

# 30th International Symposium on Temporal Representation and Reasoning

TIME 2023, September 25–26, 2023, NCSR Demokritos, Athens,  
Greece

Edited by

Alexander Artikis

Florian Bruse

Luke Hunsberger



*Editors*

**Alexander Artikis** 

University of Piraeus & NCSR Demokritos, Greece  
a.artikis@unipi.gr

**Florian Bruse** 

University of Kassel, Germany  
florian.bruse@uni-kassel.de

**Luke Hunsberger**

Vassar College, Poughkeepsie, NY, USA  
hunsberger@vassar.edu

*ACM Classification 2012*

Theory of computation; Information systems; Computing methodologies → Artificial intelligence; Applied computing

**ISBN 978-3-95977-298-3**

*Published online and open access by*

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <https://www.dagstuhl.de/dagpub/978-3-95977-298-3>.

*Publication date*

September, 2023

*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 <https://portal.dnb.de>.

*License*

This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0): <https://creativecommons.org/licenses/by/4.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.TIME.2023.0

ISBN 978-3-95977-298-3

ISSN 1868-8969

<https://www.dagstuhl.de/lipics>

## LIPICs – Leibniz International Proceedings in Informatics

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

### *Editorial Board*

- Luca Aceto (*Chair*, Reykjavik University, IS and Gran Sasso Science Institute, IT)
- Christel Baier (TU Dresden, DE)
- Roberto Di Cosmo (Inria and Université de Paris, FR)
- Faith Ellen (University of Toronto, CA)
- Javier Esparza (TU München, DE)
- Daniel Král' (Masaryk University, Brno, CZ)
- Meena Mahajan (Institute of Mathematical Sciences, Chennai, IN)
- Anca Muscholl (University of Bordeaux, FR)
- Chih-Hao Luke Ong (University of Oxford, GB)
- Phillip Rogaway (University of California, Davis, US)
- Eva Rotenberg (Technical University of Denmark, Lyngby, DK)
- Raimund Seidel (Universität des Saarlandes, Saarbrücken, DE and Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, DE)
- Pierre Senellart (ENS, Université PSL, Paris, FR)

**ISSN 1868-8969**

**<https://www.dagstuhl.de/lipics>**



## ■ Contents

Preface	
<i>Alexander Artikis, Florian Bruse, and Luke Hunsberger</i> .....	0:vii
TIME Steering Committee	
.....	0:ix
Program Committee Members	
.....	0:xi–0:xii
Authors	
.....	0:xiii–0:xiv

### Invited Talk

Learning Temporal Logic Formulas from Time-Series Data	
<i>Laura Nenzi</i> .....	1:1–1:2

### Regular Papers

LTL over Finite Words Can Be Exponentially More Succinct Than Pure-Past LTL, and <i>vice versa</i>	
<i>Alessandro Artale, Luca Geatti, Nicola Gigante, Andrea Mazzullo, and Angelo Montanari</i> .....	2:1–2:14
LSCPM: Communities in Massive Real-World Link Streams by Clique Percolation Method	
<i>Alexis Baudin, Lionel Tabourier, and Clémence Magnien</i> .....	3:1–3:18
Discovering Predictive Dependencies on Multi-Temporal Relations	
<i>Beatrice Amico, Carlo Combi, Romeo Rizzi, and Pietro Sala</i> .....	4:1–4:19
Prime Scenarios in Qualitative Spatial and Temporal Reasoning	
<i>Yakoub Salhi and Michael Sioutis</i> .....	5:1–5:14
Bounded-Memory Runtime Enforcement of Timed Properties	
<i>Saumya Shankar, Srinivas Pinisetty, and Thierry Jéron</i> .....	6:1–6:22
More Than 0s and 1s: Metric Quantifiers and Counting over Timed Words	
<i>Hsi-Ming Ho and Khushraj Madnani</i> .....	7:1–7:15
Analyzing Complex Systems with Cascades Using Continuous-Time Bayesian Networks	
<i>Alessandro Bregoli, Karin Rathsman, Marco Scutari, Fabio Stella, and Søren Wengel Mogensen</i> .....	8:1–8:21
A Sound and Complete Tableau System for Fuzzy Halpern and Shoham’s Interval Temporal Logic	
<i>Willem Conradie, Riccardo Monego, Emilio Muñoz-Velasco, Guido Sciavicco, and Ionel Eduard Stan</i> .....	9:1–9:14

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The Calculus of Temporal Influence <i>Florian Bruse, Marit Kastaun, Martin Lange, and Sören Möller</i> .....	10:1–10:19
Detecting Causality in the Presence of Byzantine Processes: The Synchronous Systems Case <i>Anshuman Misra and Ajay D. Kshemkalyani</i> .....	11:1–11:14
Embarrassingly Greedy Inconsistency Resolution of Qualitative Constraint Networks <i>Michael Sioutis</i> .....	12:1–12:12
Optimization of Nonsequenced Queries Using Log-Segmented Timestamps <i>Curtis E. Dyreson</i> .....	13:1–13:15

## Extended Abstracts

An Event Calculus for Run-Time Reasoning <i>Periklis Mantenoglou</i> .....	14:1–14:3
SSTRESED: Scalable Semantic Trajectory Extraction for Simple Event Detection over Streaming Movement Data <i>Nikos Giatrakos</i> .....	15:1–15:4
A Decomposition Framework for Inconsistency Handling in Qualitative Spatial and Temporal Reasoning <i>Yakoub Salhi and Michael Sioutis</i> .....	16:1–16:3
Answer Set Automata: A Learnable Pattern Specification Framework for Complex Event Recognition <i>Nikos Katzouris and Georgios Paliouras</i> .....	17:1–17:3
A Benchmark for Early Time-Series Classification <i>Petro-Foti Kamberi, Evgenios Kladis, and Charilaos Akasiadis</i> .....	18:1–18:3
Time-Aware Robustness of Temporal Graph Neural Networks for Link Prediction <i>Marco Sälzer and Silvia Beddar-Wiesing</i> .....	19:1–19:3
Converting Simple Temporal Networks with Uncertainty into Dispatchable Form – Faster <i>Luke Hunsberger and Roberto Posenato</i> .....	20:1–20:3
Towards Infinite-State Verification and Planning with Linear Temporal Logic Modulo Theories <i>Luca Geatti, Alessandro Gianola, and Nicola Gigante</i> .....	21:1–21:3
Qualitative past Timeline-Based Games <i>Renato Acampora, Luca Geatti, Nicola Gigante, and Angelo Montanari</i> .....	22:1–22:3

## ■ Preface

After three years of online events, the 30th edition of the International Symposium on Temporal Representation and Reasoning (TIME 2022) will take place as a physical event at NCSR Demokritos in Athens, Greece.

Since its first edition in 1994, the TIME Symposium is quite unique in the panorama of the scientific conferences as its main goal is to bring together researchers from distinct research areas involving the management and representation of temporal data as well as reasoning about temporal aspects of information. Moreover, the TIME Symposium aims to bridge theoretical and applied research, as well as to serve as an interdisciplinary forum for exchange among researchers from the areas of artificial intelligence, database management, logic and verification and beyond.

As a novelty this year, TIME was also open to systems papers focusing on the development, deployment and evaluation of systems for temporal reasoning, as opposed to traditional theoretical contributions. Moreover, we also solicited extended abstracts presenting work-in-progress or summarising work that has been published elsewhere, qualifying for presentation at the symposium and inclusion in the proceedings.

The authors of the of the top-ranked papers will be invited to submit an extended version of their contribution to a special issue in Information and Computation.

We received a total of 16 submissions for regular papers, and another 9 extended abstracts, representing a wide range of research topics in the areas of artificial intelligence, databases and theoretical computer science. Submissions came from Africa, Asia, Europe and North America. We would like to thank all the authors of the submitted papers, as they have helped to build a successful TIME 2023 Symposium.

As a result of the review process and the following discussions, coordinated by the Program Committee chairs, 12 regular papers were selected for presentation at the symposium. The range of their topics is wide, including, among others, predicting temporal functional dependencies, interval temporal logic, and qualitative constraint networks. All 9 extended abstracts were deemed suitable to be presented at the symposium after light reviewing by the Program Committee chairs. The accepted papers and extended abstracts are very interesting and we are confident that we will have lively discussions during the symposium.

We are very pleased to include invited talks by leading scholars in our scientific communities: Thomas Eiter (TU Vienna, Austria) and Laura Nenzi (University of Trieste, Italy). We believe that the invited talks, the selected papers and extended abstracts, and their presentations will help to stimulate and improve several research efforts in the area of temporal representation and reasoning, and motivate members of under-represented research communities to participate in the TIME Symposia.

We would like to thank all the members of the Program Committee and the additional reviewers, who volunteered their time and expertise to set up the final program. We want also to thank Periklis Mantenoglou for his efforts in maintaining the web page of the symposium.

Alexander Artikis, University of Piraeus & NCSR Demokritos, Greece

Florian Bruse, University of Kassel, Germany

Luke Hunsberger, Vassar College, United States

TIME 2023 Program Committee Co-Chairs

July 21st, 2023

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





## ■ TIME Steering Committee

Alexander Artikis  
University of Piraeus & NCSR Demokritos  
Greece  
a.artikis@unipi.gr

Patricia Bouyer  
CNRS & ENS Paris-Saclay  
France  
bouyer@lsv.fr

Carlo Combi (Chair)  
University of Verona  
Italy  
carlo.combi@univr.it

Johann Eder  
University of Klagenfurt  
Austria  
johann.eder@aau.at

Thomas Guyet  
IRISA  
France  
thomas.guyet@irisa.fr

Luke Hunsberger  
Vassar College  
United States  
hunsberger@vassar.edu

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

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

Shankara Narayanan Krishna (Krishna S.)  
IIT Bombay  
India  
krishnas@cse.iitb.ac.in

Mark Reynolds  
University of Western Australia  
Australia  
mark.reynolds@uwa.edu.au

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger



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



## ■ Program Committee Members

S. Akshay  
IIT Bombay  
India  
akshayss@cse.iitb.ac.in

Elias Alevizos  
NCSR Demokritos  
Greece  
alevizos.elias@iit.demokritos.gr

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

Bartosz Bednarczyk  
TU Dresden & University of Wrocław  
Germany & Poland  
bartosz.bednarczyk@cs.uni.wroc.pl

Jim Boerkoel  
Harvey Mudd College  
United States  
boerkoel@hmc.edu

Patricia Bouyer  
CNRS & ENS Paris-Saclay  
France  
bouyer@lsv.fr

Davide Bresolin  
University of Padua  
Italy  
bresolin@math.unipd.it

Jaewook Byun  
Sejong University  
South Korea  
jwbyun@sejong.ac.kr

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

Paulo Cortez  
University of Minho  
Portugal  
pcortez@dsi.uminho.pt

Clare Dixon  
University of Manchester  
United Kingdom  
clare.dixon@manchester.ac.uk

Christos Doukeridis  
University of Piraeus  
Greece  
cdoulk@unipi.gr

Curtis Dyreson  
Utah State University  
United States  
curtis.dyreson@usu.edu

Johann Eder  
University of Klagenfurt  
Austria  
eder@acm.org

Marco Franceschetti  
University of St. Gallen  
Switzerland  
marco.franceschetti@unisg.ch

Silvia García  
University of Vigo  
Spain  
sgarcia@gti.uvigo.es

Gopal Gupta  
The University of Texas at Dallas  
United States  
gupta@utdallas.edu

Sylvain Hallé  
University of Quebec at Chicoutimi  
Canada  
shalle@acm.org

Nikos Katzouris  
NCSR Demokritos  
Greece  
nkatz@iit.demokritos.gr

Mourad Khayati  
University of Fribourg  
Switzerland  
mkhayati@exascale.info

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).  
Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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

Francois Laroussinie  
LIAFA, Univ. Paris 7, CNRS  
France  
francoisl@liafa.univ-paris-diderot.fr

Danh Le Phuoc  
TU Berlin  
Germany  
danh.lephuoc@tu-berlin.de

Stephane Le Roux  
ENS Paris-Saclay  
France  
leroux@lsv.fr

Jianwen Li  
East China Normal University  
China  
lijwen2748@gmail.com

Peter Lucas  
University of Twente  
The Netherlands  
peterl@cs.ru.nl

Ruizhe Ma  
University of Massachusetts Lowell  
United States  
ruizhe\_ma@uml.edu

Nicolas Markey  
IRISA, CNRS & INRIA & Univ. Rennes 1  
France  
nicolas.markey@irisa.fr

Andrea Micheli  
Fondazione Bruno Kessler  
Italy  
amicheli@fbk.eu

Daniel Neider  
TU Dortmund  
Germany  
daniel.neider@cs.tu-dortmund.de

Andrea Orlandini  
CNR  
Italy  
andrea.orlandini@istc.cnr.it

Paritosh Pandya  
TIFR & IIT Bombay  
India  
pandya@tifr.res.in

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

Manfred Reichert  
University of Ulm  
Germany  
manfred.reichert@uni-ulm.de

Matteo Rossi  
Polytechnic University of Milan  
Italy  
matteo.rossi@polimi.it

Pietro Sala  
University of Verona  
Italy  
pietro.sala@univr.it

Spiros Skiadopoulos  
University of Peloponnese  
Greece  
spiros@uop.gr

Stefano Tonetta  
Fondazione Bruno Kessler  
Italy  
tonettas@fbk.eu

Hazem Torfah  
University of California, Berkeley  
United States  
torfah@berkeley.edu

Patrick Totzke  
University of Liverpool  
United Kingdom  
totzke@liverpool.ac.uk

Matteo Zavatteri  
University of Padua  
Italy  
matteo.zavatteri@unipd.it

## ■ List of Authors

Renato Acampora (22)  
University of Udine, Italy

Charilaos Akasiadis (18)  
Institute of Informatics & Telecommunications,  
NCSR "Demokritos", Athens, Greece

Beatrice Amico (4)  
Department of Computer Science, University of  
Verona, Italy

Alessandro Artale (2)  
Free University of Bozen-Bolzano, Italy

Alexis Baudin (3)  
Sorbonne Université, CNRS, LIP6, F-75005  
Paris, France

Silvia Beddar-Wiesing (19)  
School of Electrical Engineering and Computer  
Science, University of Kassel, Germany

Alessandro Bregoli (8)  
Department of Informatics, Systems and  
Communication, University of Milano-Bicocca,  
Italy

Florian Bruse (10)  
Theoretical Computer Science / Formal  
Methods, University of Kassel, Germany

Carlo Combi (4)  
Department of Computer Science, University of  
Verona, Italy

Willem Conradie (9)  
School of Mathematics, University of the  
Witwatersrand, Johannesburg, South Africa

Curtis E. Dyreson (13)  
Department of Computer Science, Utah State  
University, Logan, UT, USA

Luca Geatti (2, 21, 22)  
University of Udine, Italy

Alessandro Gianola (21)  
Free University of Bozen-Bolzano, Italy

Nikos Giatrakos (15)  
Technical University of Crete, Chania, Greece;  
Athena RC, Marousi, Greece

Nicola Gigante (2, 21, 22)  
Free University of Bozen-Bolzano, Italy

Hsi-Ming Ho (7)  
Department of Informatics, University of Sussex,  
UK

Luke Hunsberger (20)  
Vassar College, Poughkeepsie, NY, USA

Thierry Jéron (6)  
Univ Rennes, Inria, IRISA, France

Petro-Foti Kamberi (18)  
Institute of Informatics & Telecommunications,  
NCSR "Demokritos", Athens, Greece

Marit Kastaun (10)  
Didactics of Biology, University of Kassel,  
Germany

Nikos Katzouris (17)  
Institute of Informatics, National Center for  
Scientific Research "Demokritos", Athens,  
Greece

Evgenios Kladis (18)  
Institute of Informatics & Telecommunications,  
NCSR "Demokritos", Athens, Greece

Ajay D. Kshemkalyani (11)  
University of Illinois at Chicago, IL, USA

Martin Lange (10)  
Theoretical Computer Science / Formal  
Methods, University of Kassel, Germany

Khushraj Madnani (7)  
Max Planck Institute for Software Systems,  
Kaiserslautern, Germany

Clémence Magnien (3)  
Sorbonne Université, CNRS, LIP6, F-75005  
Paris, France

Periklis Mantenoglou (14)  
National and Kapodistrian University of Athens,  
Greece; NCSR "Demokritos", Athens, Greece

Andrea Mazzullo (2)  
Free University of Bozen-Bolzano, Italy

Anshuman Misra (11)  
University of Illinois at Chicago, IL, USA

Søren Wengel Mogensen (8)  
Department of Automatic Control, Lund  
University, Sweden

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- Riccardo Monego  (9)  
Department of Mathematics and Computer  
Science, University of Ferrara, Italy
- Angelo Montanari  (2, 22)  
University of Udine, Italy
- Sören Möller (10)  
Theoretical Computer Science / Formal  
Methods, University of Kassel, Germany
- Laura Nenzi  (1)  
Department of Engineering and Architecture,  
University of Trieste, Italy
- Georgios Paliouras  (17)  
Institute of Informatics, National Center for  
Scientific Research "Demokritos", Athens,  
Greece
- Srinivas Pinisetty  (6)  
Indian Institute of Technology Bhubaneswar,  
India
- Roberto Posenato  (20)  
University of Verona, Italy
- Karin Rathsman  (8)  
European Spallation Source ERIC, Lund,  
Sweden
- Romeo Rizzi  (4)  
Department of Computer Science, University of  
Verona, Italy
- Pietro Sala  (4)  
Department of Computer Science, University of  
Verona, Italy
- Yakoub Salhi  (5, 16)  
CRIL UMR 8188, Université d'Artois & CNRS,  
France
- Guido Sciavicco  (9)  
Department of Mathematics and Computer  
Science, University of Ferrara, Italy
- Marco Scutari  (8)  
Istituto Dalle Molle di Studi sull'Intelligenza  
Artificiale (IDSIA), Lugano, Switzerland
- Saumya Shankar  (6)  
Indian Institute of Technology Bhubaneswar,  
India
- Michael Sioutis  (5, 12, 16)  
LIRMM UMR 5506, Université de Montpellier &  
CNRS, France
- Ionel Eduard Stan  (9)  
Faculty of Engineering, Free University of  
Bozen-Bolzano, Italy
- Fabio Stella  (8)  
Department of Informatics, Systems and  
Communication, University of Milano-Bicocca,  
Italy
- Marco Sälzer  (19)  
School of Electrical Engineering and Computer  
Science, University of Kassel, Germany;  
marcosaelzer.github.io
- Lionel Tabourier (3)  
Sorbonne Université, CNRS, LIP6, F-75005  
Paris, France
- Emilio Muñoz-Velasco  (9)  
Department of Applied Mathematics, University  
of Málaga, Spain

# Learning Temporal Logic Formulas from Time-Series Data

Laura Nenzi  

Department of Engineering and Architecture, University of Trieste, Italy

---

## Abstract

In this talk, we provide an overview of recent advancements in the field of mining formal specifications from time-series data, with a specific focus on learning Signal Temporal Logic (STL) formulae.

