ coefficients. For example, to predict the *Home* sub-flow of $M^*_{val}(d,t)$, we apply the coefficients inferred from the S system trained with *Home* values at time t. For the *Leisure* sub-flow we apply the coefficients obtained with the *Leisure* values system S at

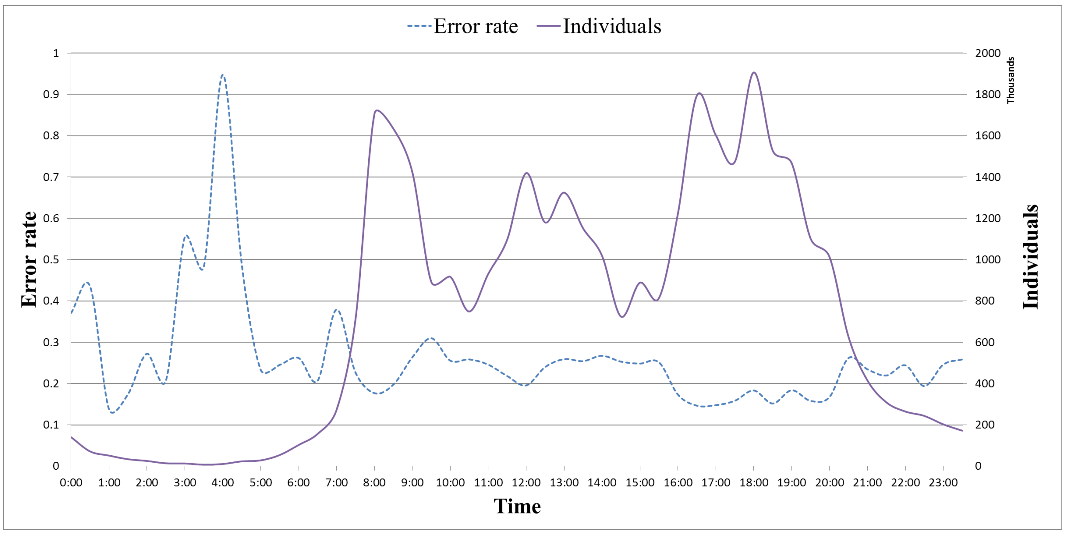


Figure 2 Error rate of the PFD algorithm.

time t, etc. It is important to note that these coefficients are not directly applied to the flow itself, but to the ground indicators $ground_i(d)$ at destination zone d. We thus obtain a theoretical displacement value $M_{theo}(d, t, p)$ for each purpose. We can then compare this theoretical value to the real displacement value $M_{val}(d, t, p)$ and estimate the error of our model. We call that model the Purpose Flow Decomposition algorithm (PFD).

Then, we write the computation of the theoretical value for purpose *Home* and timestamp 13:30 with:

$$M_{theo}(d) = \alpha_1.Res_d + \alpha_2.Emp_d + \alpha_3.Stu_d + \alpha_4.Meg_d + \alpha_5.Sup_d + \alpha_6.Sto_d \ . \quad (3)$$

For readability reasons the value $M_{theo}(d)$ corresponds in our example to the value M_{theo}(d,13:30,*Home*), and α_i to α_i(13:30,*Home*) coefficients.

3.2.5 Results

We operate Equation (3) on all aggregated flows of M^*_{val} and for all purposes *Home, Work, School, Leisure, Shopping* and *Lunch*. As a reminder, the flows of M_{val} have not been used to determine the α coefficients. We then estimate for every timestamp the number of wrongly predicted flows. For that, we compute the sum of the absolute values between theoretical $M_{theo}(d, t, p)$ and real value $M_{val}(d, t, p)$, divided by the total flow size at this timestamp. This gives us an error rate between [0, 1]. In Figure 2 we represent in dotted line the evolution of the error rate for the PFD algorithm. The solid line represents the total flow size along the day. We observe that the error rate is relatively stable over the day, except during nighttime (from 03:00 to 04:00) where the error rate jumps at 95%. During this period the number of traveling individuals is really small (around 15,000), thus the sampling is not large and individuals behavior is thus less predictable. The predicting rate of the algorithm is robust even when the number of individuals traveling increases greatly. It even reaches its optimum from 16:00 to 20:00 with almost 85% of accurate prediction whereas the number of individuals' displacements is the highest. The total daily error rate for all zones and all timestamps taken together is around 21.21%. Finally, we can state that the more individuals are traveling, the best we can predict mobility purposes.

3.3 γ-PFD, a first optimized approach

The reference PFD algorithm aims at setting down the relationship between distinct ground characteristics and different purposes of mobility. It means that for any destination d whom the land use is known, the PFD model can predict purposes of mobility with almost 78% of accuracy. We propose now to introduce in the main Equation (2) a new indicator. This new indicator considers the flow size as a determining variable. Actually, Equation (2) predicts a flow size, but do not take into account the scaling effect and flow amplitude at time t. And we see in Figure 2 that flow sizes adopt different behaviors at different times of the day. We propose then to add the total flow size $M^*_{tr}(d,t)$ as a seventh indicator in our main equation. This variable is associated to a new coefficient that we call γ. Now, by adding this new variable, one equation of the overdetermined system S becomes:

$$\forall d, \forall p, \forall t, \sum_{i=1}^{n} (\alpha_i(t,p)\,.\,n\_ground_i(d)) + (\gamma(t,p)\,.\,n\_M^*_{tr}(d,t)) = M_{tr}(d,t,p) \tag{4}$$

where

- $\gamma(t,p)$ is a coefficient to find for every timestamp t and purpose p,
- $n\_M^*_{tr}(d,t)$ is the normalized value of $M^*_{tr}(d,t)$ relatively to

  $\max(M^*_{tr}(d,t))$ for all d in Z_{tr} .

- and the other components are identical to the ones in equation (2).

3.3.1 Results

As for the reference PFD algorithm, we solve the overdetermined linear equations system and we compute the $\alpha_i(t,p)$ and $\gamma(t,p)$ coefficients with a least squares approach. This means that now, the equation from our example in (3) with the couple (t,p) =(13:30,*Home*) becomes:

$$M_{theo}(d) = \alpha_1.Res_d+\alpha_2.Emp_d+\alpha_3.Stu_d+\alpha_4.Meg_d+\alpha_5.Sup_d+\alpha_6.Sto_d+\gamma.n\_M^*_{tr}(d). \tag{5}$$

We operate Equation (5) on all aggregated flows of the matrix M^*_{val} and for all purposes *Home, Work, School, Leisure, Shopping* and *Lunch*. In Figure 3 we represent in dashed line the evolution of the error rate for the γ-PFD algorithm. The dotted line represents the evolution of the error rate for the reference algorithm. The total error rate over time and zones for this γ-PFD algorithm is around 17.86%. As a reminder, the total daily error rate for PFD algorithm was 21.21%. This means that the introduction of a flow amplitude coefficient increases the mean prediction accuracy of our algorithm by 3.3 points. We observe that the error rate is relatively stable over the day, except again during nighttime (from 03:00 to 04:00) where the error rate jumps at 70%. However, by introducing the flow amplitude coefficient we reduce the error during that period by almost 25 points. The γ-PFD algorithm reaches a maximum prediction rate during the period [18:00, 20:00] with nearly 90% of good prediction.

3.4 γ-PFD*, a second optimized approach

The underlying effect of solving an overdetermined system of linear equations is that the generated coefficients are adapted to give the best average solution from a training set. This means that the solver tries to give the best solutions taking into account the all

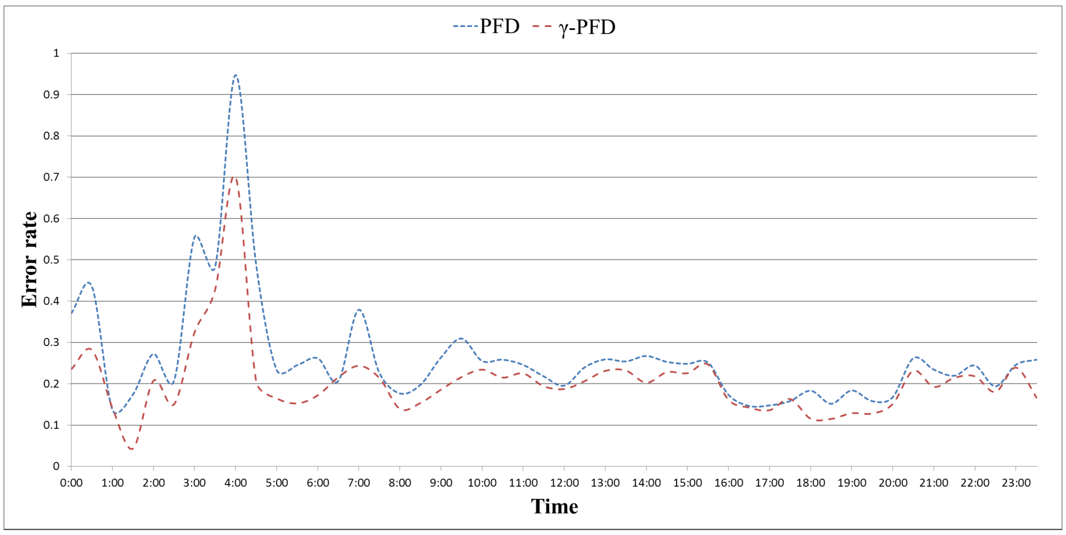


Figure 3 Error rate of the γ-PFD algorithm.

Table 2 Summary of the overdetermined systems S_1 and S_2.

System	S_1	S_2
Ground indicators	Residents, Employees, Students	Megastores, Supermarkets, Stores
Purposes	Home, Work, School	Shopping, Leisure, Lunch

different input variables. Here these variables (ground characteristics and flow size) cannot be compared directly, that is why we use normalized values. However, the solver here tries to link numbers of individuals (*Residents, Employees, Students*) and infrastructures (*Megastores, Supermarkets, Stores)* with purposes of mobility that allegedly are more attracted by specific land characteristics. As a matter of fact, individuals traveling with purpose *Work* will statistically be more attracted to a zone with more *Employees*. The same applies to the rest of the purposes. We propose here to split the system S into two twin systems S_1 and S_2, to differ primary mobility purposes and secondary mobility purposes. The primary set will address the purposes *Home, Work, School*, while the secondary set will be in charge of purposes *Shopping, Leisure, Lunch*. By doing so, the α coefficients will be more adapted to the mobility purposes inside their respective subset of learning data. We propose a summary of these two systems in table 2.

We proceed to the S_1, S_2 separation, and Equation (4) becomes:

$$\forall d, \forall p, \forall t, \sum_{i=1}^{q} (\alpha_i(t,p) \, . \, n\_ground_i(d)) + (\gamma(t,p) \, . \, n\_M^*_{tr}(d,t)) = M_{tr}(d,t,p) \qquad (6)$$

where

- q is the number of ground indicators (3 for S_1 and 3 for S_2),
- with $p \in \{Home, Work, School\}$ for S_1,
- and $p \in \{Shopping, Leisure, Lunch\}$ for S_2.

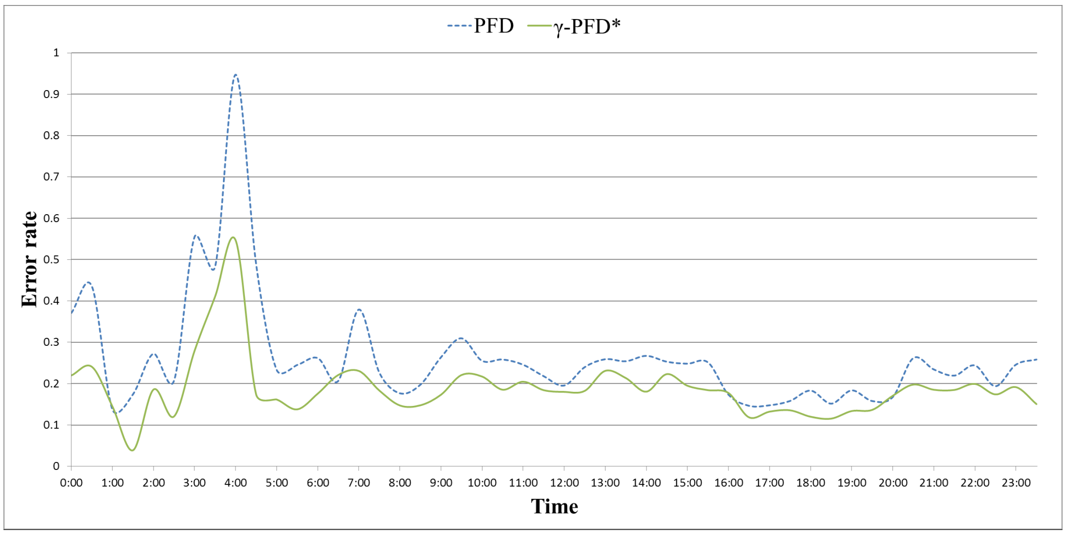


Figure 4 Error rate of the γ-PFD* algorithm.

3.4.1 Results

Now that we split our global system in two sub-systems, the equation from our example in (5) with the couple $(t, p) = (13{:}30, \textit{Home})$ is given by the equation of the system S_1 for primary mobility purposes:

$$M_{theo}(d) = \alpha_1.Res_d + \alpha_2.Emp_d + \alpha_3.Stu_d + \gamma.n\_M^*_{tr}(d) \quad . \tag{7}$$

And by the system S_2 for secondary mobility purposes:

$$M_{theo}(d) = \alpha_1.Meg_d + \alpha_2.Sup_d + \alpha_3.Sto_d + \gamma.n\_M^*_{tr}(d) \quad . \tag{8}$$

As for the reference algorithm PFD, we compute the $\alpha_i(d, t)$ and $\gamma(t, p)$ coefficients by solving these overdetermined systems, and we apply these coefficients on all aggregated flows of the matrix M^*_{val} and for all purposes. In other word, we apply either Equation (7) or (8) to the destination zone of M^*_{val} flows. In Figure 4 we represent in solid line the evolution of the error rate for the γ-PFD* algorithm and in dotted line the evolution of the error rate for the PFD algorithm. The total daily error rate for the γ-PFD* algorithm is around 16.84%. With this system separation we increase the prediction accuracy of our algorithm by 4.3 points.

4 Analysis

4.1 Application on the training set

The γ-PFD* algorithm gives a correct average prediction rate of 83% for the validation set M^*_{val}. We now wonder how the algorithms behave when used on their own training set M^*_{tr}. Figure 5 shows the error rates of the three algorithms when used with M^*_{tr}. We observe that all three algorithms adopt the same behavior, with an average good prediction rate of 86%. This means that the linear combinations of the ground indicators generated by the supervised learning are well adapted to the training set. However, when confronted with another set of data, introducing the flow amplitude indicator is beneficial.

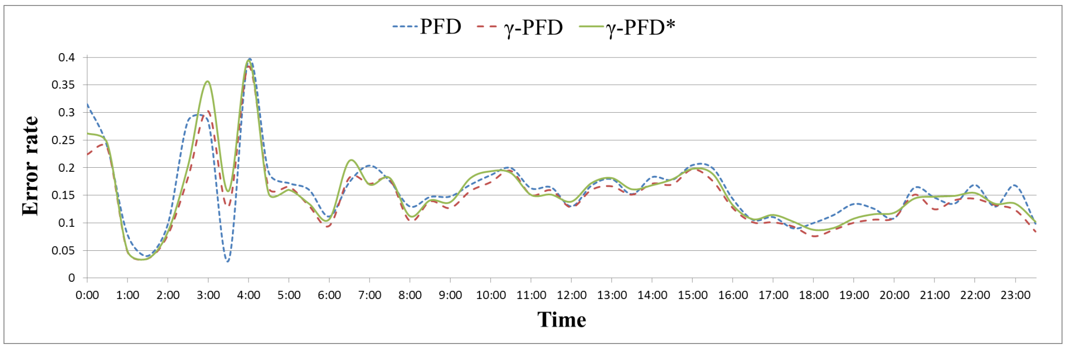


Figure 5 Error rates of the 3 algorithms when used on M^*_{tr}.

4.2 Geographical analysis

As explained before, our model gives predictions based on ground characteristics. So when the model wrongly assigns purposes to individuals flows, it means that the model has been tricked by the ground characteristics of the destination zone. We propose here to study the prevalence of some zones to give wrong results. For that we study the correlation between the daily error rate of each zone and the ground characteristics for all zones in Z_{val}. Figure 6 shows these correlations. On the abscissa we represent the ground indicator values, and on the ordinate we display the daily error rates. Each point corresponds to one of the 59 zones of Z_{val}. We observe that for all land uses, the correlation curve adopts a $\frac{1}{\log(x)}$ like pattern. Next to each graph we display the Spearman's rank correlation coefficient which is adapted to describe the correlation between two variables without linear relation [16]. All curves have a good Spearman correlation (value close to -1) and show the same trend. The more a destination zone has important land indicator value, the more the prediction rate is good. In other words, the more residents, students or supermarkets in a zone, the more the γ-PFD* model accurately predicts the purposes of mobility. We note that similar results are obtained from the training set Z_{tr}.

To validate this point, we apply the γ-PFD* algorithm on the zones having more than 40,000 inhabitants. In the *Residents* figure a vertical line shows this 40,000 limit. As a reminder, the median number of inhabitants per zone is 58,000 in our data set. With this parameter the daily error rate with the γ-PFD* algorithm is 16.01%. The gain is not important (0.8%), but it opens an interesting way for future improvements. This zone selection has been done on the other ground indicators with similar results.

5 Conclusions

Purpose decomposition of aggregated mobility flows is a delicate problem that has recently been treated from mobile network databases analysis. In this paper, we propose three different algorithms predicting purpose distributions of aggregated mobility flows, with different prediction results. The reference algorithm that we propose uses supervised learning to infer purposes of mobility from raw ground indicators. We then propose two improvements to this reference algorithm using freely available databases. We notably add a variable linked to the mobility flow size, and we propose to split the system into two sets managing distinct purposes. *Home, Work* and *School* purposes are inferred from *Residents, Employees* and *Students* indicators, while *Leisure, Shopping* and *Lunch* purposes are inferred from *Megastores, Supermarkets* and *Stores* information.

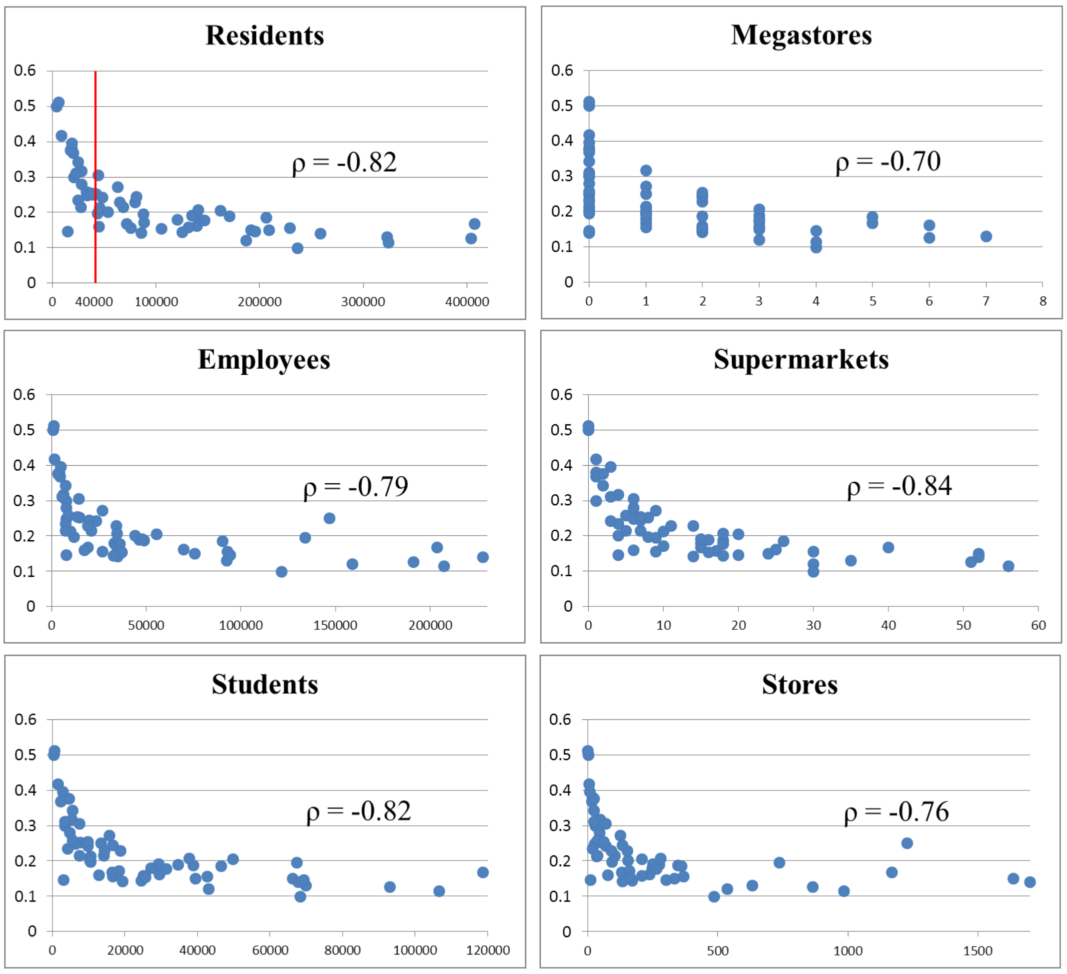


Figure 6 Correlations between ground characteristics and daily error rate.

The last improvement of the initial algorithm accurately predicts purposes of mobility in 83% of cases. And even when the number of individuals in displacement increases significantly, the prediction rate stays stable. It even reaches an optimum during the period [16:00, 18:00] with nearly 90% of success. These are promising results, as they allow a purpose decomposition of aggregated mobility flows without anything more than freely accessible sociological and geographical databases.

Furthermore, we see an interesting geographical correlation between the daily error estimation rate of the studied zones and their land use. The more infrastructures are present in a zone, the better are the prediction results. The same applies to the number of individuals moving. The more individuals are traveling, the better are the prediction results. This means that we can estimate the confidence rate of our results according to the input variables. This opens a new way for the analysis of aggregated mobility flows, especially from mobile network data.

References

1 Mariano G. Beiró, André Panisson, Michele Tizzoni, and Ciro Cattuto. Predicting human mobility through the assimilation of social media traces into mobility models. *EPJ Data Science*, 5(1):30, October 2016. `doi:10.1140/epjds/s13688-016-0092-2`.

2 Michele Berlingerio, Francesco Calabrese, Giusy Di Lorenzo, Rahul Nair, Fabio Pinelli, and Marco Luca Sbodio. AllAboard: A System for Exploring Urban Mobility and Optimizing Public Transport Using Cellphone Data. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, number 8190 in Lecture Notes in Computer Science, pages 663–666. Springer Berlin Heidelberg, January 2013.

3 F. Calabrese, G. Di Lorenzo, and C. Ratti. Human mobility prediction based on individual and collective geographical preferences. In *2010 13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 312–317, September 2010. `doi:10.1109/ITSC.2010.5625119`.

4 Cynthia Chen, Jingtao Ma, Yusak Susilo, Yu Liu, and Menglin Wang. The promises of big data and small data for travel behavior (aka human mobility) analysis. *Transportation Research Part C: Emerging Technologies*, 68:285–299, 2016. `doi:10.1016/j.trc.2016.04.005`.

5 Cathal Coffey and Alexei Pozdnoukhov. Temporal Decomposition and Semantic Enrichment of Mobility Flows. In *Proceedings of the 6th ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, LBSN'13, pages 34–43, New York, NY, USA, 2013. ACM. `doi:10.1145/2536689.2536806`.

6 Mi Diao, Yi Zhu, Joseph Ferreira, and Carlo Ratti. Inferring individual daily activities from mobile phone traces: A Boston example. *Environment and Planning B: Planning and Design*, 43(5):920–940, September 2016. `doi:10.1177/0265813515600896`.

7 Zhanwei Du, Bo Yang, and Jiming Liu. Understanding the Spatial and Temporal Activity Patterns of Subway Mobility Flows. *arXiv:1702.02456 [cs]*, February 2017. arXiv: 1702.02456.

8 Li Gong, Xi Liu, Lun Wu, and Yu Liu. Inferring trip purposes and uncovering travel patterns from taxi trajectory data. *Cartography and Geographic Information Science*, 43(2):103–114, March 2016. `doi:10.1080/15230406.2015.1014424`.

9 Sibren Isaacman, Richard Becker, Ramón Cáceres, Stephen Kobourov, Margaret Martonosi, James Rowland, and Alexander Varshavsky. Identifying Important Places in People's Lives from Cellular Network Data. In Kent Lyons, Jeffrey Hightower, and Elaine M. Huang, editors, *Pervasive Computing*, number 6696 in Lecture Notes in Computer Science, pages 133–151. Springer Berlin Heidelberg, January 2011.

10 S. Jiang, J. Ferreira, and M. C. Gonzalez. Activity-Based Human Mobility Patterns Inferred from Mobile Phone Data: A Case Study of Singapore. *IEEE Transactions on Big Data*, PP(99), 2017. `doi:10.1109/TBDATA.2016.2631141`.

11 Shan Jiang, Marta C. Gonzalez, and Joseph Ferreira. Understanding the Link between Urban Activity Destinations and Human Travel Pattern. *MIT web domain*, July 2011.

12 M. Katranji, E. Thuillier, S. Kraiem, L. Moalic, and F. H. Selem. Mobility data disaggregation: A transfer learning approach. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1672–1677, Nov 2016. `doi:10.1109/ITSC.2016.7795783`.

13 Anastasios Noulas. *Human urban mobility in location-based social networks : analysis, models and applications*. PhD thesis, University of Cambridge, UK, 2013.

14 Lucas M. Silveira, Jussara M. de Almeida, Humberto T. Marques-Neto, Carlos Sarraute, and Artur Ziviani. MobHet: Predicting human mobility using heterogeneous data sources. *Computer Communications*, 95:54–68, 2016. `doi:10.1016/j.comcom.2016.04.013`.

15 Zbigniew Smoreda, Ana-Maria Olteanu, and Thomas Couronné. Spatiotemporal data from mobile phones for personal mobility assessment. *Transport Survey Methods: Best Practice for Decision Making*, 2013.

16 C. Spearman. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology*, 15(1):72–101, 1904. doi:10.2307/1412159.

17 Jameson L. Toole, Michael Ulm, Marta C. González, and Dietmar Bauer. Inferring land use from mobile phone activity. In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing, UrbComp@KDD 2012, Beijing, China, August 12, 2012*, pages 1–8. ACM Press, 2012. doi:10.1145/2346496.2346498.

18 Peter Widhalm, Yingxiang Yang, Michael Ulm, Shounak Athavale, and Marta C. González. Discovering urban activity patterns in cell phone data. *Transportation*, 42(4):597–623, July 2015. doi:10.1007/s11116-015-9598-x.

Time Dependent Policy-Based Access Control[*]

Panagiotis Vasilikos[1], Flemming Nielson[2], and Hanne Riis Nielson[3]

1 **Department of Applied Mathematics and Computer Science, Technical University of Denmark, Lyngby, Denmark**
panva@dtu.dk
2 **Department of Applied Mathematics and Computer Science, Technical University of Denmark, Lyngby, Denmark**
fnie@dtu.dk
3 **Department of Applied Mathematics and Computer Science, Technical University of Denmark, Lyngby, Denmark**
hrni@dtu.dk

Abstract

Access control policies are essential to determine who is allowed to access data in a system without compromising the data's security. However, applications inside a distributed environment may require those policies to be dependent on the actual content of the data, the flow of information, while also on other attributes of the environment such as the time.

In this paper, we use systems of Timed Automata to model distributed systems and we present a logic in which one can express time-dependent policies for access control. We show how a fragment of our logic can be reduced to a logic that current model checkers for Timed Automata such as UPPAAL can handle and we present a translator that performs this reduction. We then use our translator and UPPAAL to enforce time-dependent policy-based access control on an example application from the aerospace industry.

1998 ACM Subject Classification D.4.6 Security and Protection, D.2.4 Software/Program Verification, C.2.4 Distributed Systems, F.4.1 Mathematical Logic

Keywords and phrases Access Control, Timed Automata, Time-Dependent Policies, UPPAAL

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.21

1 Introduction

Motivation. Cyberphysical systems play an increasingly important role in the technology development in many industries such as the aerospace, the automotive and the medical. Embedded systems are key components to cyberphysical systems and while verifying their safety goals has received a significant focus until now, security has been left for later. As more and more cyberphysical systems are integrated with real-time hardware, complex software, and internet connected devices through wireless connections, ensuring security goals of those systems is becoming essential. Particularly, assuring the confidentiality or the integrity of the information manipulated by the different components of a cyberphysical system is a crucial security goal.

Information security is usually achieved by access control policies, which formally specify desired flows of information inside a system. Access requests to the resources/data (objects)

[*] The authors are supported in part by the IDEA4CPS Reseearch Centre studying the Foundations for Cyber-Physical Systems and granted by the Danish Research Foundation for Basic Research (DNRF86-10).

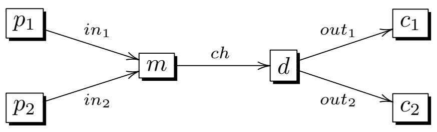


Figure 1 The processes and channels of the gateway example.

of the system by users (subjects) are then either denied or allowed by a monitor that enforces the access control policies. The literature offers a vast number of access control models, where among all, the most used in practice are the discretionary access control (DAC) [29], mandatory access control (MAC) [28] and role-based access control (RBAC) [12], while lately a great attention has been given to the attribute-based access control (ABAC) model [17], wherein access control may depend on the attributes of the accessed data or the attributes of the environment such as the time.

Although access control policies is a well-established approach for information security at the subject-object level, distributed systems require precise policies that express also the desired information flows that occur at the application level, such as *explicit* flows. As an example, consider the explicit flow from the variable y to the variable x that arises from the assignment $x := y$ and the access control policy "x can only be modified by p", where p is a trusted process in the system. Although the assignment $x := y$ executed by p does not violate the access control policy of the system, the fact that p resides in a distributed environment together with potential bugs inside its source code gives no guarantees that the value of y was written by p. For instance, the value of y could have been influenced by an untrusted process p' after a communication between p and p' and consequently p' would have also influenced the value of x. To see more challenges that arise inside distributed systems consider the work of [25] where it illustrates that security policies may need to depend on the actual content of the data, while ABAC [17] models also address the need for time-dependent security policies.

Contribution. It is natural then to extend the enforcement of safety properties of embedded systems with enforcement of access control policies. The idea is that having an abstract model of an embedded system, one could eliminate possible security violations in the trusted part of the system before the actual run of the system happens. We use *Timed Automata* [3, 1] to model distributed systems and we specify an information flow instrumented semantics that allows us to record information about the accesses being performed; we call this information a behaviour of the system. To deal with formal definitions of security policies we present a behaviour logic (based on the behaviours of the system) that supports the specification of content, time and information dependent access control policies. Verification of Timed Automata has been successfully achieved by model checkers based on the timed computation tree logic (TCTL) [2], and consequently, we propose a reduction of a substantial fragment of our logic to a logic that can be handled by the well-established model checker UPPAAL [30]. Finally, we present a translator that performs this reduction and we illustrate our development using an example from the aerospace industry. Figure 1 sketches the example: *A gateway with two processes, each of them produces data for different targets, uses a multiplexer and a demultiplexer to successfully deliver the data to the intended target. The multiplexer merges the data from the producers and sends it to the demultiplexer who is responsible for delivering it to the right target. The target of the data depends both on the time of the system while also on the content of it and thus it is challenging to express the appropriate security policy.* The example is based on the secure gateway presented in [22], where a seperation kernel is used to allow the accesses to the resources of the system by the system's processes to be *temporal* (based on time) and based on an *information flow policy*. The seperation of the resources is used to ensure that untrusted processes such as passenger's devices can have access to onboard communication systems, without alerting the safe operation of the aircraft.

Related work. There are many other papers dealing with access control [27, 19, 15, 14, 34, 32], however without considering time-dependent security policies; a survey of access control models is available at [11].

A rich logic that allows reasoning about time-dependent policies, together with a proof checker for the logic is considered in [10], however there are no information flow considerations at the application level such as explicit flows. SecPAL [5] is another logic that supports time-dependent policies, as well as the encoding of many well-known policy idioms such as DAC, MAC, RBAC, and ABAC, however the enforcement of time constraints is external to the language. A somewhat different approach has been taken in [4], where a monitor is used to enforce time-dependent access control, by checking a system's logs that records the different actions of the users in a database system. Our contribution focuses on the challenges of time-dependent access control for embedded systems modeled as Timed Automata.

The work of [20] presents a formal specification and verification of the temporal role-based access control (TRBAC) model [6], a flexible model in which the roles of the users of the system are enabled or disabled depending on the time of the system. They then use UPPAAL [30] to model a TRBAC system and verify the desired security policies. The same authors of this paper, present in [21], an extension of this model which is based on the generalized-TRBAC (GTRBAC) model [18]. The work of [13] considers spatial-TRBAC (STRBAC) models [7] in which the rights of the user may depend on the time as well as on the location of the user; again the different roles of the system are modeled as timed automata and verified in UPPAAL. Carlo Combi et.al in [9] merges temporal role-based access control with workflows, while in [8] he defines access-controlled temporal networks, an extension of the conditional simple temporal networks with uncertainty which allows you to model users and temporal authorization constraints. Although all of those models deal with an important number of access control policies at the subject-object level considering time dependencies, they are not able to express time-dependent policies with information flow considerations that occur at the application level of the system (e.g does a process running on behalf of a user respects the access control policy?).

The work of [26] formalizes the timed decentralised label model (TDLM) an extension of the traditional and well-established decentralised label model (DLM) [23], which deals with both information flow and time-dependent security policies; however, their work does not consider an enforcement mechanism for the policies. Our key contribution is to develop a logic that allows the specification of time, data's content and information flow dependent policies for access control, and to make use of current model checkers such as UPPALL [30] for the enforcement of the policies.

Organisation. The remainder of this paper is organized as follows. In Section 2, we give the definition of a Timed System (a system of Timed Automata) and in Section 3 we define an information flow instrumented operational semantics for Timed Systems. Section 4 presents the syntax and the semantics of our behaviour logic called **BTCTL** (behaviour TCTL) and we illustrate how we can successfully express security policies for our gateway example. In Section 5 we present the reduction of the **BTCTL** logic to a variation of the TCTL [2], called $\mathbf{TCTL}^{+}$ and in Section 6 we present our translator. Finally, in Section 7, we give our conclusions and we outline our future work, while in appendix A we give the proof of our main theorem.

2 Systems of Timed Automata

2.1 Timed Systems

A *Timed System TS*

$$p_1 : \mathsf{TA}_1 \mid\mid \ldots \mid\mid p_n : \mathsf{TA}_n \ (n \geq 1)$$

which sometimes we will call system, is the parallel composition of n timed automata called processes, written as $TS{=}(TA_i)_{i \leq n}$.

The processes are able to exchange information via synchronous message passing, using polyadic channels from the finite set **Chan**. Each of the processes in the timed system, is labelled with a unique identifier $p \in \mathbf{P}{=}\{p_1, ..., p_n\}$ and we write $\mathbf{Var}_p$ and $\mathbf{Clock}_p$ for the data variables and clocks appearing in the process p. We also require that the sets of data variables and clocks for the processes are mutually disjoint ($\forall i \neq j : \mathbf{Var}_{p_i} \cap \mathbf{Var}_{p_j} = \emptyset \wedge \mathbf{Clock}_{p_i} \cap \mathbf{Clock}_{p_j} = \emptyset$) and we write $\mathbf{Var}{=}\bigcup_i \mathbf{Var}_{p_i}$ and $\mathbf{Clock}{=}\bigcup_i \mathbf{Clock}_{p_i}$ for the overall data variables and clocks apperaring in the timed system.

2.2 Timed Automata

Formally, we model a *Timed Automaton* [3, 1] TA as a 4-tuple $(q_\circ, \mathsf{E}, \mathsf{I}, \mathsf{Q})$ where $q_\circ$ is the initial location of the automaton, E is a finite set of edges, I is mapping from the automaton's locations to conditions that impose invariants, and Q is the set of the automaton's locations.

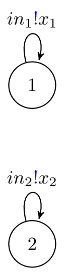
The edges are labelled with actions $g \rightarrow act\colon \vec{r}$ and take the form $(q_s, g \rightarrow act\colon \vec{r}, q_t)$ where the syntax of the act is given by

$$act ::= \vec{x} := \vec{e} \mid ch!\vec{e} \mid ch?\vec{x}$$

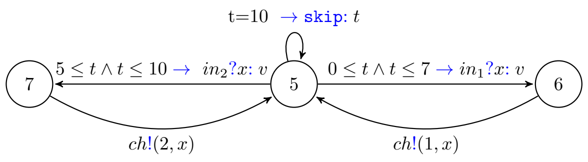
and $q_s \in \mathsf{Q}$ is the source location and $q_t \in \mathsf{Q}$ is the target location.

Every action $g \rightarrow act\colon \vec{r}$ consists of a guard g which has to be true in order for the action to be performed and it ends with a reset on the clocks $\vec{r}$. The assignment action $g \rightarrow \vec{x} := \vec{e}\colon \vec{r}$ performs multiple assignments $\vec{x} := \vec{e}$, while the action $g \rightarrow ch!\vec{e}\colon \vec{r}$ is used to communicate the data of the expressions in $\vec{e}$ using the channel ch and the action $g \rightarrow ch?\vec{x}\colon \vec{r}$ is used to receive data and store it in the variables of the vector $\vec{x}$. We shall assume that the sequences $\vec{x}$ and $\vec{e}$ of data variables and expressions, respectively, have the same length and that $\vec{x}$ does not contain any repetitions. Finally, we write $\vec{x}(i)$ (and also $\vec{e}(i)$) for the i-th element of the vector $\vec{x}$ (and $\vec{e}$ respectively). To cater for special cases of the assignment action, we shall allow to write $g \rightarrow \texttt{skip}\colon \vec{r}$ when $\vec{x}$ (and hence $\vec{e}$) is empty; also for any kind of action we shall allow to omit the guard g when it equals to $\texttt{tt}$ and to omit the clock resets when $\vec{r}$ is empty. If it is the case that all of the above take place together we omit the whole action.

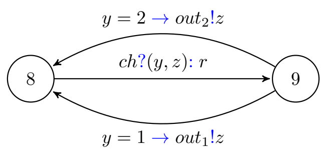
▶ **Example 1.** The timed system of our gateway example is given in Figure 2. The timed system consists of six processes $\mathbf{P} = \{p_1, p_2, m, d, c_1, c_2\}$. Two producers p_1 and p_2 send their data via the channels in_1 and in_2 respectively. The multiplexer m collects the data from the producers using the channel ch and then forwards it to the demultiplexer d, who then distributes it to the consumers c_1 and c_2 via the channels out_1 and out_2 respectively. The access policy that we want to impose here is that the consumers c_1 and c_2 read data only from the producers p_1 and p_2, respectively.



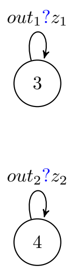
(a) The producers p_1 (top) and p_2 (bottom)


t=10 → skip: t
$5 \leq t \wedge t \leq 10 \rightarrow in_2?x\colon v$
$0 \leq t \wedge t \leq 7 \rightarrow in_1?x\colon v$
7
5
6
$ch!(2,x)$
$ch!(1,x)$

(b) The multiplexer m



(c) The demultiplexer d



(d) The two consumers c_1 (top) and c_2 (bottom)

Figure 2 The timed system for the gateway example.

We use clocks to model the temporal accesses to the resources of the system as required in [22]. In particular we use two clocks v (in the multiplexer) and r (in the demultiplexer) to model instantaneous transitions (time does not pass) and we use a clock t (in the multiplexer) to split the overall execution time of the timed system into periods of 10-time units. In each period the multiplexer m reads data from the channel in_1 only whenever $t \in [0, 7]$, while it reads data from the channel in_2 only whenever $t \in [5, 10]$. Whenever $t \in [5, 7]$ the multiplexer chooses non-deterministically to read either from in_1 or in_2. The multiplexer then transports the data together with a constant, using the dyadic channel ch to the demultiplexer d; the constant is used as a mark to indicate the source of the data. Finally, the demultiplexer delivers the data to the right consumer according to the constant.

The expressions e, guards g and conditions c that label the locations are defined as follows using boolean tests b:

$$\begin{array}{lll}
e & ::= & e_1 \,\mathsf{op}_a\, e_2 \mid x \mid n \\
b & ::= & \mathsf{tt} \mid \mathsf{ff} \mid e_1 \,\mathsf{op}_r\, e_2 \mid \neg b \mid b_1 \wedge b_2 \\
g & ::= & b \mid r \,\mathsf{op}_c\, n \mid (r_1 - r_2) \,\mathsf{op}_c\, n \mid g_1 \wedge g_2 \\
c & ::= & b \mid r \,\mathsf{op}_d\, n \mid (r_1 - r_2) \,\mathsf{op}_d\, n \mid c_1 \wedge c_2
\end{array}$$

The arithmetic operators op_a and the relational operators op_r are as usual. For comparisons of clocks we use the operators $\mathsf{op}_c \in \{<, \leq, =, \geq, >\}$ in guards and the less permissive set of operators $\mathsf{op}_d \in \{<, \leq, =\}$ in conditions.

3 Information Flow Instrumented Semantics

3.1 Behaviours

The transitions of the timed systems are labeled with behaviours. A *behaviour* records information relevant to the action that has occurred and also information about the processes that were involved in the action. Formally a behavior takes the form

$$\mathfrak{b} \in \mathcal{B}_{local} \cup \mathcal{B}_{com}$$

where

$$\mathcal{B}_{local} = \mathbf{P} \times \overrightarrow{\mathbf{Var}} \times \overrightarrow{\mathbf{Exp}}$$

are the behaviours that occur due to assignments and

$$\mathcal{B}_{com} = \mathbf{P} \times \mathbf{Chan} \times \overrightarrow{\mathbf{Var}} \times \overrightarrow{\mathbf{Exp}} \times \mathbf{P}$$

are the behaviours that occur due to the communication between two processes. We write $\overrightarrow{\mathbf{Var}}$ and $\overrightarrow{\mathbf{Exp}}$ for the sets of vectors with elements over the data variables and arithmetic expressions respectively.

For instance, the local behaviour $p : (\vec{x}, \vec{e})$ records that the process p has performed an assignment in which the vector $\vec{e}$ is used to modify the variables of the vector $\vec{x}$, while the behaviour $p : ch(\vec{x}, \vec{e}) : p'$ records that a communication between the processes p (the sender) and p' (the receiver) has happened, using the channel ch, and the vector $\vec{e}$ is the vector of expressions whose values have been communicated and have been bound to the variables of the vector $\vec{x}$.

In all of the behaviours, the vectors that are being used must have the same length while for the delay action we will write the empty behaviour ϵ.

3.2 Operational Semantics

To specify the semantics of timed systems, let σ be a state mapping data variables to values (which we take to be integers), let δ be a clock assignment mapping clocks to non-negative reals and let κ be a mapping from data variables to sets of processes which we will call *writers*. The mapping κ is used to monitor the explicit flows that occur from the assignments and the communication between two processes in the system; we explain the use of κ in more detail in a while. We then have total semantic functions $[\![.]\!]$ for evaluating the expressions, boolean tests, guards, and conditions; we evaluate expressions either with a state σ or the mapping κ, where for the first case the evaluation returns a value and in the second it returns the writers of the expression. The evaluation of boolean expressions only depends on the states, whereas that of guards and conditions also depend on the clock assignments.

The configurations of a timed system $TS = (TA_i)_{i \leq n}$ are of the form $\langle \vec{q}, \sigma, \delta, \kappa \rangle$, where $\vec{q}$ is a vector of nodes and we write $\vec{q}(i)$ for the i-th element of the vector $\vec{q}$, $\vec{q}[q'/q]$ to substitute the node q with the node q' in $\vec{q}$, we have that $\forall i : \vec{q}(i) \in \mathsf{Q}_i$ and finally we shall assume that the sets of nodes of the processes are mutually disjoint ($\forall i \neq j : \mathsf{Q}_i \cap \mathsf{Q}_j = \emptyset$).

The transitions of a timed system take the form

$$\langle \vec{q}_s, \sigma, \delta, \kappa \rangle \xRightarrow{\mathfrak{b}} \langle \vec{q}_t, \sigma', \delta', \kappa' \rangle$$

and the initial configurations are of the form $\langle \vec{q}_\circ, \sigma, \lambda r.0, \kappa_0 \rangle$ where $\vec{q}_\circ$ is the vector whose elements are the initial locations of the timed automata of the system and κ_0 maps each variable to the process that it belongs to ($\kappa_0(x) = \{p\}$ iff $x \in \mathbf{Var}_p$). The transition relation is given in Table 1.

The rule for the assignment, ensures that the guard is satisfied in the starting configuration and updates the mappings σ, δ, κ and the location of the process p_j and finally ensures that the invariant is satisfied in the resulting configuration. The behaviour $p_j : (\vec{x}, \vec{e})$ records

Table 1 Semantics for Timed Systems.

$$\langle \vec{q}_s, \sigma, \delta, \kappa\rangle \overset{p_j:(\vec{x},\vec{e})}{\Longrightarrow} \langle \vec{q}_t, \sigma', \delta', \kappa'\rangle \quad \text{if} \begin{cases} (q, g \rightarrow \vec{x} := \vec{e}\colon \vec{r}, q') \text{ is in } \mathsf{E}_j \\ [\![g]\!](\sigma,\delta) = \mathsf{tt} \\ \sigma' = \sigma[\vec{x} \mapsto [\![\vec{e}]\!]\sigma] \\ \delta' = \delta[\vec{r} \mapsto \vec{0}] \\ \kappa' = \kappa[\vec{x} \mapsto [\![\vec{e}]\!]\kappa] \\ \vec{q}_t = \vec{q}_s[q'/q] \\ \bigwedge_{i=1}^{n} [\![\mathsf{I}_i(\vec{q}_t(i))]\!](\sigma',\delta') = \mathsf{tt} \end{cases}$$

$$\langle \vec{q}_s, \sigma, \delta, \kappa\rangle \overset{p_h:ch(\vec{x},\vec{e}):p_l}{\Longrightarrow} \langle \vec{q}_t, \sigma', \delta', \kappa'\rangle \quad \text{if} \begin{cases} h \neq l \\ (q_1, g_1 \rightarrow ch!\vec{e}\colon \vec{r}_1, q_1') \text{ is in } \mathsf{E}_h \\ (q_2, g_2 \rightarrow ch?\vec{x}\colon \vec{r}_2, q_2') \text{ is in } \mathsf{E}_l \\ \sigma' = \sigma[\vec{x} \mapsto [\![\vec{e}]\!]\sigma] \\ \delta' = (\delta[\vec{r}_1 \mapsto \vec{0}])[\vec{r}_2 \mapsto \vec{0}] \\ \kappa' = \kappa[\vec{x} \mapsto [\![\vec{e}]\!]\kappa] \\ \vec{q}_t = \vec{q}_s[q_1'/q_1][q_2'/q_2] \\ \bigwedge_{i=1}^{n} [\![\mathsf{I}_i(\vec{q}_t(i))]\!](\sigma',\delta') = \mathsf{tt} \end{cases}$$

$$\langle \vec{q}, \sigma, \delta, \kappa\rangle \overset{\epsilon}{\Longrightarrow} \langle \vec{q}, \sigma, \delta', \kappa\rangle \quad \text{if} \begin{cases} \exists\, d > 0 : \delta' = \lambda r.\, \delta(r) + d, \\ \bigwedge_{i=1}^{n} [\![\mathsf{I}_i(\vec{q}(i))]\!](\sigma,\delta') = \mathsf{tt} \end{cases}$$

that the process p_j is performing an assignment to the vector $\vec{x}$ using the vector $\vec{e}$, and κ' records the information flow that occurs due to this behaviour, by updating the writers of each variable $\vec{x}(i)$ with the writers of the expression $\vec{e}(i)$, where for a single expression e', $[\![e']\!]\kappa = \bigcup_{y \in fv(e')} \kappa(y)$ and $fv(e')$ is the set of free variables occuring in e'.

To understand the rule for the communication one could see it as an assignment of the form $\vec{x} := \vec{e}$ where $\vec{e}$ are the expressions which are used at the channel output action and $\vec{x}$ the variables that are used in the channel input action.

Finally, the delay rule only modifies the clock assignment with a delay d ensuring that the invariant is satisfied in the resulting configuration. The mapping κ remains the same since the delay action produces the empty behaviour ϵ.

▶ **Example 2.** To see how the semantics for the κ mapping works, return to Example 1 and consider the transition

$$\langle \vec{q}, \sigma, \delta, \kappa\rangle \overset{p_1:in_1(x,x_1):m}{\Longrightarrow} \langle \vec{q}[6/5], \sigma[x \mapsto \sigma(x_1)], \delta[v \mapsto 0], \kappa[x \mapsto \{p_1\}]\rangle$$

which corresponds to the communication between the the producer p_1 and the multiplexer m. We have that $\vec{q} = (1, 2, 5, 8, 3, 4)$, and let

$$\kappa = [x_1 \mapsto \{p_1\}, x_2 \mapsto \{p_2\}, x \mapsto \{m\}, y \mapsto \{m\}, z \mapsto \{d\}, z_1 \mapsto \{c_1\}, z_2 \mapsto \{c_2\}]$$

and the resulting mapping $\kappa[x \mapsto \{p_1\}]$ records that p_1 has written its value into the variable x, since there is an explicit flow from the variable x_1 to the variable x and x_1 has previously been written by p_1.

4 Time Dependent Policies in BTCTL

In this section, we present our behaviour based logic **BTCTL** which serves to specify time-dependent security policies for access control, based on the behaviours of the system. The access control policies can then be enforced statically before the execution of the system.

4.1 The Logic

The syntax of the **BTCTL** formulas ϕ is given by

$$\phi ::= g \mid set_1 \,\mathsf{rel}\, set_2 \mid \forall\Box_{\mathfrak{b}}(\phi_1, \phi_2) \mid \phi_1 \wedge \phi_2 \mid \neg\phi$$

where

$$set ::= \underline{e} \mid W\,.$$

We have basic formulas which can be either a guard g, or relations between two sets of writers, set_1 rel set_2, where $\mathsf{rel} = \{\subseteq, \supseteq\}$. The underlined set expression $\underline{e}$ denotes the set of writers of the expression e and $W \in \mathbb{P}(\mathbf{P})$ is a set of writers. We use the box operator $\forall\Box_{\mathfrak{b}}(\phi_1, \phi_2)$ to speak about pre- and post-conditions whenever the non-empty behaviour $\mathfrak{b} \neq \epsilon$ happens. Informally speaking, a configuration γ will satisfy the $\forall\Box_{\mathfrak{b}}(\phi_1, \phi_2)$ formula whenever for all of the system runs starting at γ, if a transition labelled with the behaviour $\mathfrak{b}$ occurs, then ϕ_1 should hold at the configuration before the transition and ϕ_2 at the configuration after it. As we will see shortly, the box operator will be the key formula to express access control policies. The $\neg\phi$ and $\phi_1 \wedge \phi_2$ cases are the usual ones. Finally, we sometimes write $\phi_1 \Rightarrow \phi_2$ for $\neg(\phi_1 \wedge \neg\phi_2)$.

► **Example 3.** Going back to Example 1, each of the variables has a time-dependent policy which specifies the maximum set of permitted writers of the variable. We are interested in the policies of the variables of the multiplexer, the demultiplexer and the two consumers:

$$\begin{aligned}
P_x &= (0 \le t \wedge t < 5 \Rightarrow \underline{x} \subseteq \{p_1\})\wedge\\
&\quad\;(5 \le t \wedge t \le 7 \Rightarrow \underline{x} \subseteq \{p_1, p_2\})\wedge\\
&\quad\;(7 < t \wedge t \le 10 \Rightarrow \underline{x} \subseteq \{p_2\})\,,\\
P_y &= \underline{y} \subseteq \{m\}\ ,\\
P_z &= (0 \le t \wedge t \le 7 \wedge y = 1 \Rightarrow \underline{z} \subseteq \{p_1\})\wedge\\
&\quad\;(5 \le t \wedge t \le 10 \wedge y = 2 \Rightarrow \underline{z} \subseteq \{p_2\})\,,\\
P_{z_1} &= \underline{z_1} \subseteq \{p_1\}\ ,\\
P_{z_2} &= \underline{z_2} \subseteq \{p_2\}\ .
\end{aligned}$$

The first line of the policy for the variable x, expresses that whenever $t \in [0, 5)$, only the process p_1 is allowed to write data to x, while both p_1 and p_2 may write to x if $t \in [5, 7]$ and similarly to the first line, if $t \in (7, 10]$ then only p_2 can write to x. On the other hand, looking at the policy for the variable y, a write action to y is allowed only by the multiplexer. The rest of the policies can be explained accordingly.

We then perform the enforcement of the access control policies by checking the following formulas:

$$\begin{aligned}
\Phi_x &= \forall\Box_{p_1:in_1(x,x_1):m}(\mathsf{tt}, P_x) \wedge \forall\Box_{p_2:in_2(x,x_2):m}(\mathsf{tt}, P_x)\,,\\
\Phi_{y,z} &= \forall\Box_{m:ch((y,z),(1,x)):d}(\mathsf{tt}, P_y \wedge P_z) \wedge \forall\Box_{m:ch((y,z),(2,x)):d}(\mathsf{tt}, P_y \wedge P_z)\,,\\
\Phi_{z_1} &= \forall\Box_{d:out_1(z_1,z):c_1}(\mathsf{tt}, P_{z_1})\,,\\
\Phi_{z_2} &= \forall\Box_{d:out_2(z_2,z):c_2}(\mathsf{tt}, P_{z_2})\,.
\end{aligned}$$

Each of the formulas express that whenever someone is writing to the variable (or variables) appearing as a subscript, then the policy of the variable (or variables) is imposed as a post condition. The variable x is accessed (someone is writing data to x) whenever p_1 and p_2 communicates with the multiplexer and thus we have to impose the policy of the variable x for both of those actions, while the variables y and z are being accessed whenever the multiplexer communicates with the demultiplexer and that happens with two communication actions. Similarly, we define the formula for the variables z_1 and z_2.

4.2 Semantics of BTCTL

The formal rules that define whenever a configuration γ satisfies a **BTCTL** formula ϕ are given below:

$$\begin{array}{lll}
\gamma \models g & \text{iff} & \gamma = \langle \vec{q}, \sigma, \delta, \kappa \rangle \Rightarrow [\![g]\!](\sigma, \delta) \\
\gamma \models set_1 \,\mathsf{rel}\, set_2 & \text{iff} & \gamma = \langle \vec{q}, \sigma, \delta, \kappa \rangle \Rightarrow [\![set_1]\!]\kappa \,\mathsf{rel}\, [\![set_2]\!]\kappa \\
\gamma \models \forall\Box_{\mathfrak{b}}(\phi_1, \phi_2) & \text{iff} & \forall \gamma_0 \overset{\mathfrak{b}_1}{\Rightarrow} \gamma_1 \overset{\mathfrak{b}_2}{\Rightarrow} .. \in \mathbf{Trace}_\gamma : \\
 & & \forall i \geq 1 : \mathfrak{b}_i = \mathfrak{b} \Rightarrow \gamma_{i-1} \models \phi_1 \text{ and } \gamma_i \models \phi_2 \\
\gamma \models \phi_1 \wedge \phi_2 & \text{iff} & \gamma \models \phi_1 \text{ and } \gamma \models \phi_2 \\
\gamma \models \neg\phi & \text{iff} & \gamma \not\models \phi
\end{array}$$

A guard g is then satisfied by a configuration γ whenever g holds in γ. For the case of the set relation rel, γ satisfies it whenever $set_1 \,\mathsf{rel}\, set_2$ evaluates to true and we do that check by lifting the definition of $[\![.]\!]\kappa$ to set expressions, by $[\![\underline{e}]\!]\kappa = [\![e]\!]\kappa$ and $[\![W]\!]\kappa = W$. A configuration γ satisfies the box formula $\forall\Box_{\mathfrak{b}}(\phi_1, \phi_2)$ whenever for all the execution paths that start from γ, if a behaviour $\mathfrak{b}'$ occurs and $\mathfrak{b}'$ is syntactically equal to $\mathfrak{b}$, then the pre-condition ϕ_1 has to hold at the configuration before the behaviour and the post-condition ϕ_2 at the configuration after it. The rest of the cases are the usual ones.

▶ **Example 4.** Consider now the prefix of an execution trace of the timed system from Example 1

$$pr = \gamma_0 \overset{p_1:in_1(x,x_1):m}{\Longrightarrow} \gamma_1 \overset{m:ch((y,z),(1,x)):d}{\Longrightarrow} \gamma_2 \overset{d:out_1(z_1,z):c_1}{\Longrightarrow} \gamma_3$$

where for the initial configuration $\gamma_0 = \langle \vec{q_0}, \sigma_0, \delta_0, \kappa_0 \rangle$, we have that $\vec{q_0} = (1, 2, 5, 8, 3, 4)$, σ_0 is arbitrary, $\delta_0 = \lambda c.0$ and

$$\kappa_0 = [x_1 \mapsto \{p_1\}, x_2 \mapsto \{p_2\}, x \mapsto \{m\}, y \mapsto \{d\}, z \mapsto \{d\}, z_1 \mapsto \{c_1\}, z_2 \mapsto \{c_2\}]$$

and for the rest of the configurations

$$\begin{array}{l}
\gamma_1 = \langle \vec{q_1}, \sigma_1, \delta_1, \kappa_1 \rangle, \vec{q_1} = \vec{q_0}[6/5], \sigma_1 = \sigma_0[x \mapsto \sigma_0(x_1)], \delta_1 = \delta_0[v \mapsto 0], \kappa_1 = \kappa_0[x \mapsto \{p_1\}] \\
\gamma_2 = \langle \vec{q_2}, \sigma_2, \delta_2, \kappa_2 \rangle, \vec{q_2} = \vec{q_1}[5/6][9/8], \sigma_2 = \sigma_1[y \mapsto 1, z \mapsto \sigma_1(x)], \delta_2 = \delta_1[r \mapsto 0], \\
\qquad \kappa_2 = \kappa_1[y \mapsto \emptyset, z \mapsto \{p_1\}] \\
\gamma_3 = \langle \vec{q_3}, \sigma_3, \delta_3, \kappa_3 \rangle, \vec{q_3} = \vec{q_1}[8/9], \sigma_3 = \sigma_2[z_1 \mapsto \sigma_2(z)], \delta_3 = \delta_2, \kappa_3 = \kappa_2[z_1 \mapsto \{p_1\}].
\end{array}$$

Now consider the formulas Φ_x, $\Phi_{y,z}$, Φ_{z_1} from Example 3 and to illustrate how the semantics work for the box operator, we will do the appropriate checks for those formulas on pr.

The formula Φ_x is the conjuction of two box operators, where for the first one because of the behaviour $p_1 : in_1(x, x_1) : m$ of the transition $\gamma_0 \overset{p_1:in_1(x,x_1):m}{\Longrightarrow} \gamma_1$, we have to check that $\gamma_1 \models P_x$ where

$$\begin{array}{l}
P_x = (0 \leq t \wedge t < 5 \Rightarrow \underline{x} \subseteq \{p_1\}) \wedge \\
\qquad (5 \leq t \wedge t \leq 7 \Rightarrow \underline{x} \subseteq \{p_1, p_2\}) \wedge \\
\qquad (7 < t \wedge t \leq 10 \Rightarrow \underline{x} \subseteq \{p_2\}) .
\end{array}$$

This check evaluates to true, since γ satisfies only the guard of the first line of the policy and $\kappa_1(x) = \{p_1\}$. For the transition $\gamma_1 \overset{m:ch((y,z),(1,x)):d}{\Longrightarrow} \gamma_2$, because of the formula $\Phi_{y,z}$ we have to check that $\gamma_2 \models P_y \wedge P_z$ where

$$\begin{aligned} P_y &= \underline{y} \subseteq \{m\}\ , \\ P_z &= (0 \leq t \wedge t \leq 7 \wedge y = 1 \Rightarrow \underline{z} \subseteq \{p_1\}) \wedge \\ &\quad\ (5 \leq t \wedge t \leq 10 \wedge y = 2 \Rightarrow \underline{z} \subseteq \{p_2\})\ . \end{aligned}$$

This check evaluates to true, since $\kappa_2(y) = \emptyset$ and γ_2 satisfies only the condition at the first line of the policy P_z and also $\kappa_2(z) = \{p_1\}$. Finally for the last transition $\gamma_2 \overset{d:out_1(z_1,z):c_1}{\Longrightarrow} \gamma_3$, because of the Φ_{z_1} formula, we have to check that $\gamma_3 \models P_z$, and this check evaluates to true, since $\kappa_3(z_1) = \{p_1\}$ and $P_z = \underline{z_1} \subseteq \{p_1\}$.

5 Reduction of BTCTL to TCTL$^+$

In this section, we perform a transformation of the original time system, and of the **BTCTL** formulas. The transformation is based on the work done in [16], where the action-based logic ATCTL (action-TCTL) is being reduced to TCTL [2]. A transformed formula produces a formula in **TCTL**$^+$, a logic based on TCTL and in the next section we show how a fragment of **TCTL**$^+$ can be handled by the model checker UPPAAL [30].

5.1 Behaviour Automata

A timed system $TS = (TA_i)_{i \leq n}$ yields a behaviour automaton $BA = (v_\circ, \mathsf{E}, \mathsf{I}, \mathsf{Q}, \mathsf{L})$, which is a kind of timed automaton in that it is the product automaton of the system, extended to contain *auxiliary vertices* that represent the actions of the system and a labelling function L that assigns to each vertex a property. A property is either a behaviour or a location vector of the system TS; auxiliary vertices of the system will be labeled with the behaviour that corresponds to the particular action of the vertex, while *genuine vertices* that represent locations of the system TS are labeled with a location vector. The initial vertex $v_\circ$ will be labeled with the initial location vector of the system $\vec{q}_\circ$. The behaviour automaton BA has the same set of variables as the timed system TS, while for the clock variables it has an extra clock t. Similarly to the timed automata, E is a finite set of edges, the mapping I imposes an invariant on each vertex and Q is the finite set of vertices.

The algorithm for constructing the edges E, the labelling functions I and L and the set of vertices $\mathsf{Q} = Q_{gen} \cup Q_{aux}$ ($Q_{gen} \cap Q_{aux} = \emptyset$) where Q_{gen} and Q_{aux} contain the genuine and auxiliary vertices respectively, is given in Figure 3.

In the first step, we create the genuine vertices and we label them with the invariant of the location vector that they represent; each of those vertices is inserted in Q_{gen}, which will be used in the next steps to create the auxiliary vertices.

In step 2 we create the auxiliary vertices and the edges that correspond to the assignment actions of the system. For each process p_i, we start looking at all of its assigment edges $(q_i, g \rightarrow \vec{x} := \vec{e} \colon \vec{r}, q'_i) \in E_i$. For each one of those edges and for all the vertices $v_s \in Q_{gen}$ and $v_t \in Q_{gen}$, where the label of v_s, $\mathsf{L}(v_s)$ corresponds to a vector location where this assignment could have been performed and would have moved the system to the location $\mathsf{L}(v_t)$, we create the edges $(v_s, g \rightarrow \texttt{skip} \colon t, v)$ and $(v, \vec{x} := \vec{e} \colon \vec{r}, v_t)$, where v is a fresh auxiliary vertex; whereas in the construction of the product automaton one would have constructed only the edge $(v_s, g \rightarrow \vec{x} := \vec{e} \colon r, v_t)$. The auxiliary vertex v is labelled with the assignment behaviour

```
(1)  let $Q_{gen} = \emptyset$; let $Q_{aux} = \emptyset$;
     for all $\vec{q}$: create fresh $v$; let $\mathsf{L}(v) = \vec{q}$; let $\mathsf{I}(v) = \bigwedge_{i=1}^{n} \mathsf{I}_i(\vec{q}(i))$; insert $v$ in $Q_{gen}$
(2)  for all $(q_i, g \to \vec{x} := \vec{e}\colon \vec{r}, q_i') \in E_i$:
       for all $v_s \in Q_{gen}$, $v_t \in Q_{gen}$ such that $\begin{cases} \mathsf{L}(v_s)(i) = q_i \\ \mathsf{L}(v_t)(i) = q_i' \\ \forall j : j \neq i : \mathsf{L}(v_s)(j) = \mathsf{L}(v_t)(j) \end{cases}$ :
         create fresh $v$;
         insert $(v_s, g \to \texttt{skip}\colon t, v)$ in $\mathsf{E}$; insert $(v, \vec{x} := \vec{e}\colon \vec{r}, v_t)$ in $\mathsf{E}$ ;
         let $\mathsf{L}(v) = p_i : (\vec{x}, \vec{e})$; let $\mathsf{I}(v) = (t = 0) \wedge \mathsf{I}(v_t)[\vec{e}/\vec{x}][\vec{0}/\vec{r}]$; insert $v$ in $Q_{aux}$
(3)  for all $(q_i, g_1 \to ch!\vec{e}\colon \vec{r}_1, q_i') \in E_i$ and $(q_j, g_2 \to ch?\vec{x}\colon \vec{r}_2, q_j') \in E_j$ such that $i \neq j$:
       for all $v_s \in Q_{gen}$, $v_t \in Q_{gen}$ such that $\begin{cases} \mathsf{L}(v_s)(i) = q_i \wedge \mathsf{L}(v_s)(j) = q_j \\ \mathsf{L}(v_t)(i) = q_i' \wedge \mathsf{L}(v_t)(j) = q_j' \\ \forall l : l \neq i \wedge l \neq j : \mathsf{L}(v_s)(l) = \mathsf{L}(v_t)(l) \end{cases}$ :
         create fresh $v$; let $g = g_1 \wedge g_2$; let $\vec{r} = \vec{r}_1\vec{r}_2$;
         insert $(v_s, g \to \texttt{skip}\colon t, v)$ in $\mathsf{E}$; insert $(v, \vec{x} := \vec{e}\colon \vec{r}, v_t)$ in $\mathsf{E}$;
         let $\mathsf{L}(v) = p_i : ch(\vec{x}, \vec{e}) : p_j$; let $\mathsf{I}(v) = (t = 0) \wedge \mathsf{I}(v_t)[\vec{e}/\vec{x}][\vec{0}/\vec{r}]$; insert $v$ in $Q_{aux}$
(4)  let $\mathsf{Q} = Q_{gen} \cup Q_{aux}$
```

Figure 3 The algorithm for constructing E, I, Q and L.

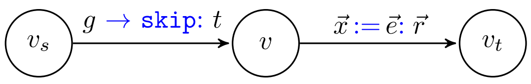

$g \to \texttt{skip}\colon t$
$\vec{x} := \vec{e}\colon \vec{r}$
v_s
v
v_t

Figure 4 Edge construction of BA.

$p_i : (\vec{x}, \vec{e})$ and its invariant is being set to $(t = 0) \wedge \mathsf{I}(v_t)[\vec{e}/\vec{x}][\vec{0}/\vec{r}]$, to first ensure that the action of the edge leaving v will be performed instantenous and secondly that we can not get stuck at an auxiliary vertex. Figure 4 illustrates the construction and note that each auxiliary vertex v has *exactly one predecessor* and *exactly one successor*.

Similarly to step 2, in step 3 we construct the auxiliary vertices for the communication actions of the system and finally in step 4 we define the set Q.

5.2 Trace Equivalence

From the construction of the behaviour automaton BA, it is essential that every execution trace in the original system TS can be interpreted as an execution trace in the behaviour automaton BA and vice versa. Particularly, each transition in the system TS is equivalent to a single step transition (in the case of a delay) or a two-step transition (in the case of an action) in its behaviour automaton BA. To overcome the vagueness of this explanation we will later define an equivalence relation between execution traces of the system TS and the behaviour automaton BA.

First, we give the operational semantics of the behaviour automata in Table 2. The semantics is similar to the semantics of the timed automata, however now, the transitions are not labelled with behaviours.

Table 2 Semantics for Behaviour Automata.

$$\langle v_s,\sigma,\delta,\kappa\rangle \longrightarrow \langle v_t,\sigma',\delta',\kappa'\rangle \quad \text{if} \begin{cases} (v_s, g \rightarrow \vec{x}:=\vec{e}\colon \vec{r}, v_t) \text{ is in } \mathsf{E} \\ [\![g]\!](\sigma,\delta) = \mathsf{tt} \\ \sigma' = \sigma[\vec{x}\mapsto [\![\vec{e}]\!]\sigma] \\ \delta' = \delta[\vec{r}\mapsto \vec{0}] \\ \kappa' = \kappa[\vec{x}\mapsto [\![\vec{e}]\!]\kappa] \\ [\![\mathsf{I}(v_t)]\!](\sigma',\delta') = \mathsf{tt} \end{cases}$$

$$\langle v,\sigma,\delta,\kappa\rangle \longrightarrow \langle v,\sigma,\delta',\kappa\rangle \quad \text{if} \begin{cases} \exists\, d>0 : \delta' = \lambda r.\,\delta(r)+d, \\ [\![\mathsf{I}(v)]\!](\sigma,\delta') = \mathsf{tt} \end{cases}$$

Now let γ and γ' to be two configurations of a timed system TS and its behaviour automaton BA respectively. We define the relation $\cong\colon \mathbf{Config}_{TS} \times \mathbf{Config}_{BA} \to \{\mathsf{tt},\mathsf{ff}\}$ to be

$$\langle \vec{q},\sigma,\delta,\kappa\rangle \cong \langle v,\sigma',\delta',\kappa'\rangle \quad \text{iff} \quad \begin{array}{l} \vec{q} = \mathsf{L}(v)\,, \\ \sigma = \sigma'\,, \\ \forall r\in \mathbf{Clock} : \delta(r) = \delta'(r)\,, \\ \kappa = \kappa'\,, \end{array}$$

where we recall that **Clock** is the set of the clocks appearing in the system TS and thus the clock t of the behaviour automaton BA is *not* included in **Clock**. It is straightforward by the definition of $\cong$ that configurations of the system TS can only be related with configurations that correspond to genuine vertices in the behaviour automaton BA.