**2012 ACM Subject Classification** Theory of computation → Modal and temporal logics

**Keywords and phrases** Temporal Logic, Mining Specifications

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.1

**Category** Invited Talk

**Funding** This research was partially supported by the Austrian FWF projects ZK-35, and by the consortium iNEST funded by PNRR – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS\_00000043).

**Acknowledgements** The research surveyed in this talk is joint work with Luca Bortolussi, Eric Medvet, Jyotirmoy V. Deshmukh, Simone Silveti, Federico Pigozzi, Patrick Indri, Sara Mohammadinejad, Ezio Bartocci, and Alberto Bartoli.

## 1 Extended Abstract

The abundance of available data has led to a significant increase in the utilization of machine learning techniques for describing and analyzing systems. Although these methods are adept at generating powerful black-box models and performing well in complex, high-dimensional scenarios, they often lack measures of uncertainty for individual estimates and fail to fully understand the underlying mechanisms driving the obtained outcomes. This limitation becomes particularly crucial when dealing with safety-critical systems like smart healthcare and self-driving cars, where failures can have severe consequences and incur substantial costs. Therefore, it is imperative for designers to gain a comprehensive understanding of the phenomena these systems capture and to extract interpretable information from the data they produce.

To tackle this challenge, recent research has explored the application of learning Temporal Logic (TL) formulae as a powerful approach for extracting human-interpretable information from data. TL is a formal language that offers precise specifications which are easily comprehensible to humans, allowing for the expression of complex system properties in a clear manner. Furthermore, it provides verification algorithms that can automatically evaluate the satisfaction of these properties. In this talk, we aim to provide a comprehensive overview of the latest advancements in this field. We focus in particular on the learning of Signal Temporal Logic (STL) formulae [2]. STL is a linear-time TL very suitable to describe properties associated with real-time trajectories.

First, we present a framework for a supervised learning scenario [4]. We focus on a two-label classification problem, specifically targeting the discrimination between regular and anomalous trajectories. The objective is to develop a technique that learns a Signal Temporal Logic (STL) formula capable of effectively distinguishing between the two sets of labeled data. The desired formula should be satisfied by the regular trajectories to the greatest extent



© Laura Nenzi;

licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 1; pp. 1:1–1:2

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

possible, while not being satisfied by the anomalous ones. To accomplish this, the proposed methodology leverages genetic programming, a form of Evolutionary Computation (EC), to extract the structure of the formula, and Bayesian optimization to learn the formula's parameters.

Second, we explore a semi-supervised scenario, which poses a more complex problem as it considers datasets comprising solely regular behaviors [1]. We introduce an algorithm that utilizes the Grammar-Guided Genetic Programming (G3P) technique to learn an ensemble of STL formulas. This ensemble is then utilized for effectively detecting anomalous behaviors within the system.

Third, we explore the integration of spatial considerations within the study of system behavior. To achieve this, we introduce the Spatio-Temporal Reach and Escape Logic (STREL) [3], an extension of STL that incorporates a variety of spatial operators. This extension enables the modeling and analysis of spatial-temporal properties, providing a comprehensive framework for capturing complex behaviors that involve both temporal and spatial aspects. We present a method that leverages a Parametric STREL (PSTREL) for automatic feature extraction from the given spatio-temporal data. This is accomplished by projecting the data onto the parameter space of PSTREL. Using an agglomerative hierarchical clustering technique, we ensure the satisfaction of a distinct STREL formula in each cluster.

We demonstrate the versatility of these techniques through multiple case studies spanning various domains, including naval surveillance, train speed regulation, secure water treatment, urban transportation, and epidemiological analysis. We conclude the talk by discussing the remaining challenges and future prospects, while also providing an overview of the latest ongoing research.



---

## References

- 1 Patrick Indri, Alberto Bartoli, Eric Medvet, and Laura Nenzi. One-shot learning of ensembles of temporal logic formulas for anomaly detection in cyber-physical systems. In *Genetic Programming - 25th European Conference, EuroGP 2022, Held as Part of EvoStar 2022, Madrid, Spain, April 20-22, 2022, Proceedings*, pages 34–50. Springer, 2022. doi:10.1007/978-3-031-02056-8\_3.
- 2 Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In Yassine Lakhnech and Sergio Yovine, editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings*, Lecture Notes in Computer Science, pages 152–166. Springer, 2004. doi:10.1007/978-3-540-30206-3\_12.
- 3 Laura Nenzi, Ezio Bartocci, Luca Bortolussi, and Michele Loreti. A logic for monitoring dynamic networks of spatially-distributed cyber-physical systems. *Log. Methods Comput. Sci.*, 18(1), 2022. doi:10.46298/lmcs-18(1:4)2022.
- 4 Laura Nenzi, Simone Silvetti, Ezio Bartocci, and Luca Bortolussi. A robust genetic algorithm for learning temporal specifications from data. In *Quantitative Evaluation of Systems - 15th International Conference, QEST 2018, Beijing, China, September 4-7, 2018, Proceedings*, pages 323–338. Springer, 2018. doi:10.1007/978-3-319-99154-2\_20.



# LTL over Finite Words Can Be Exponentially More Succinct Than Pure-Past LTL, and *vice versa*

Alessandro Artale  

Free University of Bozen-Bolzano, Italy

Luca Geatti  



University of Udine, Italy

Nicola Gigante  

Free University of Bozen-Bolzano, Italy

Andrea Mazzullo  

Free University of Bozen-Bolzano, Italy

Angelo Montanari  

University of Udine, Italy

---

## Abstract

---

Linear Temporal Logic over finite traces ( $LTL_f$ ) has proved itself to be an important and effective formalism in formal verification as well as in artificial intelligence. Pure past  $LTL_f$  ( $pLTL$ ) is the logic obtained from  $LTL_f$  by replacing each (future) temporal operator by a corresponding past one, and is naturally interpreted at the end of a finite trace. It is known that each property definable in  $LTL_f$  is also definable in  $pLTL$ , and *vice versa*. However, despite being extensively used in practice, to the best of our knowledge, there is no systematic study of their *succinctness*.

In this paper, we investigate the succinctness of  $LTL_f$  and  $pLTL$ . First, we prove that  $pLTL$  can be exponentially more succinct than  $LTL_f$  by showing that there exists a property definable with a  $pLTL$  formula of size  $n$  such that the size of all  $LTL_f$  formulas defining it is at least *exponential* in  $n$ . Then, we prove that  $LTL_f$  can be exponentially more succinct than  $pLTL$  as well. This result shows that, although being expressively equivalent,  $LTL_f$  and  $pLTL$  are incomparable when succinctness is concerned. In addition, we study the succinctness of **Safety-LTL** (the syntactic safety fragment of LTL over infinite traces) with respect to its canonical form  $G(pLTL)$ , whose formulas are of the form  $G(\alpha)$ ,  $G$  being the *globally* operator and  $\alpha$  a  $pLTL$  formula. We prove that  $G(pLTL)$  can be exponentially more succinct than **Safety-LTL**, and that the same holds for the dual cosafety fragment.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Modal and temporal logics; Theory of computation  $\rightarrow$  Logic and verification

**Keywords and phrases** Temporal Logic, Succinctness,  $LTL_f$ , Finite Traces, Pure past LTL

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.2

**Funding** Luca Geatti and Angelo Montanari acknowledge the support from the 2022 Italian INdAM-GNCS project “*Elaborazione del Linguaggio Naturale e Logica Temporale per la Formalizzazione di Testi*”, ref. no. CUP\_E55F22000270001. Andrea Mazzullo and Angelo Montanari acknowledge the support of the MUR PNRR project FAIR - Future AI Research (PE00000013) funded by the NextGenerationEU. Nicola Gigante acknowledges the support of the PURPLE project, in the context of the AIPlan4EU project’s First Open Call for Innovators.

## 1 Introduction

In this paper, we study the succinctness of Linear Temporal Logic over finite words ( $LTL_f$ ) with respect to pure past  $LTL_f$  ( $pLTL$ ) and prove two lower bounds that show the incomparability of  $LTL_f$  and  $pLTL$  as far as succinctness is concerned. In addition, we investigate some succinctness properties of the safety and cosafety fragments of Linear Temporal Logic over infinite words (LTL) with respect to their canonical forms (resp.,  $G(pLTL)$  and  $F(pLTL)$ ).



© Alessandro Artale, Luca Geatti, Nicola Gigante, Andrea Mazzullo, and Angelo Montanari; licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 2; pp. 2:1–2:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

LTL<sub>f</sub> is a modal logic that extends classic Boolean Logic with *temporal modalities* for reasoning about time and it is interpreted over *finite* sequences of states (called traces or words). LTL<sub>f</sub> is extensively used in many areas of Artificial Intelligence (AI), like automated synthesis [10, 12, 23], planning [4–6], and business process management [19, 20]. In last years, also pLTL, the *pure past* version of LTL<sub>f</sub>, gained momentum in AI. As a matter of fact, while all properties expressible in LTL<sub>f</sub> are also expressible in pLTL and *vice versa* (pLTL and LTL<sub>f</sub> have been shown to be expressively equivalent [9, 16, 25]), some properties like, e.g., those characterizing planning problems (“*to reach a goal while always obeying to a safety rule*”) are more natural and easy to express using past modalities [16]. Moreover, pLTL has been advocated as a suitable declarative, logic programming language [3, 14]. Last but not least, arguably the most important feature of pLTL is enjoying a compilation into deterministic finite automata of singly exponential size [8, 9], a result that cannot be achieved for LTL<sub>f</sub> [11].

In spite of the success of LTL<sub>f</sub> and pLTL, to the best of our knowledge, there is no systematic study of their *succinctness*, that is, the study of which properties (if any) are definable in one logic with formulas of small, polynomial size, but such that all formulas in the other logic would require exponential size or more. The importance of studying succinctness is twofold. On the one hand, it is an important theoretical tool, that joins the study of computational complexity and expressive power (cf. e.g. the work by Hella and Vilander [15], comparing first-order logic with basic modal logic and  $\mu$ -calculus in terms of succinctness, by means of formula size games). On the other hand, it may help in choosing the right formalism when solving problems like model checking and reactive synthesis.

The main contributions of the paper are the following ones.

First, we prove that pLTL *can be exponentially more succinct than* LTL<sub>f</sub>, that is, there exists a family of properties definable with pLTL formulas of size  $n$  such that the size of all LTL<sub>f</sub> formulas defining them is at least exponential in  $n$ .

Second, by exploiting the fact that each trace recognized by a pLTL formula is the *reverse*<sup>1</sup> of a trace recognized by an LTL<sub>f</sub> formula, we derive that LTL<sub>f</sub> *can be exponentially more succinct than* pLTL as well. This has three important consequences:

1. it shows that, despite being expressively equivalent, LTL<sub>f</sub> and pLTL are *incomparable* when succinctness is concerned;
2. it confirms the conjecture formulated in [2], derived from the complexity gap between the realizability problem of LTL<sub>f</sub>, which is 2EXPTIME-complete, and pLTL, which is EXPTIME-complete;
3. it proves that *any* translation from LTL<sub>f</sub> to pLTL (and *vice versa*), for which we only have a triply exponential upper bound [9], has at least an exponential complexity in the size of the initial formula.

Third, we study the succinctness of the syntactic safety fragment of LTL over infinite traces (denoted as Safety-LTL) with respect to its *canonical form*  $G(\text{pLTL})$ , which is the set of formulas of the form  $G(\alpha)$ , where  $G$  is the *globally* modality of LTL and  $\alpha$  is a pLTL formula [7]. We show that  $G(\text{pLTL})$  *can be exponentially more succinct than* Safety-LTL. By a duality argument, we derive the same result for the syntactic cosafety fragment of LTL (coSafety-LTL) and its canonical form  $F(\text{pLTL})$ . Whether Safety-LTL (resp., coSafety-LTL) can be exponentially more succinct than  $G(\text{pLTL})$  (resp.,  $F(\text{pLTL})$ ) is, to the best of our knowledge, still an open question.

---

<sup>1</sup> By “reverse” of a trace  $\sigma$ , we mean the trace obtained by  $\sigma$  considering its last state as the first one.

The paper is organized as follows. In Section 2, we provide the necessary background. Sections 3 and 4 prove, respectively, that pLTL can be exponentially more succinct than LTL<sub>f</sub>, and *vice versa*. In Section 5, we show the succinctness of G(pLTL) and F(pLTL) with respect to the safety and cosafety fragments of LTL, respectively. We conclude with Section 6, where we recap the results of the paper and we point out some future research directions.

## 2 Background

In this section, we give the necessary background on linear-time temporal logic and finite-state automata.

### 2.1 Linear-time Temporal Logic

Given a set  $\Sigma$  of proposition letters, an LTL+P *formula*  $\phi$  is generated as follows:

$\phi := p \mid \neg p \mid \phi \vee \phi \mid \phi \wedge \phi$	Boolean connectives
$\mid X\phi \mid \tilde{X}\phi \mid \phi U \phi \mid \phi R \phi$	future modalities
$\mid Y\phi \mid \tilde{Y}\phi \mid \phi S \phi \mid \phi T \phi$	past modalities

where  $p \in \Sigma$ , and we call:  $X$ , *next*;  $\tilde{X}$ , *weak next*;  $U$ , *until*;  $R$ , *releases*;  $Y$ , *yesterday*;  $\tilde{Y}$ , *weak yesterday*;  $S$ , *since*;  $T$ , *triggers*. Note that, *w.l.o.g.*, our definition of LTL+P considers formulas already in Negation Normal Form (NNF), that is, negations are applied only to proposition letters. For any formula  $\phi$ , the *size* of  $\phi$  (denoted with  $|\phi|$ ) is the size of the (smallest) syntax tree of  $\phi$ .

Let  $\sigma \in (2^\Sigma)^+ \cup (2^\Sigma)^\omega$  be a *word over*  $2^\Sigma$  (or *trace over*  $2^\Sigma$ ). We define the *length* of  $\sigma$  as  $|\sigma| = n$ , if  $\sigma = \langle \sigma_0, \dots, \sigma_{n-1} \rangle \in (2^\Sigma)^+$  (in this case we say that  $\sigma$  is a *finite trace*); or  $|\sigma| = \omega$ , if  $\sigma \in (2^\Sigma)^\omega$  (in this case we say that  $\sigma$  is an *infinite trace*). We call any subset of  $(2^\Sigma)^*$  a *language of finite words over*  $2^\Sigma$ . Similarly, a *language of infinite words over*  $2^\Sigma$  is any subset of  $(2^\Sigma)^\omega$ .