For the behaviour automata BA, we define a *macro transition* t to be a single step delay transition $\gamma_s' \longrightarrow \gamma_t'$ or a two-step transition $\gamma_s' \longrightarrow \gamma_{aux} \longrightarrow \gamma_t'$, where γ_{aux} is an *auxiliary configuration* (a configuration that corresponds to an auxiliary vertex) and γ_s and γ_t are *genuine configurations* (configurations that correspond to genuine vertices). We then lift the definition of $\cong$ to single step transitions of the system TS and macro transitions of the behaviour automaton BA as

$$\gamma_s \overset{\epsilon}{\Longrightarrow} \gamma_t \cong \gamma_s' \longrightarrow \gamma_t' \text{ iff } \begin{cases} \gamma_s \cong \gamma_s' \\ \gamma_t \cong \gamma_t' \end{cases}$$

$$\gamma_s \overset{\mathfrak{b}}{\Longrightarrow} \gamma_t \cong \gamma_s' \longrightarrow \gamma_{aux} \longrightarrow \gamma_t' \text{ iff } \begin{cases} \gamma_s \cong \gamma_s' \\ \gamma_t \cong \gamma_t' \\ \gamma_{aux} = \langle v,\sigma,\delta,\kappa\rangle \Rightarrow \mathfrak{b} = \mathsf{L}(v) \end{cases}$$

Now for each genuine configuration γ' of the BA, we have that every execution trace $tr' = \gamma_0' \longrightarrow \gamma_1'.... \in \mathbf{Trace}_{\gamma'}$ of γ', with length greater than 0, can be parsed as a *macro transition trace* $T_{tr'} = t_1't_2't_3'.....$ where each t_j' is a macro transition. For example the finite execution trace $\gamma_0' \longrightarrow \gamma_1' \longrightarrow \gamma_{aux_1} \longrightarrow \gamma_2' \longrightarrow \gamma_{aux_2} \longrightarrow \gamma_3' \longrightarrow \gamma_4'$, which is a sequence of a delay, action(two-step), action(two-step), delay, will produce the macro transition trace $t_1't_2't_3't_4'$ where $t_1' = \gamma_0' \longrightarrow \gamma_1'$, $t_2' = \gamma_1' \longrightarrow \gamma_{aux_1} \longrightarrow \gamma_2'$, $t_3' = \gamma_2' \longrightarrow \gamma_{aux_2} \longrightarrow \gamma_3'$ and $t_4' = \gamma_3' \longrightarrow \gamma_4'$.

Similarly to the macro transition traces, for each configuration γ of the system TS, we can write each nonzero-length execution trace, $tr = \gamma_0 \overset{\mathfrak{b}_1}{\Longrightarrow} \gamma_1 \overset{\mathfrak{b}_2}{\Longrightarrow} \gamma_2... \in \mathbf{Trace}_\gamma$ of γ, as a *transition trace* $T_{tr} = t_1t_2...$ where $t_i = \gamma_{i-1} \overset{\mathfrak{b}_i}{\Longrightarrow} \gamma_i$ (for all $i \geq 1$).

Finally we lift the definition of $\cong$ to execution traces of length $n > 0$, that start in genuine configurations inside the timed system TS and its behaviour automaton BA as

$$tr \cong tr' \text{ iff } \begin{cases} T_{tr} \text{ and } T_{tr'} \text{ have the same length} \\ \forall i \geq 1 : T_{tr}(i) \cong T_{tr'}(i) \end{cases}$$

$tr \cong tr'$, then results to true if and only if the transition trace T_{tr} of tr and the macro transition trace $T_{tr'}$ of tr' have the same length and they are equivalent stepwise.

The following fact follows from the method of constructing a behaviour automaton and states that equivalent configurations in the timed system TS and its behaviour automaton BA, produce equivalent execution traces.

▶ **Fact 5.** *For every timed system TA, its behaviour automaton BA and two configurations γ and γ' such that $\gamma \cong \gamma'$ we have that:*

- $\forall tr \in \textbf{\textit{Trace}}_{\gamma} : \exists tr' \in \textbf{\textit{Trace}}_{\gamma'} : tr \cong tr'$,
- $\forall tr' \in \textbf{\textit{Trace}}_{\gamma'} : \exists tr \in \textbf{\textit{Trace}}_{\gamma} : tr \cong tr'$.

5.3 TCTL$^+$

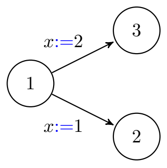
For the behaviour automata, we define a new logic called **TCTL**$^+$ patterned after TCTL [2], and the syntax of a **TCTL**$^+$ formula ψ is given by

$$\psi ::= prop \mid g \mid set_1 \,\mathsf{rel}\, set_2 \mid \forall\Box\psi \mid \exists(\psi_1 U \psi_2) \mid \neg\psi \mid \psi_1 \wedge \psi_2 \,.$$

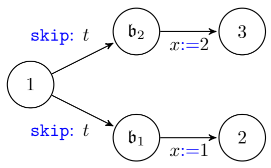
The basic formula *prop* is a proposion which is either a behaviour or a location vector and it holds in a configuration if its vertex is labelled with *prop*; the rest of the basic formulas are the same as in **BTCTL** . The $\forall\Box\psi$ formula holds in a configuration if for all of its execution traces, ψ holds in all the configurations of the trace, while for the $\exists(\psi_1 U \psi_2)$ to hold, it is sufficient that there exists an execution trace where ψ_1 holds for a prefix of the trace and eventually ψ_2 also holds. The rest of the operators are the same as in **BTCTL** . The formal semantics of the **TCTL**$^+$ is given by:

$$\begin{array}{lll}
\gamma' \models prop & \text{iff} & \gamma' = \langle v, \sigma, \delta, \kappa \rangle \Rightarrow \mathsf{L}(v) = prop \\
\gamma' \models g & \text{iff} & \gamma' = \langle v, \sigma, \delta, \kappa \rangle \Rightarrow [\![g]\!](\sigma, \delta) \\
\gamma' \models set_1 \,\mathsf{rel}\, set_2 & \text{iff} & \gamma' = \langle v, \sigma, \delta, \kappa \rangle \Rightarrow [\![set_1]\!]\kappa \,\mathsf{rel}\, [\![set_2]\!]\kappa \\
\gamma' \models \forall\Box\psi & \text{iff} & \forall \gamma'_0 \longrightarrow \gamma'_1 \longrightarrow \gamma'_2 .... \in \textbf{Trace}_{\gamma'} : \forall i \geq 0 : \gamma'_i \models \psi \\
\gamma' \models \exists(\psi_1 U \psi_2) & \text{iff} & \exists \gamma'_0 \longrightarrow \gamma'_1 \longrightarrow \gamma'_2 .... \in \textbf{Trace}_{\gamma'} : \\
 & & \exists i : \gamma'_i \models \psi_2 \text{ and } \forall j < i : \gamma'_j \models \psi_1 \\
\gamma' \models \psi_1 \wedge \psi_2 & \text{iff} & \gamma' \models \psi_1 \text{ and } \gamma' \models \psi_2 \\
\gamma' \models \neg\psi & \text{iff} & \gamma' \not\models \psi
\end{array}$$

Our goal is to transform a **BTCTL** formula ϕ into a **TCTL**$^+$ formula ψ and then show that for two equivalent configurations γ and γ' of a timed system TS and its behaviour automaton BA respectively, checking the formula ϕ in γ it is sufficient to check the transformed formula ψ in γ' and vice versa. We perform the transformation of the formulas using a function $\mathcal{T}[\![.]\!]$



(a) The timed automaton of the process p



(b) The behaviour automaton of p

as follows

$$
\begin{aligned}
\mathcal{T}[\![g]\!] &= g\,,\\
\mathcal{T}[\![set_1 \,\mathsf{rel}\, set_2]\!] &= set_1 \,\mathsf{rel}\, set_2\,,\\
\mathcal{T}[\![\forall\Box_{\mathfrak{b}}(\phi_1,\phi_2)]\!] &= \forall\Box(\mathfrak{b} \Rightarrow (\mathcal{T}[\![\phi_1]\!] \;\wedge \exists(\mathfrak{b}\; U(\neg\mathfrak{b} \wedge \mathcal{T}[\![\phi_2]\!]))))\,,\\
\mathcal{T}[\![\phi_1 \wedge \phi_2]\!] &= \mathcal{T}[\![\phi_1]\!] \wedge \mathcal{T}[\![\phi_2]\!]\,,\\
\mathcal{T}[\![\neg\phi]\!] &= \neg\mathcal{T}[\![\phi]\!]\,.
\end{aligned}
$$

For the special cases $\forall\Box_{\mathfrak{b}}(\mathtt{tt},\phi_2)$ (ϕ_2 is not $\mathtt{tt}$) and $\forall\Box_{\mathfrak{b}}(\phi_1,\mathtt{tt})$ (ϕ_1 is not $\mathtt{tt}$) we shall omit the transformed formula that corresponds to the trivial formula $\mathtt{tt}$, by writting $\mathcal{T}[\![\forall\Box_b(\mathtt{tt},\phi_2)]\!] = \forall\Box(\mathfrak{b} \Rightarrow \mathfrak{b}\; U(\neg\mathfrak{b} \wedge \mathcal{T}[\![\phi_2]\!]))$ for the first case and $\mathcal{T}[\![\forall\Box_{\mathfrak{b}}(\phi_1,\mathtt{tt})]\!] = \forall\Box(\mathfrak{b} \Rightarrow \mathcal{T}[\![\phi_1]\!])$ for the second case. Finally, we shall assume that formulas in the pre-condition of the $\forall\Box_{\mathfrak{b}}(\phi_1,\phi_2)$ are not nested. To justify this assumption consider the following example

▶ **Example 6.** Consider the timed automaton of a process p (Figure 5a) with a variable x and a clock r, and its behaviour automaton BA (Figure 5b), where $\mathfrak{b}_1 = p : (x,1)$ and $\mathfrak{b}_2 = p : (x,2)$ are the behaviours of the actions x:=1 and x:=2 respectively, and all the location invariants in the timed automaton of p are $\mathtt{tt}$.

Now let $\phi = \forall\Box_{\mathfrak{b}_1}(\forall\Box_{\mathfrak{b}_2}(\mathtt{tt}, x=1),\mathtt{tt})$ and observe that every initial configuation of the process p does not satisfy ϕ, whereas every initial configuration of the behaviour automaton does satisfy the transformed formula $\mathcal{T}[\![\phi]\!] = \forall\Box(\mathfrak{b}_1 \Rightarrow (\forall\Box(\mathfrak{b}_2 \Rightarrow \exists(\mathfrak{b}_2\; U(\neg\mathfrak{b}_2 \wedge x=1)))))$.

Since the proposed formula transformation is sufficient to express and enforce access control policies of our interest we leave the development of transformations that support the entire **BTCTL** as future work.

Finally, we state the correctness of the function $\mathcal{T}[\![.]\!]$ with the following theorem

▶ **Theorem 7.** *For a timed system TS, its behaviour automaton BA, a **BTCTL** formula ϕ and for every configuration γ and γ' of TS and BA respectively, we have that if $\gamma \cong \gamma'$ then*

$$\gamma \models \phi \textit{ iff } \gamma' \models \mathcal{T}[\![\phi]\!]$$

The proof of Theorem 7 can be found in Appendix A.

5.4 Reduction Complexity

We give a computation bound for the algorithm of Figure 3, that given a timed system $TS = (TA_i)_{i\leq n}$ constructs the behaviour automaton $BA = (v_\circ, \mathsf{E}, \mathsf{I}, \mathsf{Q}, \mathsf{L})$. Assuming that the computation time of all the simple operations (creation of fresh vertices, setting of invariants e.t.c) is constant, we have that : let $K = |\mathsf{Q}_1| + ... + |\mathsf{Q}_n|$ and $E = |\mathsf{E}_1| + ... + |\mathsf{E}_n|$ then the first part of the algorithm is bounded by K^n. The second part iterates over the assignement edges and all the pairs of the auxiliary vertices and that is bounded by $E \times K^{2n} \times n$, where

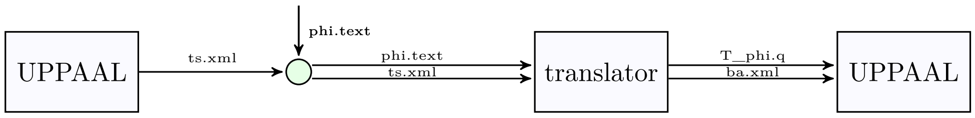


Figure 6 Architecture of the Translator.

n corresponds to the computation bound of checking the third condition of the branch of the for-loop. Similarly to the second part of the algorithm the third part is bounded by $E^2 \times K^{2n} \times n$ and therefore for the total sum of those bounds we obtain a complexity of $O(E^2 \times K^{2n} \times n)$. Finally, for a **BTCTL** formula ϕ the complexity of the transformation $\mathcal{T}[\![\phi]\!]$ is linear to the size of ϕ.

6 The Translator

We have implemented a translator in Java that works together with the model checker UPPAAL version 4.0 [30]. Figure 6 depicts the architecture of the translator.

UPPAAL is using a graphical interface in which one can model (draw) a system of timed automata. We first do that and next UPPAAL saves it as a file in the eXtensible Markup Language (XML) [33]; the xml file together with a text file that contains the desired property ϕ that we want to check, are being passed to the translator. The translator parses the two files and produces an xml file which contains the behaviour automaton of the system together with a UPPAAL query file that includes the property $\mathcal{T}[\![\phi]\!]$. The two files are imported to UPPAAL and then one can check if the desired property holds.

Since UPPALL does not allow nested formulas nor supports the operator $\exists \phi_1 U \phi_2$, we had to find a workaround for some of the transformed formulas. The guards g are translated directly; for the set_1 rel set_2, we model a set as a bit array since UPPALL supports multidimensional integer arrays and then we check the bit version of the relation rel. In case of the $\mathcal{T}[\![\forall\Box_{\mathfrak{b}}(\phi_1, \phi_2)]\!] = \forall\Box(\mathfrak{b} \Rightarrow (\mathcal{T}[\![\phi_1]\!] \;\wedge \exists(\mathfrak{b}\, U(\neg\mathfrak{b} \wedge \mathcal{T}[\![\phi_2]\!]))))$, UPPAAL allows labelling a vertex with a string (the name of the vertex) and thus auxiliary vertices with label $\mathfrak{b}$ have as a name a string that corresponds to the behaviour $\mathfrak{b}$. For the part $\mathfrak{b}\, U(\neg\mathfrak{b} \wedge \mathcal{T}[\![\phi_2]\!])$ we annotate the outgoing edges of the auxiliary vertices with an assignment to a fresh variable a that works as a switch. We switch on by $a := 1$, only when we leave the auxiliary vertex, and we switch off by $a := 0$, whenever we leave the successor of the auxiliary vertex. Thus the formula $\mathfrak{b}\, U(\neg\mathfrak{b} \wedge \mathcal{T}[\![\phi_2]\!])$ is transformed into the formula $a = 1 \Rightarrow \mathcal{T}[\![\phi_2]\!]$.

Finally, since the mapping κ is not part of the timed automata of UPPAAL, we first enumerate each variable and each process of the system and we then model κ as a two-dimensional array, whose first index corresponds to a variable and whose second to a process. For instance, if a variable x is enumerated with 1 and $\kappa(x) = \{p\}$, where p is a process of the system and p is enumerated with 2, then $\kappa[1][2] = 1$, while for any other index $j \neq 2$, $\kappa[1][j] = 0$, modelling in that way that only p has written data in x. The edges of the automaton are also annotated with assignments to κ to capture the updates to it whenever the system performs an action.

7 Conclusions

We have successfully shown how to enforce access control policies on Systems of Timed Automata using a behaviour-based logic. The logic allows specification of time, data's content and information flow dependent security policies, an essential need in the modern world of cyberphysical systems. We have developed a sound reduction of a substantial fragment of our logic to a logic based on TCTL [2], so that the model checking of the formulas can be performed by existing model checkers such as UPPAAL [30]. We implemented a translator which performs the reduction and together with UPPAAL it enforces access control policies. Finally, we illustrated our development using an example from the aerospace industry, where ensuring data's integrity is a life critical goal.

There are several ways in which we can extend our work. We are currently exploring how our development can be extended to capture more complex information flows such as *implicit* flows [31]. We have shown in [24] that the time aspect, as well as the non-deterministic semantics of Timed Automata, poses a challenge for that.

We are considering extensions to our logic that allow expressing richer access control policies and also how to develop a reduction which supports the entire syntax of the **BTCTL** logic. Another possibility is to explore new algorithms for determining if a formula of our logic holds in a timed system rather than reducing the formula to current TCTL-based logics.

References

1 Luca Aceto, Anna Ingolfsdottir, Kim Guldstrand Larsen, and Jiri Srba. *Reactive Systems: Modelling, Specification and Verification.* Cambridge University Press, 2007.

2 Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking in dense real-time. *Inf. Comput.*, 104(1):2–34, 1993.

3 Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

4 David A. Basin, Matús Harvan, Felix Klaedtke, and Eugen Zalinescu. Monitoring usage-control policies in distributed systems. *TIME*, pages 88–95, 2011.

5 Moritz Y. Becker, Cédric Fournet, and Andrew D. Gordon. Secpal: Design and semantics of a decentralized authorization language. *Journal of Computer Security*, 18(4):619–665, 2010.

6 Elisa Bertino, Piero A. Bonatti, and Elena Ferrari. Trbac: A temporal role-based access control model. *ACM Trans. Inf. Syst. Secur.*, 4(3):191–233, 2001.

7 Hsing-Chung Chen, Shiuh-Jeng Wang, Jyh-Horng Wen, Yung-Fa Huang, and Chung-Wei Chen. A generalized temporal and spatial role-based access control model. *JNW*, 5(8):912–920, 2010.

8 Carlo Combi, Roberto Posenato, Luca Viganò, and Matteo Zavatteri. Access controlled temporal networks. *ICAART (2)*, pages 118–131, 2017.

9 Carlo Combi, Luca Viganò, and Matteo Zavatteri. Security constraints in temporal role-based access-controlled workflows. *CODASPY*, pages 207–218, 2016.

10 Henry DeYoung, Deepak Garg, and Frank Pfenning. An authorization logic with explicit time. *CSF*, pages 143–165, 2008.

11 Sabrina De Capitani di Vimercati, Pierangela Samarati, and Ravi Sandhu. Access control. *Computing Handbook, 3rd ed.*, 1:47:1–25, 2014.

12 David F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–264, 2001.

13 Emsaieb Geepalla, Behzad Bordbar, and Kozo Okano. Verification of spatio-temporal role based access control using timed automata. *NESEA*, pages 1–6, 2012.

14 Yong-Zhong He, Zhen Han, and Ye Du. Context active rbac and its applications. *ISECS*, pages 1041–1044, 2008.

15 Xuezhen Huang, Jiqiang Liu, and Zhen Han. A privacy-aware access model on anonymized data. *INTRUST*, pages 201–212, 2014.

16 David N. Jansen and Roel Wieringa. Extending ctl with actions and real time. *J. Log. Comput.*, 12(4):607–621, 2002.

17 Xin Jin, Ram Krishnan, and Ravi S. Sandhu. A unified attribute-based access control model covering dac, mac and rbac. *DBSec*, pages 41–45, 2012.

18 James Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, 17(1):4–23, 2005.

19 M. Fahim Ferdous Khan and Ken Sakamura. A discretionary delegation framework for access control systems. *OTM Conferences*, pages 865–882, 2016.

20 Samrat Mondal and Shamik Sural. Security analysis of temporal-rbac using timed automata. *IAS*, pages 37–40, 2008.

21 Samrat Mondal, Shamik Sural, and Vijayalakshmi Atluri. Security analysis of gtrbac and its variants using model checking. *Computers and Security*, 30(2-3):128–147, 2011.

22 Kevin Mueller, Michael Paulitsch, Sergey Tverdyshev, and Holger Blasum. Mils-related information flow control in the avionic domain: A view on security-enhancing software architectures. *DSN Workshops*, pages 1–6, 2012.

23 Andrew C. Myers and Barbara Liskov. A decentralized model for information flow control. In *ACM Symposium on Operating System Principles, SOSP 1997*, pages 129–142. ACM, 1997.

24 Flemming Nielson, Hanne Riis Nielson, and Panagiotis Vasilikos. Information flow for timed automata. *Accepted for Springer Lecture Notes in Computer Science*, 2017.

25 Hanne Riis Nielson and Flemming Nielson. Content dependent information flow control. *J. Log. Algebr. Meth. Program.*, 87:6–32, 2017.

26 Martin Leth Pedersen, Michael Hedegaard Sørensen, Daniel Lux, Ulrik Nyman, and René Rydhof Hansen. The timed decentralised label model. *NordSec*, pages 27–43, 2015.

27 Carlos Ribeiro, Andre Zuquete, Paulo Ferreira, and Paulo Guedes. Spl: An access control language for security policies and complex constraints. *NDSS*, 2001.

28 Ravi S. Sandhu. Lattice-based access control models. *IEEE Computer*, 1993.

29 Ravi S. Sandhu and P. Samarati. Access control: Principles and practice. *IEEE Com. Mag.*, 1996.

30 UPPALL. `http://www.uppaal.com/index.php?sida=200&rubrik=95`.

31 Dennis M. Volpano, Geoffrey Smith, and Cynthia E. Irvine. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(2/3):167–188, 1996.

32 OASIS eXtensible Access Control Markup Language. `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`.

33 eXtensible Markup Language(XML) . `https://www.w3.org/XML/`.

34 Wenrong Zeng, Yuhao Yang, and Bo Luo. Content-based access control: Use data content to assist access control for large-scale content-centric databases. *BigData Conference*, pages 701–710, 2014.

A Proof of Theorem 7

Proof. The proof proceeds by structural induction on ϕ. The base cases are trivial since $\mathcal{T}[\![\phi]\!] = \phi$, the formula ϕ does not include any constraint about the clock t, and $\gamma \cong \gamma'$.

- The case $\forall\Box_{\mathfrak{b}}(\phi_1, \phi_2)$.
  Assume that $\gamma \models \forall\Box_{\mathfrak{b}}(\phi_1, \phi_2)$ and thus by definition:

$$\forall \gamma_0 \overset{\mathfrak{b}_1}{\Rightarrow} \gamma_1 \overset{\mathfrak{b}_2}{\Rightarrow} .. \in \mathbf{Trace}_\gamma : \forall i \geq 1 : \mathfrak{b}_i = \mathfrak{b} \Rightarrow \gamma_{i-1} \models \phi_1 \text{ and } \gamma_i \models \phi_2 \,. \tag{1}$$

Now take arbitrary trace $tr' = \gamma'_0 \longrightarrow \gamma'_1 \longrightarrow ... \in \mathbf{Trace}_{\gamma'}$, where $\gamma' = \gamma'_0$ and prove that

$$\forall j \geq 0 : \gamma'_j \models \mathfrak{b} \Rightarrow (\mathcal{T}[\![\phi_1]\!] \wedge \exists(\mathfrak{b}\, U(\neg \mathfrak{b} \wedge \mathcal{T}[\![\phi_2]\!]))) \,.$$

Now if tr' has length 0 then $tr' = \gamma'$ and the proof is trivial since γ' is a genuine configuration and thus $\gamma' \not\models \mathfrak{b}$. Similarly, if tr' has length greater than 0 and γ'_j is a genuine configuration then the proof holds. Now if γ'_j is an auxiliary configuration, consider the macro tranistion trace $T_{tr'} = t'_1 t'_2 ...$ of the trace tr' and let t'_i be the macro tranistion which corresponds to the transition in which γ'_j is being involved and thus we have that $T_{tr'}(i) = \gamma'_{j-1} \longrightarrow \gamma'_j \longrightarrow \gamma'_{j+1}$. Next, let $\gamma'_j = \langle v, \sigma, \delta, \kappa \rangle$ and using Fact 5 we have that $\exists tr \in \mathbf{Trace}_{\gamma} : tr \cong tr'$ and thus

$$\begin{aligned}
& \forall h \geq 1 : T_{tr}(h) \cong T_{tr'}(h) \\
\Rightarrow \quad & T_{tr}(i) \cong T_{tr'}(i) \\
\Leftrightarrow \quad & \gamma_{i-1} \overset{\mathfrak{b}_i}{\Longrightarrow} \gamma_i \cong \gamma'_{j-1} \longrightarrow \gamma'_j \longrightarrow \gamma'_{j+1} \\
\Leftrightarrow \quad & \gamma_{i-1} \cong \gamma'_{j-1} \text{ and } \gamma_i \cong \gamma'_{j+1} \text{ and } \mathsf{L}(v) = \mathfrak{b}_i
\end{aligned} \tag{2}$$

Now if $\gamma'_j \not\models \mathfrak{b}$ then the proof is trivial. Otherwise, because of (2) ($\mathsf{L}(v) = \mathfrak{b}_i$) we have that also $\mathfrak{b}_i = \mathfrak{b}$ and using (1) we have that $\gamma_{i-1} \models \phi_1$ and $\gamma_i \models \phi_2$. Next, using (2) ($\gamma_{i-1} \cong \gamma'_{j-1}$) and our induction hypothesis we have also that $\gamma'_{j-1} \models \mathcal{T}[\![\phi_1]\!]$ and since ϕ_1 does not contain any nested formulas we also have that $\gamma'_j \models \mathcal{T}[\![\phi_1]\!]$ as required. Finally, using (2) ($\gamma_i \cong \gamma'_{j+1}$) and our induction hypothesis we have also that $\gamma'_{j+1} \models \mathcal{T}[\![\phi_2]\!]$ and thus $\gamma'_j \models \exists(\mathfrak{b}\; U\; (\neg\mathfrak{b} \wedge \mathcal{T}[\![\phi_2]\!]))$ as required.
For the other direction now assume that $\gamma' \models \forall\Box\mathfrak{b} \Rightarrow (\mathcal{T}[\![\phi_1]\!] \;\wedge \exists(\mathfrak{b}\, U(\neg\mathfrak{b} \wedge \mathcal{T}[\![\phi_2]\!])))$ and thus

$$\forall \gamma'_0 \longrightarrow \gamma'_1 \longrightarrow \gamma'_2 .... \in \mathbf{Trace}_{\gamma'} : \forall i \geq 0 : \gamma'_i \models \mathfrak{b} \Rightarrow (\mathcal{T}[\![\phi_1]\!] \;\wedge \exists(\mathfrak{b}\, U(\neg\mathfrak{b} \wedge \mathcal{T}[\![\phi_2]\!]))) \tag{3}$$

and take arbitrary trace $tr = \gamma_0 \overset{\mathfrak{b}_1}{\Longrightarrow} \gamma_1 \overset{\mathfrak{b}_2}{\Longrightarrow} ... \in \mathbf{Trace}_{\gamma}$, where $\gamma_0 = \gamma$ and prove that

$$\forall j \geq 1 : \mathfrak{b}_j = \mathfrak{b} \Rightarrow \gamma_{j-1} \models \phi_1 \text{ and } \gamma_j \models \phi_2 \,.$$

For the cases where the length of tr is 0 or $\mathfrak{b}_j \neq \mathfrak{b}$ then the proof is trivial. Therefore take j such that $\mathfrak{b}_j = \mathfrak{b}$ and consider the transition trace $T_{tr} = t_1 t_2 ....$ of the trace tr and thus, using Fact 5 we have that $\exists tr' \in \mathbf{Trace}_{\gamma'} : tr \cong tr'$ and consequently

$$\begin{aligned}
& \forall h \geq 1 : T_{tr}(h) \cong T_{tr'}(h) \\
\Rightarrow \quad & T_{tr}(j) \cong T_{tr'}(j) \\
\Leftrightarrow \quad & \gamma_{j-1} \overset{\mathfrak{b}_j}{\Longrightarrow} \gamma_j \cong \gamma'_s \longrightarrow \gamma_{aux} \longrightarrow \gamma'_t \\
\Leftrightarrow \quad & \gamma_{j-1} \cong \gamma'_s \text{ and } \gamma_j \cong \gamma'_t \text{ and if } \gamma_{aux} = \langle v, \sigma, \delta, \kappa \rangle \text{ then } \mathsf{L}(v) = \mathfrak{b}_j \,.
\end{aligned} \tag{4}$$

Therefore because of (4) ($\mathsf{L}(v) = \mathfrak{b}_j$) and (3) we have that $\gamma_{aux} \models \mathcal{T}[\![\phi_1]\!] \wedge \exists(\mathfrak{b}\; U\; (\neg\mathfrak{b} \wedge \mathcal{T}[\![\phi_2]\!]))$ and thus since ϕ_1 does not contain any nested formulas, $\gamma'_s \models \mathcal{T}[\![\phi_1]\!]$ and $\gamma'_t \models \mathcal{T}[\![\phi_2]\!]$; but then using (4) ($\gamma_{j-1} \cong \gamma'_s$ and $\gamma_j \cong \gamma'_t$) and our induction hypothesis we get the required result.

- The cases $\phi_1 \wedge \phi_2$ and $\neg\phi$ can be proved straightforwardly using structural induction on ϕ_1, ϕ_2 and ϕ. ◀

On Expressiveness of Halpern-Shoham Logic and its Horn Fragments[*]

Przemysław Andrzej Wałęga

University of Warsaw, Warsaw, Poland
p.a.walega@gmail.com

Abstract

Halpern and Shoham's modal logic of time intervals (HS in short) is an elegant and highly influential propositional interval-based logic. Its Horn fragments and their hybrid extensions have been recently intensively studied and successfully applied in real-world use cases. Detailed investigation of their decidability and computational complexity has been conducted, however, there has been significantly less research on their expressive power. In this paper we make a step towards filling this gap. We (1) show what time structures are definable in the language of HS, and (2) determine which HS fragments are capable of expressing: hybrid machinery, i.e., *nominals* and *satisfaction operators*, and *somewhere*, *difference*, and *everywhere* modal operators. These results enable us to classify HS Horn fragments according to their expressive power and to gain insight in the interplay between their decidability/computational complexity and expressiveness.

1998 ACM Subject Classification I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases Temporal Logic, Interval Logic, Expressiveness, Hybrid Logic

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.22

1 Introduction

The aim of this paper is to investigate the expressive power of the temporal logic of Halpern and Shoham (HS in short) [13] and its Horn fragments [9, 8]. The latter are especially interesting due to their relatively low computational complexity [9, 17, 3] and the range of potential applications, e.g, real-world use cases in temporal ontology-based data access (OBDA) [14]. Although decidability and computational complexity of these fragments have been intensively studied [9, 17, 3], their expressive power is yet to be studied in any significant depth. Our research aims at filling this gap and enabling a better understanding of the interplay between decidability/computational complexity and expressive power of HS fragments.

Halpern-Shoham logic is a propositional multimodal logic which enables reasoning about relations between time-intervals in a one dimensional timeline. The HS language contains 12 modal operators, each corresponding to one of the Allen's binary relations between intervals [1], namely *adjacent to*, *begins*, *during*, *ends*, *later than*, *overlaps*, and their inverses (Allen's algebra contains also *identity* as the 13th relation). A model of HS is a linear ordering of time-points (a temporal frame), where propositional variables are interpreted by sets of intervals over this temporal frame. The HS language is very expressive and the satisfiability problem of its formulas is undecidable over a range of interesting linear orders including $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ [13]. As a result, restrictions on HS have been intensively investigated in order to establish fragments of relatively low computational complexity, whose expressive

[*] This work was partially supported by the NCN grant DEC-2011/02/A/HS1/00395.

Table 1 Semantics definable in HS under irreflexive and reflexive semantics, respectively.

	Under irreflexive semantics:	**Under reflexive semantics:**
Definable semantics:	(Dis), (Den), (S), (Non-S)	(Non-S)

power is high enough for a variety of applications. A number of methods to specify HS fragments have been proposed, e.g., restricting the set of modal operators occurring in the language [11, 12, 6], softening semantics of modal operators [15], and restricting the nesting-depth of modal operators [7]. Recently, a twofold method has been proposed to obtain HS fragments [9]: firstly, by imposing restrictions on the use of classical propositional connectives in the language, giving rise to fragments called *Horn*, *Krom*, and *core*, and secondly, by additionally disallowing diamond (or box) modal operators. This new approach has led to the identification of tractable fragments (precisely P-complete) [3], which were already applied in real-world use cases within temporal OBDA [14]. The success of this approach motivated applying the same technique in other logics in order to establish their low complexity fragments. Namely, it was applied in modal logics K, T, K4, S4 [10], and in Metric Temporal Logic [5].

In this paper we investigate the expressiveness of HS and its Horn fragments, and show how it depends on the structure of a temporal frame. Namely, we will follow an idea from [8] and distinguish between HS-models (i) with irreflexive ($<$) (originally introduced by Halpern and Shoham in [13]) and reflexive ($\leq$) (obtained by softening semantics as described in [8, 7]) semantics of relations between intervals, (ii) over discrete (Dis) and dense (Den) frames, and (iii) under strict (S) (i.e., without punctual-intervals) and non-strict (Non-S) (i.e., with punctual-intervals) semantics. Combinations of the above 3 lines of distinction give us 8 distinct semantics. A precise description of HS, its Horn fragments, as well as all 8 semantics, is presented in Section 2. Our contributions are as follows.

First, in Section 3 we study which semantics are definable in the language of HS, where a semantics is definable by a formula, if the formula is true exactly in frames satisfying conditions imposed on this semantics. We show that if the irreflexive semantics of relations is assumed, then not only (Dis) and (Den) semantics are definable (as proved in [13]) but also (S), and (Non-S). Moreover, we show that under reflexive semantics (Non-S) is definable but it is an open question if (Dis), (Den), and (S) are definable or not – see Table 1.

Second, in Section 4 we study the expressive power of HS, Horn fragment HS_{horn}, and its further restrictions, namely $\mathsf{HS}^{\Diamond}_{horn}$ obtained by deleting box modal operators from the language and $\mathsf{HS}^{\Box}_{horn}$ in which diamond operators are deleted. We show which of the following expressions are expressible in these languages: *somewhere* p ($\mathsf{E}p$), *in a different interval* p ($\mathsf{D}p$), and *everywhere* p ($\mathsf{A}p$), where p is a propositional variable. Moreover, we show in which fragments *nominals* (i), and *satisfaction operators* ($@_i$) are expressible – see Table 2. Nominals and satisfaction operators constitute a standard hybrid machinery exploited in order to overcome the local nature of a modal language [2, 4].