The *satisfaction* of an LTL+P formula  $\phi$  by  $\sigma$  at time  $0 \leq i < |\sigma|$ , denoted by  $\sigma, i \models \phi$ , is defined as follows:

- $\sigma, i \models p$  iff  $p \in \sigma_i$ ;
- $\sigma, i \models \neg p$  iff  $p \notin \sigma_i$ ;
- $\sigma, i \models \phi_1 \vee \phi_2$  iff  $\sigma, i \models \phi_1$  or  $\sigma, i \models \phi_2$ ;
- $\sigma, i \models \phi_1 \wedge \phi_2$  iff  $\sigma, i \models \phi_1$  and  $\sigma, i \models \phi_2$ ;
- $\sigma, i \models X\phi$  iff  $i + 1 < |\sigma|$  and  $\sigma, i + 1 \models \phi$ ;
- $\sigma, i \models \tilde{X}\phi$  iff either  $i + 1 = |\sigma|$  or  $\sigma, i + 1 \models \phi$ ;
- $\sigma, i \models Y\phi$  iff  $i > 0$  and  $\sigma, i - 1 \models \phi$ ;
- $\sigma, i \models \tilde{Y}\phi$  iff either  $i = 0$  or  $\sigma, i - 1 \models \phi$ ;
- $\sigma, i \models \phi_1 U \phi_2$  iff there exists  $i \leq j < |\sigma|$  such that  $\sigma, j \models \phi_2$ , and  $\sigma, k \models \phi_1$  for all  $k$ , with  $i \leq k < j$ ;
- $\sigma, i \models \phi_1 S \phi_2$  iff there exists  $j \leq i$  such that  $\sigma, j \models \phi_2$ , and  $\sigma, k \models \phi_1$  for all  $k$ , with  $j < k \leq i$ ;
- $\sigma, i \models \phi_1 R \phi_2$  iff either  $\sigma, j \models \phi_2$  for all  $i \leq j < |\sigma|$ , or there exists  $i \leq k < |\sigma|$  such that  $\sigma, k \models \phi_1$  and  $\sigma, j \models \phi_2$  for all  $i \leq j \leq k$ ;
- $\sigma, i \models \phi_1 T \phi_2$  iff either  $\sigma, j \models \phi_2$  for all  $0 \leq j \leq i$ , or there exists  $k \leq i$  such that  $\sigma, k \models \phi_1$  and  $\sigma, j \models \phi_2$  for all  $i \geq j \geq k$ .

We say that  $\sigma$  is a *model* of  $\phi$  (written as  $\sigma \models \phi$ ) iff  $\sigma, 0 \models \phi$ . The *language of infinite* (resp., *finite*) *traces* of  $\phi$ , denoted by  $\mathcal{L}(\phi)$ , is the set of traces  $\sigma \in (2^\Sigma)^\omega$  (resp.,  $\sigma \in (2^\Sigma)^+$ ) such that  $\sigma \models \phi$ .

We use the standard shortcuts for  $\top := p \vee \neg p$ ,  $\perp := p \wedge \neg p$  (for some  $p \in \Sigma$ ) and other temporal operators:  $F\phi := \top \text{ U } \phi$  (*eventually*),  $G\phi := \perp \text{ R } \phi$  (*globally*),  $O\phi := \top \text{ S } \phi$  (*once*), and  $H\phi := \perp \text{ T } \phi$  (*historically*).

From now on, given a linear-time temporal logic  $\mathbb{L}$ , with some abuse of notation, we denote with  $\mathbb{L}$  also the set of formulas of  $\mathbb{L}$ . A *pure future* (resp., *pure past*) *formula* is an LTL+P formula without occurrences of past (resp., future) modalities. We denote by LTL (resp., pLTL) the set of pure future (resp., pure past) formulas. In the following, we use the subscript *f* to denote a logic interpreted on finite traces. Thus, e.g., with LTL<sub>f</sub> we denote LTL interpreted on finite traces. Note that, if  $\phi$  belongs to pLTL (*i.e.* pure past fragment of LTL+P), then we interpret  $\phi$  only on *finite words* and we say that  $\sigma \in (2^\Sigma)^+$  is a model of  $\phi$  if and only if  $\sigma, |\sigma| - 1 \models \phi$ , that is, each  $\phi$  in pLTL is interpreted at the *last* state of a finite word. It holds that LTL<sub>f</sub> and pLTL are expressively equivalent.

► **Proposition 1** (see [9, 16, 25]). *For any alphabet  $\Sigma$  and for any language  $\mathcal{L} \subseteq \Sigma^\omega$ , it holds that: there exists a formula  $\phi \in \text{LTL}_f$  such that  $\mathcal{L}(\phi) = \mathcal{L}$  iff there exists a formula  $\phi' \in \text{pLTL}$  such that  $\mathcal{L}(\phi') = \mathcal{L}$ .*

In the following, we denote by **Safety-LTL** (also called the syntactic safety fragment of LTL) the set of LTL formulas whose temporal operators are restricted to  $\tilde{X}$ ,  $G$ , and  $R$  [7, 21, 24]. Similarly, we define **coSafety-LTL** (the syntactic cosafety fragment of LTL) as the set of LTL formulas whose temporal operators are restricted to  $X$ ,  $F$ , and  $U$ . Finally, we denote by  $G(\text{pLTL})$  (resp.,  $F(\text{pLTL})$ ) the set of LTL+P formulas of the form  $G\alpha$  (resp.,  $F\alpha$ ), with  $\alpha \in \text{pLTL}$ . A fundamental theorem by Chang, Manna, and Pnueli [7], based on the results found by Zuck [25], establishes the expressive equivalence of **Safety-LTL** with  $G(\text{pLTL})$ , and of **coSafety-LTL** and  $F(\text{pLTL})$ , when interpreted over infinite traces.

We now define what it means, for two linear-time temporal logics  $\mathbb{L}$  and  $\mathbb{L}'$ , that  $\mathbb{L}$  *can be exponentially more succinct than*  $\mathbb{L}'$ . We use the  $\Omega$ -notation  $f(n) \in \Omega(g(n))$  to denote that the function  $f$  is asymptotically bounded from below by  $g$ . Similarly, we use the  $\mathcal{O}$ -notation  $f(n) \in \mathcal{O}(g(n))$  to denote that  $f$  is asymptotically bounded from above by  $g$ .

► **Definition 2.** *Given two linear-time temporal logics  $\mathbb{L}$  and  $\mathbb{L}'$ , we say that  $\mathbb{L}$  can be exponentially more succinct than  $\mathbb{L}'$  over infinite trace (resp., over finite traces) iff there exists an alphabet  $\Sigma$  and a family of languages  $\{\mathcal{L}_n\}_{n>0} \subseteq (2^\Sigma)^\omega$  (resp.,  $\{\mathcal{L}_n\}_{n>0} \subseteq (2^\Sigma)^*$ ) such that, for any  $n > 0$ :*

- *there exists a formula  $\phi \in \mathbb{L}$  over  $\Sigma$  such that its language over infinite traces (resp., over finite traces) is  $\mathcal{L}_n$  and  $|\phi| \in \mathcal{O}(n)$ ; and*
- *for all formulas  $\phi' \in \mathbb{L}'$  over  $\Sigma$ , if the language of  $\phi'$  over infinite traces (resp., finite traces) is  $\mathcal{L}_n$ , then  $|\phi'| \in 2^{\Omega(n)}$ .*

## 2.2 Finite-state Automata

A *Nondeterministic Finite Automaton* (NFA, for short) is a tuple  $A = (2^\Sigma, Q, I, \Delta, F)$ , where:  $2^\Sigma$  is a finite (nonempty) *alphabet*;  $Q$  is a finite set of *states*;  $I \subseteq Q$  is the set of *initial states*;  $\Delta \subseteq Q \times 2^\Sigma \times Q$  is the *transition relation*;  $F \subseteq Q$  is the set of *final states*. We define the *size* of  $A$ , denoted with  $|A|$ , as the number of its states ( $|Q|$ ).

A *run*  $\pi$  of  $A$  over the word  $\sigma = \langle \sigma_0, \sigma_1, \dots, \sigma_{n-1} \rangle \in (2^\Sigma)^*$  is a finite sequence of states  $\pi = \langle q_0, q_1, \dots, q_n \rangle$  such that  $(q_i, \sigma_i, q_{i+1}) \in \Delta$ , for all  $0 \leq i < n-1$ . A run  $\pi = \langle q_0, q_1, \dots, q_n \rangle$  is *accepting* if  $q_n$  is a final state of  $A$ , that is  $q_n \in F$ .

Given an NFA  $\mathcal{A} = (2^\Sigma, Q, I, \Delta, F)$ , a word  $\sigma \in (2^\Sigma)^*$  is *accepted by*  $\mathcal{A}$  iff there exists an accepting run of  $\mathcal{A}$  over  $\sigma$ . The *language of*  $\mathcal{A}$ , denoted with  $\mathcal{L}(\mathcal{A})$ , is the set (finite) words accepted by  $\mathcal{A}$ .

For each LTL<sub>f</sub>+P formula  $\phi$  of size  $n$  over the set of proposition letters  $\Sigma$ , we can effectively build an NFA whose language is exactly  $\mathcal{L}(\phi)$  and its size is at most exponential in  $n$  [11].

► **Proposition 3** (see [11]). *For any formula  $\phi$  of LTL<sub>f</sub>+P of size  $n$ , there exists an NFA  $\mathcal{A}$  such that  $\mathcal{L}(\phi) = \mathcal{L}(\mathcal{A})$  and  $|\mathcal{A}| \in 2^{\mathcal{O}(n)}$ .*

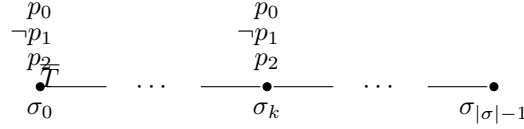
### 3 pLTL can be exponentially more succinct than LTL<sub>f</sub>

In this section, we prove the first main result of this paper, *i.e.* that pLTL can be exponentially more succinct than LTL<sub>f</sub>.

Let  $\Sigma = \{p_0, p_1, \dots, p_n\}$  be a finite set of proposition letters. Consider the following family of languages over the alphabet  $2^\Sigma$ , where  $n > 0$ .

$$A_n := \{\sigma \in (2^\Sigma)^+ \mid \exists k > 0 . (\bigwedge_{i=0}^n (p_i \in \sigma_k \leftrightarrow p_i \in \sigma_0))\} \quad (1)$$

For any  $n > 0$ , the language  $A_n$  is the set of finite words over  $2^\Sigma$  containing a position which agrees with the initial state on the evaluation of all proposition letters in  $\Sigma$  (cf. Figure 1).



■ **Figure 1** Example of a word  $\sigma$  in  $A_2$ .

We shall prove that all formulas of LTL<sub>f</sub> defining  $A_n$  are at least of size exponential in  $n$ . Conversely, as shown by the following lemma,  $A_n$  can be expressed in pLTL with formulas of linear size in  $n$ , for any  $n > 0$ .

► **Lemma 4.** *For any  $n > 0$ , there exists a formula  $\phi \in \text{pLTL}$  such that  $\mathcal{L}(\phi) = A_n$  and  $|\phi| \in \mathcal{O}(n)$ .*

**Proof.** For any  $n > 0$ , we define the formula  $\phi_{A_n}$  as

$$\mathcal{O}\left(\bigwedge_{i=0}^n (p_i \leftrightarrow \text{YO}(\tilde{\text{Y}}\perp \wedge p_i))\right)$$

Note the crucial role of the *weak yesterday* operator, and in particular of the subformula  $\tilde{\text{Y}}\perp$ , for hooking the initial state of a word. We prove that  $\mathcal{L}(\phi_{A_n}) = A_n$ . For any  $\sigma \in (2^\Sigma)^+$  and for any  $n > 0$ , it holds that  $\sigma \in A_n$  if and only if  $\exists k > 0 . \bigwedge_{i=1}^n (\sigma_k \models p_i \leftrightarrow \sigma_0 \models p_i)$ . This, in turn, is equivalent to  $\exists k < |\sigma| . (k \neq 0 \wedge \bigwedge_{i=1}^n (\sigma_k \models p_i \leftrightarrow \sigma_0 \models p_i))$  and thus to  $\exists k < |\sigma| . \bigwedge_{i=1}^n (\sigma_k \models p_i \leftrightarrow (\exists h . (h < k \wedge h = 0 \wedge \sigma_h \models p_i)))$ . Therefore,  $\sigma \models \mathcal{O}(\bigwedge_{i=0}^n (p_i \leftrightarrow \text{YO}(\tilde{\text{Y}}\perp \wedge p_i)))$ . Clearly,  $|\phi_{A_n}| \in \mathcal{O}(n)$ . ◀

To prove that  $A_n$  is not expressible in LTL<sub>f</sub> with formulas of size less than  $2^{\Omega(n)}$  (for any  $n > 0$ ), we make use of an auxiliary family of languages. For each  $n > 0$ , we define the language  $B_n$  over the alphabet  $2^\Sigma$  with  $\Sigma = \{p_0, p_1, \dots, p_n\}$  as follows:

$$B_n := \{\sigma \in (2^\Sigma)^+ \mid \exists h \geq 0 . \exists k > h . (\bigwedge_{i=0}^n (p_i \in \sigma_k \leftrightarrow p_i \in \sigma_h))\}$$

For any  $n > 0$ ,  $B_n$  is the set of finite words over  $2^\Sigma$  containing two (distinct) positions that agree on the interpretation of all the proposition letters in  $\Sigma$  (cf. Figure 4). Clearly,  $A_n \subseteq B_n$ , for any  $n > 0$ .



■ **Figure 2** Example of a word  $\sigma$  in  $B_2$ .

We now show that, if  $A_n$  was expressible in LTL<sub>f</sub> in space less than exponential in  $n$ , then the property  $B_n$  would be expressible in LTL<sub>f</sub> in space less than exponential as well.

► **Lemma 5.** *If there exists a formula of LTL<sub>f</sub> for  $A_n$  of size less than exponential in  $n$ , then there exists a formula of LTL<sub>f</sub> for  $B_n$  of size less than exponential in  $n$ .*

**Proof.** Let  $\psi_{A_n}$  be a formula of LTL<sub>f</sub> for  $A_n$  of size less than exponential in  $n$ . Consider the formula  $F(\psi_{A_n})$ : we prove that its language is exactly  $B_n$ . For any  $\sigma \in (2^\Sigma)^+$  and for any  $n > 0$ , it holds that  $\sigma \models F(\psi_{A_n})$  iff  $\exists k \geq 0 \cdot \sigma_{[k,-]} \models \psi_{A_n}$ , where  $\sigma_{[k,-]}$  is the suffix of  $\sigma$  starting from  $i$ . This means:  $\exists k \geq 0 \cdot \exists h > k \cdot (\bigwedge_{i=0}^n (\sigma_k \models p_i \leftrightarrow \sigma_h \models p_i))$ . Equivalently,  $\sigma \in B_n$ . Moreover  $F(\psi_{A_n})$  belongs to LTL<sub>f</sub> and it is of size less than exponential in  $n$ . ◀

We show that there cannot exist formulas of LTL<sub>f</sub> (and, in general, of LTL<sub>f</sub>+P) defining  $B_n$  whose size is less than exponential in  $n$ . In order to prove it, we first show that any NFA accepting  $B_n$  is of size at least *doubly exponential* in  $n$ .

► **Lemma 6.** *For any  $n > 0$  and for any NFA  $\mathcal{A}$  over the alphabet  $2^\Sigma$ , if  $\mathcal{L}(\mathcal{A}) = B_n$  then  $|\mathcal{A}| \in 2^{2^{\Omega(n)}}$ .*

**Proof.** Let  $n > 0$  and let  $\langle a_0, \dots, a_{2^n-1} \rangle$  be any permutation of the  $2^n$  subsets of  $\{p_1, \dots, p_n\}$  (note that this set does not include the proposition letter  $p_0 \in \Sigma$ ). Let  $K$  be any subset of  $\{0, \dots, 2^n - 1\}$  and let  $\bar{K}$  be the complement set of  $K$ . We define  $b_i^K$  in this way:  $b_i^K := a_i$ , if  $i \in \bar{K}$ ; and  $b_i^K := a_i \cup \{p_0\}$ , otherwise. We define  $\sigma_K$  as the sequence  $\langle b_0^K, b_1^K, \dots, b_{2^n-1}^K \rangle$ .

Suppose by contradiction that there exists an NFA  $\mathcal{A}$  for  $B_n$  of size less than doubly exponential in  $n$ . Consider the words  $\sigma_K \cdot \sigma_K$  (obtained by concatenating  $\sigma_K$  with itself),  $\sigma_{\bar{K}} \cdot \sigma_{\bar{K}}$  (the concatenation of  $\sigma_{\bar{K}}$  with itself), and  $\sigma_K \cdot \sigma_{\bar{K}}$  (the concatenation of  $\sigma_K$  with  $\sigma_{\bar{K}}$ ). By construction, both  $\sigma_K \cdot \sigma_K$  and  $\sigma_{\bar{K}} \cdot \sigma_{\bar{K}}$  contain (at least) two positions that agree on the interpretation of all symbols in  $\Sigma$  and thus they both belong to  $B_n$ , while  $\sigma_K \cdot \sigma_{\bar{K}}$  contains no such positions and so it does *not* belong to  $B_n$ . Therefore, for any  $K \subseteq \{0, \dots, 2^n - 1\}$ :

1.  $\sigma_K \cdot \sigma_K$  is accepted by  $\mathcal{A}$ ;
2.  $\sigma_{\bar{K}} \cdot \sigma_{\bar{K}}$  is accepted by  $\mathcal{A}$ ;
3.  $\sigma_K \cdot \sigma_{\bar{K}}$  is *not* accepted by  $\mathcal{A}$ .

Now let  $\pi$  (resp.,  $\pi'$ ) be any *accepting* run of  $\mathcal{A}$  over the word  $\sigma_K \cdot \sigma_K$  (resp.,  $\sigma_{\bar{K}} \cdot \sigma_{\bar{K}}$ ). Let  $q$  (resp.,  $q'$ ) be the  $2^n$ -th state of  $\pi$  (resp.,  $\pi'$ ). Suppose that  $q = q'$  and let  $\pi''$  be the sequence obtained by appending the suffix of  $\pi'$  starting from its  $2^n$ -th state to the prefix of  $\pi$  of length  $2^n - 1$ , i.e.:  $\pi'' := \langle \pi_0, \dots, \pi_{2^n-1}, \pi'_{2^n}, \pi'_{2^n+1}, \dots \rangle$ . By construction,  $\pi''$  is an *accepting run* of the automaton  $\mathcal{A}$  over the word  $\sigma_K \cdot \sigma_{\bar{K}}$ , which is a contradiction. Therefore, the  $2^n$ -th states of  $\pi$  and  $\pi'$  must be distinct. This means that the automaton  $\mathcal{A}$  has to contain at least a state for choice of  $K \subseteq \{0, \dots, 2^n - 1\}$ . Since there are  $2^{2^n}$  of such possible choices, this means that  $\mathcal{A}$  has to contain at least  $2^{2^{\Omega(n)}}$  states. ◀

By exploiting the singly exponential translation of  $\text{LTL}_f + \text{P}$  formulas into equivalent NFAs (Proposition 3), we can prove that (for any  $n > 0$ ) the language  $B_n$  is not expressible in  $\text{LTL}_f + \text{P}$  (and, in particular, in  $\text{LTL}_f$ ) in space less than exponential.

► **Lemma 7.** *For any formula  $\phi \in \text{LTL}_f + \text{P}$ , if  $\mathcal{L}(\phi) = B_n$  then  $|\phi| \in 2^{\Omega(n)}$ .*

**Proof.** Suppose by contradiction that this does not hold, *i.e.* there exists a formula  $\phi \in \text{LTL}_f + \text{P}$  such that  $\mathcal{L}(\phi) = B_n$  and  $|\phi|$  is less than exponential in  $n$ . Then, by Proposition 3, it holds that there exists an NFA  $\mathcal{A}$  such that  $\mathcal{L}(\mathcal{A}) = B_n$  and  $|\mathcal{A}|$  is *less than doubly exponential* in  $n$ , which is a contradiction with Lemma 6. ◀

Directly from Lemmas 5 and 7, it follows that the family of languages  $A_n$  cannot be expressed in  $\text{LTL}_f$  with formulas of size less than exponential in  $n$ .

► **Theorem 8.** *For any  $n > 0$  and for any formula  $\phi \in \text{LTL}_f$ , if  $\mathcal{L}(\phi) = A_n$  then  $|\phi| \in 2^{\Omega(n)}$ .*

The following corollary is a direct consequence of Lemma 4 and Theorem 8.

► **Corollary 9.** *pLTL can be exponentially more succinct than  $\text{LTL}_f$ .*

### Comparison with Markey's proof about $\text{LTL} + \text{P}$ and $\text{LTL}$

In [18], Markey proves that  $\text{LTL} + \text{P}$  can be exponentially more succinct than  $\text{LTL}$ . In particular, he exploits the result by Etesami, Vardi, and Wilke [13] that there are no Büchi automata of size less than doubly exponential for the family of languages  $I_n$  (for all  $n > 0$ ), defined as the language of *infinite* traces in which any two positions that agree on  $p_1, \dots, p_n$ , agree also on  $p_0$ .

One could, in principle, use  $I_n$  interpreted over finite trace (let us call it  $I_n^{<\omega}$ ) to prove that any NFA recognizing  $I_n^{<\omega}$  is at least of doubly exponential size in  $n$ , and use it as a base for proving that pLTL can be exponentially more succinct than  $\text{LTL}_f$ . This would require to restate and reprove the theorem by Etesami, Vardi, and Wilke [13] to work over finite traces. While we believe this is possible, we followed a simpler (and more useful) path by showing that there is another family of properties, in our case  $B_n$  (which is arguably simpler than  $I_n$  and  $I_n^{<\omega}$ ), for which each NFA explodes double-exponentially.

## 4 LTL<sub>f</sub> can be exponentially more succinct than pLTL

In this section, we show the second main result of this paper, *i.e.* that  $\text{LTL}_f$  can be exponentially more succinct than pLTL. Together with Corollary 9, this shows that  $\text{LTL}_f$  and pLTL, despite being expressively equivalent, are *incomparable* when succinctness is considered.

### 4.1 The Reverse Lemma

We first define the notions of *reverse language* and *reverse logic*. Given an alphabet  $\Sigma$  and a language  $\mathcal{L} \subseteq (2^\Sigma)^+$  of finite words over  $2^\Sigma$ , we define the *reverse language* of  $\mathcal{L}$  as the set:

$$\mathcal{L}^- = \{\sigma' \in (2^\Sigma)^+ \mid \sigma'_i = \sigma_{n-i}, \text{ for } \sigma = \sigma_0 \dots \sigma_n \in \mathcal{L} \text{ and } 0 \leq i \leq n\}.$$

We then define *reverse logics* as follows.

► **Definition 10 (Reverse Logics).** *Given two linear-time temporal logics  $\mathbb{L}$  and  $\mathbb{L}^-$ , we say that  $\mathbb{L}^-$  is a reverse logic of  $\mathbb{L}$  iff:*

1. *for any formula  $\phi \in \mathbb{L}$ , there is a formula  $\phi' \in \mathbb{L}^-$  so that  $\mathcal{L}(\phi) = \mathcal{L}(\phi')^-$  and  $|\phi'| = |\phi|$ ;*
2. *for any formula  $\phi' \in \mathbb{L}^-$ , there is a formula  $\phi \in \mathbb{L}$  so that  $\mathcal{L}(\phi') = \mathcal{L}(\phi)^-$  and  $|\phi| = |\phi'|$ .*



Clearly, being a reverse logic is a symmetric property:  $\mathbb{L}$  is a reverse logic of  $\mathbb{L}^-$  iff  $\mathbb{L}^-$  is a reverse logic of  $\mathbb{L}$ .

As an example, consider the logic pLTL and any formula  $\phi \in \text{pLTL}$ . By replacing in  $\phi$  the temporal operators  $Y, \tilde{Y}, S,$  and  $T$  with  $X, \tilde{X}, U,$  and  $R$ , respectively, one obtains a formula  $\phi'$  such that: (i) it belongs to LTL<sub>f</sub>; (ii) its size is  $|\phi|$ ; (iii) it is such that  $\mathcal{L}(\phi) = \mathcal{L}(\phi')^-$ . Therefore, LTL<sub>f</sub> is a reverse logic of pLTL, and *vice versa*.

The next lemma proves that, for any two linear-time temporal logics  $\mathbb{L}$  and  $\mathbb{L}^-$  such that  $\mathbb{L}$  is a reverse logic of  $\mathbb{L}^-$ , if a language  $\mathcal{L}$  with a compact definition in  $\mathbb{L}$  is not succinctly definable in  $\mathbb{L}^-$ , then  $\mathcal{L}^-$  (*i.e.*, the reverse language of  $\mathcal{L}$ ) is compactly definable in  $\mathbb{L}^-$ , but its definitions exponentially blow-up in  $\mathbb{L}$ .

► **Lemma 11 (Reverse Lemma).** *Let  $\mathbb{L}$  and  $\mathbb{L}^-$  be two linear-time temporal logics such that  $\mathbb{L}^-$  is a reverse logic of  $\mathbb{L}$ . Moreover, let  $\phi \in \mathbb{L}$  be such that, for every  $\psi \in \mathbb{L}^-$ ,  $\mathcal{L}(\psi) = \mathcal{L}(\phi)$  implies  $|\psi| \in 2^{\Omega(|\phi|)}$ . Then, for some  $\phi' \in \mathbb{L}^-$ , we have: (i)  $\mathcal{L}(\phi') = \mathcal{L}(\phi)^-$ ; (ii)  $|\phi'| = |\phi|$ ; and (iii) for every  $\psi \in \mathbb{L}$ ,  $\mathcal{L}(\psi) = \mathcal{L}(\phi')$  implies  $|\psi| \in 2^{\Omega(|\phi'|)}$ .*

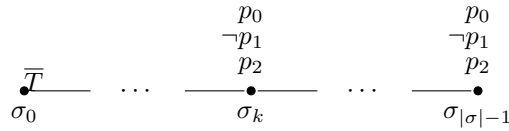
**Proof.** Suppose that  $\phi \in \mathbb{L}$  is a formula such that, for any  $\psi \in \mathbb{L}^-$ , if  $\mathcal{L}(\psi) = \mathcal{L}(\phi)$  then  $|\psi| \in 2^{\Omega(|\phi|)}$ . Now suppose by contradiction that for any formula  $\phi' \in \mathbb{L}'$ , such that  $\mathcal{L}(\phi') = \mathcal{L}(\phi)^-$  and  $|\phi'| = |\phi|$ , there exists a formula  $\psi \in \mathbb{L}$  such that  $\mathcal{L}(\psi) = \mathcal{L}(\phi')$  and  $|\psi|$  is sub-exponential in  $|\phi'|$ . Since  $\mathbb{L}^-$  is a reverse logic of  $\mathbb{L}$ , this means that for any formula  $\phi' \in \mathbb{L}^-$ , such that  $\mathcal{L}(\phi') = \mathcal{L}(\phi)^-$  and  $|\phi'| = |\phi|$ , there exists a formula  $\phi_R \in \mathbb{L}$  and a formula  $\psi_R \in \mathbb{L}^-$  such that:

- $\mathcal{L}(\phi_R)^- = \mathcal{L}(\phi')$  and thus  $\mathcal{L}(\phi_R) = \mathcal{L}(\phi)$ ;
- $\mathcal{L}(\psi_R)^- = \mathcal{L}(\phi')$  and thus  $\mathcal{L}(\psi_R) = \mathcal{L}(\phi)$ ;
- $|\phi_R| = |\phi'|$  and thus  $|\phi_R| = |\phi|$ ;
- $|\psi_R| = |\psi|$  and thus  $|\psi_R|$  is sub-exponential in  $|\phi_R|$ .

It follows that for any  $\phi_R \in \mathbb{L}$  there exists  $\psi_R \in \mathbb{L}'$  such that  $\mathcal{L}(\phi_R) = \mathcal{L}(\psi_R)$  and  $|\psi_R|$  is sub-exponential in  $|\phi_R|$ . But this is a contradiction with the hypothesis. Therefore, it has to hold that there exists a formula  $\phi' \in \mathbb{L}'$ , such that  $\mathcal{L}(\phi') = \mathcal{L}(\phi)^-$  and  $|\phi'| = |\phi|$  and, for all  $\psi \in \mathbb{L}$ , if  $\mathcal{L}(\psi) = \mathcal{L}(\phi')$  then  $|\psi| \in 2^{\Omega(|\phi'|)}$ . ◀

From Lemma 11, one obtains a concrete family of languages that are definable with LTL<sub>f</sub> formulas of polynomial size but such that any pLTL formula for them requires at least an exponential amount of space. In particular, for any  $n > 0$ , recall  $A_n$  from the previous section, and consider  $A_n^-$  (*cf.* Figure 3):

$$A_n^- := \{\sigma \in (2^\Sigma)^+ \mid \exists k < |\sigma| - 1 . (\bigwedge_{i=0}^n (p_i \in \sigma_k \leftrightarrow p_i \in \sigma_{|\sigma|-1}))\}$$



■ **Figure 3** Example of a word  $\sigma$  in  $A_2^-$ .

For each  $n > 0$ ,  $A_n^-$  can be expressed in LTL<sub>f</sub> in space linear in  $n$  with the formula

$$F\left(\bigwedge_{i=0}^n (p_i \leftrightarrow XF(\tilde{X}\perp \wedge p_i))\right).$$

However, since LTL<sub>f</sub> is a reverse logic of pLTL, by Lemma 11 every formula of pLTL for  $A_n^-$  requires an amount of space at least exponential in  $n$ . This leads directly to the following.



► **Theorem 12.** For any  $n > 0$  and for any formula  $\phi \in \text{pLTL}$ , if  $\mathcal{L}(\phi) = A_n^-$  then  $|\phi| \in 2^{\Omega(n)}$ .

► **Corollary 13.**  $\text{LTL}_f$  can be exponentially more succinct than  $\text{pLTL}$ .

## 4.2 Some meaningful implications of the incomparability

We have shown that the logics  $\text{LTL}_f$  and  $\text{pLTL}$ , despite being expressively equivalent (Proposition 1), are incomparable when succinctness is considered. Here below, we point out some implications of this incomparability that are worth discussing.

### Succinctness and Realizability

*Realizability* is the problem of establishing whether there is a strategy implementing a given formula. That is, given a formula  $\phi \in \text{LTL}_f$  (resp.,  $\phi \in \text{pLTL}$ ) over a set of variables  $\mathcal{C} \cup \mathcal{U}$  (with  $\mathcal{C}$  and  $\mathcal{U}$  sets of *controllable* and *uncontrollable* variables, respectively), the realizability problem of  $\text{LTL}_f$  (resp.,  $\text{pLTL}$ ) is the problem of establishing whether there exists a strategy  $s : (2^{\mathcal{U}})^+ \rightarrow 2^{\mathcal{C}}$  such that, for all sequences  $\langle \mathbf{U}_0, \mathbf{U}_1, \dots \rangle \in (2^{\mathcal{U}})^+$ , it holds that there exists  $k \in \mathbb{N}$  so that the prefix from 0 up to  $k$  of  $\langle \mathbf{U}_0 \cup s(\langle \mathbf{U}_0 \rangle), \mathbf{U}_1 \cup s(\langle \mathbf{U}_0, \mathbf{U}_1 \rangle), \dots \rangle$  is a model of  $\phi$ .

Despite having the same expressive power,  $\text{LTL}_f$  and  $\text{pLTL}$  have different complexity for the realizability problem: while  $\text{LTL}_f$  realizability is 2EXPTIME-complete [12],  $\text{pLTL}$  realizability is EXPTIME-complete [2]. This is due to the fact that, starting from any  $\text{LTL}_f$  formula  $\phi$  of size  $n$ , it is not possible to construct a *Deterministic Finite Automaton* (DFA) recognizing  $\mathcal{L}(\phi)$  of singly exponential size in  $n$ , whereas for  $\text{pLTL}$  formulas this is possible, thanks to the fact that “since past already happened”, there is no need to introduce nondeterminism [2, 8, 9].

In [2], the exponential gap between the two complexities, and the fact that  $\text{LTL}_f$  and  $\text{pLTL}$  are expressively equivalent, led to the conjecture that any translation from  $\text{LTL}_f$  formulas to equivalent  $\text{pLTL}$  ones requires at least an exponential blowup in the size of the resulting formulas. The results proved in this paper (in particular Theorem 12) *confirm* this conjecture: any translation in  $\text{pLTL}$  of the  $\text{LTL}_f$  formula  $F(\bigwedge_{i=0}^n (p_i \leftrightarrow \text{XF}(\bar{X}\perp \wedge p_i)))$ , which defines the language  $A_n^-$ , requires at least an exponential blowup.