Furthermore, we show that the expressive power of $\mathsf{HS}^{\Box,i}_{horn}$ (which stands for $\mathsf{HS}^{\Box}_{horn}$ whose language is extended with nominals) and $\mathsf{HS}^{\Box,i,@}_{horn}$ (i.e., $\mathsf{HS}^{\Box,i}_{horn}$ extended by satisfaction operators) is the same in any semantics. A Hasse diagram of HS fragments is depicted in Figure 1a where an arrow indicates a syntactical extension. Our research resulted in a classification of these fragments according to their expressive power. A map of expressiveness under all semantics except ($\leq$,Dis,Non-S) and ($\leq$,Den,Non-S) is presented in Figure 1b, where $\approx$ stands for the same expressive power and $\succcurlyeq$ for greater-or-equal expressive power.

Table 2 Summarized expressiveness results.

	Ep	Dp	Ap	i	i and $@_i$
HS	✓	✓*	✓	✓*	✓*
HS_{horn}	✓	?	✓	✓*	✓*
$\mathsf{HS}^{\Diamond}_{horn}$	✓	–	✓	✓*	✓*
$\mathsf{HS}^{\Box}_{horn}$	–*	–*	✓	–*	–*

✓	:	definable in all semantics;
✓*	:	definable in all semantics except (≤,Dis,Non-S) and (≤,Den,Non-S);
–	:	undefinable in any semantics;
–*	:	undefinable in (<,Den,S), (<,Den,Non-S), (≤,Dis,Non-S), (≤,Den,S), (≤,Den,Non-S);
?	:	unknown.

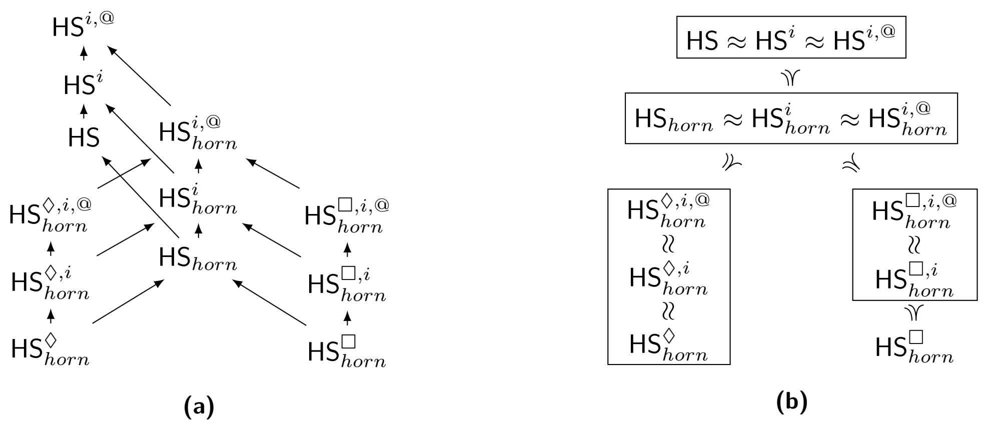

$\mathsf{HS}^{i,@}$
HS^{i}
HS
$\mathsf{HS}^{i,@}_{horn}$
HS^{i}_{horn}
HS_{horn}
$\mathsf{HS}^{\Diamond,i,@}_{horn}$
$\mathsf{HS}^{\Diamond,i}_{horn}$
$\mathsf{HS}^{\Diamond}_{horn}$
$\mathsf{HS}^{\Box,i,@}_{horn}$
$\mathsf{HS}^{\Box,i}_{horn}$
$\mathsf{HS}^{\Box}_{horn}$
(a)
$\mathsf{HS} \approx \mathsf{HS}^{i} \approx \mathsf{HS}^{i,@}$
$\mathsf{HS}_{horn} \approx \mathsf{HS}^{i}_{horn} \approx \mathsf{HS}^{i,@}_{horn}$
$\mathsf{HS}^{\Diamond,i,@}_{horn} \approx \mathsf{HS}^{\Diamond,i}_{horn} \approx \mathsf{HS}^{\Diamond}_{horn}$
$\mathsf{HS}^{\Box,i,@}_{horn} \approx \mathsf{HS}^{\Box,i}_{horn}$
$\mathsf{HS}^{\Box}_{horn}$
(b)

Figure 1 Syntactical dependencies of HS fragments (a) and expressive power dependencies in HS fragments under all semantics except (≤,Dis,Non-S) and (≤,Den,Non-S) (b).

2 Halpern-Shoham Logic and its Horn Fragments

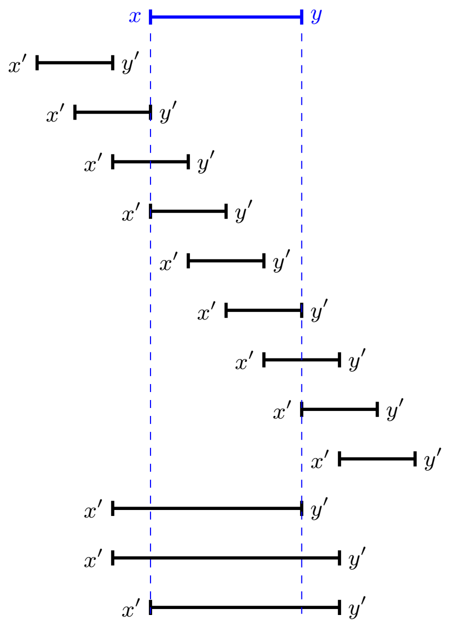
The language of Halpern-Shoham logic consists of a set of propositional variables PROP, propositional constants $\top$ (true) and $\bot$ (false), classical propositional connectives $\neg, \wedge, \vee, \rightarrow$, 12 modal operators of the form $\langle \mathsf{R} \rangle$, called *diamonds*, and their duals of the form $[\mathsf{R}]$, called *boxes*, where $\mathsf{R} \in \{\mathsf{B}, \overline{\mathsf{B}}, \mathsf{D}, \overline{\mathsf{D}}, \mathsf{E}, \overline{\mathsf{E}}, \mathsf{O}, \overline{\mathsf{O}}, \mathsf{A}, \overline{\mathsf{A}}, \mathsf{L}, \overline{\mathsf{L}}\}$ (in what follows, we denote this set by $\mathsf{HS}_{\mathsf{rel}}$). Well-formed HS-formulas are defined by the following abstract grammar:

$$\varphi := \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \langle \mathsf{R} \rangle \varphi \mid [\mathsf{R}]\varphi,$$

where $p \in \mathsf{PROP}$ and $\mathsf{R} \in \mathsf{HS}_{\mathsf{rel}}$. We follow [8] and define an HS-model as a pair $(\mathbb{D}, V)$ such that $\mathbb{D} = (D, \leq)$, called a *temporal frame* (or simply a *frame*), is a non-strict linear order (reflexive, antisymmetric, transitive, and total relation) of time-points, $I(\mathbb{D})$ is a set of intervals over $\mathbb{D}$ (defined in what follows), and $V : \mathsf{PROP} \rightarrow \mathcal{P}(I(\mathbb{D}))$ assigns a set of intervals to each propositional variable (notice that the original definition from [13] of an HS-model is more general). In what follows, we also use "$x < y$" as an abbreviation for "$x \leq y$ and $x \neq y$". In the paper we restrict attention to orderings $\mathbb{D}$ which contain an infinitely ascending and descending chains. As proposed in [8], semantics of HS may be specified according to the following lines of division. The first distinction deals with a definition of binary relations between intervals, namely *begins* ($\mathsf{rel}_{\mathsf{B}}$), *during* ($\mathsf{rel}_{\mathsf{D}}$), *ends* ($\mathsf{rel}_{\mathsf{E}}$), *overlaps* ($\mathsf{rel}_{\mathsf{O}}$), *adjacent to* ($\mathsf{rel}_{\mathsf{A}}$), *later than* ($\mathsf{rel}_{\mathsf{L}}$), and opposite relations: $\mathsf{rel}_{\overline{\mathsf{B}}}$, $\mathsf{rel}_{\overline{\mathsf{D}}}$, $\mathsf{rel}_{\overline{\mathsf{E}}}$, $\mathsf{rel}_{\overline{\mathsf{O}}}$, $\mathsf{rel}_{\overline{\mathsf{A}}}$, $\mathsf{rel}_{\overline{\mathsf{L}}}$ ("dashed" relations are not always inverses of their "not dashed" counterparts, e.g., $\mathsf{rel}_{\overline{\mathsf{A}}}$ is not an inverse of $\mathsf{rel}_{\mathsf{A}}$ in non-strict semantics):

(<) *Irreflexive* semantics: a relation between intervals $[x, y]$ and $[x', y']$ is defined as:

$[x,y]\mathsf{rel}_{\overline{\mathsf{L}}}[x',y']$	iff	$y' < x$
$[x,y]\mathsf{rel}_{\overline{\mathsf{A}}}[x',y']$	iff	$x' < y',\ y' = x$
$[x,y]\mathsf{rel}_{\overline{\mathsf{O}}}[x',y']$	iff	$x' < x < y' < y$
$[x,y]\mathsf{rel}_{\mathsf{B}}[x',y']$	iff	$x = x',\ y' < y$
$[x,y]\mathsf{rel}_{\mathsf{D}}[x',y']$	iff	$x < x',\ y' < y$
$[x,y]\mathsf{rel}_{\mathsf{E}}[x',y']$	iff	$x < x',\ y = y'$
$[x,y]\mathsf{rel}_{\mathsf{O}}[x',y']$	iff	$x < x' < y < y'$
$[x,y]\mathsf{rel}_{\mathsf{A}}[x',y']$	iff	$y = x',\ x' < y'$
$[x,y]\mathsf{rel}_{\mathsf{L}}[x',y']$	iff	$y < x'$
$[x,y]\mathsf{rel}_{\overline{\mathsf{E}}}[x',y']$	iff	$x' < x,\ y = y'$
$[x,y]\mathsf{rel}_{\overline{\mathsf{D}}}[x',y']$	iff	$x' < x,\ y < y'$
$[x,y]\mathsf{rel}_{\overline{\mathsf{B}}}[x',y']$	iff	$x = x',\ y < y'$



(≤) *Reflexive* semantics: each occurrence of "<" is replaced by "≤" with respect to the definition of relations under irreflexive semantics.

The second distinction is between:

(Dis) *Discrete* frames: any $x \in D$ has an immediate $<$-successor, and an immediate $<$-predecessor;

(Den) *Dense* frames: for any $x, y \in D$ such that $x < y$ there is $z \in D$ such that $x < z < y$.

The third distinction differentiates between:

(S) *Strict* semantics: punctual intervals are disallowed, i.e., a set of all intervals over $\mathbb{D}$ is defined as $I(\mathbb{D}) = \{[x, y] \mid x, y \in D \text{ and } x < y\}$;

(Non-S) *Non-Strict* semantics: punctual intervals are allowed, i.e., a set of all intervals over $\mathbb{D}$ is defined as $I(\mathbb{D}) = \{[x, y] \mid x, y \in D \text{ and } x \leq y\}$;

where $[x, y] = \{z \mid z \in D \text{ and } x \leq z \leq y\}$. Independently of the semantics, the relations between non-identical intervals are exhaustive in the sense that between any two non-identical intervals necessarily holds some relation. In irreflexive semantics the relations are also disjoint, and consequently exactly one relation holds between any two non-identical intervals.

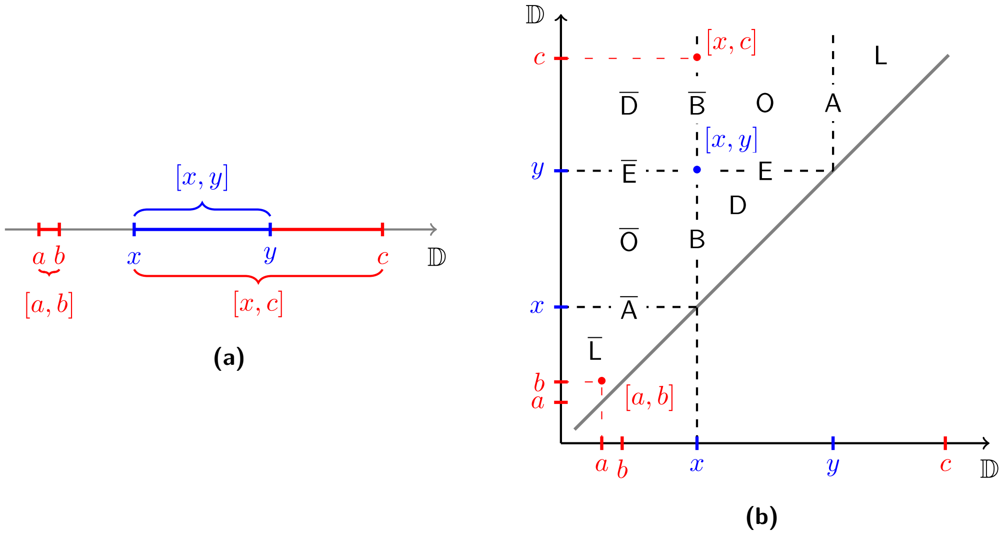


Figure 2 One-dimensional (a) and two-dimensional (b) representations of the same HS-model, in which $[x,y]\overline{\mathsf{L}}[a,b]$ ($[a,b]$ is earlier than $[x,y]$) and $[x,y]\overline{\mathsf{B}}[x,c]$ ($[x,c]$ is begun by $[x,y]$).

The satisfaction relation for an HS-model $\mathcal{M}$ and an interval $[x,y]$ is defined as follows:

$\mathcal{M},[x,y] \models \top$		for all $[x,y] \in I(\mathbb{D})$;
$\mathcal{M},[x,y] \not\models \bot$		for all $[x,y] \in I(\mathbb{D})$;
$\mathcal{M},[x,y] \models p$	iff	$[x,y] \in V(p)$, for any $p \in \mathsf{PROP}$;
$\mathcal{M},[x,y] \models \neg\varphi$	iff	$\mathcal{M},[x,y] \not\models \varphi$;
$\mathcal{M},[x,y] \models \varphi \wedge \psi$	iff	$\mathcal{M},[x,y] \models \varphi$ and $\mathcal{M},[x,y] \models \psi$;
$\mathcal{M},[x,y] \models \varphi \vee \psi$	iff	$\mathcal{M},[x,y] \models \varphi$ or $\mathcal{M},[x,y] \models \psi$;
$\mathcal{M},[x,y] \models \varphi \rightarrow \psi$	iff	if $\mathcal{M},[x,y] \models \varphi$, then $\mathcal{M},[x,y] \models \psi$;
$\mathcal{M},[x,y] \models \langle\mathsf{R}\rangle\varphi$	iff	there exists $[x',y'] \in I(\mathbb{D})$ such that $[x,y]\mathsf{rel}_{\mathsf{R}}[x',y']$ and $\mathcal{M},[x',y'] \models \varphi$;
$\mathcal{M},[x,y] \models [\mathsf{R}]\varphi$	iff	for every $[x',y'] \in I(\mathbb{D})$ such that $[x,y]\mathsf{rel}_{\mathsf{R}}[x',y']$ it holds that $\mathcal{M},[x',y'] \models \varphi$;

for any $\mathsf{R} \in \mathsf{HS}_{\mathsf{rel}}$, and any HS-formulas φ, ψ. An HS-formula φ is true in an HS-model $\mathcal{M}$ (in symbols: $\mathcal{M} \models \varphi$) iff for all $[x,y] \in I(\mathbb{D})$ it holds that $\mathcal{M},[x,y] \models \varphi$. An HS-formula φ is true in a frame $\mathbb{D}$ (in symbols: $\mathbb{D} \models \varphi$) iff for any HS-model $\mathcal{M}$ based on $\mathbb{D}$ it holds that $\mathcal{M} \models \varphi$. An HS-formula φ is valid (in symbols: $\models \varphi$) iff for any HS-frame $\mathbb{D}$ we have $\mathbb{D} \models \varphi$.

A convenient representation of a temporal frame (e.g., for decidability and computational complexity proofs [8]) is obtained by treating an interval $[x,y]$ as a point in a two-dimensional Cartesian space $D \times D$, where the abscissa has a value x and the ordinate has a value y [16]. In the two-dimensional representation, intervals correspond to the points in the north-western half-plane of $D \times D$, and points on the diagonal correspond to punctual-intervals. In such a representation relations between intervals obtain a spatial interpretation. Let $[x,y]$ be a fixed interval, then a relation between $[x,y]$ and any other interval $[x',y']$ may be determined on the basis of a relative position of points (x,y) and (x',y') as presented in Figure 2.

Decidability of the HS-formulas satisfiability problem (HS-satisfiability) depends on the semantics. However, for most interesting frames it is undecidable, e.g., it was already shown in [13] that the problem is undecidable under irreflexive and non-strict semantics for any class of temporal frames that contains an infinite ascending chain (e.g., $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$). These negative results motivated searching for decidable and tractable fragments of HS which would be still interesting from the expressiveness point of view.

As observed in [8], any HS-formula can be transformed into an equisatisfiable formula defined by the following grammar, and vice versa:

$$\varphi := \lambda \mid \neg\lambda \mid [\mathsf{U}](\lambda \wedge \ldots \wedge \lambda \rightarrow \lambda \vee \ldots \vee \lambda) \mid \varphi \wedge \varphi, \tag{1}$$

where $[\mathsf{U}]$ is the universal modality, i.e., $[\mathsf{U}]\psi$ is satisfied iff ψ is satisfied in every $[x, y] \in I(\mathbb{D})$ whereas λ, the so-called *positive temporal literal*, is a formula defined by the grammar:

$$\lambda := \top \mid \bot \mid p \mid \langle\mathsf{R}\rangle\lambda \mid [\mathsf{R}]\lambda, \tag{2}$$

where $p \in \mathsf{PROP}$ and $\mathsf{R} \in \mathsf{HS}_{\mathsf{rel}}$. Horn fragments of HS (proposed in [9, 8]) are obtained by imposing limitations on the grammars (1) and (2) as follows:

- First, the limitation of the grammar (1) gives rise to an HS fragment denoted by HS_{horn}, in which (1) is restricted to the grammar:

$$\varphi := \lambda \mid [\mathsf{U}](\lambda \wedge \ldots \wedge \lambda \rightarrow \lambda) \mid \varphi \wedge \varphi.$$

- Second, additional limitations on the grammar of positive temporal literals (2) in HS_{horn} give rise to fragments denoted by $\mathsf{HS}^{\Diamond}_{horn}$ and $\mathsf{HS}^{\Box}_{horn}$. In the case of $\mathsf{HS}^{\Diamond}_{horn}$ the grammar of positive temporal literals is restricted to:

$$\lambda := \top \mid \bot \mid p \mid \langle\mathsf{R}\rangle\lambda,$$

whereas in the case $\mathsf{HS}^{\Box}_{horn}$ grammar (2) is restricted to:

$$\lambda := \top \mid \bot \mid p \mid [\mathsf{R}]\lambda.$$

$\mathsf{HS}^{\Box}_{horn}$ is particularly interesting, as it is both decidable and tractable (P-complete) under ($<$,Den,S), ($<$,Den,Non-S), ($\leq$,Dis,Non-S), ($\leq$,Den,S), and ($\leq$,Den,Non-S) semantics [9, 8].

3 Defining Semantics

A formula φ defines a semantics *Sem* if the following are equivalent:

- φ is true in a frame $\mathbb{D}$;
- $\mathbb{D}$ satisfies the conditions imposed by *Sem*.

We will show that under the standard, i.e., irreflexive semantics ($<$), the semantics (Dis), (Den), (S), and (Non-S) are definable but under the softened semantics, i.e., reflexive ($\leq$) we only know how to define (Non-S). We treat existence and non-existence of punctual-intervals as a property of a frame. At first assume that the semantics is irreflexive. As showed in [13], (Dis) is definable in the language of HS by the formula:

$$[\mathsf{B}]\bot \vee (\langle\mathsf{B}\rangle\top \wedge [\mathsf{B}][\mathsf{B}]\bot) \vee \Big(\langle\mathsf{B}\rangle(\langle\mathsf{B}\rangle\top \wedge [\mathsf{B}][\mathsf{B}]\bot) \wedge \langle\mathsf{E}\rangle(\langle\mathsf{B}\rangle\top \wedge [\mathsf{B}][\mathsf{B}]\bot)\Big);$$

and (Den) by the formula:

$$\neg(\langle\mathsf{B}\rangle\top \wedge [\mathsf{B}][\mathsf{B}]\bot).$$

We show that (S) and (Non-S) are also definable in the language of HS.

▶ **Theorem 1.** (S) *is definable in the language of* HS *under irreflexive semantics by:*

$$\varphi_{<,S} := \langle\mathsf{A}\rangle[\mathsf{B}]\bot \vee [\mathsf{A}]\neg\langle\mathsf{B}\rangle[\mathsf{B}]\bot.$$

Proof. Fix any HS-frame $\mathbb{D}$. First, assume that $\mathbb{D} \models \varphi_{<,\mathsf{S}}$. We will show that the semantics is strict. Fix any time-point y. To show that $[y, y]$ is not an interval. Fix any $x < y$ and any HS-model $\mathcal{M}$ based on $\mathbb{D}$. It follows that $\mathcal{M}, [x, y] \models \varphi_{<,\mathsf{S}}$.

(Case 1) $\mathcal{M}, [x, y] \models \langle \mathsf{A} \rangle [\mathsf{B}] \bot$. Hence, there is $z > y$ such that $\mathcal{M}, [y, z] \models [\mathsf{B}] \bot$. If $[y, y]$ was an interval, we would have $\mathcal{M}, [y, z] \not\models [\mathsf{B}] \bot$. Hence, $[y, y]$ cannot be an interval.

(Case 2) $\mathcal{M}, [x, y] \models [\mathsf{A}] \neg \langle \mathsf{B} \rangle [\mathsf{B}] \bot$. Suppose that $[y, y]$ is an interval, so $\mathcal{M}, [y, y] \models [\mathsf{B}] \bot$. Hence, for any $z > y$ we have $[y, z] \mathsf{rel}_{\mathsf{B}} [y, y]$. Then $\mathcal{M}, [x, y] \models [\mathsf{A}] \langle \mathsf{B} \rangle [\mathsf{B}] \bot$. Since $\mathbb{D}$ contains an infinite ascending chain, we have reached a contradiction, so $[y, y]$ is not an interval.

As a result, for any time-point y, $[y, y]$ is not an interval, hence the semantics is strict.

Second, assume that the semantics is non-strict. We will show that $\mathbb{D} \not\models \varphi_{<,\mathsf{S}}$. Fix any punctual-interval $[y, y]$ and any HS-model $\mathcal{M}$ based on $\mathbb{D}$. To show that $\mathcal{M}, [y, y] \not\models \varphi_{<,\mathsf{S}}$, i.e., $\mathcal{M}, [y, y] \not\models \langle \mathsf{A} \rangle [\mathsf{B}] \bot \vee [\mathsf{A}] \neg \langle \mathsf{B} \rangle [\mathsf{B}] \bot$.

$\mathcal{M}, [y, y] \not\models \langle \mathsf{A} \rangle [\mathsf{B}] \bot$, because an interval $[u, w]$ is in relation $\mathsf{rel}_{\mathsf{A}}$ with $[y, y]$ whenever $u = y$ and $w > y$. It follows that $w > u$, so $\mathcal{M}, [u, w] \not\models [\mathsf{B}] \bot$. $\mathcal{M}, [y, y] \not\models [\mathsf{A}] \neg \langle \mathsf{B} \rangle [\mathsf{B}] \bot$, because if $z > y$ then $[y, y] \mathsf{rel}_{\mathsf{A}} [y, z]$, $[y, z] \mathsf{rel}_{\mathsf{B}} [y, y]$, and $\mathcal{M}, [y, y] \models [\mathsf{B}] \bot$. ◀

▶ **Theorem 2.** (Non-S) *is definable in the language of* HS *under irreflexive semantics by:*

$$\varphi_{<,\mathit{Non\text{-}S}} := [\mathsf{A}] \neg [\mathsf{B}] \bot \wedge [\mathsf{A}] \langle \mathsf{B} \rangle [\mathsf{B}] \bot.$$

Proof. Fix any HS-frame $\mathbb{D}$. First, assume that $\mathbb{D} \models \varphi_{<,\mathsf{Non\text{-}S}}$. To show that the semantics is non-strict. Fix any time-point y. We will show that $[y, y]$ is an interval. Fix any HS-model $\mathcal{M}$ based on $\mathbb{D}$ and $x < y$. It follows that $\mathcal{M}, [x, y] \models \varphi_{<,\mathsf{Non\text{-}S}}$.

From the one hand, $\mathcal{M}, [x, y] \models [\mathsf{A}] \neg [\mathsf{B}] \bot$, hence for any $z > y$ we have $\mathcal{M}, [y, z] \not\models [\mathsf{B}] \bot$. On the other hand, $\mathcal{M}, [x, y] \models [\mathsf{A}] \langle \mathsf{B} \rangle [\mathsf{B}] \bot$. Fix any $w > y$. Then $\mathcal{M}, [y, w] \models \langle \mathsf{B} \rangle [\mathsf{B}] \bot$. Hence, for all u such that $y < u < w$ we have $\mathcal{M}, [y, u] \models \langle \mathsf{B} \rangle [\mathsf{B}] \bot$, which leads to a contradiction

Second, assume that the semantics is strict. To show that $\mathbb{D} \not\models \varphi_{<,\mathsf{Non\text{-}S}}$. Fix any interval $[x, y]$ and any HS-model $\mathcal{M}$ based on $\mathbb{D}$. To show that $\mathcal{M}, [x, y] \not\models \varphi_{<,\mathsf{Non\text{-}S}}$.

(Case 1) y has an immediate $>$-successor. Let z be the immediate $>$-successor of y. Then $\mathcal{M}, [y, z] \models [\mathsf{B}] \bot$, so $\mathcal{M}, [x, y] \not\models [\mathsf{A}] \neg [\mathsf{B}] \bot$. Hence $\mathcal{M}, [x, y] \not\models \varphi_{<,\mathsf{Non\text{-}S}}$.

(Case 2) y does not have an immediate $>$-successor. Hence, there is no $z > y$ such that $\mathcal{M}, [y, z] \models [\mathsf{B}] \bot$. As a result $\mathcal{M}, [x, y] \not\models [\mathsf{A}] \langle \mathsf{B} \rangle [\mathsf{B}] \bot$, so $\mathcal{M}, [x, y] \not\models \varphi_{<,\mathsf{Non\text{-}S}}$. ◀

Finally, we show that under reflexive semantics HS is expressive enough to define non-strict semantics.

▶ **Theorem 3.** (Non-S) *is definable in the language of* HS *under reflexive semantics by:*

$$\varphi_{\leq,\mathit{Non\text{-}S}} := [\mathsf{E}] p \to \langle \mathsf{A} \rangle p.$$

Proof. Fix any HS-frame $\mathbb{D}$. First, assume that the semantics is non-strict. To show that $\mathbb{D} \models \varphi_{\leq,\mathsf{Non\text{-}S}}$. Fix any interval $[x, y]$ and any HS-model $\mathcal{M}$ based on $\mathbb{D}$. Assume that $\mathcal{M}, [x, y] \models [\mathsf{E}] p$. It follows that $\mathcal{M}, [y, y] \models p$. Since $[x, y] \mathsf{rel}_{\mathsf{A}} [y, y]$, we have $\mathcal{M}, [x, y] \models \langle \mathsf{A} \rangle p$. It follows that $\mathbb{D} \models \varphi_{\leq,\mathsf{Non\text{-}S}}$.

Second, assume that the semantics is strict. We will show that $\mathbb{D} \not\models \varphi_{\mathsf{S}}$. Fix any interval $[x, y]$ and any HS-model $\mathcal{M} = (\mathbb{D}, V)$ such that $V(p) = \{[z, y] \mid z \geq x\}$. To show that $\mathcal{M}, [x, y] \not\models \varphi_{\geq,\mathsf{Non\text{-}S}}$. By the definition of V it follows that $\mathcal{M}, [x, y] \models [\mathsf{E}] p$. However, $\mathcal{M}, [x, y] \not\models \langle \mathsf{A} \rangle p$, so $\mathcal{M}, [x, y] \not\models \varphi_{\geq,\mathsf{Non\text{-}S}}$. ◀

4 Hybrid Machinery and Additional Operators

In this section we will study whether HS and its Horn fragments are expressive enough to define hybrid machinery (*nominals* and *satisfaction operators*) and the *somewhere* (E), *difference* (D), and *universal* (A) operators (notice that, following the standard notation, we use symbols E, D, and A in two meanings, namely as elements of $\mathsf{HS}_{\mathsf{rel}}$ and as above mentioned modal operators). The satisfaction relation in an HS-model $\mathcal{M}$ and an interval $[x, y]$ is defined for E, D, and A as follows:

$$\begin{array}{lll} \mathcal{M},[x,y] \models \mathsf{E}\varphi & \text{iff} & \text{there is } [x',y'] \in I(\mathbb{D}) \text{ such that } \mathcal{M},[x',y'] \models \varphi; \\ \mathcal{M},[x,y] \models \mathsf{D}\varphi & \text{iff} & \text{there is } [x',y'] \in I(\mathbb{D}) \text{ such that} \\ & & [x',y'] \neq [x,y] \text{ and } \mathcal{M},[x',y'] \models \varphi; \\ \mathcal{M},[x,y] \models \mathsf{A}\varphi & \text{iff} & \text{for all } [x',y'] \in I(\mathbb{D}) \text{ it holds that } \mathcal{M},[x',y'] \models \varphi; \end{array}$$

where φ is any HS-formula. Notice that A has the same meaning as [U] occurring in (1).

Hybrid machinery is obtained by enriching the language with the second sort of atoms, called *nominals* (we denote the set of all nominals by NOM) and *satisfaction operators* $@_i$ indexed by nominals. Intuitively, a nominal is a special kind of atom which is satisfied in exactly one interval, whereas an expression of the form $@_i\varphi$ is satisfied if φ is satisfied in the interval in which i is satisfied.

Formally, a hybrid HS-model $\mathcal{M}$ is a pair $(\mathbb{D}, V)$, such that $V : \mathsf{ATOM} \to \mathcal{P}(I(\mathbb{D}))$ assigns a set of intervals to each atom ($\mathsf{ATOM} = \mathsf{PROP} \cup \mathsf{NOM}$) with an additional restriction that $V(i)$ is a singleton for any $i \in \mathsf{NOM}$. The satisfaction relation conditions for nominals and @ operators are as follows:

$$\begin{array}{lll} \mathcal{M},[x,y] \models i & \text{iff} & V(i) = \{[x,y]\}, \text{ for any } i \in \mathsf{NOM}; \\ \mathcal{M},[x,y] \models @_i\varphi & \text{iff} & \mathcal{M},[x',y'] \models \varphi, \text{ where } V(i) = \{[x',y']\} \text{ and } i \in \mathsf{NOM}. \end{array}$$

Hybrid extensions of HS fragments were introduced in [17] and their Hasse diagram is depicted in Figure 1a, where “i” in the superscript of a fragment's symbol means that an expression of the form i (for $i \in \mathsf{NOM}$) is added to the grammar of positive temporal literals, whereas “$i, @$” in the superscript denotes a further extension obtained by adding an expression of the form $@_i\lambda$ (for $i \in \mathsf{NOM}$) to the grammar of positive temporal literals.

Interestingly, it has been shown in [17] that the satisfiability problems in $\mathsf{HS}^{\Box,i,@}_{horn}$ and in $\mathsf{HS}^{\Box,i}_{horn}$ are NP-complete under (<,Den,S), (<,Den,Non-S), (≤,Dis,Non-S), (≤,Den,S), and (≤,Den,Non-S) semantics, whereas the satisfiability problem in $\mathsf{HS}^{\Box}_{horn}$ is known to be P-complete under these semantics [9, 8]. However, differences in expressive power of $\mathsf{HS}^{\Box}_{horn}$, $\mathsf{HS}^{\Box,i}_{horn}$, and $\mathsf{HS}^{\Box,i,@}_{horn}$ have, thus far, not been investigated.

4.1 The Case of Full Halpern-Shoham Logic

Our aim is to determine if the hybrid machinery is definable in the full HS language, and whether the choice of semantics affects the answer to this problem. We start by recalling a general result established in [2], stating that we can eliminate all occurrences of nominals and @ operators in a hybrid modal logic formula by simulating them using the D operator .

▶ **Theorem 4** (Areces, Blackburn, Marx [2]). *There is a polynomial reduction which preserves satisfaction from any hybrid language enabling unrestricted use of classical propositional connectives, containing nominals and @ operators, to the fragment without nominals and @ operators but enabling to polynomially define* D.

Theorem 4 holds if there are no restrictions on the use of classical connectives. As noticed in [2] in the case of HS under irreflexive semantics, i.e., $(<,\mathsf{Dis},\mathsf{S})$, $(<,\mathsf{Den},\mathsf{S})$, $(<,\mathsf{Dis},\mathsf{Non\text{-}S})$, and $(<,\mathsf{Den},\mathsf{Non\text{-}S})$, operator the D is definable as follows:

$$\mathsf{D}\varphi := \bigvee_{\mathsf{R}\in\mathsf{HS}_{\mathsf{rel}}} \langle \mathsf{R}\rangle\varphi. \tag{3}$$

Hence, by Theorem 4 nominals and @ operators are definable in the language of HS under any irreflexive semantics. However, in the reflexive semantics the definition (3) would not be correct because it is not the case that $\mathsf{rel}_{\mathsf{R}}$ is irreflexive for any $\mathsf{R} \in \mathsf{HS}_{\mathsf{rel}}$. Nevertheless, under $(\leq,\mathsf{Dis},\mathsf{S})$ and $(\leq,\mathsf{Den},\mathsf{S})$ we can define D as follows:

$$\mathsf{D}\varphi := \langle\overline{\mathsf{B}}\rangle\langle\mathsf{B}\rangle\langle\mathsf{A}\rangle\varphi \vee \langle\overline{\mathsf{A}}\rangle\langle\overline{\mathsf{B}}\rangle\langle\mathsf{B}\rangle\varphi \vee \langle\mathsf{A}\rangle\langle\overline{\mathsf{E}}\rangle\langle\mathsf{E}\rangle\varphi \vee \langle\overline{\mathsf{E}}\rangle\langle\mathsf{E}\rangle\langle\overline{\mathsf{A}}\rangle\varphi. \tag{4}$$

The key is to observe that under $(\leq,\mathsf{Dis},\mathsf{S})$ and $(\leq,\mathsf{Den},\mathsf{S})$ if $[x,y]\mathsf{rel}_{\mathsf{A}}[x',y']$ then $y' > y$. Moreover, if $[x,y]\mathsf{rel}_{\overline{\mathsf{A}}}[x',y']$ then $x' < x$. To describe what formula (4) means let the current interval be $[x,y]$. Then (4) states that φ holds in some $[x',y']$ such that (i) $x' > x$, or (ii) $x' < x$, or (iii) $y' > y$, or (iv) $y' < y$. As a result, (4) states that φ holds in some $[x',y']$ such that $[x',y'] \neq [x,y]$.