### Succinctness helps in choosing the most convenient formalism for realizability

The succinctness results between  $\text{LTL}_f$  and  $\text{pLTL}$  can help in choosing the right formalism to express a property when the time complexity of realizability is considered. As a matter of fact, consider the family of languages  $A_n$  (Equation (1)), and suppose one wants to solve the realizability problem for  $A_n$ , for a given partition of the variables  $p_0, \dots, p_n$  into controllable and uncontrollable. There are two possibilities:

- (i) either formalize  $A_n$  in  $\text{pLTL}$  (in linear size) and use  $\text{pLTL}$  realizability algorithms (which are singly exponential in the worst case);
- (ii) or formalize the language in  $\text{LTL}_f$  (with at least an exponential blowup, by Theorem 8) and use  $\text{LTL}_f$  realizability algorithms (which are doubly exponential in the worst case).

While the former point requires only a *singly exponential* amount of time in the worst case, the latter requires a *triply exponential* amount of time, in the worst case. This shows how the results on the succinctness of  $\text{LTL}_f$  and  $\text{pLTL}$  can tremendously help choosing the best performing algorithm.

The fact that  $\text{LTL}_f$  can be exponentially more succinct than  $\text{pLTL}$  has an important implication as well. The realizability problem for the family of language  $A_n^-$  has the same worst-case time complexity (doubly exponential in  $n$ ) irrespectively of whether we choose to formalize  $A_n^-$  in  $\text{LTL}_f$  or we choose  $\text{pLTL}$  as the target formalism. In other words, the family of languages  $A_n^-$  cancels out the advantages of the past in realizability.

### Translation of LTL<sub>f</sub> into pLTL

Recall that LTL<sub>f</sub> and pLTL are expressively equivalent (Proposition 1). To the best of our knowledge, the most efficient translation of LTL<sub>f</sub> into pLTL is the one reported in [9] that performs the following steps:

1. build the corresponding NFA for the initial formula;
  2. determinize the NFA into a DFA;
  3. build a pLTL formula from the DFA using the Krohn-Rhodes cascaded decomposition [17];
- Since all three steps may introduce an exponential blow up in the worst case, the whole translation is *triply exponential* in the size of the initial formula. Maler and Pnueli prove that this translation (in particular the third step) has an exponential lower bound [17]. In this respect, Corollary 13 proves that *any* translation from LTL<sub>f</sub> to pLTL (not only the above one) has at least an exponential lower bound.

## 5 Succinctness of safety and cosafety fragments of LTL+P

In this section, we show our last main results, *i.e.* that, when interpreted over infinite traces, G(pLTL) can be exponentially more succinct than Safety-LTL, and that F(pLTL) can be exponentially more succinct than coSafety-LTL.

### 5.1 G(pLTL) can be exponentially more succinct than Safety-LTL

The proof of this case follows from the result by Markey that LTL+P can be exponentially more succinct than LTL, when interpreted over infinite traces [18]. In the following, we show the details of the proof.

Let  $\Sigma = \{p_0, \dots, p_n\}$  be a finite set of proposition symbols. Consider the family of languages  $M_n$  over the alphabet  $2^\Sigma$  proposed by Markey in [18]: for each  $n > 0$ ,  $M_n$  comprises all and only those infinite traces in which any position of the trace that agrees on  $p_1, \dots, p_n$  with the initial state also agrees on  $p_0$ . Formally, for each  $n > 0$ , we define:

$$M_n := \{\sigma \in (2^\Sigma)^\omega \mid \forall k > 0 (\forall i, 1 \leq i \leq n (p_i \in \sigma_k \leftrightarrow p_i \in \sigma_0) \leftrightarrow (p_0 \in \sigma_k \leftrightarrow p_0 \in \sigma_0))\}$$

In [18], Markey proves that, for any  $n > 0$ , any formula of LTL expressing  $M_n$  is at least of size exponential in  $n$ . Since Safety-LTL is a proper subfragment of LTL (*i.e.* each Safety-LTL formula is also an LTL formula), it follows that, for any  $n > 0$ , any formula of Safety-LTL expressing  $M_n$  is at least of size exponential in  $n$ .

► **Lemma 14.** *For any  $n > 0$  and any formula  $\phi \in \text{Safety-LTL}$ , if  $\mathcal{L}(\phi) = M_n$  then  $|\phi| \in 2^{\Omega(n)}$ .*

However, for each  $n > 0$ , there is a formula in G(pLTL) of size linear in  $n$  expressing  $M_n$ , such as the following:

$$G\left(\left(\bigwedge_{i=1}^n (p_i \leftrightarrow O(\tilde{Y} \perp \wedge p_i))\right) \leftrightarrow (p_0 \leftrightarrow O(\tilde{Y} \perp \wedge p_0))\right).$$

Note again the crucial role of the subformula  $\tilde{Y} \perp$  for hooking the initial state of the trace. This theorem directly follows.

► **Theorem 15.** *G(pLTL) can be exponentially more succinct than Safety-LTL.*

## 5.2 F(pLTL) can be exponentially more succinct than coSafety-LTL

We now dualize the previous result to the cosafety case, and obtain the succinctness lower bound of F(pLTL) with respect to coSafety-LTL. We first prove a more general result on “dual” temporal logics (which we define here below), and then we instantiate the result to the specific case of F(pLTL) and G(pLTL). We define *dual logics* as follows.

► **Definition 16** (Dual Logics). *Given two linear-time temporal logics  $\mathbb{L}$  and  $\overline{\mathbb{L}}$ , we say that  $\overline{\mathbb{L}}$  is a dual logic of  $\mathbb{L}$  iff:*

1. *for any formula  $\phi \in \overline{\mathbb{L}}$ , the transformation in negation normal form of  $\neg\phi$  (denoted as  $\text{nnf}(\neg\phi)$ ) belongs to  $\mathbb{L}$ , and*
2. *for any formula  $\phi \in \mathbb{L}$ , the transformation in negation normal form of  $\neg\phi$  (denoted as  $\text{nnf}(\neg\phi)$ ) belongs to  $\overline{\mathbb{L}}$ .*

As for the case of reverse logics, also being a dual logic is a symmetric property.

The following lemma proves that *duality* (as for Definition 16) preserves succinctness.

► **Lemma 17** (Duality Lemma). *For any linear-time temporal logics  $\mathbb{L}$  and  $\mathbb{L}'$ , if  $\mathbb{L}$  can be exponentially more succinct than  $\mathbb{L}'$ , then  $\overline{\mathbb{L}}$  can be exponentially more succinct than  $\overline{\mathbb{L}'}$ , where  $\overline{\mathbb{L}}$  (resp.,  $\overline{\mathbb{L}'}$ ) is a dual logic of  $\mathbb{L}$  (resp.,  $\mathbb{L}'$ ).*

**Proof.** Since by hypothesis  $\mathbb{L}$  can be exponentially more succinct than  $\mathbb{L}'$ , there exists a formula  $\phi \in \mathbb{L}$  of size  $n$  such that, for all  $\phi' \in \mathbb{L}'$ , if  $\mathcal{L}(\phi') = \mathcal{L}(\phi)$  then  $|\phi'| \in 2^{\Omega(n)}$ .

Let  $\overline{\phi}$  be the negation normal form of  $\neg\phi$ , i.e.  $\overline{\phi} = \text{nnf}(\neg\phi)$ . By definition:

1.  $\overline{\phi}$  belongs to  $\overline{\mathbb{L}}$ ;
2.  $\mathcal{L}(\overline{\phi}) = \mathcal{L}(\neg\phi)$ ; and
3.  $|\overline{\phi}| \in \mathcal{O}(n)$ .

Suppose by contradiction that the thesis does not hold, that is, for all formulas  $\overline{\psi} \in \overline{\mathbb{L}}$  of size  $s = |\overline{\psi}|$  there exists a formula  $\overline{\psi'} \in \overline{\mathbb{L}'}$  such that  $\mathcal{L}(\overline{\psi}) = \mathcal{L}(\overline{\psi'})$  and  $|\overline{\psi'}|$  is less than exponential in  $s$ . In particular, for  $\overline{\psi} := \overline{\phi}$ , this means that there exists a formula  $\overline{\psi'} \in \overline{\mathbb{L}'}$  such that  $\mathcal{L}(\overline{\psi'}) = \mathcal{L}(\overline{\phi})$  and  $|\overline{\psi'}|$  is less than exponential in  $n$  (recall that  $n$  is the size of  $\overline{\phi}$ ).

Now, let  $\chi'$  be the negation normal form of  $\neg\overline{\psi'}$  in negated normal form. It holds that:

1.  $\chi'$  is a formula in  $\mathbb{L}$ ;
2.  $\mathcal{L}(\chi') = \mathcal{L}(\phi)$ ; and
3.  $|\chi'| \in \mathcal{O}(|\overline{\psi'}|)$ .

Since the size of  $\overline{\psi'}$  is less than exponential in  $n$ , the size of  $|\chi'|$  is less than exponential in  $n$  as well. This means that  $\mathcal{L}(\phi)$  can be defined in  $\mathbb{L}'$  with a formula of size less than exponential in  $n$ , which is a contradiction with the hypothesis. ◀

Since, by definition, F(pLTL) and coSafety-LTL are dual logics of G(pLTL) and Safety-LTL, respectively, by Lemma 17 and Theorem 15, this result follows.

► **Theorem 18.** *F(pLTL) can be exponentially more succinct than coSafety-LTL.*

## 5.3 Open Problems

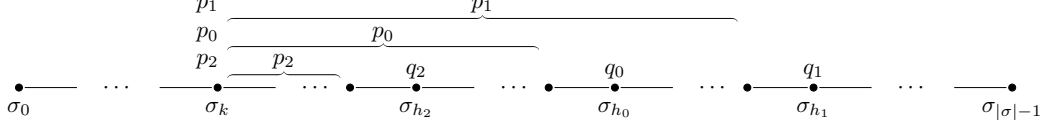
To complete the picture, we give a conjecture on the succinctness of the (co)safety fragments of LTL. To the best of our knowledge, it is still an open question whether coSafety-LTL (resp., Safety-LTL) can be exponentially more succinct than F(pLTL) (resp., G(pLTL)).

We conjecture that coSafety-LTL can be  $n!$  ( $n$  factorial) more succinct than F(pLTL). Let  $\Sigma = \{p_i\}_{i=1}^n \cup \{q_i\}_{i=1}^n$  be a finite alphabet. Consider the following family of languages  $C_n$  over the alphabet  $\Sigma$ , where  $n > 0$ :

$$C_n := \{\sigma \in (2^\Sigma)^\omega \mid \exists k \geq 0 . \bigwedge_{i=1}^n (\exists h > k . (q_i \in \sigma_h \wedge \forall k \leq l < h . p_i \in \sigma_l))\}.$$

## 2:12 LTL<sub>f</sub> Can Be Exponentially More Succinct Than pLTL, and *vice versa*

For any  $n > 0$ ,  $C_n$  comprises the infinite traces for which there exists a time point  $k$  such that, for each  $i \in \{1, \dots, n\}$ ,  $q_i$  will eventually be realized in the future of  $k$  and  $p_i$  holds until (and excluding) that point (cf. Figure 4).



■ **Figure 4** Example of a word  $\sigma$  in  $C_2$ .

While  $C_n$  is definable in coSafety-LTL with a formula of linear size in  $n$ , for example  $F(\bigwedge_{i=1}^n (p_i \cup q_i))$ , using F(pLTL) one is forced to enumerate all possible orders between  $q_1, \dots, q_n$  with a formula of this type:

$$F\left(\bigvee_{\pi \in \Pi} (q_{\pi(1)} \wedge Y(p_{\pi(1)}) S(p_{\pi(1)} \wedge q_{\pi(2)} \wedge Y(p_{\pi(1)} \wedge p_{\pi(2)}) S(p_{\pi(1)} \wedge p_{\pi(2)} \wedge q_{\pi(3)} \wedge \dots Y(\bigwedge_{i=1}^{n-1} p_{\pi(i)}) S(q_{\pi(n)} \wedge \bigwedge_{i=1}^n p_{\pi(i)})) \dots))\right)$$

where  $\Pi$  is the set of permutations of  $\{1, \dots, n\}$ . This, in turn, forces the formula to be at least of size  $n!$ .

► **Conjecture 19.** *For any  $n > 0$ , the language  $C_n$  is not expressible in F(pLTL) with a formula of size less than  $n!$ .*

We conjecture the same for dual case of Safety-LTL and G(pLTL).

## 6 Conclusions

We proved the incomparability between the succinctness of LTL<sub>f</sub> and of pLTL. We started by proving that the family of properties  $A_n$  admits a formalization in pLTL with formulas of linear size, while all formulas in LTL<sub>f</sub> for  $A_n$  are at least of exponential size. By using the Reverse Lemma, we derived that also the *vice versa* holds, that is, LTL<sub>f</sub> can be exponentially more succinct than pLTL. This result allowed us to confirm the conjecture left open in [2] about the lower bound for the complexity of translating LTL<sub>f</sub> into pLTL. We finally showed that G(pLTL) and F(pLTL) (*i.e.* the canonical forms of the safety and cosafety fragments of LTL) can be exponentially more succinct than Safety-LTL and coSafety-LTL, respectively.

The study of the maximal fragment of LTL<sub>f</sub> that does not incur in the exponential blow-up in the translation into pLTL is surely a problem worth studying, both for its theoretical implications and for its applications in reactive synthesis.

Proving Conjecture 19 is also an interesting future direction, which may require more sophisticated techniques for proving the lower bound, such as Ehrenfeucht-Fraïssé games [22] or Adler-Immermann games [1].

Finally, while we know that the lower bound between the translation of LTL<sub>f</sub> into pLTL is at least exponential, we have an upper bound which is triply exponential. The possibility of tighter lower bounds, or more efficient algorithms for this problem, is worth investigating.

---

**References**

---

- 1 Micah Adler and Neil Immerman. An  $n!$  lower bound on formula size. *ACM Transactions on Computational Logic (TOCL)*, 4(3):296–314, 2003.
- 2 Alessandro Artale, Luca Geatti, Nicola Gigante, Andrea Mazzullo, and Angelo Montanari. Complexity of safety and cosafety fragments of Linear Temporal Logic. In *Proc. of the 36th AAAI Conf. on Artificial Intelligence (AAAI-22)*. AAAI Press, 2023.
- 3 Howard Barringer, Michael Fisher, Dov Gabbay, Graham Gough, and Richard Owens. Metatem: A framework for programming in temporal logic. In *Stepwise Refinement of Distributed Systems Models, Formalisms, Correctness: REX Workshop, Mook, The Netherlands May 29–June 2, 1989 Proceedings*, pages 94–129. Springer, 1990.
- 4 Ronen I. Brafman and Giuseppe De Giacomo. Planning for LTLf/LDLf goals in non-markovian fully observable nondeterministic domains. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 1602–1608. ijcai.org, 2019.
- 5 Alberto Camacho, Jorge Baier, Christian Muise, and Sheila McIlraith. Finite ltl synthesis as planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, pages 29–38, 2018.
- 6 Alberto Camacho, Eleni Triantafyllou, Christian J. Muise, Jorge A. Baier, and Sheila A. McIlraith. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In Satinder Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3716–3724. AAAI Press, 2017.
- 7 Edward Y. Chang, Zohar Manna, and Amir Pnueli. Characterization of temporal property classes. In Werner Kuich, editor, *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, volume 623 of *Lecture Notes in Computer Science*, pages 474–486. Springer, 1992.
- 8 Alessandro Cimatti, Luca Geatti, Nicola Gigante, Angelo Montanari, and Stefano Tonetta. Extended bounded response LTL: a new safety fragment for efficient reactive synthesis. *Formal Methods in System Design*, pages 1–49, 2021.
- 9 Giuseppe De Giacomo, Antonio Di Stasio, Francesco Fuggitti, and Sasha Rubin. Pure-past linear temporal and dynamic logic on finite traces. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4959–4965, 2021.
- 10 Giuseppe De Giacomo, Antonio Di Stasio, Lucas M. Tabajara, Moshe Y. Vardi, and Shufang Zhu. Finite-trace and generalized-reactivity specifications in temporal synthesis. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1852–1858. ijcai.org, 2021.
- 11 Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 854–860. IJCAI/AAAI, 2013.
- 12 Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for LTL and LDL on finite traces. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1558–1564. AAAI Press, 2015.
- 13 Kousha Etessami, Moshe Y Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Information and computation*, 179(2):279–295, 2002.
- 14 Dov Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In *Temporal Logic in Specification: Altrincham, UK, April 8–10, 1987 Proceedings*, pages 409–448. Springer, 1989.
- 15 Lauri Hella and Miikka Vilander. Formula size games for modal logic and  $\mu$ -calculus. *J. Log. Comput.*, 29(8):1311–1344, 2019.
- 16 Orna Lichtenstein, Amir Pnueli, and Lenore Zuck. The glory of the past. In *Workshop on Logic of Programs*, pages 196–218. Springer, 1985.