Next, we will show that in the remaining semantics, i.e., $(\leq,\mathsf{Dis},\mathsf{Non\text{-}S})$ and $(\leq,\mathsf{Den},\mathsf{Non\text{-}S})$ neither D nor nominals are definable. For this purpose, we will construct a *bisimulation* (see, e.g., [4, Chapter 2]), which is a relation between models, which makes them indistinguishable by any modal formula. We say that HS-models $\mathcal{M} = (\mathbb{D}, V)$ and $\mathcal{M}' = (\mathbb{D}', V')$ are bisimilar (in symbols $\mathcal{M} \leftrightarroweq \mathcal{M}'$) if there is a relation (a bisimulation) $Z \subseteq I(\mathbb{D}) \times I(\mathbb{D}')$, which satisfies the following conditions:

(atom) For any intervals $[x,y]$, $[x',y']$ such that $[x,y]Z[x',y']$ it holds that $\mathcal{M},[x,y] \models p$ iff $\mathcal{M}',[x',y'] \models p$, for any $p \in \mathsf{PROP}$;

(zig) If $[x,y]Z[x',y']$ and $[x,y]\mathsf{rel}_{\mathsf{R}}[u,w]$, then there exists $[u',w']$ (in $\mathcal{M}'$) such that $[x',y']\mathsf{rel}_{\mathsf{R}}{}'[u',w']$ and $[u,w]Z[u',w']$, for any $\mathsf{R} \in \mathsf{HS}_{\mathsf{rel}}$;

(zag) If $[x,y]Z[x',y']$ and $[x',y']\mathsf{rel}_{\mathsf{R}}{}'[u',w']$, then there exists $[u,w]$ (in $\mathcal{M}$) such that $[x,y]\mathsf{rel}_{\mathsf{R}}[u,w]$ and $[u,w]Z[u',w']$, for any $\mathsf{R} \in \mathsf{HS}_{\mathsf{rel}}$.

It is easy to show by induction on an HS-formula construction that HS is *invariant for bisimulation* in the following sense.

▶ **Lemma 5.** *Let $\mathcal{M}$, $\mathcal{M}'$ be any HS-models, and Z a bisimulation between them. For any intervals $[x,y]$, $[x',y']$ if $[x,y]Z[x',y']$, then for any HS-formula φ the following are equivalent:*

1. *$\mathcal{M},[x,y] \models \varphi$;*
2. *$\mathcal{M}',[x',y'] \models \varphi$.*

▶ **Lemma 6.** *Nominals are not definable in HS under $(\leq,\mathsf{Dis},\mathsf{Non\text{-}S})$ and $(\leq,\mathsf{Den},\mathsf{Non\text{-}S})$.*

Proof. Fix an HS-model $\mathcal{M} = (\mathbb{D}, V)$ under $(\leq,\mathsf{Dis},\mathsf{Non\text{-}S})$ or $(\leq,\mathsf{Den},\mathsf{Non\text{-}S})$ such that in the case of the former semantics $\mathbb{D}$ is $\mathbb{Z}$ (i.e., the standard ordering of integers), whereas in the latter case $\mathbb{D}$ is $\mathbb{Q}$ (i.e., the standard ordering of rational numbers). Fix a nominal and towards a contradiction suppose that there is an HS-formula φ which under $(\leq,\mathsf{Dis},\mathsf{Non\text{-}S})$ or $(\leq,\mathsf{Den},\mathsf{Non\text{-}S})$ enables us to simulate this nominal. It follows that φ is satisfied in exactly one interval in $\mathcal{M}$, say $[a,b]$.

In what follows, we will construct an HS-model $\mathcal{M}' = (\mathbb{D}', V')$ and a bisimulation Z between $\mathcal{M}$ and $\mathcal{M}'$ such that $[a,b]$ is bisimilar with more than one interval. Then, by Lemma 5, φ is satisfied in more than one interval in $\mathcal{M}'$ hence we will obtain a contradiction with the statement that φ simulates a nominal.

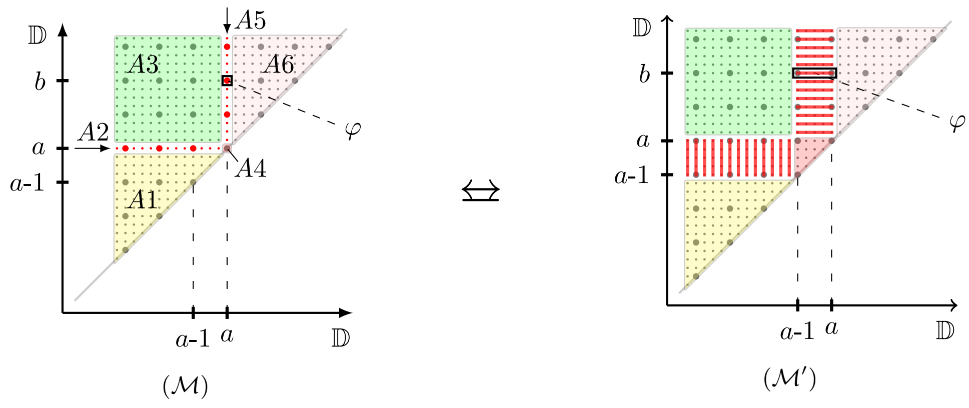


Figure 3 Bisimulation Z between models $\mathcal{M}$ and $\mathcal{M}'$.

First, we divide $\mathbb{D}$ into areas $A1$–$A6$ as follows:

$[x, y] \in A1$ iff $(x < a$ and $y < a)$; $[x, y] \in A4$ iff $(x = a$ and $y = a)$;
$[x, y] \in A2$ iff $(x < a$ and $y = a)$; $[x, y] \in A5$ iff $(x = a$ and $y > a)$;
$[x, y] \in A3$ iff $(x < a$ and $y > a)$; $[x, y] \in A6$ iff $(x > a$ and $y > a)$.

Let $\mathcal{M}' = (\mathbb{D}', V')$ be such that $\mathbb{D}' = \mathbb{D}$. Then, we exploit areas $A1$–$A6$ to define the intended bisimulation $Z \subseteq I(\mathbb{D}) \times I(\mathbb{D}')$ between intervals in $\mathcal{M}$ and $\mathcal{M}'$ (see Figure 3) as follows (we use a standard functional notation below, i.e., $Z([x, y]) = \{[x', y'] \mid [x, y]Z[x', y']\}$):

$$\begin{aligned}
&\forall[x, y] \in A1 \quad Z([x, y]) = \{[x-1, y-1]\};\\
&\forall[x, y] \in A2 \quad Z([x, y]) = \{[x, y'] \mid a-1 \leq y' \leq a\};\\
&\forall[x, y] \in A3 \quad Z([x, y]) = \{[x-1, y]\};\\
&\forall[x, y] \in A4 \quad Z([x, y]) = \{[x', y'] \mid a-1 \leq x' \leq a, \text{ and } a-1 \leq y' \leq a\};\\
&\forall[x, y] \in A5 \quad Z([x, y]) = \{[x', y] \mid a-1 \leq x' \leq a\};\\
&\forall[x, y] \in A6 \quad Z([x, y]) = \{[x, y]\}.
\end{aligned}$$

To finish defining $\mathcal{M}'$ let V' be such that for any $[x, y] \in I(\mathbb{D})$ and any $p \in \mathsf{PROP}$:

$$[x, y] \in V'(p) \quad \text{iff} \quad Z^{-1}[x, y] \in V(p).$$

Let's check if Z is a bisimulation between $\mathcal{M}$ and $\mathcal{M}'$. Condition (atom) follows directly from the definition of V'. To show that (zig) and (zag) hold for Z, observe that we may restrict the set of modal operators in the language of HS to $\langle \mathsf{R} \rangle$ and $[\mathsf{R}]$ such that $\mathsf{R} \in \{\mathsf{B}, \overline{\mathsf{B}}, \mathsf{E}, \overline{\mathsf{E}}, \mathsf{A}, \overline{\mathsf{A}}\}$. Other operators are definable as follows:

$\langle \mathsf{D} \rangle\varphi := \langle \mathsf{E} \rangle\langle \mathsf{B} \rangle\varphi$; $\langle \mathsf{O} \rangle\varphi := \langle \mathsf{E} \rangle\langle \overline{\mathsf{B}} \rangle\varphi$; $\langle \mathsf{L} \rangle\varphi := \langle \mathsf{A} \rangle\langle \mathsf{E} \rangle\varphi$;
$\langle \overline{\mathsf{D}} \rangle\varphi := \langle \overline{\mathsf{E}} \rangle\langle \overline{\mathsf{B}} \rangle\varphi$; $\langle \overline{\mathsf{O}} \rangle\varphi := \langle \mathsf{B} \rangle\langle \overline{\mathsf{E}} \rangle\varphi$; $\langle \overline{\mathsf{L}} \rangle\varphi := \langle \overline{\mathsf{A}} \rangle\langle \mathsf{B} \rangle\varphi$;

where φ is any HS-formula and the translation for box modalities is obtained by replacing $\langle \mathsf{R} \rangle$ with $[\mathsf{R}]$ for any $\mathsf{R} \in \mathsf{HS}_{\mathsf{rel}}$ in the above definitions. Hence it remains to perform a routine inspection of all $\mathsf{rel}_{\mathsf{R}}$ such that $\mathsf{R} \in \{\mathsf{B}, \overline{\mathsf{B}}, \mathsf{E}, \overline{\mathsf{E}}, \mathsf{A}, \overline{\mathsf{A}}\}$ against (zig) and (zag) conditions. We leave this inspection to the reader.

Hence, φ does not simulate a nominal. It follows that a nominal cannot be defined in HS under ($\leq$,Dis,Non-S) and ($\leq$,Den,Non-S). ◀

Let us summarize the results obtained so far in this subsection.

▶ **Theorem 7.** *Nominals and @ operators are definable in* HS *under all semantics except* $(\leq,\mathsf{Dis},\mathsf{Non\text{-}S})$ *and* $(\leq,\mathsf{Den},\mathsf{Non\text{-}S})$ *in which even nominals are not definable.*

Notice that discreteness/density of a time frame has no influence on definability of the hybrid machinery in HS. Next, we examine whether expressions of the form $\mathsf{E}p$, $\mathsf{D}p$, and $\mathsf{A}p$, where p is a propositional variable are definable in HS and if the choice of semantics affects the answer to this problem.

▶ **Theorem 8.** *For any* $p \in \mathsf{PROP}$ *expressions of the form* $\mathsf{E}p$ *and* $\mathsf{A}p$ *are definable in* HS *under all semantics.* $\mathsf{D}p$ *is definable in* HS *under all semantics except* $(\leq,\mathsf{Dis},\mathsf{Non\text{-}S})$ *and* $(\leq,\mathsf{Den},\mathsf{Non\text{-}S})$.

Proof. Expressions of the form $\mathsf{E}p$ and $\mathsf{A}p$ are definable in HS as follows:

$$\mathsf{E}p := \langle \mathsf{L}\rangle\langle \overline{\mathsf{L}}\rangle p; \tag{5}$$

$$\mathsf{A}p := [\mathsf{L}][\overline{\mathsf{L}}]p. \tag{6}$$

An expression of the form $\mathsf{D}p$ is definable in HS under $(<,\mathsf{Dis},\mathsf{S})$, $(<,\mathsf{Den},\mathsf{S})$, $(<,\mathsf{Dis},\mathsf{Non\text{-}S})$, and $(<,\mathsf{Den},\mathsf{Non\text{-}S})$ by (3), whereas under $(\leq,\mathsf{Dis},\mathsf{S})$ and $(\leq,\mathsf{Den},\mathsf{S})$ by (4).

To show that $\mathsf{D}p$ is not definable in HS under $(\leq,\mathsf{Dis},\mathsf{Non\text{-}S})$ and $(\leq,\mathsf{Den},\mathsf{Non\text{-}S})$ suppose that $\mathsf{D}p$ is definable in these semantics. Then by Theorem 4 we obtain that nominals are definable in HS under $(\leq,\mathsf{Dis},\mathsf{Non\text{-}S})$ and $(\leq,\mathsf{Den},\mathsf{Non\text{-}S})$. However, by Lemma 6 we know that it is not the case, so we obtain a contradiction. ◀

4.2 The Case of Horn Fragments of Halpern-Shoham Logic

In what follows, we show that hybrid machinery is definable in $\mathsf{HS}^{\Diamond}_{horn}$ under all semantics except $(\leq,\mathsf{Dis},\mathsf{Non\text{-}S})$ and $(\leq,\mathsf{Den},\mathsf{Non\text{-}S})$. However, we can no longer use Theorem 4, which holds if there are no restrictions on the use of classical propositional connectives $(\neg, \wedge, \vee, \rightarrow)$. As a result, we need to conduct the proof exploiting other techniques.

▶ **Lemma 9.** *Nominals and satisfaction operators are definable in the language of* $\mathsf{HS}^{\Diamond}_{horn}$ *under* $(<,\mathsf{Dis},\mathsf{S})$, $(<,\mathsf{Den},\mathsf{S})$, $(<,\mathsf{Dis},\mathsf{Non\text{-}S})$, $(<,\mathsf{Den},\mathsf{Non\text{-}S})$, $(\leq,\mathsf{Dis},\mathsf{S})$, *and* $(\leq,\mathsf{Den},\mathsf{S})$.

Proof. Let φ be a formula in the language of $\mathsf{HS}^{\Diamond,i,@}_{horn}$. We show how to construct under $(<,\mathsf{Dis},\mathsf{S})$, $(<,\mathsf{Den},\mathsf{S})$, $(<,\mathsf{Dis},\mathsf{Non\text{-}S})$, $(<,\mathsf{Den},\mathsf{Non\text{-}S})$, $(\leq,\mathsf{Dis},\mathsf{S})$, and $(\leq,\mathsf{Den},\mathsf{S})$ an equisatisfiable formula φ' in the language of $\mathsf{HS}^{\Diamond}_{horn}$ of size linear with respect to $|\varphi|$ (where $|\varphi|$ is the length of φ).

We will simulate any nominal i occurring in φ with a propositional variable p_i. We introduce a formula ψ_i expressing that p_i is satisfied in exactly one interval. In the case of any irreflexive semantics define:

$$\psi_i := \langle \mathsf{L}\rangle\langle \overline{\mathsf{L}}\rangle p_i \wedge \tag{7}$$

$$[\mathsf{U}](p_i \wedge \langle \overline{\mathsf{B}}\rangle\langle \overline{\mathsf{E}}\rangle\langle \mathsf{E}\rangle p_i \rightarrow \bot) \wedge \tag{8}$$

$$[\mathsf{U}](p_i \wedge \langle \overline{\mathsf{E}}\rangle\langle \overline{\mathsf{B}}\rangle\langle \mathsf{B}\rangle p_i \rightarrow \bot). \tag{9}$$

In the case of $(\mathsf{Dis},\leq,\mathsf{S})$ and $(\mathsf{Den},\leq,\mathsf{S})$ define ψ_i as:

$$\psi_i := \langle \mathsf{L}\rangle\langle \overline{\mathsf{L}}\rangle p_i \wedge \tag{10}$$

$$[\mathsf{U}](p_i \wedge \langle \mathsf{A}\rangle\langle \overline{\mathsf{E}}\rangle\langle \mathsf{E}\rangle p_i \rightarrow \bot) \wedge \tag{11}$$

$$[\mathsf{U}](p_i \wedge \langle \overline{\mathsf{A}}\rangle\langle \overline{\mathsf{B}}\rangle\langle \mathsf{B}\rangle p_i \rightarrow \bot). \tag{12}$$

Under any irreflexive semantics (7) states that p_i is satisfied somewhere, (8) expresses that p_i cannot be satisfied in any two intervals $[x, y]$, $[x', y']$ such that $y' > y$, whereas (9) disallows p_i being satisfied in any two intervals $[x, y]$, $[x', y']$ such that $x' < x$. Formulas (10)–(12) have the analogous meaning under ($\leq$,Dis,S) and ($\leq$,Den,S). As a result, (7)–(9) as well as (10)–(12) enable us to simulate a nominal i with a propositional variable p_i.

Importantly, in ($\leq$,Dis,$\mathsf{Non\text{-}S}$) and ($\leq$,Den,$\mathsf{Non\text{-}S}$) none of the above encodings enable us to state that p_i holds in exactly one interval. (7)–(9) would not work because in reflexive semantics $\mathsf{rel}_{\mathsf{E}}$, $\mathsf{rel}_{\overline{\mathsf{E}}}$, $\mathsf{rel}_{\mathsf{B}}$, and $\mathsf{rel}_{\overline{\mathsf{B}}}$ are reflexive. On the other hand, (10)–(12) would also fail because under ($\leq$,Dis,$\mathsf{Non\text{-}S}$) and ($\leq$,Den,$\mathsf{Non\text{-}S}$) relations $\mathsf{rel}_{\mathsf{A}}$ and $\mathsf{rel}_{\overline{\mathsf{A}}}$ are reflexive in punctual-intervals, i.e., for any interval of the form $[x, x]$ we have $[x, x]\mathsf{rel}_{\mathsf{A}}[x, x]$, and $[x, x]\mathsf{rel}_{\overline{\mathsf{A}}}[x, x]$.

Let φ_1 be obtained from φ by replacing each occurrence of a nominal i (except symbols of nominals occurring in the index of a satisfaction operator) by a propositional variable p_i and by adding to the obtained formula a conjunction of the form $\bigwedge_{i \in \mathsf{NOM}(\varphi)} \psi_i$, where $\mathsf{NOM}(\varphi)$ is a set of nominals occurring in φ.

Next, we show how to replace occurrences of satisfaction operators $@_i$ in φ_1 in order to obtain an equisatisfiable formula in the language of $\mathsf{HS}^{\Diamond}_{horn}$. The construction is by induction on the number of $@_i$ operators occurring in φ_1. In each step of the construction choose any positive temporal literal λ in the so far constructed formula, such that some satisfaction operator occurs in λ. Find the left-most satisfaction operator occurring in λ, i.e., an operator $@_i$ such that $\lambda = \langle \mathsf{R}_1 \rangle \ldots \langle \mathsf{R}_n \rangle @_i \eta$ with $\langle \mathsf{R}_1 \rangle \ldots \langle \mathsf{R}_n \rangle$ being a (possibly empty) sequence of diamond modal operators and η a subformula of λ. Replace this occurrence of $@_i \eta$ by $\langle \mathsf{L} \rangle \langle \overline{\mathsf{L}} \rangle p_{@_i \eta}$, where $p_{@_i \eta}$ is a fresh propositional variable which did not occur in the so far constructed formula. Moreover, add to the constructed formula the following conjunction:

$$[\mathsf{U}](p_i \land \eta \to p_{@_i \eta}) \land [\mathsf{U}](p_{@_i \eta} \to p_i) \land [\mathsf{U}](p_{@_i \eta} \to \eta). \tag{13}$$

Formula (13) states that $p_{@_i \eta}$ is satisfied exactly in an interval in which $p_i \land \eta$ is satisfied. Hence, $\langle \mathsf{L} \rangle \langle \overline{\mathsf{L}} \rangle p_{@_i \eta}$ – stating that $p_{@_i \eta}$ is satisfied somewhere – is equisatisfiable with $@_i \eta$.

The construction terminates when all occurrences of satisfaction operators are eliminated and we denote the finally obtained formula by φ'. It is easy to see that φ' is equisatisfiable with φ and, since we have eliminated all occurrences of nominals and satisfaction operators, φ' is in the language of $\mathsf{HS}^{\Diamond}_{horn}$. Moreover, the size of φ' is linear in the size of φ because within the construction of φ' for each occurrence of a nominal and a satisfaction operator we have added only a constant number of symbols to the formula. ◀

As a corollary to Lemma 6 we obtain that nominals are not definable in $\mathsf{HS}^{\Diamond}_{horn}$ under ($\leq$,Dis,$\mathsf{Non\text{-}S}$) and ($\leq$,Den,$\mathsf{Non\text{-}S}$). Hence, we get the following result.

▶ **Theorem 10.** *Nominals and satisfaction operators are definable in* $\mathsf{HS}^{\Diamond}_{horn}$ *under all semantics except* ($\leq$,Dis,$\mathsf{Non\text{-}S}$) *and* ($\leq$,Den,$\mathsf{Non\text{-}S}$) *in which nominals are not definable.*

Interestingly, although nominals and satisfaction operators are definable in $\mathsf{HS}^{\Diamond}_{horn}$ (except ($\leq$,Dis,$\mathsf{Non\text{-}S}$) and ($\leq$,Den,$\mathsf{Non\text{-}S}$)), we will show now that D is not.

▶ **Theorem 11.** *For any* $p \in \mathsf{PROP}$ *expressions of the form* $\mathsf{E}p$ *and* $\mathsf{A}p$ *are definable in* $\mathsf{HS}^{\Diamond}_{horn}$ *but* $\mathsf{D}p$ *is not. This result holds for all semantics.*

Proof Sketch. An expression of the form $\mathsf{E}p$ is definable by (5), whereas $\mathsf{A}p$ is definable as follows: $\mathsf{A}p := [\mathsf{U}](\top \to p)$. To show that an expression of the form $\mathsf{D}p$ is not definable

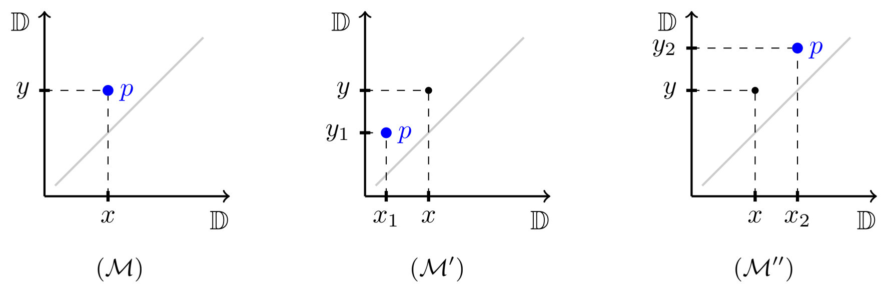


Figure 4 Isomorphic HS-models $\mathcal{M}$, $\mathcal{M}'$, and $\mathcal{M}''$.

in $\mathsf{HS}^{\Diamond}_{horn}$ consider HS-models $\mathcal{M} = (\mathbb{D}, V)$, $\mathcal{M}' = (\mathbb{D}', V')$, and $\mathcal{M}'' = (\mathbb{D}'', V'')$ (in any semantics), where $\mathbb{D} = (D, \leq)$, $\mathbb{D}' = (D', \leq)$, $\mathbb{D}'' = (D'', \leq)$, and:

$$V(p) = \{[x, y]\}; \qquad V'(p) = \{[x_1, y_1]\}; \qquad V''(p) = \{[x_2, y_2]\};$$

where $V(q) = V'(q) = V''(q) = \emptyset$ for any propositional variable $q \neq p$. D' is an isomorphic translation by an integer $c < 0$ (to the left) of D ($[x_1, y_1]$ is an image of $[x, y]$ with respect to this translation). Analogously, D'' is an isomorphic translation by an integer $c > 0$ (to the right) of D ($[x_2, y_2]$ is an image of $[x, y]$ with respect to this translation) – see Figure 4. Towards a contradiction suppose that there is an $\mathsf{HS}^{\Diamond}_{horn}$-formula $\varphi_{\mathsf{D}p}$ expressing $\mathsf{D}p$. Hence, $\mathcal{M}, [x, y] \not\models \varphi_{\mathsf{D}p}$, $\mathcal{M}', [x, y] \models \varphi_{\mathsf{D}p}$, and $\mathcal{M}'', [x, y] \models \varphi_{\mathsf{D}p}$. We will show that ($\star$) for any $\mathsf{HS}^{\Diamond}_{horn}$-formula φ if $\mathcal{M}', [x, y] \models \varphi$ and $\mathcal{M}'', [x, y] \models \varphi$, then $\mathcal{M}, [x, y] \models \varphi$ which will give rise to a contradiction and finish the proof.

Let ψ be any conjunct of φ. Then ψ is of a form $[\mathsf{U}](\lambda_1 \wedge \ldots \wedge \lambda_n \to \lambda_{n+1})$ or λ, where λ_i and λ are generated by the grammar $\lambda := \top \mid \bot \mid r \mid \langle \mathsf{R} \rangle \lambda$. (Case 1): ψ is of a form $[\mathsf{U}](\lambda_1 \wedge \ldots \wedge \lambda_n \to \lambda_{n+1})$. Since the formula is preceded by $[\mathsf{U}]$ and models $\mathcal{M}$, $\mathcal{M}'$, and $\mathcal{M}''$ are isomorphic, then ψ is true in all three models or false in all of them, hence ($\star$) holds. (Case 2): ψ is of the form $\top$, $\bot$, $\langle \mathsf{R}_1 \rangle \ldots \langle \mathsf{R}_n \rangle \top$, $\langle \mathsf{R}_1 \rangle \ldots \langle \mathsf{R}_n \rangle \bot$, p, q, or $\langle \mathsf{R}_1 \rangle \ldots \langle \mathsf{R}_n \rangle q$, where $q \neq p$ and $\mathsf{R}_i \in \mathsf{HS}_{\mathsf{rel}}$. Then it is easy to see that ($\star$) holds. (Case 3) $\psi = \langle \mathsf{R}_1 \rangle \ldots \langle \mathsf{R}_n \rangle p$ for any $\mathsf{R}_1, \ldots, \mathsf{R}_n \in \mathsf{HS}_{\mathsf{rel}}$. As showed previously we may consider only $\mathsf{R}_i \in \{\mathsf{B}, \overline{\mathsf{B}}, \mathsf{E}, \overline{\mathsf{E}}, \mathsf{A}, \overline{\mathsf{A}}\}$. Assume that $\mathcal{M}', [x, y] \models \psi$ and $\mathcal{M}'', [x, y] \models \psi$. To show that $\mathcal{M}, [x, y] \models \psi$ it suffices to prove the following statement:

($\star\star$) For any sequence $\mathsf{R}_1, \ldots, \mathsf{R}_n$ of relations from $\{\mathsf{B}, \overline{\mathsf{B}}, \mathsf{E}, \overline{\mathsf{E}}, \mathsf{A}, \overline{\mathsf{A}}\}$ the following holds: if for some intervals $[x, y]$, $[x', y']$, $[x'', y'']$ we have $[x, y]\mathsf{rel}_{\mathsf{R}_1} \circ \ldots \circ \mathsf{rel}_{\mathsf{R}_n}[x', y']$ and $[x, y]\mathsf{rel}_{\mathsf{R}_1} \circ \ldots \circ \mathsf{rel}_{\mathsf{R}_n}[x'', y'']$, then for any interval $[s, t]$ such that $\min(x', x'') \leq s \leq \max(x', x'')$, and $\min(y', y'') \leq t \leq \max(y', y'')$, and $t - s \geq \min(y' - x', y'' - x'')$ it holds that $[x, y]\mathsf{rel}_{\mathsf{R}_1} \circ \ldots \circ \mathsf{rel}_{\mathsf{R}_n}[s, t]$,

where $\circ$ is the composition operator and for any interval $[x, y]$ we use "$x - y$" to denote the number of time-points between x and y (notice that in the case of a dense frame $x - y$ equals 0 or infinity). Because of space limits we leave the proof of ($\star\star$) to the reader. The proof may be conducted by an induction on the number n. Showing that ($\star\star$) finishes the proof. ◀

In the remaining part of this subsection we consider expressiveness of $\mathsf{HS}^{\Box}_{horn}$. We will show that under (<,Den,S), (<,Den,Non-S), (≤,Dis,Non-S), (≤,Den,S), and (≤,Den,Non-S), $\mathsf{HS}^{\Box}_{horn}$ is not expressive enough to define the *somewhere* and *difference* modalities, as well as nominals. However, the *universal* modality is still definable in $\mathsf{HS}^{\Box}_{horn}$.

▶ **Theorem 12.** *Nominals and expressions of the form* $\mathsf{E}p$ *and* $\mathsf{D}p$ *for* $p \in \mathsf{PROP}$ *are not definable in* $\mathsf{HS}^{\Box}_{horn}$ *under* (<,Den,S), (<,Den,Non-S), (≤,Dis,Non-S), (≤,Den,S), *and* (≤,Den,Non-S). *Whereas,* $\mathsf{A}p$ *is definable in* $\mathsf{HS}^{\Box}_{horn}$ *under all semantics.*

Proof. An expression of the form $\mathsf{A}p$ is definable as $\mathsf{A}p := [\mathsf{U}](\top \to p)$. To show that nominals, $\mathsf{E}p$, and $\mathsf{D}p$ are not definable under ($<$,Den,S), ($<$,Den,Non-S), ($\leq$,Dis,Non-S), ($\leq$,Den,S), and ($\leq$,Den,Non-S) we will use a result from [8], where under these semantics a construction of a *canonical* $\mathsf{HS}^{\Box}_{horn}$-model $\mathcal{K}^{[a,b]}_{\varphi}$ of an HS-formula φ in an interval $[a,b]$ is presented. The model is canonical in the following sense [8, Theorem 3.2]:

(a) If in some HS-model $\mathcal{M}$ it holds that $\mathcal{M},[a,b] \models \varphi$, then $\mathcal{K}^{[a,b]}_{\varphi},[a,b] \models \varphi$, and

(b) For any interval $[x,y]$ and any $p \in \mathsf{PROP}$ if $\mathcal{K}^{[a,b]}_{\varphi},[x,y] \models p$, then in any HS-model $\mathcal{M}$ such that $\mathcal{M},[a,b] \models \varphi$ we have $\mathcal{M},[x,y] \models p$.

Towards a contradiction let us suppose that nominals, $\mathsf{E}p$, and $\mathsf{D}p$ are definable in $\mathsf{HS}^{\Box}_{horn}$ under ($<$,Den,S), ($<$,Den,Non-S), ($\leq$,Dis,Non-S), ($\leq$,Den,S), and ($\leq$,Den,Non-S). Let φ be an $\mathsf{HS}^{\Box}_{horn}$-formula expressing that (i) $p \in \mathsf{PROP}$ simulates a nominal i, or (ii) $\mathsf{E}p$, or (iii) $\mathsf{D}p$. We will reach a contradiction no matter in which of these forms φ is.

Let $\mathcal{M} = (\mathbb{D}, V)$ and $\mathcal{M}' = (\mathbb{D}, V')$ be HS-models such that $V(p) = \{[x,y]\}$, $V'(p) = \{[x',y']\}$, and $[x,y] \neq [x',y']$. Let $[a,b]$ be an interval distinct from $[x,y]$ and $[x',y']$. Hence, $\mathcal{M},[a,b] \models \varphi$ and $\mathcal{M}',[a,b] \models \varphi$. By (a) $\mathcal{K}^{[a,b]}_{\varphi},[a,b] \models \varphi$. From the definition of φ – see (i), (ii), and (iii) – it follows that there is $[u,w]$ such that $\mathcal{K}^{[a,b]}_{\varphi},[u,w] \models p$. Then by (b) we obtain that $\mathcal{M},[u,w] \models p$ and $\mathcal{M}',[u,w] \models p$. (Case 1): $[u,w] \neq [x,y]$. Then $\mathcal{M},[u,w] \not\models p$. (Case 2): $[u,w] \neq [x',y']$. Then $\mathcal{M}',[u,w] \not\models p$. We have obtained a contradiction in both cases, hence nominals, $\mathsf{E}p$, and $\mathsf{D}p$ are not definable in $\mathsf{HS}^{\Box}_{horn}$ under ($<$,Den,S), ($<$,Den,Non-S), ($\leq$,Dis,Non-S), ($\leq$,Den,S), and ($\leq$,Den,Non-S). ◀

As a corollary to the above theorem we obtain that adding nominals to the language of $\mathsf{HS}^{\Box}_{horn}$ (which results in obtaining $\mathsf{HS}^{\Box,i}_{horn}$) strictly increases expressive power of the language (in the listed semantics). In what follows, we will show that the further extension of $\mathsf{HS}^{\Box,i}_{horn}$ obtained by adding satisfaction operators (i.e., reaching $\mathsf{HS}^{\Box,i,@}_{horn}$) does not increase its expressiveness in any semantics.

▶ **Theorem 13.** *@ operators are definable in the language of* $\mathsf{HS}^{\Box,i}_{horn}$ *under all semantics.*

Proof. Let φ be an $\mathsf{HS}^{\Box,i,@}_{horn}$-formula. Construction of an equisatisfiable $\mathsf{HS}^{\Box,i}_{horn}$-formula φ' is by induction on the number of $@_i$ operators occurring in φ. In each step of the construction choose any positive temporal literal λ in which occurs a satisfaction operator and find the left-most satisfaction operator occurring in λ, i.e., an operator $@_i$ such that $\lambda = [\mathsf{R}_1]\dots[\mathsf{R}_n]@_i\eta$ with $[\mathsf{R}_1]\dots[\mathsf{R}_n]$ being a (possibly empty) sequence of box modal operators and η a subformula of λ. We replace this occurrence of $@_i\eta$ by $[\mathsf{L}][\bar{\mathsf{L}}]p_{@i\eta}$, where $p_{@i\eta}$ is a fresh propositional variable that did not occur in the so far constructed formula. Moreover, add to the constructed formula the following conjunction:

$$[\mathsf{U}](i \wedge \eta \to [\mathsf{L}][\bar{\mathsf{L}}]p_{@i\eta}) \wedge [\mathsf{U}]([\mathsf{L}][\bar{\mathsf{L}}]p_{@i\eta} \wedge i \to \eta), \tag{14}$$

where (14) allows us to simulate $@_i\eta$ with $[\mathsf{L}][\bar{\mathsf{L}}]p_{@i\eta}$. The construction terminates when all occurrences of satisfaction operators are eliminated. The resulting formula is in the language of $\mathsf{HS}^{\Box,i}_{horn}$ and is equisatisfiable with φ. ◀

5 Conclusions

Recently, decidability and computational complexity of Horn fragments of HS have been intensively investigated. However, significantly less research has focused on their expressive power. In this paper we address this gap by showing which semantics are definable in full HS and which operators (nominals, satisfaction operators, *somewhere*, *difference*, and *everywhere* operators) are definable in various Horn fragments of HS. Our results on the relative expressive power of HS Horn fragments are summarised diagrammatically in Figure 1b.

There are still numerous open problems concerning the expressive power of HS fragments. For instance, not much is known about the expressive power of *core* fragments of HS [8, 9], which are obtained by imposing further restrictions on Horn fragments. In particular, an interesting open question is whether the expressive power diagram for such fragments is analogous to the one for Horn fragments presented in Figure 1b.

Acknowledgements. The author is grateful to Michał Zawidzki and Carl Schultz for valuable substantive an presentational comments, respectively, as well as to anonymous reviewers.

References

1 James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

2 Carlos Areces, Patrick Blackburn, and Maarten Marx. The computational complexity of hybrid temporal logics. *Logic Journal of IGPL*, 8(5):653–679, 2000.

3 Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. Tractable interval temporal propo-sitional and description logics. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 1417–1423. 2015.