- 17 Oded Maler and Amir Pnueli. Tight bounds on the complexity of cascaded decomposition of automata. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 672–682. IEEE, 1990.
- 18 Nicolas Markey. Temporal logic with past is exponentially more succinct. *Bull. EATCS*, 79:122–128, 2003.
- 19 Maja Pesic, Helen Schonenberg, and Wil MP Van der Aalst. Declare: Full support for loosely-structured processes. In *11th IEEE international enterprise distributed object computing conference (EDOC 2007)*, pages 287–287. IEEE, 2007.
- 20 Maja Pesic and Wil MP Van der Aalst. A declarative approach for flexible business processes management. In *Business Process Management Workshops: BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, Vienna, Austria, September 4-7, 2006. Proceedings 4*, pages 169–180. Springer, 2006.
- 21 A. Prasad Sistla. On characterization of safety and liveness properties in temporal logic. In *Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, pages 39–48, 1985.
- 22 Wolfgang Thomas. On the ehrenfeucht-fraïssé game in theoretical computer science. In *TAPSOFT'93: Theory and Practice of Software Development: 4th International Joint Conference CAAP/FASE Orsay, France, April 13–17, 1993 Proceedings*, pages 559–568. Springer, 2005.
- 23 Shufang Zhu, Giuseppe De Giacomo, Geguang Pu, and Moshe Y. Vardi. Ltlf synthesis with fairness and stability assumptions. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3088–3095. AAAI Press, 2020.
- 24 Shufang Zhu, Lucas M. Tabajara, Jianwen Li, Geguang Pu, and Moshe Y. Vardi. A Symbolic Approach to Safety LTL Synthesis. In Ofer Strichman and Rachel Tzoref-Brill, editors, *Proceedings of the 13th International Haifa Verification Conference*, volume 10629 of *Lecture Notes in Computer Science*, pages 147–162. Springer, 2017. doi:10.1007/978-3-319-70389-3\_10.
- 25 Lenore Zuck. Past temporal logic. *Weizmann Institute of Science*, 67, 1986.

# LSCPM: Communities in Massive Real-World Link Streams by Clique Percolation Method

Alexis Baudin  

Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Lionel Tabourier  

Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Clémence Magnien  

Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

---

## Abstract

Community detection is a popular approach to understand the organization of interactions in static networks. For that purpose, the Clique Percolation Method (CPM), which involves the percolation of  $k$ -cliques, is a well-studied technique that offers several advantages. Besides, studying interactions that occur over time is useful in various contexts, which can be modeled by the link stream formalism. The Dynamic Clique Percolation Method (DCPM) has been proposed for extending CPM to temporal networks.

However, existing implementations are unable to handle massive datasets. We present a novel algorithm that adapts CPM to link streams, which has the advantage that it allows us to speed up the computation time with respect to the existing DCPM method. We evaluate it experimentally on real datasets and show that it scales to massive link streams. For example, it allows to obtain a complete set of communities in under twenty-five minutes for a dataset with thirty million links, what the state of the art fails to achieve even after a week of computation. We further show that our method provides communities similar to DCPM, but slightly more aggregated. We exhibit the relevance of the obtained communities in real world cases, and show that they provide information on the importance of vertices in the link streams.

**2012 ACM Subject Classification** Theory of computation → Dynamic graph algorithms

**Keywords and phrases** Temporal network, Link stream,  $k$ -clique, Community detection, Clique Percolation Method, Real-world interactions

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.3

**Supplementary Material** *Software (Source code)*: <https://gitlab.lip6.fr/audin/lscpm>  
archived at `swh:1:dir:f8f7300a4825f880f58596790f80e24dce2b4d47`

**Funding** This work is funded by the ANR (French National Agency of Research) through the ANR FiT LabCom.

**Acknowledgements** The authors thank the SocioPatterns<sup>1</sup> and Icon<sup>2</sup> communities, for making their datasets available.

## 1 Introduction

Detecting communities in complex networks has been a focus of interest since the early years of the field to the point that even the number of surveys on the topic is large. While the first ones aimed at giving a general view of the landscape (*e.g.*, [9]), more recent ones tend to focus on a particular issue, for instance the underlying purpose of the community detection [24] or a specific family of networks [16].

---

<sup>1</sup> <http://www.sociopatterns.org>

<sup>2</sup> <https://icon.colorado.edu>



© Alexis Baudin, Lionel Tabourier, and Clémence Magnien;  
licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 3; pp. 3:1–3:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



The Clique Percolation Method (CPM) proposed by Palla *et al.* [20] is a well studied technique to detect communities in a graph. It is appreciated for the advantages that its definition confers: the communities are defined locally and in a deterministic way and there is no need to use heuristics or optimization functions that are hard to interpret. Also, it allows the communities to overlap each other, by contrast with most other techniques which lead to a partition of the nodes. It is a desirable property in general as the frontier between communities is often difficult to decide. While the early implementations were computationally costly, it was later improved [14, 22] and the most recent one scales up to graphs with hundreds of millions of nodes and edges [2].

Considering temporal networks, where the structure evolves dynamically, a standard approach consists in examining them as a sequence of snapshots and run graph community detection algorithms on each of them. Then, it makes sense to match the communities obtained from a time step to the next to obtain consistent groups through time [1]. Another approach following the same purpose is to design communities that ensure cohesive structure continuity over a time interval [25]. This strategy has been investigated for various applications, including mobile communication networks [18] or social networks analysis [23]. However, the instability of many community detection methods makes this task hard to achieve properly [6]. In addition, some works stress the importance of achieving online community detection, in which case the communities are updated at each time step, by aggregating new information to the existing communities [7, 21]. If the method is fast enough, it is possible to achieve a streaming community analysis of the data. However, this type of method often comes at the cost of losing part of the long term meaning that a community might have.

CPM can be implemented on graph snapshots and avoid instabilities from one step to the next due to its deterministic nature; it thus appears as a good candidate for the approach described above. It was in fact proposed quite early on, in [19]; we refer to this approach as the *Dynamic Clique Percolation Method* (DCPM). Recently, Boudebza *et al.* [4] introduced a faster algorithm to do this, called *Online Clique Percolation Method* (OCPM). However, describing a temporal network as a sequence of snapshots has shortcomings. Indeed, it misses the fact that the time step of analysis chosen is frequently arbitrary, while in general the data do not exhibit an obvious timescale of analysis. This is why formalisms to describe temporal networks have been developed to circumvent these limitations [5, 11, 15]. Among those, the link stream formalism stresses the symmetric roles of structure and time in the representation of data, and aim at describing temporal networks at the intersection of graph theory and time series analysis [15]. As the notion of clique has been recently extended to this formalism and the search for cliques is implemented efficiently [3, 28], it is now possible to investigate extensions of CPM to link streams. This makes possible to design faster algorithms than the state-of-the-art DCPM implementations. In this way, we obtain a community detection technique that can process data online and maintain relevant communities which naturally spread through time.

This is the main contribution of this paper: we propose an algorithm for this goal as well as an open source implementation that scales to large link streams<sup>3</sup>. In Section 2, we give the necessary background definitions and notations. Then, we describe our method in Section 3 and derive an expression of its theoretical complexity; note that this method uses a novel  $k$ -clique enumeration algorithm in link streams. Finally, we provide in Section 4 an extensive experimental investigation which shows its efficiency on several real-world instances, compares the obtained communities to the DCPM ones, and illustrates its relevance to draw information about the data examined.

---

<sup>3</sup> <https://gitlab.lip6.fr/baudin/lscpm>



## 2 Definitions and notations

First, we start with some basic reminders about (static) graphs. A graph is a pair  $G = (V, E)$ , where  $V$  is a set of vertices, and  $E$  is a set of edges of the form  $\{u, v\}$ , where  $u, v \in V$  and  $u \neq v$ . A  $k$ -clique in  $G$  is a set of  $k$  vertices that are all connected pairwise by an edge. Finally, the definition of CPM communities is given by the fact that two  $k$ -cliques are adjacent when they have  $k - 1$  vertices in common. Then, a CPM community of  $G$  is the set of vertices belonging to a maximal set of  $k$ -cliques that can be reached from one to another by a series of adjacent  $k$ -cliques (it forms a connected component of the graph whose vertices are the  $k$ -cliques of  $G$  and edges are defined by the adjacency relation just explained).

### 2.1 Cliques in link streams

In this article, we work with the link stream model, which represents interactions over time. Formally, a *link stream* is a triplet  $L = (T, V, E)$  where  $T$  is a time interval,  $V$  a set of vertices and  $E \subseteq T \times T \times V \times V$  a set of links  $(b, e, u, v)$  such that  $e \geq b$ ; we call  $e - b$  the duration of such a link. Throughout the paper, we consider link streams with no self-loop, i.e. for any link  $(b, e, u, v) \in E$ , then  $u \neq v$ . Moreover, links on the same vertices exist over disjoint time intervals, i.e. if  $(b, e, u, v), (b', e', u, v) \in E$ , with  $b \neq b'$  or  $e \neq e'$ , then  $[b, e] \cap [b', e'] = \emptyset$ .

We use the definition of a clique in a link stream which follows the one in [27], with a minor difference to avoid cliques over time intervals of null length: a *clique* is a pair  $(C, [t_0, t_1])$ , where  $C \subseteq V$ ,  $|C| \geq 2$  and  $t_0, t_1 \in T$ ,  $t_0 < t_1$ , such that for all  $u, v \in C$ ,  $u \neq v$ , there is a link  $(b, e, u, v)$  in  $E$  such that  $[t_0, t_1] \subseteq [b, e]$ . A *k-clique* is a clique containing  $k$  vertices. Notice that if  $(C, [t_0, t_1])$  is a  $k$ -clique, then  $(C, [t'_0, t'_1])$  is also a  $k$ -clique for all  $t'_0, t'_1$  such that  $t_0 \leq t'_0 < t'_1 \leq t_1$ . We are therefore interested in maximal  $k$ -cliques:

► **Definition 1** (maximal  $k$ -clique). *For  $k \in \llbracket 2, +\infty \rrbracket$ , a maximal  $k$ -clique is a clique  $(C, [t_0, t_1])$  having  $k$  vertices ( $|C| = k$ ), and such that its time interval is maximal: there is no  $t'_0 < t_0$  nor  $t'_1 > t_1$  such that  $(C, [t'_0, t_1])$  or  $(C, [t_0, t'_1])$  is a clique.*

With this definition, we can introduce the notion of  $k$ -clique adjacency, which will allow defining a generalization of CPM communities to link streams: two maximal  $k$ -cliques  $(C, [t_0, t_1])$  and  $(C', [t'_0, t'_1])$  are said to be *adjacent* if they share  $k - 1$  vertices and overlap over a time interval with strictly positive length, i.e. ,  $|C \cap C'| = k - 1$  and  $|[t_0, t_1] \cap [t'_0, t'_1]| > 0$ , where  $|I|$  denotes the length of interval  $I$ .

### 2.2 Communities in link streams

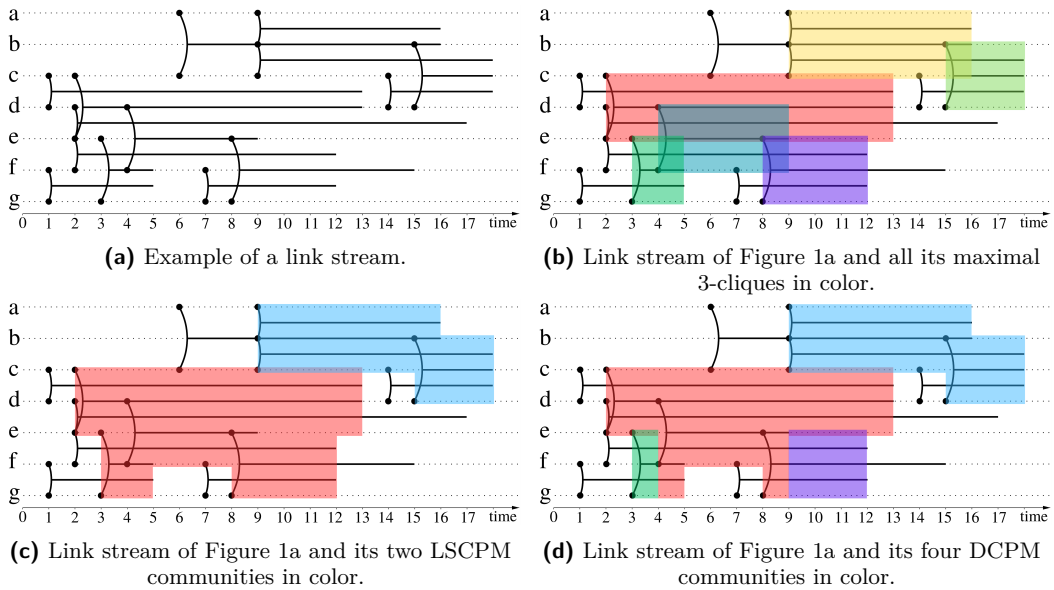
In a dynamical context, it is natural to define a *temporal community* as a set of *temporal vertices* of the form  $(u, I)$ , where  $u$  is a vertex, and  $I$  is a set of disjoint time intervals, which are the time intervals during which  $u$  is present in the community. Then, the notion of LSCPM community is similar to the one of CPM community in graphs, but with the notion of maximal  $k$ -clique adjacency adapted to link streams:

► **Definition 2** (LSCPM community). *A LSCPM community is a temporal community whose temporal vertices belong to a maximal set of  $k$ -cliques that can be reached from one to another by a series of adjacent  $k$ -cliques.*

A few observations can be made about this definition. First, as  $k$  increases, the communities may only split and/or lose temporal vertices. In other words, if  $k_1 < k_2$ , each community computed with  $k_2$  is included in a community computed with  $k_1$ . This property is illustrated and further discussed in Section 4.5.

Second, let us recall the definition of the dynamic CPM (DCPM) communities introduced in [19]. The idea is to compute the CPM communities at each snapshot. Then, comparing the communities obtained for two consecutive snapshots allows finding whether each community evolves by gaining vertices, losing vertices, dying (disappearing or merging with a larger one), or being born (appearing or being detached from a larger one). Notice that DCPM communities can be considered as temporal communities: given two consecutive snapshots at  $t_i$  and  $t_{i+1}$ , a vertex of a CPM community at  $t_i$  belongs to the DCPM community on  $[t_i, t_{i+1}[$ . Most importantly, a LSCPM community is a union of DCPM communities. Indeed, the temporal vertices of a CPM community in a snapshot are all included in a same LSCPM community; and a CPM community that gains or loses vertices from one snapshot to the next remain included in the same LSCPM community. Note that two DCPM communities that are merged (resp. split) in the next snapshot belong to the same LSCPM community.

To illustrate these definitions, we show in Figure 1a an example of a link stream, with time on the X-axis and vertices on the Y-axis. Links are represented as connections between two vertices, and the horizontal line indicate their duration. For example, there is a link between vertices  $c$  and  $d$  over the time interval  $[1, 13]$ . In Figure 1b, we represent its maximal  $k$ -cliques in color, in Figure 1c its LSCPM communities, and in Figure 1d its DCPM communities. The background of each vertex is colored according to the time during which it belongs to its clique or community. Notice that the red LSCPM community is composed of three DCPM communities: the red one, but also the green one because  $\{e, f, g\}$  is not adjacent to  $\{c, d, e\}$  at time  $t = 3$ ; and the purple one because  $\{e, f, g\}$  is no longer adjacent to  $\{d, e, f\}$  for  $t \geq 9$ .



■ **Figure 1** Example of a link stream with its maximal  $k$ -cliques for  $k = 3$  and the associated LSCPM communities and DCPM communities.

### 3 Algorithms

Our main idea to compute efficiently LSCPM communities in link streams is to use techniques similar to those developed for enumerating maximal cliques in link streams [3, 28]. Indeed, it is possible to enumerate  $k$ -cliques in a link stream efficiently by going through each link

only once and aggregate adjacent ones using a temporal Union-Find data structure. In the DCPM case however, the comparison of communities at each time step to detect their evolution is more demanding. We present an algorithm to efficiently enumerate  $k$ -cliques in Section 3.1. We then adapt the best existing method for computing CPM communities, and propose an efficient algorithm that, given as input the  $k$ -cliques of a link stream, computes its communities, in Section 3.2.

### 3.1 $k$ -clique enumeration algorithm in link streams

We take inspiration from our recent work [3] which enumerates the maximal cliques to design the algorithm for enumerating  $k$ -cliques in a link stream. The key idea is to use graph  $k$ -clique enumeration, as very efficient algorithms have been designed for this task [8]. Then, we compute the starting and ending time of each induced maximal  $k$ -clique in the link stream.

■ **Algorithm 1**  $k$ -clique enumeration in link streams.