4 Patrick Blackburn, Maarten De Rijke, and Yde Venema. *Modal Logic*, volume 53. Cambridge University Press, 2002.

5 Sebastian Brandt, E. Güzel Kalaycı, Roman Kontchakov, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. Ontology-based data access with a horn fragment of metric temporal logic. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1070–1076, 2017.

6 Davide Bresolin, Dario Della Monica, Angelo Montanari, Pietro Sala, and Guido Sciavicco. Interval temporal logics over strongly discrete linear orders: Expressiveness and complexity. *Theoretical Computer Science*, 560:269–291, 2014.

7 Davide Bresolin, Dario Della Monica, Angelo Montanari, Pietro Sala, and Guido Sciavicco. The light side of interval temporal logic: The Bernays-Schönfinkel's fragment of CDT. *Annals of Mathematics and Artificial Intelligence*, 71:11–39, 2014.

8 Davide Bresolin, Agi Kurucz, Emilio Muñoz-Velasco, Vladislav Ryzhikov, Guido Sciavicco, and Michael Zakharyaschev. Horn fragments of the Halpern-Shoham interval temporal logic, Forthcoming. URL: `https://arxiv.org/pdf/1604.03515v1.pdf`.

9 Davide Bresolin, Emilio Muñoz-Velasco, and Guido Sciavicco. Sub-propositional fragments of the interval temporal logic of Allen's relations. In *European Workshop on Logics in Artificial Intelligence*, pages 122–136. Springer, 2014.

10 Davide Bresolin, Emilio Munoz-Velasco, and Guido Sciavicco. On the complexity of fragments of horn modal logics. In *Temporal Representation and Reasoning (TIME), 2016 23rd International Symposium on*, pages 186–195. IEEE, 2016.

11 Dario Della Monica, Valentin Goranko, Angelo Montanari, Guido Sciavicco, et al. Interval temporal logics: a journey. *Bulletin of EATCS*, 3(105), 2013.

12 Valentin Goranko, Angelo Montanari, and Guido Sciavicco. A road map of interval temporal logics and duration calculi. *Journal of Applied Non-Classical Logics*, 14(1-2):9–54, 2004.

13 Joseph Y. Halpern and Yoav Shoham. A propositional modal logic of time intervals. *Journal of the ACM (JACM)*, 38(4):935–962, 1991.

14 Roman Kontchakov, Laura Pandolfo, Luca Pulina, Vladislav Ryzhikov, and Michael Zakharyaschev. Temporal and spatial OBDA with many-dimensional Halpern-Shoham logic. In *IJCAI*, pages 1160–1166, 2016.

15 Angelo Montanari, Ian Pratt-Hartmann, and Pietro Sala. Decidability of the logics of the reflexive sub-interval and super-interval relations over finite linear orders. In *Temporal Representation and Reasoning (TIME), 2010 17th International Symposium on*, pages 27–34. IEEE, 2010.

16 Yde Venema. Expressiveness and completeness of an interval tense logic. *Notre Dame journal of formal logic*, 31(4):529–547, 1990.

17 Przemysław A. Wałęga. Computational complexity of a hybridized Horn fragment of Halpern-Shoham logic. In *Indian Conference on Logic and Its Applications*, pages 224–238. Springer, 2017.

A Appendix (proofs)

Proof of Lemma 5. Fix HS-models $\mathcal{M}$, $\mathcal{M}'$, a bisimulation Z between them and bisimilar intervals $[x,y]$, $[x',y']$, i.e., intervals such that $[x,y]Z[x',y']$. We show by induction on a construction of an HS-formula that for any HS-formula φ the following conditions are equivalent:

1. $\mathcal{M},[x,y] \models \varphi$;
2. $\mathcal{M}',[x',y'] \models \varphi$.

(Case 1) $\varphi \in \mathsf{PROP}$. By (atom) it follows that $\mathcal{M},[x,y] \models \varphi$ iff $\mathcal{M}',[x',y'] \models \varphi$.

(Case 2) $\varphi = \neg\psi$ for any HS-formula ψ. By the inductive assumption $\mathcal{M},[x,y] \models \psi$ iff $\mathcal{M}',[x',y'] \models \psi$. As a result, $\mathcal{M},[x,y] \models \varphi$ iff $\mathcal{M}',[x',y'] \models \varphi$.

(Case 3) $\varphi = \psi \wedge \xi$ for any HS-formulas ψ, ξ. By the inductive assumption $\mathcal{M},[x,y] \models \psi$ iff $\mathcal{M}',[x',y'] \models \psi$, and $\mathcal{M},[x,y] \models \xi$ iff $\mathcal{M}',[x',y'] \models \xi$. Hence, $\mathcal{M},[x,y] \models \varphi$ iff $\mathcal{M}',[x',y'] \models \varphi$.

(Case 4) $\varphi = \langle \mathsf{R} \rangle \psi$ for any $\mathsf{R} \in \mathsf{HS}_{\mathsf{rel}}$ and any HS-formula ψ.

(1 $\Rightarrow$ 2) Assume $\mathcal{M},[x,y] \models \varphi$. Then, there is $[u,w]$ such that $[x,y]\mathsf{rel}_{\mathsf{R}}[u,w]$ and $\mathcal{M},[u,w] \models \psi$. $[x,y]Z[x',y']$ so by (zig) there is $[u',w']$ such that $[x',y']\mathsf{rel}_{\mathsf{R}}{}'[u',w']$ and $[u,w]Z[u',w']$. Therefore, by the inductive assumption $\mathcal{M}',[u',w'] \models \psi$, so $\mathcal{M}',[x',y'] \models \varphi$.

(1 $\Leftarrow$ 2) Is proved analogously as (1 $\Rightarrow$ 2) but using (zag) instead of (zig).

(Case 5) $\varphi = [\mathsf{R}]\psi$ for any $\mathsf{R} \in \mathsf{HS}_{\mathsf{rel}}$ and any HS-formula ψ.

(1 $\Rightarrow$ 2) Assume $\mathcal{M},[x,y] \models \varphi$. To show $\mathcal{M}',[x',y'] \models \varphi$. Fix any $[u',w']$ such that $[x',y']\mathsf{rel}_{\mathsf{R}}{}'[u',w']$. To show that $\mathcal{M}',[u',w'] \models \psi$. $[x,y]Z[x',y']$ so by (zag) there is $[u,w]$ such that $[x,y]\mathsf{rel}_{\mathsf{R}}[u,w]$ and $[u,w]Z[u',w']$. $\mathcal{M},[x,y] \models [\mathsf{R}]\psi$, so $\mathcal{M},[u,w] \models \psi$. By the inductive assumption $\mathcal{M}',[u',w'] \models \psi$, hence we obtain $\mathcal{M}',[x',y'] \models \varphi$.

(1 $\Leftarrow$ 2) Is proved analogously as (1 $\Rightarrow$ 2) but using (zig) instead of (zag). ◀

Theorem 4 was stated in [2] without a proof. In what follows, we present a proof of this theorem.

Proof of Theorem 4. Let $\mathcal{L}$ be a modal language in which the difference operator D is polynomially definable. Then, let $\mathcal{L}^{ext}$ be an extension of $\mathcal{L}$ with nominals and @ operators. Fix any $\varphi^{ext} \in \mathcal{L}^{ext}$. To prove the theorem we show a polynomial (with respect to the size of φ^{ext}) translation of φ^{ext} into $\varphi \in \mathcal{L}$ such that φ^{ext} is satisfiable iff φ is.

First, we show that E and A are polynomially definable in $\mathcal{L}$. Indeed, they are defined as:

$$\mathsf{E}\psi := \psi \vee \mathsf{D}\psi;$$
$$\mathsf{A}\psi := \neg\mathsf{E}\neg\psi.$$

Now, we show a step by step construction of φ. For any i occurring in φ^{ext} as an nominal or an index of some @ operator occurring in φ^{ext}, we introduce a fresh propositional variable p_i (i.e., not occurring in φ^{ext}) such that all p_i's are pairwise different. Then we proceed as follows.

First, inductively apply the following procedure to φ^{ext}. Let $@_i\psi$ be any subformula of φ^{ext} such that ψ does not contain any occurrence of @ operators. Construct φ' by replacing $@_i\psi$ with a new variable q_k and let:

$$\begin{aligned}\theta_k = &(\mathsf{A}q_k \lor \mathsf{A}\neg q_k) \land \\ &(\mathsf{A}q_k \to \mathsf{E}(p_i \land \psi)) \land \\ &(\mathsf{A}\neg q_k \to \mathsf{A}\neg\psi)\,; \\ \chi_i = &\ \mathsf{E}p_i \land \mathsf{A}(p_i \to \neg\mathsf{D}p_i).\end{aligned}$$

It is easy to see that φ^{ext} is satisfiable iff $\varphi' \land \theta_k \land \chi_i$ is. The procedure finishes when no more @ operators are in the constructed formula. Let φ'' be a conjunction of the finally constructed formula with all θ_k's and all χ_i's constructed so far. Obviously, φ'' does not contain @ operators and φ^{ext} is satisfiable iff φ'' is.

Second, for any nominal j occurring in φ'' replace all its occurrences in φ'' by p_j and let

$$\gamma_j = \mathsf{E}p_j \land \mathsf{A}(p_j \to \neg\mathsf{D}p_j).$$

A conjunction of the obtained formula with all γ_j's is the final formula φ.

Formula φ does not contain @ operators nor nominals, and φ^{ext} is satisfiable iff φ is. Furthermore, the translation of φ^{ext} to φ is polynomial. More precisely, it is at most quadratic in the size of φ^{ext}. ◀

Completion of the proof of Theorem 11. To finish the proof it remains to prove the following statement:

($\star\star$) For any sequence $\mathsf{R}_1, \dots, \mathsf{R}_n$ of relations from $\{\mathsf{B}, \overline{\mathsf{B}}, \mathsf{E}, \overline{\mathsf{E}}, \mathsf{A}, \overline{\mathsf{A}}\}$ the following holds: if for some intervals $[x, y]$, $[x', y']$, $[x'', y'']$ we have $[x, y]\mathsf{rel}_{\mathsf{R}_1} \circ \dots \circ \mathsf{rel}_{\mathsf{R}_n}[x', y']$ and $[x, y]\mathsf{rel}_{\mathsf{R}_1} \circ \dots \circ \mathsf{rel}_{\mathsf{R}_n}[x'', y'']$, then for any interval $[s, t]$ such that $\min(x', x'') \leq s \leq \max(x', x'')$, and $\min(y', y'') \leq t \leq \max(y', y'')$, and $t - s \geq \min(y' - x', y'' - x'')$ it holds that $[x, y]\mathsf{rel}_{\mathsf{R}_1} \circ \dots \circ \mathsf{rel}_{\mathsf{R}_n}[s, t]$,

where $\circ$ is the composition operator and for any interval $[x, y]$ we use "$x - y$" to denote a number of time-points between x and y (in the case of dense time frame $x - y$ equals 0 or infinity).

We will prove ($\star\star$) by induction on the number n of relations. Let us fix an interval $[x, y]$ and assume that the statement holds for k relations, i.e., for any $\mathsf{R}_1, \dots, \mathsf{R}_k \in \{\mathsf{B}, \overline{\mathsf{B}}, \mathsf{E}, \overline{\mathsf{E}}, \mathsf{A}, \overline{\mathsf{A}}\}$. Let

$$X = \{[x', y'] \mid [x, y]\mathsf{rel}_{\mathsf{R}_1} \circ \dots \circ \mathsf{rel}_{\mathsf{R}_k}[x', y']\}.$$

We will shows that the statement holds for any $k + 1$ relations, i.e., for any $\mathsf{R}_1, \dots, \mathsf{R}_{k+1} \in \{\mathsf{B}, \overline{\mathsf{B}}, \mathsf{E}, \overline{\mathsf{E}}, \mathsf{A}, \overline{\mathsf{A}}\}$. Let us fix any $\mathsf{R}_{k+1} \in \{\mathsf{B}, \overline{\mathsf{B}}, \mathsf{E}, \overline{\mathsf{E}}, \mathsf{A}, \overline{\mathsf{A}}\}$ and define

$$X' = \{[x', y'] \mid [x, y]\mathsf{rel}_{\mathsf{R}_1} \circ \dots \circ \mathsf{rel}_{\mathsf{R}_{k+1}}[x', y']\}.$$

Fix any intervals $[x', y'], [x'', y''] \in X'$ and any interval $[s, t]$ such that $\min(x', x'') \leq s \leq \max(x', x'')$ and $\min(y', y'') \leq t \leq \max(y', y'')$. We need to show that $[s, t] \in X'$. In what follows we will assume that the semantics is irreflexive but for any reflexive semantics the proof is analogous (namely by replacing "$<$" with "$\leq$" in the remaining part of the proof).

(Case 1) $\mathsf{R}_{k+1} = \mathsf{B}$. Then there are intervals $[u', w'], [u'', w''] \in X$ such that $[u', w']\mathsf{rel}_{\mathsf{B}}[x', y']$ and $[u'', w'']\mathsf{rel}_{\mathsf{B}}[x'', y'']$. By the definition of $\mathsf{rel}_{\mathsf{B}}$ it follows that $u' = x'$, $w' > y'$, $u'' = x''$, and $w'' > y''$. Let $z = \max(w', w'')$, then by the inductive assumption we obtain $[s, z] \in X$. It follows that $z > y'$ and $z > y''$, so $z > t$. Then $[s, z]\mathsf{rel}_{\mathsf{B}}[s, t]$, so $[s, t] \in X'$.

(Case 2) $\mathsf{R}_{k+1} = \overline{\mathsf{B}}$. Then there are intervals $[u', w'], [u'', w''] \in X$ such that $[u', w']\mathsf{rel}_{\overline{\mathsf{B}}}[x', y']$ and $[u'', w'']\mathsf{rel}_{\overline{\mathsf{B}}}[x'', y'']$. By the definition of $\mathsf{rel}_{\overline{\mathsf{B}}}$ it follows that $u' = x'$, $w' < y'$, $u'' = x''$, and $w'' < y''$. Let $[s, z]$ be such an interval that $z - s = \min(w' - u', w'' - u'')$, then by the inductive assumption $[s, z] \in X$. Since $w' < y'$ and $w'' < y''$, $z - s < \min(y' - x', y'' - x'')$. Hence $z - s < t - s$, so $z < t$. It follows that $[s, z]\mathsf{rel}_{\overline{\mathsf{B}}}[s, t]$ and so $[s, t] \in X'$.

(Case 3) $\mathsf{R}_{k+1} = \mathsf{E}$. Then there are intervals $[u', w'], [u'', w''] \in X$ such that $[u', w']\mathsf{rel}_{\mathsf{E}}[x', y']$ and $[u'', w'']\mathsf{rel}_{\mathsf{E}}[x'', y'']$. By the definition of $\mathsf{rel}_{\mathsf{E}}$ it follows that $u' < x'$, $w' = y'$, $u'' < x''$, and $w'' = y''$. Let $z = \min(u', u'')$, then by the inductive assumption we obtain $[z, t] \in X$. It follows that $z < x'$ and $z < x''$, so $z < s$. Then $[z, t]\mathsf{rel}_{\mathsf{E}}[s, t]$, so $[s, t] \in X'$.

(Case 4) $\mathsf{R}_{k+1} = \overline{\mathsf{E}}$. Then there are intervals $[u', w'], [u'', w''] \in X$ such that $[u', w']\mathsf{rel}_{\overline{\mathsf{E}}}[x', y']$ and $[u'', w'']\mathsf{rel}_{\overline{\mathsf{E}}}[x'', y'']$. By the definition of $\mathsf{rel}_{\overline{\mathsf{E}}}$ it follows that $u' > x'$, $w' = y'$, $u'' > x''$, and $w'' = y''$. Let $[z, t]$ be such an interval that $t - z = \min(w' - u', w'' - u'')$, then by the inductive assumption $[z, t] \in X$. Since $u' > x'$ and $u'' > x''$, $t - z < \min(y' - x', y'' - x'')$. Hence $t - z < t - s$, so $s < z$. It follows that $[z, t]\mathsf{rel}_{\overline{\mathsf{E}}}[s, t]$ and so $[s, t] \in X'$.

(Case 5) $\mathsf{R}_{k+1} = \mathsf{A}$. Then there are intervals $[u', w'], [u'', w''] \in X$ such that $[u', w']\mathsf{rel}_{\mathsf{A}}[x', y']$ and $[u'', w'']\mathsf{rel}_{\mathsf{A}}[x'', y'']$. By the definition of $\mathsf{rel}_{\mathsf{A}}$ it follows that $w' = x'$ and $w'' = x''$. Let $z = \min(u', u'')$, then by the inductive assumption we obtain $[z, s] \in X$. Then $[z, s]\mathsf{rel}_{\mathsf{A}}[s, t]$, so $[s, t] \in X'$.

(Case 6) $\mathsf{R}_{k+1} = \overline{\mathsf{A}}$. Then there are intervals $[u', w'], [u'', w''] \in X$ such that $[u', w']\mathsf{rel}_{\overline{\mathsf{A}}}[x', y']$ and $[u'', w'']\mathsf{rel}_{\overline{\mathsf{A}}}[x'', y'']$. By the definition of $\mathsf{rel}_{\overline{\mathsf{A}}}$ it follows that $u' = y'$ and $u'' = y''$. Let $z = \max(w', w'')$, then by the inductive assumption we obtain $[t, z] \in X$. Then $[t, z]\mathsf{rel}_{\overline{\mathsf{A}}}[s, t]$, so $[s, t] \in X'$. ◀

Conditional Simple Temporal Networks with Uncertainty and Decisions*

Matteo Zavatteri

Department of Computer Science, University of Verona, Verona, Italy
`matteo.zavatteri@univr.it`

Abstract

A conditional simple temporal network with uncertainty (CSTNU) is a framework able to model temporal plans subject to both conditional constraints and uncertain durations. The combination of these two characteristics represents the uncontrollable part of the network. That is, before the network starts executing, we do not know completely which time points and constraints will be taken into consideration nor how long the uncertain durations will last. Dynamic controllability (DC) implies the existence of a strategy scheduling the time points of the network in real time depending on how the uncontrollable part behaves. Despite all this, CSTNUs fail to model temporal plans in which a few conditional constraints are under control and may therefore influence (or be influenced by) the uncontrollable part. To bridge this gap, this paper proposes *conditional simple temporal networks with uncertainty and decisions (CSTNUDs)* which introduce *decision time points* into the specification in order to operate on this conditional part under control. We model the dynamic controllability checking (DC-checking) of a CSTNUD as a two-player game in which each player makes his moves in his turn at a specific time instant. We give an encoding into timed game automata for a sound and complete DC-checking. We also synthesize memoryless execution strategies for CSTNUDs proved to be DC. The proposed approach is fully automated.

1998 ACM Subject Classification G.2.2 Graph Theory, F.1.1 Models of Computation, I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases cstnud, dynamic controllability, timed game automata, controller synthesis, advanced temporal planning under uncertainty.

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.23

1 Introduction

Temporal networks are a framework to model temporal plans and check the coherence of their temporal constraints which impose a minimal and maximal temporal distance between the occurrence of the events specified in the plan. Temporal plans mainly divide in plans having everything under control and plans having something out of control. The main components of a temporal network are *time points* and *constraints*. Time points are variables having continuous domain and model the occurrence of events as soon as these variables are assigned real values (i.e., executed). Constraints regulate the minimal and maximal temporal distance between the occurrence of pairs of events and are formalized as linear inequalities.

Whenever both these two components are under control we simply deal with a *consistency* problem asking us to find an assignment of real values to *all* time points satisfying all constraints. Simple temporal networks (STNs) model exactly this case [10], whereas Drake [9]

* The author wants to thank Luca Viganò for his valuable feedback.

addresses temporal plans with choices that are, however, under control; therefore, we keep dealing with a consistency problem asking us to further find suitable values for such choices.

Instead, when some component is out of control, satisfiability is, in general, not enough. In such a case, we deal with a *controllability* problem.

Conditional simple temporal networks (CSTNs) [14, 20] address conditional constraints to enable or disable some parts of the network (i.e., a subset of time points and constraints) during execution. Conditionals are expressed as labels consisting of conjunctions of literals whose atoms are Boolean propositions. The truth value assignments to such propositions are out of control and depend on the behavior of unpredictable external events which are only observed to occur while executing the network.

Simple temporal networks with uncertainty (STNUs) [17, 18] address uncertain (but bounded) durations. Such durations are modeled by contingent links, i.e. pairs of distinct time points specifying a range of allowed values between their distance. One of these time points is called *activation* and it is under control, whereas the other one is called *contingent* and it is not. The real value assignment to the contingent one depends again on the behavior of unpredictable external events which are only observed to occur while executing the network.

Conditional simple temporal networks with uncertainty (CSTNUs) [7, 13] merge the semantics of CSTNs and STNUs addressing conditional constraints and uncertain durations.

Controllability of a temporal network implies the existence of a *strategy* operating on the controllable part such that all constraints will eventually be satisfied. Controllability mainly divides in *weak*, *strong* and *dynamic*. Weak controllability ensures the existence of a (possible different) strategy to operate on the controllable part whenever we are able to predict how the entire uncontrollable part will behave before the execution starts. Strong controllability is the opposite case ensuring the existence of a strategy operating always the same way on the controllable part no matter how the uncontrollable part will behave. However, strong controllability is "too strong". If a temporal network is not strongly controllable, it could still be executable by operating on the controllable part reacting to the uncontrollable one as soon as it becomes known. Dynamic controllability addresses exactly this case.

However, none of the formalisms mentioned so far tackles temporal plans in which some conditional constraints under control may influence (or be influenced by) some uncontrollable part. An initial discussion is given in [3] where CSTNs are extended with decision nodes regulating the truth value assignments to some propositions under control.

We give here the first attempt to address temporal plans in which decisions may influence (or be influenced by) *both* conditional and temporal uncertainty.

Toward this aim our contributions are three-fold. First, we define *conditional simple temporal networks with uncertainty and decisions* (CSTNUDs) as a unified formalism for temporal networks expressing uncontrollable parts and model dynamic controllability as a two-player game in which players make moves in their turns. Second, we provide an encoding into timed game automata for a sound and complete DC-checking and synthesize execution strategies by means of the UPPAAL-TIGA software [2]. Third, we automate our approach by discussing a proof of concept tool we came up with.

The rest of the paper is organized as follows. Section 2 provides essential background on CSTNUs, timed game automata (TGAs) and the DC-checking of CSTNUs via TGAs. Section 3 introduces our main contribution: *CSTNUs with Decisions* along with a new semantics given in *move-based strategies*. Section 4 extends the encoding given in Section 2 to address the DC-checking of CSTNUDs. Section 5 discusses our tool and a preliminary experimental evaluation. Section 6 discusses the correctness and complexity of the encoding. Section 7 discusses related work. Section 8 draws conclusions and discusses future work.

2 Background: CSTNUs, TGAs and Dynamic Controllability

2.1 Conditional Simple Temporal Networks with Uncertainty

Given a set $\mathcal{P}$ of Boolean propositions, a *label* $\ell = \lambda_1 \dots \lambda_n$ is any finite conjunction of literals λ_i, where a literal is either a proposition p (positive literal) or its negation $\neg p$ (negative literal). The *empty label* is denoted by $\boxdot$. The *label universe of* $\mathcal{P}$, denoted by $\mathcal{P}^*$, is the set of all possible (consistent) labels drawn from $\mathcal{P}$; e.g., if $\mathcal{P} = \{p, q\}$, then $\mathcal{P}^* = \{\boxdot, p, q, \neg p, \neg q, p \wedge q, p \wedge \neg q, \neg p \wedge q, \neg p \wedge \neg q\}$. Two labels $\ell_1, \ell_2 \in \mathcal{P}^*$ are *consistent* if and only if their conjunction $\ell_1 \wedge \ell_2$ is satisfiable. A label ℓ_1 *entails* a label ℓ_2 (written $\ell_1 \Rightarrow \ell_2$) if and only if all literals in ℓ_2 appear in ℓ_1 too (i.e., if ℓ_1 is more *specific* than ℓ_2). A label ℓ_1 *falsifies* a label ℓ_2 iff $\ell_1 \wedge \ell_2$ is inconsistent. For instance, if $\ell_1 = p \wedge \neg q$ and $\ell_2 = p$, then ℓ_1 and ℓ_2 are consistent since $p \wedge \neg q \wedge p$ is satisfiable, and ℓ_1 entails ℓ_2 since $p \wedge \neg q \Rightarrow p$.

▶ **Definition 1** (CSTNU). A *Conditional Simple Temporal Network with Uncertainty (CSTNU)* is a tuple $\langle \mathcal{T}, \mathcal{OT}, \mathcal{P}, O, L, \mathcal{L}, \mathcal{C} \rangle$, where:

- $\mathcal{T} = \{X, Y, \dots\}$ is a finite set of *time points* (i.e., variables with continuous domain).
- $\mathcal{OT} \subseteq \mathcal{T} = \{P?, Q?, \dots\}$ is a set of *observation time points*.
- $\mathcal{P} = \{p, q, \dots\}$ is a finite set of Boolean *propositions*.
- $O \colon \mathcal{P} \to \mathcal{OT}$ is a bijection associating a unique $P? \in \mathcal{OT}$ to each $p \in \mathcal{P}$ (i.e., $O(p) = P?$).
- $L \colon \mathcal{T} \to \mathcal{P}^*$ is a function assigning a *label* to each time point $X \in \mathcal{T}$.
- $\mathcal{L}$ is a finite set of *contingent links* (A, x, y, C), where $A, C \in \mathcal{T}$, $0 < x < y < \infty$ $(x, y \in \mathbb{R})$.
- $\mathcal{C}$ is a finite set of *labeled constraints* $(Y - X \leq k, \ell)$, where $X, Y \in \mathcal{T}$, $k \in \mathbb{R} \cup \{\pm\infty\}$ and $\ell \in \mathcal{P}^*$. If $(Y - X \leq k, \ell) \notin \mathcal{C}$ for some $\ell \in \mathcal{P}^*$, then $k = \infty$ (for *that* label).

A CSTNU is *well-defined* if and only if all the following properties hold.

- For each $X \in \mathcal{T}$, if $\lambda \in L(X)$, where $\lambda \in \{p, \neg p\}$, then $L(X) \Rightarrow L(O(p))$ and $(O(p) - X \leq -\epsilon, L(X)) \in \mathcal{C}$ for some $\epsilon > 0$ (*time point label honesty* [14]).
- For any $(A, x, y, C) \in \mathcal{L}$, $A \neq C$ and $L(A) = L(C)$
- For any pair $(A_1, x_1, y_1, C_1), (A_2, x_2, y_2, C_2) \in \mathcal{L}$ if $A_1 \neq A_2$, then $C_1 \neq C_2$.
- For each constraint $(Y - X \leq k, \ell) \in \mathcal{C}$, $\ell \Rightarrow L(Y) \wedge L(X)$ (*constraint label coherence* [14]), and for each literal $\lambda \in \ell$, where $\lambda \in \{p, \neg p\}$, $\ell \Rightarrow L(O(p))$ (*constraint label honesty* [14]).

We execute a time point by assigning it a real value (modeling the occurrence of some temporal event). We execute a CSTNU by executing all relevant non-contingent time points (see below). For any contingent link (A, x, y, C), A is the *activation time point*, whereas C is the *contingent time point*. A is under control, C is not. Once we execute A, we can merely observe the execution of C (by the environment). However, C is guaranteed to occur such that $C - A \in [x, y]$. A contingent link has a unique implicit label given by $\ell = L(A) = L(C)$.

Likewise, an observation time point $P? \in \mathcal{OT}$ is under control, whereas the truth value assignment to its associated Boolean proposition p is not. Once we execute $P?$ we can merely observe such an assignment. Before executing $P?$ the value of p is *unknown*, and after executing $P?$ is either $\top$ (true) or $\bot$ (false). As we execute observation time points, their truth value assignments to the associated propositions generate the *current partial scenario*. That is, a label $\ell_{cps} \in \mathcal{P}^*$ consisting of the conjunction of these literals. Initially, $\ell_{cps} = \boxdot$, and whenever a proposition is assigned a truth value, the resulting literal λ is appended to ℓ_{cps}. Time points and constraints are *relevant* if their labels are not falsified by ℓ_{cps}. Before executing the network all time points and constraints are relevant. If a time point turns irrelevant, we will not execute it. If a constraint does, we will not be obliged to satisfy it.

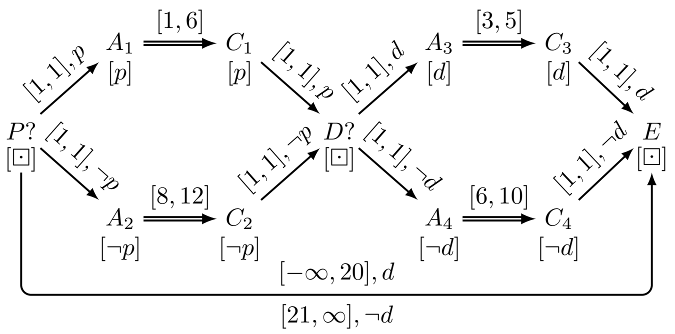


Figure 1 Example of uncontrollable CSTNU.

A CSTNU is said *dynamically controllable* (DC) if there exists a strategy executing all relevant non-contingent time points such that all (relevant) constraints are satisfied no matter which uncertain durations and truth value assignments turn out to be during execution.

We graphically represent a CSTNU as a labeled (multi)graph, where the set of nodes coincides with the set of time points (labels are shown below the nodes), whereas the set of edges divides in *contingent links* and *requirement links*. A contingent link (shown as a double arrow $A \Rightarrow C$ labeled by $[x, y]$) models $(A, x, y, C) \in \mathcal{L}$. A requirement link (shown as a single arrow $X \rightarrow Y$ labeled by $[k_1, k_2], \ell$) models the pair $(Y - X \leq k_2, \ell), (X - Y \leq -k_1, \ell) \in \mathcal{C}$.

Figure 1 shows an example of CSTNU having two observation time points $P?, D?$ and four contingent links $(A_1, 1, 6, C_1)$, $(A_2, 8, 12, C_2)$, $(A_3, 3, 5, C_3)$ and $(A_4, 6, 10, C_4)$. $P?$ is the first time point to execute, whereas E is the last. If $P?$ assigns $\top$ to p, then A_2 and C_2 along with the constraints labeled by $\neg p$ turn irrelevant as $\ell_{cps} = p$ falsifies $\neg p$. If $P?$ assigns $\bot$ to p, we will ignore A_1, C_1 and all constraints labeled by p. Likewise, if $D?$ assigns $\top$ (resp., $\bot$) to d, we will ignore A_4 and C_4 (resp., A_3 and C_3) and all constraints labeled by $\neg d$ (resp., d). The CSTNU in Figure 1 is uncontrollable. For example, assume that each contingent time point (if relevant) takes its maximal duration. If $\ell_{cps} = p \wedge \neg d$, then the execution sequence is $P? = 0$, $A_1 = 1$, $C_1 = 7$, $D? = 8$, $A_4 = 9$, $C_4 = 19$ and $E = 20$ (violating $[21, \infty], \neg d$ between $P?$ and E requiring that E is executed from 21 on.). If $\ell_{cps} = \neg p \wedge d$, then the execution sequence is $P? = 0$, $A_2 = 1$, $C_2 = 13$, $D? = 14$, $A_3 = 15$, $C_3 = 20$ and $E = 21$ (violating $[-\infty, 20], d$ between $P?$ and E requiring that E is executed within 20).

2.2 Timed Game Automata

A *timed automaton* (TA) [1] refines a finite automaton [12] by adding real-valued *clocks* and *clock constraints*. All clocks increase at the uniform rate and may by reset many times.

▶ **Definition 2** (TGA). A *Timed Automaton (TA)* is a tuple $\langle Loc, Act, \mathcal{X}, \rightarrow, Inv \rangle$, where

- Loc is a finite set of *locations* (one is initial). A location is *urgent* if time freezes in it.
- Act is a finite set of *actions* and $\mathcal{X}$ is a finite set of *real-valued clocks*.
- $\rightarrow \subseteq Loc \times \mathcal{H}(\mathcal{X}) \times Act \times 2^{\mathcal{X}} \times Loc$ is the transition relation. An edge (L_i, G, A, R, L_j) represents a transition from L_i to L_j realizing action A. $G \in \mathcal{H}(\mathcal{X})$ is a *guard* consisting of a conjunction of clock constraints having the form $c_1 \sim k$ or $c_1 - c_2 \sim k$ where $c_1, c_2 \in \mathcal{X}$, $k \in \mathbb{N}$ and $\sim \in \{<, \leq, =, >, \geq\}$. $R \subseteq 2^{\mathcal{X}}$ is the set of clocks to reset (i.e., set to 0).
- $Inv : Loc \rightarrow \mathcal{H}(\mathcal{X})$ is a function assigning an *invariant* (modeled as a conjunction of clock constraints) to each location. $Inv(L)$ says *when* the TA is allowed to remain in L.

A *Timed Game Automaton (TGA)* [15] extends a TA by dividing transitions into *controllable* and *uncontrollable*. Uncontrollable transitions have priority over controllable ones.

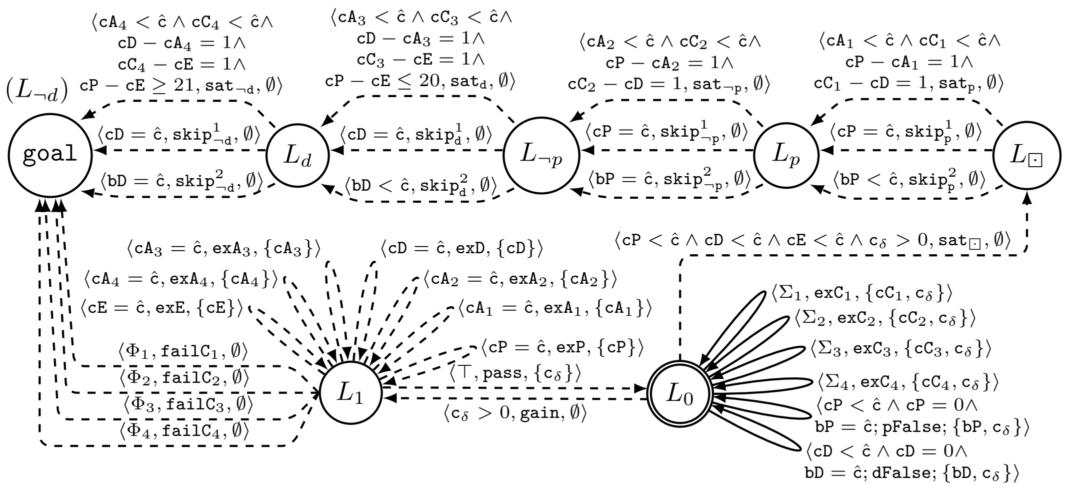

$\langle \mathtt{cA}_4 < \hat{\mathtt{c}} \land \mathtt{cC}_4 < \hat{\mathtt{c}} \land \mathtt{cD} - \mathtt{cA}_4 = 1 \land \mathtt{cC}_4 - \mathtt{cE} = 1 \land \mathtt{cP} - \mathtt{cE} \geq 21, \mathtt{sat}_{\neg d}, \emptyset \rangle$
$\langle \mathtt{cA}_3 < \hat{\mathtt{c}} \land \mathtt{cC}_3 < \hat{\mathtt{c}} \land \mathtt{cD} - \mathtt{cA}_3 = 1 \land \mathtt{cC}_3 - \mathtt{cE} = 1 \land \mathtt{cP} - \mathtt{cE} \leq 20, \mathtt{sat}_{d}, \emptyset \rangle$
$\langle \mathtt{cA}_2 < \hat{\mathtt{c}} \land \mathtt{cC}_2 < \hat{\mathtt{c}} \land \mathtt{cP} - \mathtt{cA}_2 = 1 \land \mathtt{cC}_2 - \mathtt{cD} = 1, \mathtt{sat}_{\neg p}, \emptyset \rangle$
$\langle \mathtt{cA}_1 < \hat{\mathtt{c}} \land \mathtt{cC}_1 < \hat{\mathtt{c}} \land \mathtt{cP} - \mathtt{cA}_1 = 1 \land \mathtt{cC}_1 - \mathtt{cD} = 1, \mathtt{sat}_{p}, \emptyset \rangle$
$(L_{\neg d})$
goal
L_d
$L_{\neg p}$
L_p
$L_{\boxdot}$
$\langle \mathtt{cD} = \hat{\mathtt{c}}, \mathtt{skip}^1_{\neg d}, \emptyset \rangle$
$\langle \mathtt{bD} = \hat{\mathtt{c}}, \mathtt{skip}^2_{\neg d}, \emptyset \rangle$
$\langle \mathtt{cD} = \hat{\mathtt{c}}, \mathtt{skip}^1_{d}, \emptyset \rangle$
$\langle \mathtt{bD} < \hat{\mathtt{c}}, \mathtt{skip}^2_{d}, \emptyset \rangle$
$\langle \mathtt{cP} = \hat{\mathtt{c}}, \mathtt{skip}^1_{\neg p}, \emptyset \rangle$
$\langle \mathtt{bP} = \hat{\mathtt{c}}, \mathtt{skip}^2_{\neg p}, \emptyset \rangle$
$\langle \mathtt{cP} = \hat{\mathtt{c}}, \mathtt{skip}^1_{p}, \emptyset \rangle$
$\langle \mathtt{bP} < \hat{\mathtt{c}}, \mathtt{skip}^2_{p}, \emptyset \rangle$
$\langle \mathtt{cA}_3 = \hat{\mathtt{c}}, \mathtt{exA}_3, \{\mathtt{cA}_3\} \rangle$
$\langle \mathtt{cD} = \hat{\mathtt{c}}, \mathtt{exD}, \{\mathtt{cD}\} \rangle$
$\langle \mathtt{cP} < \hat{\mathtt{c}} \land \mathtt{cD} < \hat{\mathtt{c}} \land \mathtt{cE} < \hat{\mathtt{c}} \land \mathtt{c}_\delta > 0, \mathtt{sat}_{\boxdot}, \emptyset \rangle$
$\langle \mathtt{cA}_4 = \hat{\mathtt{c}}, \mathtt{exA}_4, \{\mathtt{cA}_4\} \rangle$
$\langle \mathtt{cA}_2 = \hat{\mathtt{c}}, \mathtt{exA}_2, \{\mathtt{cA}_2\} \rangle$
$\langle \mathtt{cE} = \hat{\mathtt{c}}, \mathtt{exE}, \{\mathtt{cE}\} \rangle$
$\langle \mathtt{cA}_1 = \hat{\mathtt{c}}, \mathtt{exA}_1, \{\mathtt{cA}_1\} \rangle$
$\langle \Phi_1, \mathtt{failC}_1, \emptyset \rangle$
$\langle \Phi_2, \mathtt{failC}_2, \emptyset \rangle$
$\langle \Phi_3, \mathtt{failC}_3, \emptyset \rangle$
$\langle \Phi_4, \mathtt{failC}_4, \emptyset \rangle$
$\langle \mathtt{cP} = \hat{\mathtt{c}}, \mathtt{exP}, \{\mathtt{cP}\} \rangle$
$\langle \top, \mathtt{pass}, \{\mathtt{c}_\delta\} \rangle$
$\langle \mathtt{c}_\delta > 0, \mathtt{gain}, \emptyset \rangle$
L_1
L_0
$\langle \Sigma_1, \mathtt{exC}_1, \{\mathtt{cC}_1, \mathtt{c}_\delta\} \rangle$
$\langle \Sigma_2, \mathtt{exC}_2, \{\mathtt{cC}_2, \mathtt{c}_\delta\} \rangle$
$\langle \Sigma_3, \mathtt{exC}_3, \{\mathtt{cC}_3, \mathtt{c}_\delta\} \rangle$
$\langle \Sigma_4, \mathtt{exC}_4, \{\mathtt{cC}_4, \mathtt{c}_\delta\} \rangle$
$\langle \mathtt{cP} < \hat{\mathtt{c}} \land \mathtt{cP} = 0 \land \mathtt{bP} = \hat{\mathtt{c}}; \mathtt{pFalse}; \{\mathtt{bP}, \mathtt{c}_\delta\} \rangle$
$\langle \mathtt{cD} < \hat{\mathtt{c}} \land \mathtt{cD} = 0 \land \mathtt{bD} = \hat{\mathtt{c}}; \mathtt{dFalse}; \{\mathtt{bD}, \mathtt{c}_\delta\} \rangle$

Figure 2 TGA encoding the CSTNU in Fig. 1: L_0 is the initial location, L_1, $L_{\boxdot}$, L_p, $L_{\neg p}$, L_d, `goal` are urgent. Solid (resp., dashed) edges model controllable (resp., uncontrollable) transitions. $\Sigma_1 : \mathtt{cA}_1 < \hat{\mathtt{c}} \land \mathtt{cC}_1 = \hat{\mathtt{c}} \land \mathtt{cA}_1 \geq 1 \land \mathtt{cA}_1 \leq 6$ and $\Phi_1 : \mathtt{cA}_1 < \hat{\mathtt{c}} \land \mathtt{cC}_1 = \hat{\mathtt{c}} \land \mathtt{cA}_1 > 6$. $\Sigma_2 : \mathtt{cA}_2 < \hat{\mathtt{c}} \land \mathtt{cC}_2 = \hat{\mathtt{c}} \land \mathtt{cA}_2 \geq 8 \land \mathtt{cA}_2 \leq 12$ and $\Phi_2 : \mathtt{cA}_2 < \hat{\mathtt{c}} \land \mathtt{cC}_2 = \hat{\mathtt{c}} \land \mathtt{cA}_2 > 12$. $\Sigma_3 : \mathtt{cA}_3 < \hat{\mathtt{c}} \land \mathtt{cC}_3 = \hat{\mathtt{c}} \land \mathtt{cA}_3 \geq 3 \land \mathtt{cA}_3 \leq 5$ and $\Phi_3 : \mathtt{cA}_3 < \hat{\mathtt{c}} \land \mathtt{cC}_3 = \hat{\mathtt{c}} \land \mathtt{cA}_3 > 5$. $\Sigma_4 : \mathtt{cA}_4 < \hat{\mathtt{c}} \land \mathtt{cC}_4 = \hat{\mathtt{c}} \land \mathtt{cA}_4 \geq 6 \land \mathtt{cA}_4 \leq 10$ and $\Phi_4 : \mathtt{cA}_4 < \hat{\mathtt{c}} \land \mathtt{cC}_4 = \hat{\mathtt{c}} \land \mathtt{cA}_4 \land \mathtt{cA}_4 > 10$.

We graphically represent a TGA as a (multi)graph where the set of nodes coincides with that of locations whereas the set of edges models controllable transitions (solid edges) and uncontrollable ones (dashed edges). Figure 2 depicts a TGA encoding the CSTNU in Figure 1. In what follows we sum up how this encoding is achieved and dynamic controllability checked.

2.3 Dynamic Controllability

The DC-checking problem is the problem of deciding if a CSTNU is DC. We can answer the DC-checking problem by using *sound* and *complete* TGA reachability algorithms [4, 5]. We model the DC-checking as a two-player game between a controller (`ctrl`) and the environment (`env`). The aim of `ctrl` is to reach a specific location as soon as all relevant time points have been executed and all constraints are satisfied, whereas `env`'s goal is to prevent `ctrl` from doing that. If `ctrl` wins, the network is DC, otherwise it is not. An important aspect of this encoding is that `ctrl` is assigned *uncontrollable* transitions, whereas `env` is assigned *controllable* ones. This is necessary to allow `env`'s instantaneous reactions as in the TGA semantics, uncontrollable transitions go first [4, 5, 6]. The encoding is as follows.

Clocks. $\mathcal{X}$ contains a clock $\mathtt{cX}$ for each time point $X \in \mathcal{T}$ and a clock $\mathtt{bP}$ for each proposition $p \in \mathcal{P}$. $\mathcal{X}$ also contains two special clocks $\hat{\mathtt{c}}$ (modeling the global time) and $\mathtt{c}_\delta$ (regulating the interplay of the game). $\mathtt{cX} = \hat{\mathtt{c}}$, means that X has not been executed, whereas $\mathtt{cX} < \hat{\mathtt{c}}$ means that X was executed at time $\hat{\mathtt{c}} - \mathtt{cX}$ (when this difference is > 0). Likewise, $\mathtt{bP} = \hat{\mathtt{c}}$ means that $p = \top$, whereas $\mathtt{bP} < \hat{\mathtt{c}}$ means that $p = \bot$ (both when $\mathtt{cP} < \hat{\mathtt{c}}$). Each $\mathtt{cX}$ and $\mathtt{bP}$ may be reset at most once. For our example we have $\mathcal{X} = \{\hat{\mathtt{c}}, \mathtt{c}_\delta, \mathtt{cP}, \mathtt{cA}_1, \mathtt{cC}_1, \mathtt{cA}_2, \mathtt{cC}_2, \mathtt{cD}, \mathtt{cA}_3, \mathtt{cC}_3, \mathtt{cA}_4, \mathtt{cC}_4, \mathtt{cE}, \mathtt{bP}, \mathtt{bD}\}$.

Locations. *Loc* contains three core locations L_0 (initial), L_1 (urgent) and `goal` (urgent), and $n-1$ urgent locations $L_{\ell_1}, \ldots, L_{\ell_{n-1}}$ where n is the number of distinct labels in the CSTNU. That is, $n = |\{L(X) \mid X \in \mathcal{T}\} \cup \{\ell \mid (Y - X \leq k, \ell) \in \mathcal{C}\}|$. For our example, $Loc = \{L_0, L_1, L_{\boxdot}, L_p, L_{\neg p}, L_d, \mathtt{goal}(= L_{\neg d}\}$ as the distinct labels are $\{\boxdot, p, \neg p, d, \neg d\}$.

Transitions. $\rightarrow$ contains controllable and uncontrollable transitions to model the following:

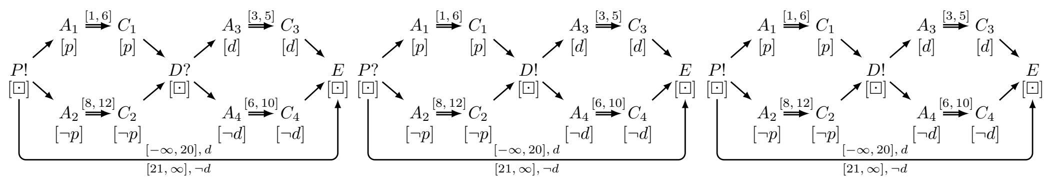
- *Game interplay.* `pass` and `gain` are uncontrollable transitions regulating the game interplay. In particular `gain` can be taken only when $\mathtt{c}_\delta > 0$ modeling the *reaction time* needed to observe how the uncontrollable part behaves.
- *Non-contingent time point executions.* For each non-contingent time point X there is an uncontrollable self-loop transition $\langle L_1, \mathtt{cX} = \hat{\mathtt{c}}, \mathtt{exX}, \{\mathtt{cX}\}, L_1\rangle$ modeling the execution of X. The guard says that X has not been executed yet, while the reset fixes the execution time of X to $\hat{\mathtt{c}} - \mathtt{cX}$ by resetting $\mathtt{cX}$.
- *Contingent time point executions.* For each contingent link $(A, x, y, C) \in \mathcal{L}$ there is a controllable self-loop transition $\langle L_0, \mathtt{cA} < \hat{\mathtt{c}} \wedge \mathtt{cC} = \hat{\mathtt{c}} \wedge \mathtt{cA} \geq x \wedge \mathtt{cA} \leq y, \mathtt{exC}, \{\mathtt{cC}, \mathtt{c}_\delta\}, L_0\rangle$ to allow `env` to execute the contingent time point C such that $C - A \in [x, y]$, and a fail transition $\langle L_0, \mathtt{cA} < \hat{\mathtt{c}} \wedge \mathtt{cC} = \hat{\mathtt{c}} \wedge \mathtt{cA} > y, \mathtt{failC}, \{\mathtt{cC}, \mathtt{c}_\delta\}, \mathtt{goal}\rangle$ to allow `ctrl` to move to `goal` if `env` fails or refuses to take the transition.
- *Truth value assignments.* For each proposition $p \in \mathcal{P}$ there is a controllable self-loop transition $\langle L_0, \mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{cP} = 0 \wedge \mathtt{bP} = \hat{\mathtt{c}}, \mathtt{pFalse}, \{\mathtt{bP}, \mathtt{c}_\delta\}, L_0\rangle$ to allow `env` to assign $\perp$ to p, if it decides so. If it does not, the truth value of p will remain forever $\top$.
- *Winning conditions.* To check that all relevant time points have been executed and all constraints are satisfied we connect each pair of locations $(L_{\ell_{i-1}}, L_{\ell_i})$ in the winning path $L_0 \rightarrow L_\boxdot \rightarrow \cdots \rightarrow L_{\ell_{n-1}} \rightarrow \mathtt{goal}$ by means of a set of uncontrollable transitions. Each set of transitions going from $L_{\ell_{i-1}}$ to L_{ℓ_i} verifies that if ℓ_{cps} does not falsify ℓ_i, then all time points labeled by ℓ_i must have been executed and all constraints labeled by ℓ_i are satisfied. If ℓ_{cps} falsifies ℓ_i, a `skip` transition allows us to ignore this check. In this way, the problem is decomposed with respect to the specific labels avoiding the combinatorial explosion of all arising cases. For example, the set of transitions going from $L_\boxdot$ to L_p is generated as follows. In the scenario where $P?$ has been executed and p assigned $\top$ (i.e., $\ell_{cps} = p$), then A_1 and C_1 must have been executed, and $A_1 - P? \leq 1$, $P? - A_1 \leq -1$, $D? - C_1 \leq 1$, $C_1 - D? \leq -1$ are satisfied. In other words, the meta conditional constraint $(\mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{bP} = \hat{\mathtt{c}}) \implies (\mathtt{cA}_1 < \hat{\mathtt{c}} \wedge \mathtt{cC}_1 < \hat{\mathtt{c}} \wedge \mathtt{cP} - \mathtt{cA}_1 = 1 \wedge \mathtt{cC}_1 - \mathtt{cD} = 1)$ refines to $\neg(\mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{bP} = \hat{\mathtt{c}}) \vee (\mathtt{cA}_1 < \hat{\mathtt{c}} \wedge \mathtt{cC}_1 < \hat{\mathtt{c}} \wedge \mathtt{cP} - \mathtt{cA}_1 = 1 \wedge \mathtt{cC}_1 - \mathtt{cD} \geq 1 = 1)$ simplifying to $(\mathtt{cP} = \hat{\mathtt{c}}) \vee (\mathtt{bP} < \hat{\mathtt{c}}) \vee (\mathtt{cA}_1 < \hat{\mathtt{c}} \wedge \mathtt{cC}_1 < \hat{\mathtt{c}} \wedge \mathtt{cP} - \mathtt{cA}_1 = 1 \wedge \mathtt{cC}_1 - \mathtt{cD} = 1)$ since TGAs do not allow negations or disjunctions of clock constraints in the guards. Finally, we generate a transition[1] for each disjunct ($\mathtt{sat}_\mathtt{p}$,$\mathtt{skip}^1_\mathtt{p}$,$\mathtt{skip}^2_\mathtt{p}$).

DC-checking is done by looking for a control strategy for `env` to always prevent `ctrl` from getting to `goal`. If such a strategy exists, the initial CSTNU is not DC, otherwise it is (as `ctrl` has a counter-strategy to react to any combination of `env`'s moves).

3 CSTNUs with Decisions

In this section we extend CSTNUs by injecting a new kind of time point: the *decision time point.* A decision time point $D!$ dualizes an observation one $P?$ as the truth value assignment to the associated proposition is *under control.* As a result, the controllable and uncontrollable part may now mutually influence one another. That is, deciding some truth value may restrict (or even exclude) some uncontrollable part and vice versa. Several interesting cases may arise depending on if a few truth values are decided before or after having full information on how

[1] We model $Y - X \leq k$ as $(\hat{\mathtt{c}} - \mathtt{cY}) - (\hat{\mathtt{c}} - \mathtt{cX}) \leq k$ simplifying to $\mathtt{cX} - \mathtt{cY} \leq k$. We might write $\mathtt{cX} - \mathtt{cY} \geq k$ as a short for $X - Y \leq -k$ and $\mathtt{cX} - \mathtt{cY} = k$ as a short for the pair $Y - X \leq k$ and $X - Y \leq -k$.



(a) *A decision before any uncontrollable part.* **(b)** *A decision after all observation and some contingent.* **(c)** *A decision after another decision and a contingent.*

Figure 3 Possible cases of the CSTNU in Figure 1 when substituting decision time points for observation ones. Missing labels on requirement links $X \to Y$ are all $[1,1], L(X) \wedge L(Y)$ (Figure 1).

the uncontrollable part will or have behaved. We go ahead with this discussion by taking Figure 3 as an example. There, we took the initial CSTNU in Figure 1 and substituted decision time points for observation ones in all possible combinations. We discuss these examples focusing on the combinations of minimal and maximal durations of contingent links only. If it works for them, then it must work for any other combination of durations.

- In Figure 3a $P!$ is a decision time point. The resulting CSTNUD is uncontrollable. If we *decide* p (i.e., assign $\top$ to p), then *observe* $\neg d$ (i.e., $D?$ assigns $\bot$ to d) and C_1, C_4 take their maximal durations, then we will have to execute E at 20 violating $(P? - E \leq -21, \neg d)$ as $P?$ is executed at 0. Conversely, if we decide $\neg p$, then observe d and C_2 and C_3 take their maximal durations, then we will have to execute E at 21 (violating $E - P? \leq 20, d$).
- In Figure 3b $D!$ is a decision time point. The resulting CSTNUD is DC. Assume that we observe p. Regardless on what duration C_1 takes, we can only decide d. Indeed, if we decided $\neg d$, regardless of the duration of C_4 we would have to execute E before time 21 violating $(P? - E \leq -21, \neg d)$. Assume now that we observe $\neg p$. If C_2 takes its minimal duration, d is the only good decision. If we decided $\neg d$ and then C_4 took its minimal duration, we would execute E at 18 violating $(P? - E \leq -21, \neg d)$. On the contrary, if C_2 takes its maximal duration then we can only decide $\neg d$. If we decided d and C_3 took its maximal duration, we would have to execute E at 21 violating $(E - P? \leq 20, d)$.
- In Figure 3c $P!$ and $D!$ are both decision time points. The resulting CSTNUD is of course[2] dynamically controllable. If we decide p, then deciding d is always going to be fine. If we decide $\neg p$, then we will decide either d or $\neg d$ depending on how long C_2 lasts. If C_2 takes its minimal duration, then we will decide d (but not $\neg d$ since C_4 could then take its minimal duration). If C_2 takes its maximal duration, then we will decide $\neg d$ (but not d since if C_3 could then take its maximal duration).

  Hence, *decisions are dynamic*.

▶ **Definition 3** (CSTNUD). A *Conditional Simple Temporal Network with Uncertainty and Decisions* (*CSTNUD*) is a tuple $\langle \mathcal{T}, \mathcal{OT}, \mathcal{DT}, \mathcal{P}, O, L, \mathcal{L}, \mathcal{C} \rangle$, where:

- $\mathcal{T}, \mathcal{OT}, \mathcal{P}, L, \mathcal{L}, \mathcal{C}$ are exactly the same of those given for CSTNUs. Furthermore, we denote the set of contingent time points as *Contingent* $= \{C \mid (A, x, y, C) \in \mathcal{L}\}$.
- $\mathcal{DT} \subseteq \mathcal{T} = \{D!, E!, \dots\}$ is a set of *decision time points* such that $\mathcal{OT} \cap \mathcal{DT} = \emptyset$.
- $O \colon \mathcal{P} \to \mathcal{DT} \cup \mathcal{OT}$ is a bijection associating a unique observation or decision time point to each proposition. If $O(p) \in \mathcal{OT}$, then p is called *observable*, whereas if $O(d) \in \mathcal{DT}$,

[2] If a network is DC (e.g., Figure 3b), then turning controllable some uncontrollable part (e.g., Figure 3c) *cannot worsen* the situation turning the network uncontrollable. The vice versa does not hold.

then d is called *decidable*. $\mathcal{OP} \subseteq \mathcal{P} = \{p \mid O(p) \in \mathcal{OT}\}$ and $\mathcal{DP} \subseteq \mathcal{P} = \{d \mid O(d) \in \mathcal{DT}\}$ shorten the sets of all observable and decidable propositions, where $\mathcal{OP} \cap \mathcal{DP} = \emptyset$.

A CSTNUD is *well-defined* if and only if the underlying CSTNU is well-defined and time point label honesty extends to decidable propositions as follows: For each $X \in \mathcal{T}$, if $\lambda \in L(X)$, where $\lambda = \{d, \neg d\}$ and $d \in \mathcal{DP}$, then $L(X) \Rightarrow L(O(d))$ and $(O(d) - X \leq 0, L(X)) \in \mathcal{C})$. That is, X can be executed at the same time of $D!$ (but instantaneously after $D!$ since time points executed at the same instant must in general follow an *order of execution*).

We model the execution semantics of a CSTNUD as a two-player game in which `Player1` models the controller and `Player2` models the environment. We employ *execution sequences* [16] to model the state of the game and define players' strategies as mappings from execution sequences considered at specific time instants to *moves*.

A *sequence* $\{x_1, x_2, \ldots, x_n\}$ is a totally ordered collection of elements such that for any pair of elements x_i, x_j, if $i < j$ (resp., $i > j$), then it means that x_i is before (resp., after) x_j. We abuse notation and write $\{x_1, x_2, \ldots, x_n\} \cup \{x_p\}$ to mean the appending operation resulting in $\{x_1, x_2, \ldots, x_n, x_p\}$ where $n < p$. We write $x_i \in \{x_1, x_2, \ldots, x_n\}$ iff there exists $j \in \mathbb{N}$, $1 \leq j \leq n$ such that $x_i = x_j$ (membership), and $|\{x_1, x_2, \ldots, x_n\}| = n$ (cardinality). A *partial schedule* for a subset of time points $\mathcal{T}' \subseteq \mathcal{T}$ is a mapping $S_{\mathcal{T}'} \colon \mathcal{T}' \to \mathbb{R}$ assigning a real value to each $X \in \mathcal{T}'$. A *partial schedule* for a subset of Boolean propositions $\mathcal{P}' \subseteq \mathcal{P}$ is a mapping $S_{\mathcal{P}'} \colon \mathcal{P}' \to \{\top, \bot\}$ assigning either $\top$ or $\bot$ to each $p \in \mathcal{P}'$. We write $\mathtt{b}$ for a generic Boolean value (i.e., $\mathtt{b} \in \{\top, \bot\}$). We write $S_{\mathcal{T}'} \cup \{S_{\mathcal{T}'}(Y) = k\}$ to shorten that the domain of $S_{\mathcal{T}'}$ extends by adding time point Y such that $S_{\mathcal{T}'}(Y) = k$. Similarly, we write $S_{\mathcal{P}'} \cup \{S_{\mathcal{P}'}(p) = \mathtt{b}\}$ for Boolean propositions.

▶ **Definition 4** (Instantiation sequence). An *instantiation sequence* is a quadruple $\langle E, K, S_E, S_K \rangle$, where E is a finite sequence of distinct time points in $\mathcal{T}$, K is a finite sequence of distinct propositions in $\mathcal{P}$, and S_E, S_K are partial schedules whose domains are E and K, respectively.

▶ **Definition 5** (Execution sequence). An *execution sequence* $Z = \langle E, K, S_E, S_K \rangle$ is an instantiation sequence satisfying the following properties:

S_E Monotonicity For any pair $X_i, X_j \in E$ if $i < j$, then $S_E(X_i) \leq S_E(X_j)$.

(Time Point Label) Honesty For each $X \in E$ and each literal $\lambda \in L(X)$ where $\lambda \in \{p, \neg p\}$, then $O(p) \in E$ and $O(p)$ is before X, $p \in K$, $S_K(p) = \top$ (if $\lambda = p$) and $S_K(p) = \bot$ (if $\lambda = \neg p$). Also, $S_E(O(p)) < S_E(X)$ (if $p \in \mathcal{OP}$) and $S_E(O(p)) \leq S_E(X)$ (if $p \in \mathcal{DP}$).

Z^* represents the set of all execution sequences. $t_{last}(Z) = \max\{S_E(X) \mid X \in E\}$ represents the last time instant in which a time point was executed in Z. $last(Z) = \{X \mid X \in E \wedge S_E(X) = t_{last}\}$ represents the set of the last executed time points.

Therefore, an execution sequence models the ordered sequence of executed time points and assigned propositions according to the well-definedness of a CSTNUD. As an example, consider again Figure 3b. Assume that we execute $P?$ at 0 and observe $\neg p$. Assume then that we execute A_2 at 1 and observe C_2 to occur at 13 (i.e., at its maximal duration). The execution sequence is $Z = \langle \{P?, A_2, C_2\}, \{p\}, \{S_E(P?) = 0, S_E(A_2) = 1, S_E(C_2) = 13\}, \{S_K(p) = \bot\} \rangle$. We can now compute the current partial scenario as the conjunction of all positive and negative literals arising from all propositions in K according to S_K and define local consistency.

▶ **Definition 6** (Current partial scenario). Given any $Z = \langle E, K, S_E, S_K \rangle$, the *current partial scenario* is given by $\ell_{cps} = \lambda_1 \wedge \cdots \wedge \lambda_k$, where for each $p_i \in K$, $\lambda_i = p_i$ (if $S_K(p_i) = \top$) and $\lambda_i = \neg p_i$ (if $S_K(p_i) = \bot$).

For Z we have that $\ell_{cps} = \neg p$ since $p \in K$ and $S_K(p) = \bot$.

▶ **Definition 7** (Local consistency)**.** An execution sequence $E = \langle E, K, S_E, S_K \rangle$, is *locally consistent* if and only if for each $(Y - X \leq k, \ell) \in \mathcal{C}$ where $X, Y \in E$ and $\ell_{cps} \Rightarrow \ell$, $S_E(Y) - S_E(X) \leq k$ holds.

Z is locally consistent since the schedule S_E satisfies $(A_2 - P? \leq 1, \neg p)$ and $(P? - A_2 \leq -1, \neg p)$. An execution sequence evolves over time according to the evolution of the game that `Player1` (the controller) plays against `Player2` (the environment). Each player follows a strategy saying what moves to make and when. Moreover, many moves can be made at the same time instant (provided that they respect an order) and sometimes moves are mandatory.

▶ **Definition 8** (Move)**.** A *move m is either X meaning "execute time point X" or* $(p, \mathtt{b})$ *meaning "assign $\mathtt{b} \in \{\top, \bot\}$ to proposition p".* A move for `Player1` requires that X is a *non-contingent* time point and p is a *decidable* proposition. A move for `Player2` requires that X is a *contingent* time point and p is an *observable* proposition. M_1^* and M_2^* represent the sets of all moves for `Player1` and `Player2`, respectively.

A *move-based strategy* is a mapping from execution sequences considered at particular time instants to moves augmented with a `wait` condition modeling the absence of move. A strategy tells a player to make a move at a particular time instant only if the move is applicable at that particular time. Therefore, a strategy specifies *applicability conditions* saying when a move *can* be made, *obligations* saying when a move *has to* be made and *postconditions* saying how the execution sequence evolves accordingly.

▶ **Definition 9** (Move-based strategy)**.** A *move-based strategy* for `Player1` is a mapping $\sigma_1 \colon Z^* \times \mathbb{R} \to M_1^* \cup \{\mathtt{wait}\}$ such that its *applicability conditions are*:

1. For any execution sequence Z and any time instant t, $\sigma_1(Z, t)$ is *applicable* iff $t \sim t_{last}(Z)$, where $\sim$ is $>$ if $last(Z)$ contains a contingent time point C or an observation time point $P?$ such that K contains its related proposition p (reaction time enforcement), $\geq$ otherwise.
2. For any execution sequence Z and any time instant t, $\sigma_1(Z, t) = X$ is applicable if (1) holds and X is an unexecuted non-contingent time point such that the current partial scenario entails $L(X)$ (i.e., $X \notin E \wedge X \notin \mathit{Contingent} \wedge \ell_{cps} \Rightarrow L(X)$).
3. For any execution sequence Z and any time instant t, $\sigma_1(Z, t) = \mathtt{wait}$ is applicable if (1) holds and there is no obligation at time t.

The unique *obligation* involves decidable propositions requiring that whenever a decision time point $D!$ has been executed and its related proposition d has not been assigned yet, then the strategy must issue a move to assign d a truth value instantaneously. In symbols: $D! \in E \wedge d \notin K \implies \sigma_1(Z, S_E(D!)) = (d, \mathtt{b})$.

A *move-based strategy* for `Player2` is a mapping $\sigma_2 \colon Z^* \times \mathbb{R} \to M_2^* \cup \{\mathtt{wait}\}$ such that its *applicability conditions* are:

1. For any execution sequence Z and any time instant t, $\sigma_2(Z, t)$ is *applicable* iff $t \geq t_{last}(Z)$.
2. For any execution sequence Z, any time instant t and any contingent link $(A, x, y, C) \in \mathcal{L}$, $\sigma_2(Z, t) = C$ is applicable if (1) holds, A has already been executed, C has not, and executing C at this time satisfies $C - A \in [x, y]$ (i.e., $A \in E \wedge C \in \mathit{Contingent} \wedge C \notin E \wedge t - S_E(A) \in [x, y]$).
3. For any execution sequence Z and any time instant t, $\sigma_2(Z, t) = \mathtt{wait}$ is applicable if (1) holds and there is no obligation at time t.

Obligations are of two kinds. The first obligation involves observable propositions requiring that whenever an observation time point $P?$ has been executed and its related proposition p has not been assigned yet, then the strategy must issue a move to assign p a truth value instantaneously. In symbols: $(P? \in E \wedge p \notin K) \implies \sigma_2(Z, S_E(P?)) = (p, \mathtt{b})$. The second obligation involves contingent links (A, x, y, C) requiring that if A has already been executed, C has not and the current time t is the last instant in which C can be executed, then the strategy must issue a move to execute C at t. In symbols: $\forall (A, x, y, C) \in \mathcal{L}, \forall t \in \mathbb{R}, (A \in E \wedge C \notin E \wedge t - S_E(A) = y) \implies \sigma_2(Z, t) = C$.

Postconditions for both σ_1 and σ_2 are the same. If the strategy tells the player to execute a time point X at time t then Z updates by appending X to E and extending S_E such that $S_E(X) = t$. If the strategy tells the player to assign the truth value $\mathtt{b}$ to the proposition p, then Z updates by appending p to K and extending S_K such that $S_K(p) = \mathtt{b}$. In symbols:

- If $\sigma_i(Z, t) = X$, then $\mathit{Post}(Z, \sigma_i, t) = \langle E \cup \{X\}, K, S_E \cup \{S_E(X) = t\}, S_K\rangle$.
- If $\sigma_i(Z, t) = (p, \mathtt{b})$, then $\mathit{Post}(Z, \sigma_i, t) = \langle E, K \cup \{p\}, S_E, S_K \cup \{S_K(p) = \mathtt{b}\}\rangle$.

Getting back to our example we have that $t_{last}(Z) = 13$ and $\mathit{last}(Z) = \{C_2\}$. Suppose that current time is $t = 14$. $\sigma_1(Z, 14) = D!$ is applicable since $t > t_{last}$ and $D!$ has not been executed yet, whereas $\sigma_1(Z, 14) = (d, \top)$ is not since $D! \notin E$. If $\sigma_1(Z, 14) = D!$ is taken into consideration (i.e., $Z' = \mathit{Post}(Z, \sigma_1, t)$), then $\sigma_1(Z', 14) = (d, \top)$ instantaneously after.

We now model `Player2` as the *most powerful player possible*. If `Player1` can beat this (worst-case of) environment, then `Player1` must be able to beat any other less powerful environment playing the same game. To achieve this purpose we model the game in *turns*. That is, at any time instant t, there exist two turns: $T_1(t)$ (occurring first) and $T_2(t)$ (occurring last). `Player1` makes his moves in $T_1(t)$, whereas `Player2` makes his in $T_2(t)$. If player i does not make any move in $T_i(t)$, then he loses forever the possibility to play at time t. As a result, `Player2`, making his moves in $T_2(t)$, is guaranteed to always have full information on what `Player1` has done in $T_1(t)$ (worst-case scenario). In what remains of this section we define the concept of *snapshot* modeling an execution sequence a particular time instant t (after the players are done in $T_1(t)$ and $T_2(t)$), *continuous game evolution* modeling how the execution sequence evolves and *winning conditions* for each player.

▶ **Definition 10** (Snapshot). Let $Z = \langle E, K, S_E, S_K\rangle$ be any execution sequence. $Z(t) = \langle E', K', S'_E, S'_K\rangle$ models the *snapshot* of Z at time t, where $E' = \{X \mid X \in E \wedge S_E(X) \le t\}$, $K' = \{p \mid p \in K \wedge O(p) \in E'\}$, $\forall X \in E', S'_E(X) = S_E(X)$, and $\forall p \in K', S'_K(p) = S_K(p)$.

To give an example, let us get back to the execution sequence we have discussed before. At $t = 11$, we have $Z(11) = \langle \{P?, A_2\}, \{p\}, \{S_E(P?) = 0, S_E(A_2) = 1\}, \{S_K(p) = \bot\}\rangle$.

▶ **Definition 11** (Continuous game evolution). Let $t \in \mathbb{R}^{\ge 0}$ be the global time. The continuous game evolution is modeled by an infinite sequence of snapshots $Z(t)$ defined as:

$$Z(t) = \begin{cases} T_2(T_1(\langle \emptyset, \emptyset, \emptyset, \emptyset\rangle, t), t) & \text{if } t = 0 \\ T_2(T_1(Z(t - \epsilon), t), t) & \text{if } t > 0 \end{cases} \quad T_i(Z, t) = \begin{cases} Z & \text{if } \sigma_i(Z, t) = \mathtt{wait} \\ T_i(\mathit{Post}(Z, \sigma_i, t), t) & \text{otherwise} \end{cases}$$

where $T_i(t)$ models the evolution of Z during turn i at time t according to σ_i, whereas $\epsilon > 0$.