---

**Input:** Link stream  $L = (T, V, E)$ ;  $k \in \llbracket 3, +\infty \rrbracket$ .  
**Output:** All  $k$ -cliques of  $L$  without duplicates.

```

1  $G \leftarrow$  empty graph
2  $\mathcal{E} \leftarrow$  empty associative array //  $\mathcal{E}$  associates ending times to edges
3 for  $(b, e, u, v) \in E$  sorted by increasing  $b$  do
4   Add edge  $\{u, v\}$  to  $G$ 
5    $\mathcal{E}(u, v) \leftarrow e$  // Record the ending time of  $\{u, v\}$ 
6   Remove from  $G$  all edges  $\{x, y\}$  with  $\mathcal{E}(x, y) < b$ 
7    $G\text{Cliques} \leftarrow$  all  $k$ -cliques of  $G$  containing  $u$  and  $v$ 
8   for  $C \in G\text{Cliques}$  do
9      $end \leftarrow \min_{x, y \in C} (\mathcal{E}(x, y))$ 
10    output  $k$ -clique  $(C, [b, end])$ 

```

---

Algorithm 1 lists all the maximal  $k$ -cliques. It starts from an empty graph (Line 1), and processes the link stream in chronological order (Line 3). For each link  $(b, e, u, v)$ , it updates the graph (Lines 4 to 6). Then, it enumerates the maximal  $k$ -cliques containing  $u$  and  $v$  induced by the links seen up till then. They match the static  $k$ -cliques in  $G$  that contain  $u$  and  $v$  (Line 7). Finally, each of these maximal  $k$ -clique starts at  $b$ , because the link between  $u$  and  $v$  does not exist before  $b$ , and lasts as long as all its links exist, so its ending time is the minimum of the ending times of the edges composing it (Line 9).

The complexity of Algorithm 1 is given by Theorem 3. Its proof is given in Appendix.

► **Theorem 3** ( $k$ -clique enumeration time complexity). *Algorithm 1 enumerates all  $k$ -cliques of the input link stream in time  $\mathcal{O}\left(m \cdot k^3 \cdot \left(\frac{d}{2}\right)^{k-2} + m \cdot d^2 + m \cdot \log(m)\right)$ , where  $m$  is the number of links and  $d$  the maximal degree of a node, that is its maximal number of neighbors at any given time.*

The largest factor in this complexity is  $m$ .  $d$  can in theory be large with respect to  $m$  but, since it is the maximal number of neighbors of a node at any given time, it is small in practice. It is therefore the value of  $k$  that determines how efficient the enumeration can be, as the factors  $k^3$  and  $\left(\frac{d}{2}\right)^{k-2}$  show that this method remains efficient only for small values of  $k$ .

### 3.2 LSCPM: CPM algorithm in link streams

To the best of our knowledge, the most efficient algorithmic implementation of CPM in graphs is [2]. In a few words, this algorithm stores each CPM community as the set of the  $(k - 1)$ -cliques composing its  $k$ -cliques. It builds these sets on the fly, by processing each  $k$ -clique one by one, testing to which community each of its  $(k - 1)$ -cliques currently belongs. Then, it merges these communities or creates a new one if needed. For this purpose, the algorithm uses a Union-Find data structure, as it is efficient to do these operations. It is a forest of trees, where each node corresponds to a  $(k - 1)$ -clique, and each tree corresponds to a CPM community, identified by its root. It has three intrinsic functions: `UF.Find(id)` that returns the root of the tree containing the node  $id$ , `UF.Union(p, q)` that performs the union of two trees by connecting their roots and returns the root of the new tree, and `UF.MakeSet()` which creates a new tree on a new root  $q$ , and returns this root.

We take inspiration from the algorithm above to extend the percolation of  $k$ -cliques to its definition in link streams. The procedure is given in Algorithm 2 and follows a similar logic: each LSCPM community is stored as the set of the temporal  $(k - 1)$ -cliques of its maximal  $k$ -cliques. A  $(k - 1)$ -clique of a maximal  $k$ -clique  $(C_k, [t_0, t_1])$  is of the form  $(C_{k-1}, [t_0, t_1])$ , with  $C_{k-1} \subseteq C_k$  containing  $k - 1$  vertices. These communities are constructed on the fly, by processing each maximal  $k$ -clique one by one, following the chronological order of their starting time, given by Algorithm 1. For each maximal  $k$ -clique  $(C_k, [t_0, t_1])$  (Line 5), Algorithm 2 checks the community to which each of its  $(k - 1)$ -clique belongs on an interval that (strictly) intersects  $[t_0, t_1]$  if it exists (Line 10), or creates a new one if needed (Line 16), then merges them (Lines 13 and 17). It also extends the membership duration of the  $(k - 1)$ -cliques in case they did not belong to this community until  $t_1$  (Line 11). To do so, in addition to the Union-Find structure `UF`, we use an associative array `TimeUF`, which associates each  $C_{k-1}$  to a list of elements of the form  $(id, [t_0, t_1])$ , where  $id$  is the identifier of a Union-Find element and  $[t_0, t_1]$  is the interval during which  $C_{k-1}$  belongs to the community of  $id$ . In these lists, the intervals are disjoint, and the pairs are sorted in ascending chronological order. Each list is initialized to  $[(-1, -1, -1)]$  (Lines 3 and 4), meaning that the corresponding  $C_{k-1}$  has not yet been added to any community. Figure 2 gives an example of the update of `TimeUF` and `UF` structures, when applying Algorithm 2 to the link stream of Figure 1.

Finally, we need to transform the Union-Find structure into the adequate format to get the output as a temporal community. This is done with a loop through the elements of `TimeUF`. Each one is of the form  $C_{k-1} \rightarrow I$ , where  $C_{k-1}$  is a set of  $k - 1$  vertices, and  $I$  is a list of pairs  $(id, [t_0, t_1])$  corresponding to a Union-Find element and a time interval. Each Union-Find element belongs to a single set, which is one of the LSCPM communities and that we obtain with the `Find` procedure. We then add each vertex of  $C_{k-1}$  to this community, on the time interval  $[t_0, t_1]$ .

The complexity of Algorithm 2 is given by Theorem 4 (demonstrated in Appendix). Note that Algorithm 2 takes as input the set of  $(k - 1)$ - and  $k$ -cliques of the link stream. Therefore, the total complexity given in Theorem 4 takes into account the time needed to perform their enumeration as well as the time needed to compute the communities.

► **Theorem 4** (LSCPM time complexity). *The time complexity of Algorithm 2 is in  $\mathcal{O}((k + \alpha(n_k)) \cdot k \cdot n_k + c(k))$ , where  $\alpha$  is the inverse Ackermann function,  $n_k$  the number of  $k$ -cliques of the link stream, and  $c(k)$  the complexity of enumerating  $k$ -cliques, given in Theorem 3. It is thus in  $\mathcal{O}\left((k + \alpha(n_k)) \cdot m \cdot k^2 \cdot \left(\frac{d}{2}\right)^{k-2} + m \cdot d^2 + m \cdot \log(m)\right)$ .*

This complexity is expressed with the inverse Ackermann function  $\alpha$ , which is known to grow extremely slowly, and can be considered as a constant at the scale of our data. Thus, we see from this theorem that the time complexity is close to  $\mathcal{O}(k^2 \cdot n_k + c(k))$ . This shows

■ **Algorithm 2** Clique Percolation Method in link streams (LSCPM).

---

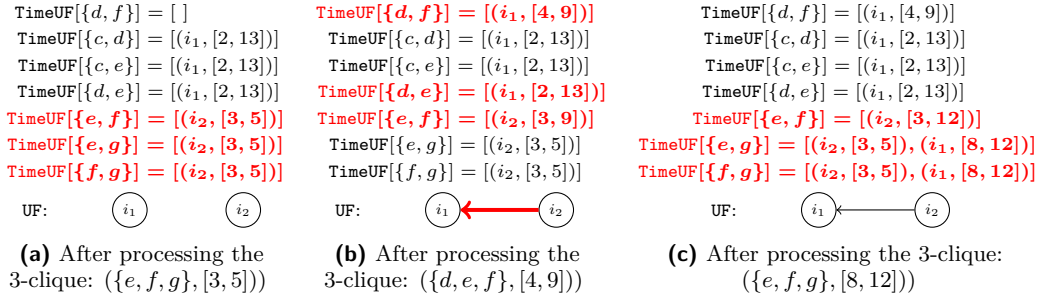
**Input:**  $(k - 1)$ -cliques then  $k$ -cliques of a link stream  $L = (T, V, E)$ ;  $k \in \llbracket 3, +\infty \rrbracket$ .  
**Output:** Union-Find structure representing all LSCPM communities of  $L$ .

```

1 UF ← empty Union-Find data structure
2 TimeUF ← empty associative array
3 for each maximal  $(k - 1)$ -clique  $(C_{k-1}, [t_0, t_1])$  of  $L$  do
4   TimeUF[ $C_{k-1}$ ] ←  $[(-1, -1, -1)]$ 
5 for each maximal  $k$ -clique  $(C_k, [t_0, t_1])$  of  $L$ , sorted by increasing  $t_0$  do
6    $p \leftarrow -1$ 
7   for each  $u \in C_k$  do
8      $C_{k-1} \leftarrow C_k \setminus \{u\}$ 
9      $(id, [t'_0, t'_1]) \leftarrow$  last element of TimeUF[ $C_{k-1}$ ]
10    if  $t_0 < t'_1$  then // is in the current community
11      last element of TimeUF[ $C_{k-1}$ ] ←  $(id, [t'_0, \max(t_1, t'_1)])$ 
12       $q \leftarrow$  UF.Find( $id$ )
13       $p \leftarrow$  UF.Union( $p, q$ ) // merge with other  $(k - 1)$ -cliques
14    else // not yet or no longer in the community
15      if  $p = -1$  then
16         $p \leftarrow$  UF.MakeSet()
17      Append  $(p, [t_0, t_1])$  to TimeUF[ $C_{k-1}$ ] // add to community of  $p$ 

```

---



■ **Figure 2** Example of updates of TimeUF and UF of Algorithm 2, during the processing of the second, third and fourth 3-cliques of the link stream of Figure 1. Note that all lists in TimeUF begin by a  $(-1, -1, -1)$  triplet which we omit for readability. We show only the part of TimeUF relevant to the cliques under study. At each time step, the  $(k - 1)$ -cliques corresponding to the added clique are shown in red. In 2a, three  $(k - 1)$ -cliques are added to the structure. In 2b, communities of  $i_1$  and  $i_2$  are merged, as the  $k$ -clique contains one  $(k - 1)$ -clique in  $i_1$  and another in  $i_2$ ; also one  $(k - 1)$ -clique is added:  $\{d, f\}$ , one is extended in time:  $\{e, f\}$ , and one remains unchanged because it is already present in the community over a longer time interval:  $\{d, e\}$ . In 2c, one  $(k - 1)$ -clique is extended in time:  $\{e, f\}$ , and as the other two,  $\{e, g\}$  and  $\{f, g\}$ , were not in the community any more, they are re-added over the time interval of the  $k$ -clique  $[8, 12]$ . At the end of the process, the structure matches the information represented by the red LSCPM community of Figure 1.

that our algorithm is efficient in the way each  $k$ -clique is processed, once the  $k$ -cliques have been computed. Indeed, each  $k$ -clique contains  $k$   $(k - 1)$ -cliques, and therefore it is not possible to process them in less than  $k \cdot n_k$  operations, using an approach similar to ours.

The second part of the theorem is obtained by replacing  $c(k)$  by the expression of Theorem 3 and  $n_k$  by a bound on its value. If we do not take into account the factor  $\alpha(n_k)$ , the complexity is in  $\mathcal{O}\left(k^3 \cdot m \cdot \left(\frac{d}{2}\right)^{k-2} + m \cdot d^2 + m \cdot \log(m)\right)$ . As noted above, this complexity depends almost linearly on  $m$  and the factor  $d$  is small in practice. Nevertheless, the cubic factor in  $k$  and the  $\left(\frac{d}{2}\right)^{k-2}$  make it manageable only for small values of  $k$ . We will see in Section 4 that small values of  $k$  allow for a fast building of the communities and are sufficient to observe interesting properties of the datasets.

In practice, Algorithm 2 needs to store in memory all the  $(k - 1)$ -cliques in the link stream. It can be limiting, for example if the input dataset contains a very large clique. For instance, if there is a clique of size 1000, and that we are looking for 6-clique communities, then there are more than  $10^{15}$  5-cliques to store from this large clique. Still, data from real-world interactions are known not to exhibit many large cliques, which makes the  $k$ -clique approach interesting for their study, and this memory feature has not been prohibitive during our experiments.

## 4 Experiments

For the experimental study, we implemented our algorithm in Python and the code is available online<sup>4</sup>. Throughout this section, we set  $k = 3$  unless otherwise specified. We will see that this value allows for a fast computation while being sufficient to provide interesting information on the datasets. Also, we present in Section 4.5 the impact of increasing  $k$  on the community structures, which induces smaller communities and therefore allows targeting their core, with more or less strength depending on the value by which  $k$  has been increased.

### 4.1 Datasets

We run our experiments on real-world link streams of various sizes and types of interactions. Even though many datasets consist of links with duration, in many cases these data are registered with regular discrete time intervals, because of the practical data acquisition protocol. This is the case for instance of proximity between individuals data, usually captured using RFID tags. Therefore, currently there is a larger range of publicly available instantaneous link streams with links of the form  $(t, u, v)$ , where  $u$  and  $v$  are vertices interacting at time  $t$ . So, we transform these link streams by adding a duration  $\Delta$  creating links of the form  $(t, t + \Delta, u, v)$ . Note that the value of  $\Delta$  will impact the number and extension of cliques in the link stream. Practically, we choose uniform  $\Delta$  values which are consistent with the typical time scales of the interactions considered for the datasets under study. These values, while consistent, remain arbitrary, and we use them to demonstrate the efficiency and relevance of our algorithm. Users can adjust the values according to their requirements and to the nature of the studied datasets.

The datasets on which we performed the experiments are described in Table 1. *Households* is a link stream representing contacts between members of five households in rural Kenya in 2012 [13]; *Highschool* corresponds to contacts between students of five classes of a preparatory school in Marseilles (France) during one week in 2012 [10] and *Infectious* consists of contacts between visitors to a museum in Dublin (Ireland) in 2009 [12]. These three datasets represent contacts between individuals, for which we have chosen to take a link duration  $\Delta = 1$  hour. The *Foursquare* dataset is extracted from the eponymous application, where users check-in in

<sup>4</sup> <https://gitlab.lip6.fr/baudin/lscpm>

venues that they visit, located in New-York City in our case [29]. It can thus be represented as a bipartite link stream between visitors and locations, where timestamps correspond to the check-in time. In this link stream, we set  $\Delta = 6$  hours, then we project it on the set of locations: if a user is connected to two locations over an overlapping time interval, this will create a link between these locations during the overlap. If the time interval of two links created in this way overlap, they are merged into a single link over the union of the initial time intervals. Finally, we use the link stream *Wikipedia*, which represents links between Wikipedia pages, timestamped by the time of the link creation, over several years in the 2000s [17]. We choose a  $\Delta = 1$  week duration, essentially to explore how our method scales to massive link streams.

■ **Table 1** Link stream datasets.  $\Delta$  is the link duration,  $m$  the number of links,  $d$  the maximal degree,  $n$  the number of nodes,  $D$  the total duration from the first to the last link and  $r$  the time resolution, that is the smallest duration between the beginning of two links.

Link stream	$\Delta$	$m$	$n$	$d$	$D$	$r$
<i>Households</i>	1 hour	2,136	75	19	3 days	1 hour
<i>Highschool</i>	1 hour	5,528	180	18	8 days	20s
<i>Infectious</i>	1 hour	44,658	10,972	43	3 months	5 min
<i>Foursquare</i>	6 hours	268,472	33,153	81	10 months	15 min
<i>Wikipedia</i>	1 week	38,953,380	1,870,709	33,217	2.3 years	20s

The link stream parameter that affects most dramatically the running time of the community detection is its number of  $k$ -cliques  $n_k$ , as the complexity depends strongly on  $k$  and  $n_k$  according to Theorem 4. In Table 2, we report the number of  $k$ -cliques for each dataset, for  $k$  ranging from 3 to 7. It allows anticipating the differences in computation time between the datasets, which are detailed in Section 4.2. We notice that for large datasets,  $n_k$  increases with  $k$ , certainly because these datasets contain some large cliques. Indeed, within a clique containing  $c$  vertices, there are  $\binom{c}{k}$  cliques with  $k$  vertices, and that quantity grows with  $k$  (as long as  $k \leq \frac{c}{2}$ ). In particular, *Foursquare* is a projection of a bipartite network, and projections are known to contain many large cliques.

■ **Table 2** Number of  $k$ -cliques ( $n_k$ ) for each dataset and for  $k$  from 3 to 7.