▶ **Definition 12** (Winning conditions). `Player1` wins the game if and only if the game evolution leads to a snapshot $Z(t)$ such that for each unexecuted time point X, ℓ_{cps} falsifies $L(X)$ and for each constraints $(Y - X \le k, \ell)$ where $X, Y \in E$ and $\ell_{cps} \Rightarrow \ell$, $S_E(Y) - S_E(X) \le k$ holds. `Player2` wins otherwise. σ_i is a *winning strategy* if player i wins the game by following σ_i.

▶ **Definition 13** (Dynamic controllability). A CSTNUD is dynamically controllable if `Player1` has a winning strategy such that for any $t > 0$ and any pair of execution sequences Z_1, Z_2, if $\sigma_2(Z_1, t') = \sigma_2(Z_2, t')$ for $0 \le t' < t$, then $\sigma_1(Z_1, t) = \sigma_1(Z_2, t)$.

In other words, whenever `Player2` has made the same (infinite) sequence of moves up to time $t - \epsilon$, then `Player1` will make the same move(s) at time t.

4 Dynamic Controllability of CSTNUDs via TGAs

In this section we extend the encoding given for CSTNUs in Section 2. As an example, we consider Figure 4 depicting the encoding of the CSTNUD in Figure 3b.

Once again, we have three core locations but this time we borrow a few names from Section 3 and rename them to $\texttt{T}_1$ (ex L_1), $\texttt{T}_2$ (ex L_0) and $\texttt{win}$ (ex $\texttt{goal}$). $\texttt{T}_1$ and $\texttt{T}_2$ model the two turns $T_1(t)$ and $T_2(t)$ when global time is > 0. $\texttt{T}_2$ is the initial location. The winning path is computed the same way only renaming each L_{ℓ_i} to $\texttt{w}_{\ell_i}$. $\texttt{gain}$ and $\texttt{pass}$ regulate the turns at any time instant t. We still have a clock $\texttt{cX}$ for each $X \in \mathcal{T}$ (considering decision time points too) and a clock $\texttt{bP}$ for each $p \in \mathcal{P}$ (considering decidable propositions too).

We optimize the guard of each uncontrollable self-loop at $\texttt{T}_1$ by exploiting what we know of the CSTNUD. That is, we extend the guards so that they enforce time point label honesty as well as the partial order among the time points when not ambiguous. This optimization was first discussed in [8] but there it dealt with disjunctive constraints and exploited internal data structures provided by the UPPAAL-TIGA software. Here, we propose a more formal definition avoiding such data structures. Moreover, [8] does not address decisions.

To give an example of this optimization, consider time points A_1 and A_4 of the CSTNUD in Figure 3b. $L(A_1) = p$ and $L(A_4) = \neg d$. Recall that the encoding models p and d as two dedicated clocks $\texttt{bP}$ and $\texttt{bD}$ such that if each of these clocks is equal to (resp., less than) $\hat{\texttt{c}}$, once its related observation or decision time point has been executed, then the related proposition is $\top$ (resp., $\bot$). Moreover, time point label honesty also requires that $P? - A_1 \leq -\epsilon$ (observation) and $D! - A_4 \leq 0$ (decision).

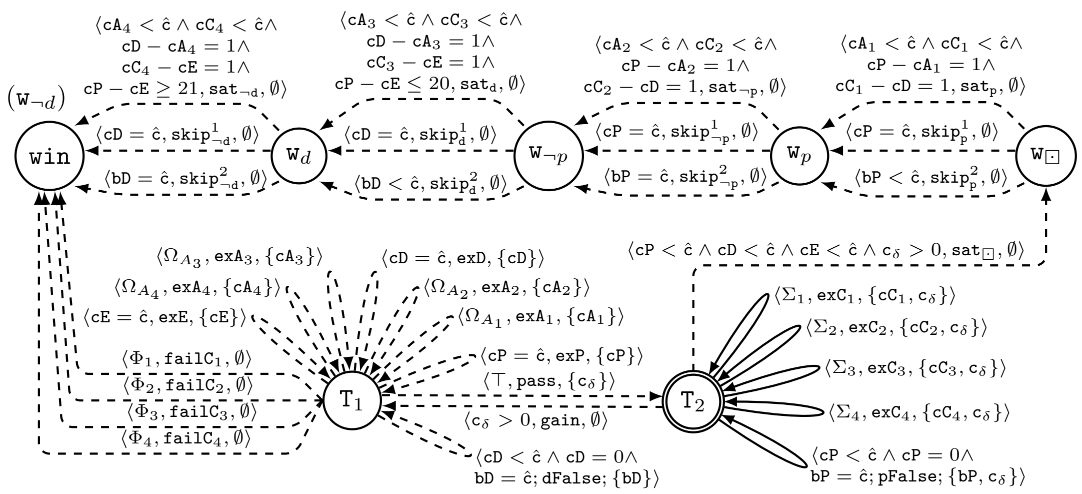
Therefore, considering the time point label honesty property for CSTNUDs, it is possible to extend the guards of $\texttt{exA}_1$ and $\texttt{exA}_4$ by appending $\texttt{bP} = \hat{\texttt{c}} \wedge \texttt{cP} < \hat{\texttt{c}} \wedge \texttt{cP} > 0$ and $\texttt{bD} < \hat{\texttt{c}} \wedge \texttt{cD} < \hat{\texttt{c}} \wedge \texttt{cD} \geq 0$, respectively. The former models the fact that A_1 must be executed if only if $p = \top$ (i.e., $\texttt{bP} = \hat{\texttt{c}}$), which also implies that A_1 must be executed after $P?$ (i.e., $P?$ have been executed ($\texttt{cP} < \hat{\texttt{c}}$)) and a positive amount of time ϵ has elapsed ($\texttt{cP} > 0$). The latter models the fact that A_4 must be executed if only if $d = \bot$ (i.e., $\texttt{bD} < \hat{\texttt{c}}$), which also implies that A_4 must be executed after $D!$ (i.e., $D!$ have been executed ($\texttt{cD} < \hat{\texttt{c}}$)) and possibly immediately or after a positive amount of time has elapsed ($\texttt{cD} \geq 0$).

▶ **Definition 14** (Encoding time point label honesty). A *label encoder* is a mapping $L_{enc}\colon \mathcal{T} \to \mathcal{H}(\mathcal{X})$ translating the label of a time point into the equivalent clock constraint $L_{enc}(X) = L^{\mathcal{OP}}_{enc}(X) \wedge L^{\mathcal{DP}}_{enc}(X)$, where $L^{\mathcal{OP}}_{enc}(X)$ and $L^{\mathcal{DP}}_{enc}(X)$ encode all literals containing observable and decidable propositions, respectively.

- $L^{\mathcal{OP}}_{enc}(X)$: $\bigwedge_{p \in L(X)}(\texttt{bP} = \hat{\texttt{c}} \wedge \texttt{cP} < \hat{\texttt{c}} \wedge \texttt{cP} > 0) \bigwedge_{\neg q \in L(X)}(\texttt{bQ} < \hat{\texttt{c}} \wedge \texttt{cQ} < \hat{\texttt{c}} \wedge \texttt{cQ} > 0)$
- $L^{\mathcal{DP}}_{enc}(X)$: $\bigwedge_{d \in L(X)}(\texttt{bD} = \hat{\texttt{c}} \wedge \texttt{cD} < \hat{\texttt{c}} \wedge \texttt{cD} \geq 0) \bigwedge_{\neg f \in L(X)}(\texttt{bF} < \hat{\texttt{c}} \wedge \texttt{cF} < \hat{\texttt{c}} \wedge \texttt{cF} \geq 0)$

We now focus on constraints. Consider the requirement link $P? \to A_1$ labeled by $[1,1], p$ in the CSTNUD that we are discussing. Such a constraint says that A_1 must be executed after 1 and within 1 since $P?$ (thus, exactly after 1 since $P?$). This requirement link has also an important characteristic: $L(A_1)$ coincides with the label of the link. Therefore, whenever A_1 is executed, the constraint must hold. Thus, we extend the original guard of $\texttt{exA}_1$ (formerly $\texttt{cA}_1 = \hat{\texttt{c}}$) to $\texttt{cA}_1 = \hat{\texttt{c}} \wedge \texttt{cP} < \hat{\texttt{c}} \wedge \texttt{cP} = 1$, where the new conjuncts say that $P?$ has already been executed ($\texttt{cP} < \hat{\texttt{c}}$) and $A_1 - P? \in [1,1]$ ($\texttt{cP} = 1$). More formally:

▶ **Definition 15** (Encoding predecessors). Given a CSTNUD, a *predecessor* of a time point $Y \in \mathcal{T}$ is a time point $X \in \mathcal{T}$ such that there exists a constraint $(X - Y \leq -x, L(Y)) \in \mathcal{C}$ where $x > 0$. $\Pi : \mathcal{T} \to 2^{\mathcal{T}}$ returns the predecessors of a given time point and it is formalized as $\Pi(Y) = \{X \mid (X - Y \leq -x, \ell) \in \mathcal{C} \wedge x > 0 \wedge \ell = L(Y)\}$. A *predecessor encoder* is a



$\langle \mathtt{cA_4} < \hat{\mathtt{c}} \wedge \mathtt{cC_4} < \hat{\mathtt{c}} \wedge \mathtt{cD} - \mathtt{cA_4} = 1 \wedge \mathtt{cC_4} - \mathtt{cE} = 1 \wedge \mathtt{cP} - \mathtt{cE} \geq 21, \mathtt{sat_{\neg d}}, \emptyset \rangle$
$\langle \mathtt{cA_3} < \hat{\mathtt{c}} \wedge \mathtt{cC_3} < \hat{\mathtt{c}} \wedge \mathtt{cD} - \mathtt{cA_3} = 1 \wedge \mathtt{cC_3} - \mathtt{cE} = 1 \wedge \mathtt{cP} - \mathtt{cE} \leq 20, \mathtt{sat_d}, \emptyset \rangle$
$\langle \mathtt{cA_2} < \hat{\mathtt{c}} \wedge \mathtt{cC_2} < \hat{\mathtt{c}} \wedge \mathtt{cP} - \mathtt{cA_2} = 1 \wedge \mathtt{cC_2} - \mathtt{cD} = 1, \mathtt{sat_{\neg p}}, \emptyset \rangle$
$\langle \mathtt{cA_1} < \hat{\mathtt{c}} \wedge \mathtt{cC_1} < \hat{\mathtt{c}} \wedge \mathtt{cP} - \mathtt{cA_1} = 1 \wedge \mathtt{cC_1} - \mathtt{cD} = 1, \mathtt{sat_p}, \emptyset \rangle$
$(\mathtt{w}_{\neg d})$
win
$\mathtt{w}_d$
$\mathtt{w}_{\neg p}$
$\mathtt{w}_p$
$\mathtt{w}_\square$
$\langle \mathtt{cD} = \hat{\mathtt{c}}, \mathtt{skip^1_{\neg d}}, \emptyset \rangle$
$\langle \mathtt{bD} = \hat{\mathtt{c}}, \mathtt{skip^2_{\neg d}}, \emptyset \rangle$
$\langle \mathtt{cD} = \hat{\mathtt{c}}, \mathtt{skip^1_d}, \emptyset \rangle$
$\langle \mathtt{bD} < \hat{\mathtt{c}}, \mathtt{skip^2_d}, \emptyset \rangle$
$\langle \mathtt{cP} = \hat{\mathtt{c}}, \mathtt{skip^1_{\neg p}}, \emptyset \rangle$
$\langle \mathtt{bP} = \hat{\mathtt{c}}, \mathtt{skip^2_{\neg p}}, \emptyset \rangle$
$\langle \mathtt{cP} = \hat{\mathtt{c}}, \mathtt{skip^1_p}, \emptyset \rangle$
$\langle \mathtt{bP} < \hat{\mathtt{c}}, \mathtt{skip^2_p}, \emptyset \rangle$
$\langle \mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{cD} < \hat{\mathtt{c}} \wedge \mathtt{cE} < \hat{\mathtt{c}} \wedge \mathtt{c_\delta} > 0, \mathtt{sat_\square}, \emptyset \rangle$
$\langle \Omega_{A_3}, \mathtt{exA_3}, \{\mathtt{cA_3}\} \rangle$
$\langle \mathtt{cD} = \hat{\mathtt{c}}, \mathtt{exD}, \{\mathtt{cD}\} \rangle$
$\langle \Omega_{A_4}, \mathtt{exA_4}, \{\mathtt{cA_4}\} \rangle$
$\langle \Omega_{A_2}, \mathtt{exA_2}, \{\mathtt{cA_2}\} \rangle$
$\langle \mathtt{cE} = \hat{\mathtt{c}}, \mathtt{exE}, \{\mathtt{cE}\} \rangle$
$\langle \Omega_{A_1}, \mathtt{exA_1}, \{\mathtt{cA_1}\} \rangle$
$\langle \Phi_1, \mathtt{failC_1}, \emptyset \rangle$
$\langle \Phi_2, \mathtt{failC_2}, \emptyset \rangle$
$\langle \Phi_3, \mathtt{failC_3}, \emptyset \rangle$
$\langle \Phi_4, \mathtt{failC_4}, \emptyset \rangle$
$\langle \mathtt{cP} = \hat{\mathtt{c}}, \mathtt{exP}, \{\mathtt{cP}\} \rangle$
$\langle \top, \mathtt{pass}, \{\mathtt{c_\delta}\} \rangle$
$\mathtt{T_1}$
$\mathtt{T_2}$
$\langle \mathtt{c_\delta} > 0, \mathtt{gain}, \emptyset \rangle$
$\langle \mathtt{cD} < \hat{\mathtt{c}} \wedge \mathtt{cD} = 0 \wedge \mathtt{bD} = \hat{\mathtt{c}}; \mathtt{dFalse}; \{\mathtt{bD}\} \rangle$
$\langle \Sigma_1, \mathtt{exC_1}, \{\mathtt{cC_1}, \mathtt{c_\delta}\} \rangle$
$\langle \Sigma_2, \mathtt{exC_2}, \{\mathtt{cC_2}, \mathtt{c_\delta}\} \rangle$
$\langle \Sigma_3, \mathtt{exC_3}, \{\mathtt{cC_3}, \mathtt{c_\delta}\} \rangle$
$\langle \Sigma_4, \mathtt{exC_4}, \{\mathtt{cC_4}, \mathtt{c_\delta}\} \rangle$
$\langle \mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{cP} = 0 \wedge \mathtt{bP} = \hat{\mathtt{c}}; \mathtt{pFalse}; \{\mathtt{bP}, \mathtt{c_\delta}\} \rangle$

Figure 4 TGA encoding the CSTNUD in Figure 3b. $\mathtt{T_2}$ (ex L_0) is the initial location (modeling $T_2(t)$ for $t > 0$). $\mathtt{T_1}$ (ex L_1) models $T_1(t)$ for $t > 0$). $\mathtt{w}_\square$, $\mathtt{w}_p$, $\mathtt{w}_{\neg p}$, $\mathtt{w}_d$, $\mathtt{win}$ model the winning path. $\Omega_{A_1}\colon \mathtt{cA_1} = \hat{\mathtt{c}} \wedge \mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{bP} = \hat{\mathtt{c}} \wedge \mathtt{cP} > 0 \wedge \mathtt{cP} = 1$. $\Omega_{A_2}\colon \mathtt{cA_2} = \hat{\mathtt{c}} \wedge \mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{bP} < \hat{\mathtt{c}} \wedge \mathtt{cP} > 0 \wedge \mathtt{cP} = 1$. $\Omega_{A_3}\colon \mathtt{cA_3} = \hat{\mathtt{c}} \wedge \mathtt{cD} < \hat{\mathtt{c}} \wedge \mathtt{bD} = \hat{\mathtt{c}} \wedge \mathtt{cD} \geq 0 \wedge \mathtt{cD} = 1$. $\Omega_{A_4}\colon \mathtt{cA_4} = \hat{\mathtt{c}} \wedge \mathtt{cD} < \hat{\mathtt{c}} \wedge \mathtt{bD} < \hat{\mathtt{c}} \wedge \mathtt{cD} \geq 0 \wedge \mathtt{cD} = 1$.

mapping $\Pi_{enc}\colon \mathcal{T} \to \mathcal{H}(\mathcal{X})$ translating each $X \in \Pi(Y)$ (along with its temporal bounds) into an equivalent clock constraint as follows. $\Pi_{enc}(Y) = \bigwedge_{X \in \Pi(Y)} \mathtt{cX} < \hat{\mathtt{c}} \wedge \mathtt{cX} \geq x \wedge \mathtt{cX} \leq y$, where $\mathtt{cX} \geq x$ models $(X - Y \leq -x, L(Y))$ and $\mathtt{cX} \leq y$ models $(Y - X \leq y, L(Y))$ (if any).

Therefore, for each non-contingent time point X, the guard of $\mathtt{exX}$ becomes $\Omega_X\colon \mathtt{cX} = \hat{\mathtt{c}} \wedge L_{enc}(X) \wedge \Pi_{enc}(X)$. In Figure 4 we shortened the guards of $\mathtt{exA_1}$, $\mathtt{exA_2}$, $\mathtt{exA_3}$ and $\mathtt{exA_4}$ as Ω_{A_1}, Ω_{A_2}, Ω_{A_3} and Ω_{A_4} and detailed them in the caption.

After optimizing the guard of each $\mathtt{exX}$ transition we now discuss how to model the truth value assignment to the decidable propositions. Dually to observable propositions, for each decidable proposition $d \in \mathcal{DP}$ we generate an uncontrollable self-loop transition $\langle \mathtt{T_1}, \mathtt{cD} < \hat{\mathtt{c}} \wedge \mathtt{cD} = 0 \wedge \mathtt{bD} = \hat{\mathtt{c}}, \mathtt{dFalse}, \{\mathtt{bD}\}, \mathtt{T_1} \rangle$ at $\mathtt{T_1}$. If we take this transition, it means that we decide $\neg d$. If we do not, it means that we decide (actually confirm) d. In the former case, such a transition has to be taken at the same instant in which $D!$ was executed but after $\mathtt{exD}$ was taken. In this way we model "how" to decide the truth values of the propositions in $\mathcal{DP}$. All other transitions remain the same of those given for CSTNUs.

5 Automated Planning: A Tool for the Experimental Evaluation

We made a tool[3] for CSTNUDs which takes as input a CSTNUD specification and allows for the automated encoding into the corresponding UPPAAL-TIGA specification as well as execution simulation. To get the UPPAAL-TIGA specification we run `./Cstnud Network.cstnud --encode TGA.xml`, where `Network.cstnud` is the CSTNUD specification and `TGA.xml` the encoding into a TGA the tool returns in output. To synthesize a strategy we use UPPAAL-TIGA by querying the TGA with `verifytga -s -q -w0 TGA.xml dc.q > strategy`, where `dc.q` contains the TCTL query `control: A[] not tga.win` and

[3] Available at `http://regis.di.univr.it/TIME2017.tar.gz` along with the case studies of this paper and further 1000 randomly generated CSTNUDs as an initial set of benchmarks.

```
$ ./Cstnud ...
P = 0.1
p = true
A1 = 1.1
C1 = 6.0
D = 7.0
d = true
A3 = 8.0
C3 = 11.7
E = 12.7
Verifying ... SAT!
```

(a) `Player1` observes p and $C_1 = 6$, therefore decides d.

```
$ ./Cstnud ...
P = 0.1
p = false
A2 = 1.1
C2 = 9.2
D = 10.2
d = true
A3 = 11.2
C3 = 14.7
E = 15.7
Verifying ... SAT!
```

(b) `Player1` observes $\neg p$ and $C_2 = 9.2$, therefore decides d.

```
$ ./Cstnud ...
P = 0.1
p = false
A2 = 1.1
C2 = 12.7
D = 13.7
d = false
A4 = 14.7
C4 = 21.5
E = 22.5
Verifying ... SAT!
```

(c) `Player1` observes $\neg p$ and $C_2 = 12.7$, therefore decides $\neg d$.

Figure 5 Execution simulations for Figure 3b (`./Cstnud Fig3b.cstnud --execute Fig3b.s`).

`strategy` is the memoryless execution strategy that UPPAAL-TIGA spits out. To execute a controllable CSTNUD we run `./Cstnud Network.cstnud --execute strategy`.

We encoded the CSTNUDs in Figure 3a, Figure 3b and Figure 3c to get the UPPAAL-TIGA specifications. We ran UPPAAL-TIGA on such specifications. We used a Linux virtual machine run on top of a VMWare ESXi hypervisor using a physical machine equipped with an Intel i7 2.80GHz and 20GB of RAM for the experimental evaluation. The VM was assigned 5GB of RAM and full CPU power. For Figure 3a the analysis took 2 minutes and 4 seconds answering `Property is satisfied` and spitting out an execution strategy of 68K for `Player2`. For both Figure 3b and Figure 3c the analysis took 1 minute and 53 second spitting out a strategy of 44K for `Player1`. Finally, we executed the latter two controllable cases. The simulator correctly scheduled all non contingent time points satisfying all constraints. We show the output of a few simulations for Figure 3b in Figure 5.

Furthermore, we randomly generated 1000 CSTNUDs as an initial set of benchmarks and ran the analysis on those networks imposing a time out of 900 seconds each. The analysis proved that 169 networks were DC and 14 non-DC. The remaining networks reached the timeout limit. Each CSTNUD proved DC was correctly executed.

6 Correctness of the Encoding

We prove the correctness and discuss the complexity of the encoding provided in Section 4.

▶ **Theorem 16.** *The encoding in Section 4 is correct.*

Proof. To prove that we start by showing that the encoding in Section 4 correctly models the move-based semantics of Section 3. A state of the TGA is given by a pair $(L, \vec{c})$, where L is a locations and $\vec{c}$ represents the values of all clocks. The state of a CSTNUD during execution is given by its execution sequence Z. We show that the game interplay correctly models the continuous game evolution given in Definition 11 for all $t > 0$. We exclude the case for $t = 0$, so `Player1` does not play in $T_1(0)$ and `Player2` does not play in $T_2(0)$.

(Invariant) At any instant $t > 0$ the snapshot $Z(t) = \langle E, K, S_E, S_K \rangle$ corresponds to a state of the TGA $(L, \vec{c})$ in which $L = \mathtt{T}_2$ and $\vec{c}$ is as follows: $\hat{\mathtt{c}} = t$, $\mathtt{c}_\delta = 0$, for each $X \in \mathcal{T}$, $\mathtt{cX} < \hat{\mathtt{c}}$ and $\hat{\mathtt{c}} - \mathtt{cX} = k$ (if $X \in E \wedge S_E(X) = k$), $\mathtt{cX} = \hat{\mathtt{c}}$ otherwise. For each $p \in \mathcal{P}$, $\mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{bP} = \hat{\mathtt{c}}$

(if $p \in K$ and $S_K(p) = \top$) and $\texttt{cP} < \hat{\texttt{c}} \wedge \texttt{bP} < \hat{\texttt{c}}$ (if $p \in K$ and $S_K(p) = \top$), $\texttt{cP} = \texttt{bP} = \hat{\texttt{c}}$ otherwise. Finally, `Player2` has finished taking controllable transitions at t.

When $t = 0$ (i.e., $\hat{\texttt{c}} = 0$) `Player2` cannot play in $\texttt{T}_2$ as no controllable transition is enabled. `Player1` cannot play either because the current location is not $\texttt{T}_1$ and he can only got there after a positive amount of time has elapsed. Therefore, at $t = 0$, $Z(0) = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$.

When $t > 0$ (i.e., $\hat{\texttt{c}} > 0$) both `Player1` and `Player2` can play in their respective turns $T_1(t)$ and $T_2(t)$. `Player1` can take `gain` to enter $\texttt{T}_1$ at time t. `Player2` cannot prevent him from doing so because `gain`, being urgent, has priority over any other controllable transition that `Player2` could take at that time. So, `Player1` plays first. Once got in $\texttt{T}_1$, `Player1` can take (in general) a non-empty sequence of transitions to execute a few non contingent time points and decide the truth values of some decidable propositions if he has executed some decision time points. Such a sequence is finite since there is a finite number of time points to execute and a finite number of propositions to assign. Furthermore, each time point (resp., proposition) can be executed (resp., assigned a value) only once. When this sequence of transitions is over, `Player1` ends his turn by taking `pass` to lead the run back to $\texttt{T}_2$. Since $\texttt{T}_1$ is urgent, time has not elapsed. Therefore, the sequence of transitions taken at $\texttt{T}_1$ corresponds to the sequence of moves made by `Player1` in $T_1(t)$. Instead, if `Player1` wants to wait at time t, he can either take `gain` and `pass` immediately after or just avoid taking `gain`. Now, at $\texttt{T}_2$, `Player2` does the same for contingent time points and observable propositions if some observation time points have been executed by `Player1` in $T_1(t)$. When `Player2` is done, the sequence of transitions taken, models the sequence of moves made in $T_2(t)$. Since `Player2` does not make any other move in $T_2(t)$, $Z(t)$ can no longer be modified.

`Player1` and `Player2` are driven by their strategies σ_1 and σ_2 which say what moves to make (i.e., transitions to take) in $T_1(t)$ and $T_2(t)$ at any time t depending on the current Z. The purpose of σ_1 is to keep $Z(t)$ locally consistent, whereas that of σ_2 is the opposite.

The strategies also satisfy their applicability conditions as `Player1` can make his moves in $T_1(t)$ according to σ_1 iff `Player2` has not played yet in $T_2(t)$, whereas `Player2` can make his moves in $T_2(t)$ according to σ_2 iff either `Player1` has not played in $T_1(t)$ or `Player2` is not done in $T_2(t)$. We have already proved that for any $t > 0$, `Player1` plays first.

The strategies satisfy their obligations as each time `Player1` executes a decision time point $D!$, he also assigns the associated decidable proposition d a truth value as well. This occurs at the same time but sequentially after the execution of $D!$. `Player1` assigns $\top$ to d by not taking `dFalse` and assigns $\bot$ to d by taking `pFalse`. If `Player1` takes the transition then he cannot take it again in the same turn (as the guard of `pFalse` invalidates). If he does not, then he will never be able to take `dFalse` in any $T_1(t')$ where $t' > t$. Likewise, σ_2 satisfies its similar obligation for observable propositions. Furthermore, σ_2 also satisfies the obligation regarding contingent time points as the encoding generates a `failC` transition for each contingent time point C (belonging to a $(A, x, y, C) \in \mathcal{L}$) allowing `Player1` to move to `win` if `Player2` does not take `exC` within its maximum upper bound y. Since `Player2` wants to prevent `Player1` from getting to `win`, σ_2 is obliged to schedule C such that $C - A \in [x, y]$.

Both σ_1 and σ_2 satisfy their postconditions: the reset of `cX` clocks says when the time points were executed, whereas the values of `bP` clocks say what truth values the propositions have been assigned. Finally, winning conditions are modeled differently with respect to the player. For `Player1` they are abstracted as a winning path checking that all time points and constraints whose labels are not falsified by ℓ_{cps} have been executed and satisfied, respectively. For `Player2` winning conditions correspond to schedule a contingent time point at a particular time or decide a truth value for an observable propositions (or any combination of these moves) such that `Player1` is unable to satisfy at least one constraint and ends up blocked somewhere while going through the winning path before entering `win`. ◀

▶ **Theorem 17.** *Any CSTNUD can be encoded into a TGA in polynomial time.*

Proof Sketch. Our encoding subsumes that for CSTNUs which runs in polynomial time [4, 5]. We “worsen” that encoding by adding a `dFalse` transition for each $d \in \mathcal{DP}$. For each $X \in \mathcal{T}$, $L_{enc}(X)$ and $\Pi_{enc}(X)$ are computed in polynomial time by analyzing $L(X)$ and $\mathcal{C}$. ◀

7 Related Work

STNs [10] and Drake [9] differ from CSTNUDs since they do not specify any uncontrollable part. Therefore, they are incomparable with CSTNUDs.

STNUs [18] specify contingent durations as the unique uncontrollable part. The execution of non-contingent time points cannot influence any contingent duration. Instead, contingent durations do influence the real-value assignment to the non-contingent time points. However, such durations never prevent any non-contingent time point from being executed. This work also addresses the influence of the controllable part over the uncontrollable one.

CSTNs [14, 20] specify conditional constraints as the unique uncontrollable part. Again, the execution of non-contingent time points cannot prevent any truth value assignment from happening. Instead, depending on what truth value a propositional variable is assigned some time point might be excluded, runtime, from the execution of the network. Similar explanations hold for CSTNUs [7, 13] which merge CSTNs and STNUs. CSTNUDs are also able to prevent uncontrollable truth value assignments and durations from happening.

In [3] CSTNs are extended with decision nodes regulating the truth value assignments to some propositions under control. That work focuses on the complexity analysis of the DC-checking problem and provides constraint-propagation algorithms for special cases in which either the network specifies only decisions and no observations or all decisions are made before any observation. Moreover, contingent durations are not addressed. This work follows a complete different direction starting from CSTNUs and it is based on TGAs.

In temporal workflow management, the difference between controllable and uncontrollable XOR splits is introduced in [11] and a technique based on PERT-nets computes internal activity deadlines in order to meet the global ones. Some missed deadlines require human interaction for recovery. We rely on DC, which guarantees that we never miss any deadline.

In [19] UPPAAL-TIGA is used to synthesize a controller for timeline-based plans which consider multivalued state variables and networks of TGAs. Apart from time points, our variables are Boolean and our encoding involves one TGA only.

8 Conclusions and Future Work

We defined *conditional simple temporal networks with uncertainty and decisions* (CSTNUDs) as a unified formalism. CSTNUDs implicitly embed all minor temporal network formalisms such as STNs (if $\mathcal{L} = \mathcal{OT} = \mathcal{DT} = \emptyset$), CSTNs (if $\mathcal{L} = \mathcal{DT} = \emptyset$), STNUs (if $\mathcal{OT} = \mathcal{DT} = \emptyset$), CSTNUs (if $\mathcal{DT} = \emptyset$), STNDs (if $\mathcal{L} = \mathcal{OT} = \emptyset$), CSTNDs (if $\mathcal{L} = \emptyset$), and STNUDs (if $\mathcal{OT} = \emptyset$). We modeled the DC-checking of a CSTNU as a two-player game where `Player1` models the controller and `Player2` models the environment and gave the execution semantics in move-based strategies. We provided an encoding from CSTNUDs into TGAs as an optimized extension of that given for CSTNUs and discussed the correctness and complexity of such an encoding. We automated the approach by making a tool we used to analyze and simulate the execution of the examples discussed in this paper. We also provided a preliminary experimental evaluation of the approach against a set of 1000 randomly generated CSTNUDs. As future work, we plan to address weak and strong controllability of CSTNUDs.

References

1 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994. doi:10.1016/0304-3975(94)90010-8.

2 Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Uppaal-tiga: Time for playing games! In *CAV 2007*. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-73368-3_14.

3 Massimo Cairo, Carlo Combi, Carlo Comin, Luke Hunsberger, Roberto Posenato, Romeo Rizzi, and Matteo Zavatteri. Incorporating decision nodes into conditional simple temporal networks. In S. Schewe, T. Schneider, and J. Wijsen, editors, *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*, volume 91 of *LIPIcs*, pages 9:1–9:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.TIME.2017.9.

4 Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Sound and complete algorithms for checking the dynamic controllability of temporal networks with uncertainty, disjunction and observation. In *21st International Symposium on Temporal Representation and Reasoning (TIME 2014)*, pages 27–36, 2014. doi:10.1109/TIME.2014.21.

5 Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Dynamic controllability via timed game automata. *Acta Informatica*, 53(6-8):681–722, 2016. doi:10.1007/s00236-016-0257-2.

6 Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, and Marco Roveri. Using timed game automata to synthesize execution strategies for simple temporal networks with uncertainty. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27-31, 2014, Québec City, Québec, Canada.*, pages 2242–2249, 2014. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8512.

7 Carlo Combi, Luke Hunsberger, and Roberto Posenato. An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty. In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence – Volume 2: ICAART*, pages 144–156. INSTICC, ScitePress, 2013. doi:10.5220/0004256101440156.

8 Carlo Combi, Roberto Posenato, Luca Viganò, and Matteo Zavatteri. Access controlled temporal networks. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence – Volume 2: ICAART*, pages 118–131. INSTICC, ScitePress, 2017. doi:10.5220/0006185701180131.

9 Patrick R. Conrad and Brian C. Williams. Drake: An efficient executive for temporal plans with choice. *J. Artif. Int. Res.*, 42(1):607–659, September 2011.

10 Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. doi:10.1016/0004-3702(91)90006-6.

11 Johann Eder, Euthimios Panagos, Heinz Pozewaunig, and Michael Rabinovich. *Time Management in Workflow Systems*, pages 265–280. Springer London, 1999. doi:10.1007/978-1-4471-0875-7_22.

12 John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

13 Luke Hunsberger, Roberto Posenato, and Carlo Combi. The Dynamic Controllability of Conditional STNs with Uncertainty. In *Workshop on Planning and Plan Execution for Real-World Systems (PlanEx) at ICAPS-2012*, pages 1–8, Atibaia, June 2012. URL: http://arxiv.org/abs/1212.2005.

14 Luke Hunsberger, Roberto Posenato, and Carlo Combi. A sound-and-complete propagation-based algorithm for checking the dynamic consistency of conditional simple temporal networks. In *22st International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 4–18, 2015. `doi:10.1109/TIME.2015.26`.

15 Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems. In Ernst W. Mayr and Claude Puech, editors, *STACS'95: 12th Annual Symposium on Theoretical Aspects of Computer Science Munich, Germany, March 2–4, 1995 Proceedings*, pages 229–242, Berlin, Heidelberg, 1995. Springer. `doi:10.1007/3-540-59042-0_76`.

16 Paul Morris. The mathematics of dispatchability revisited. In *Proceedings of the Twenty-Sixth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'16, pages 244–252. AAAI Press, 2016.

17 Paul Morris and Nicola Muscettola. Temporal Dynamic Controllability Revisited. In *Proceedings of the Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, pages 1193–1198. AAAI Pr., 2005.

18 Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 494–502, 2001.

19 Andrea Orlandini, Alberto Finzi, Amedeo Cesta, and Simone Fratini. TGA-Based Controllers for Flexible Plan Execution. In Joscha Bach and Stefan Edelkamp, editors, *KI 2011: Advances in Artificial Intelligence: 34th Annual German Conference on AI, Berlin, Germany, October 4-7,2011. Proceedings*, pages 233–245. Springer Berlin Heidelberg, 2011. `doi:10.1007/978-3-642-24455-1_22`.

20 Ioannis Tsamardinos, Thierry Vidal, and Martha E. Pollack. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints*, 8(4):365–388, 2003. `doi:10.1023/A:1025894003623`.