Link stream	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$
<i>Households</i>	3,951	4,721	3,929	2,324	987
<i>Highschool</i>	2,468	583	97	11	1
<i>Infectious</i>	79,836	128,157	202,181	274,181	300,850
<i>Foursquare</i>	571,768	2,423,011	17,823,050	155,466,085	1,302,290,726
<i>Wikipedia</i>	3,757,877	1,148,832	1,763,386	4,545,105	11,853,134

## 4.2 LSCPM: faster and scaling to massive real-world link streams

We now compare our algorithm to the DCPM one in terms of running time. Note that the comparison focuses on the running time and not on the complexity as the complexity of the DCPM method or the existing OCPM implementation are not given by their authors. In our implementation, the  $k$ -cliques are streamed to the standard input of the LSCPM algorithm, which reads them as the enumeration proceeds. These two operations are done on two different threads; but for comparison purposes with the DCPM running time, we

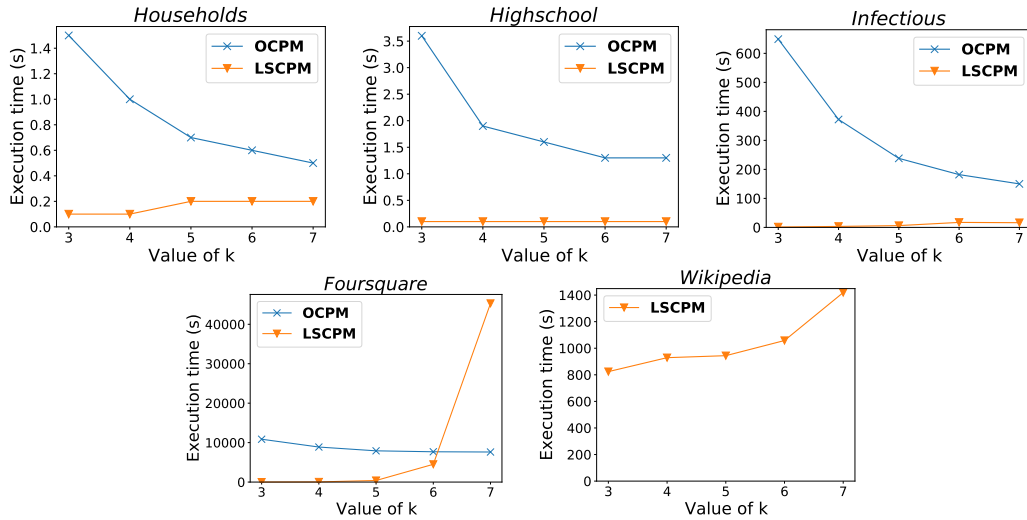


measure its computation time as the sum of the time spent on each of these two threads. The DCPM runtime is provided by the best implementation available [4]. We refer to this implementation as OCPM (standing for Online CPM). We performed all the experiments on a Linux machine, equipped with two processors Intel Xeon Silver 4210R with twenty cores each, at 2.40Ghz, and with 252Gb of RAM.

Table 3 presents the computation times of communities with the OCPM implementation of DCPM and our LSCPM implementation, on all the datasets of Table 1, for  $k$  from 3 to 7. These values are also plotted in Figure 3 for readability purposes. We observe that LSCPM is significantly faster, particularly on more massive datasets. For example, with  $k = 3$  or  $k = 4$ , it takes a few seconds with LSCPM to compute the *Foursquare* communities, while it takes a few hours with OCPM. Moreover, for the massive *Wikipedia* link stream, OCPM is unable to compute the set of communities in a week, while our algorithm provides the communities in less than 30 minutes for all  $k$  values tested. Our algorithm thus allows to study a community structure in massive datasets for which the state of the art does not provide a result.

■ **Table 3** Time computation of communities in seconds with both OCPM and LSCPM, for all our datasets, with  $k$  varying from 3 to 7. The symbol “-” means that the computation time exceeds one week.

Link stream	$k = 3$		$k = 4$		$k = 5$		$k = 6$		$k = 7$	
	OCPM	LSCPM	OCPM	LSCPM	OCPM	LSCPM	OCPM	LSCPM	OCPM	LSCPM
<i>Households</i>	1.5s	0.1s	1.0s	0.1s	0.7s	0.2s	0.6s	0.2s	0.5s	0.2s
<i>Highschool</i>	3.6s	0.1s	1.9s	0.1s	1.6s	0.1s	1.3s	0.1s	1.3s	0.1s
<i>Infectious</i>	10min49s	1.4s	6min12s	3.3s	3min58s	6.2s	3min02s	17.2s	2min30s	16.2s
<i>Foursquare</i>	3h01min	9.2s	2h28min	43s	2h12min	6min39s	2h08min	1h15mins	2h07min	12h35min
<i>Wikipedia</i>	-	13min44s	-	15min29s	-	15min44s	-	17min38s	-	23min39s



■ **Figure 3** Line charts of the running times of OCPM and LSCPM for each dataset. Values are those in Table 3.

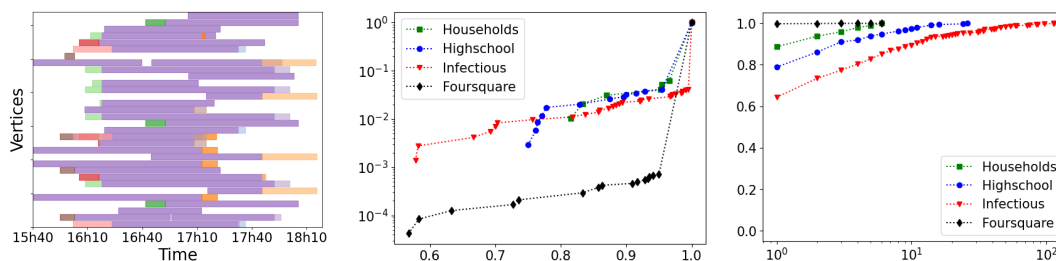
Another point of interest is that the LSCPM computation time increases with  $k$ , while it decreases with OCPM. This comes from the fact that OCPM implementation obtains its results by aggregating the maximal cliques of size *at least*  $k$ , while our method enumerates  $k$ -cliques. With larger  $k$ , there are fewer maximal cliques to enumerate and process, hence the decreasing computation time. By contrast, we have seen in Section 4.1 that  $n_k$  typically



increases with  $k$  for larger datasets, which implies that the computation time of LSCPM increases too according to Theorem 4. Notice however that in spite of this, there is only one instance where OCPM is faster than LSCPM: *Foursquare* link stream with  $k = 7$ , which as we have seen as a very large number of  $k$ -cliques.

### 4.3 Comparison between LSCPM and DCPM communities

In what follows, we compare the communities obtained with our LSCPM algorithm to those obtained with DCPM, on the four datasets where the OCPM implementation provides a result. Note that, up to our knowledge, there is no reference method to compare overlapping temporal communities. So we do not use tools such as NMI or Rand index which are designed for comparing partitions of vertices in a graph and thus would require some adaptation to the context considered in this paper. We have seen in Section 2 that each DCPM community is included in a LSCPM one and, conversely, that each LSCPM community can be seen as the union of DCPM communities. This property is illustrated in Figure 4 (left), which gives an example of a LSCPM community from *Infectious* dataset, with  $k = 3$  using the python package `tnetwork`<sup>5</sup>. Vertices are represented on the Y-axis, time on the X-axis, and each vertex belongs to the community over the period during which it is colored. Each of the DCPM communities included in this LSCPM community is represented in a different color. In what follows, we investigate to what extent DCPM communities are grouped into LSCPM communities.



■ **Figure 4** Composition of LSCPM communities in terms of DCPM communities, with  $k = 3$ . Left: a LSCPM community of *Infectious*, and the DCPM communities included in it (one color each). Center: cumulative distribution of the relative size (in number of vertices) of the largest DCPM community to the corresponding LSCPM community. Right: cumulative distribution of the number of DCPM community per LSCPM community.

To evaluate the similarity between a LSCPM community and the DCPM communities that it contains, we compute the relative size (in number of vertices) of the largest DCPM community that each LSCPM community contains. Figure 4 (center) reports the cumulative distribution of this value. We see a clear peak at the end, which shows that for all datasets, 90% of the LSCPM communities contain as many vertices as their largest DCPM community, which is the case of the example in Figure 4 (left). Only 1% of LSCPM communities have its largest DCPM community with less than 70% of the vertices, and none have its largest DCPM community with less than 50%.

We also observe that there are only a few DCPM communities per LSCPM community. It is illustrated in Figure 4 (right), which represents the cumulative distribution of the number of DCPM communities that each LSCPM community contains. In all cases, more than 70%

<sup>5</sup> <https://tnetwork.readthedocs.io/>

of LSCPM communities contain only 1 or 2 DCPM communities, and almost never more than 10. However, there are some exceptions: in *Highschool*, 2.6% of the communities contain between 10 and 26 DCPM communities, and in *Infectious*, 1.8% of LSCPM communities contain between 50 and 115 DCPM communities.

Besides, we observe that there are fewer small LSCPM communities than DCPM ones: considering communities with 5 or fewer vertices, we observe that *Households* has 17% more DCPM communities than LSCPM, *Highschool* has twice as many and *Infectious* has six times as many, (however, the sets of LSCPM and DCPM communities are very similar for *Foursquare*). This indicates that small DCPM communities tend to be aggregated into larger LSCPM ones, as observed in the example of Figure 4 (left). These observations give the typical scheme of how LSCPM communities compare to the DCPM ones: a LSCPM community is in general composed of one large DCPM community which contains almost all the vertices, and possibly a few residual communities. It also indicates that most of the meaning conveyed by the communities obtained in both cases should be closely related, but LSCPM method allows streamlining the community analysis by aggregating the smaller, less meaningful, communities into the larger ones.

#### 4.4 Insights on LSCPM communities

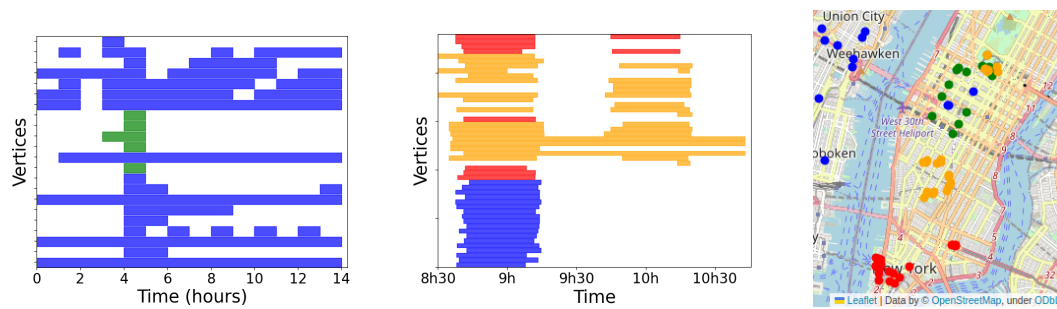
To investigate the relevance of the temporal communities obtained, we have at our disposal metadata retrieved from the dataset repositories: families of *Households*, class of *Highschool*, and GPS coordinates of locations in *Foursquare*.

In the case of the *Households* and *Highschool* datasets, which are based on person-to-person interactions, we observe that the communities are homogeneous in terms of these categories, as could be expected. Indeed, in the case of *Households* dataset, 95% of communities are composed of members of only one family, and the remaining 5% of two families. In *Highschool*, 70% of communities are composed of only one class, 23% of two classes, 6% of three classes, and 1% of four classes.

Moreover, this metadata provides interesting insight on interactions at the individual level over time, pointing out who socializes with whom and when. For instance, Figure 5 (left) is a community of *Households*, composed of 5 members of a family (in green) and 17 members of another (in blue). It highlights the existence of an at most one hour gathering between all these people except one. Similarly, Figure 5 (center) shows a community where we observe members of three different classes of *Highschool*, which are the three physics major classes of the preparatory school. We distinguish three time periods: during the first one, students from the three classes are grouped together, which suggests a period when students can meet and mix. Then, the community reduces to a few nodes of the orange class, later joined by an additional group of students, which might indicate the proximity of the students during the courses or working groups.

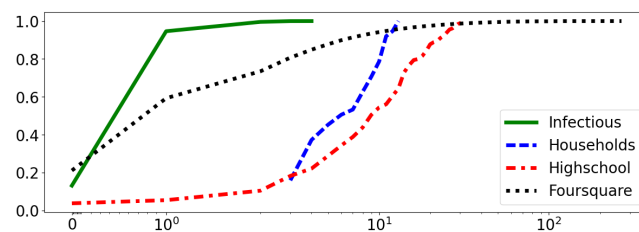
Concerning *Foursquare*, we can use the metadata to investigate the geographical distribution of locations which are visited by the same individuals within a  $\Delta$  period. Figure 5 (right) shows a map of a part of New-York City displaying a sample of four LSCPM communities. We observe that some of them are relatively clustered geographically, such as the green one, while others are more extended, which happens when one or several persons move from one part of the city to another within a  $\Delta$  period. Thus here,  $\Delta$  allows tuning the geographical extension of the communities, as lower  $\Delta$  should correspond to smaller geographical extensions.

It may also be relevant to evaluate the involvement of vertices in the communities that they belong to. Indeed, some vertices are active longer than others or belong to more communities, so it makes it possible to identify particularly important nodes in a group or



■ **Figure 5** Examples of LSCPM communities using the metadata of the datasets. Left: a community of *Households* with vertices colored according to the family. Center: a community of *Highschool* with vertices colored according to the class. Right: a map of Manhattan (NYC), where four *Foursquare* communities are represented in different colors.

nodes which act as bridges between groups. Figure 6 illustrates these two aspects: it is the cumulative distribution of the number of communities to which each vertex belongs. Points on the far left correspond to vertices that belong to no community at all. In this regard, the various datasets yield very different results. For example, in *Foursquare*, around 20% of the vertices do not belong to any community; it corresponds to locations where users rarely visit other places over the time period considered. We observe this for some specific locations such as medical centers, offices, playgrounds... By contrast, in *Households*, each vertex belongs to at least two communities, which is reasonable as it is a contact network between members of a same household, so they are in contact with each other a lot. In *Highschool*, we see that most nodes belong to many communities, which also makes sense, as students are grouped in classes, and each day makes new LSCPM communities. Finally, vertices that belong to many communities are on the far right of the distribution. It is striking on *Foursquare*, where almost 10% of vertices belong to more than 10 communities, and a few to more than 100. These can be described as central nodes of the link stream, which interact with many other nodes throughout the period of observation. For example, the location of *Foursquare* that belongs to the most communities by far (1.5 times more than the second) is the famous Pennsylvania Station, which is the main intercity train station of New York City.

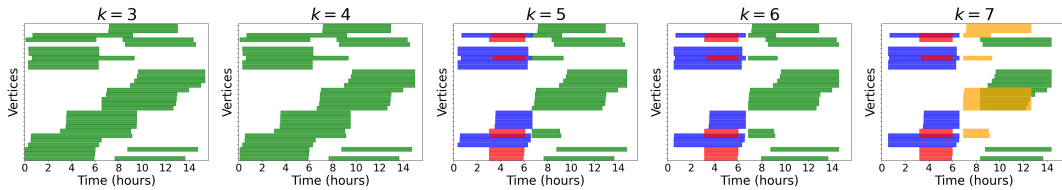


■ **Figure 6** Number of communities to which each vertex belongs, as a cumulative distribution. Note that the X-axis is in logarithmic scale, except between 0 and 1, in order to show vertices that belong to no community.

#### 4.5 Influence of $k$ on the community structure

The size  $k$  of the cliques at the base of the LSCPM communities is the key parameter of the algorithm. Here, we discuss the effect of increasing  $k$  on the community structure, in order to give an intuition to the user as to the choice to make for this value.

As we saw in Section 2, if  $k_1 < k_2$ , then each community computed with  $k_2$  is included in a community computed with  $k_1$ . This means that increasing  $k$  results in the communities splitting and/or losing temporal vertices. Figure 7 gives an example of this phenomenon on a community of the *Foursquare* dataset: from left to right we see the community computed with  $k = 3$  and how it splits and loses nodes when increasing  $k$  up to 7. We observe that the community remains almost identical for  $k = 4$ , and that it splits into three communities for  $k = 5$ , and one of these smaller communities split again for  $k = 7$ . We also observe that some nodes which belong to the community for  $k = 3$  at a given time do not belong to any of the resulting communities at this time for larger values of  $k$ . Thus, increasing  $k$  leads to more cohesive communities, but smaller in size and time extension. This allows to change the granularity of the dynamical communities that can be obtained on a dataset by focusing on the “core” of the interactions.



■ **Figure 7** Split of a *Foursquare* community as  $k$  increases from  $k = 3$  to  $k = 7$ .

This allows to identify consistent sub-communities when metadata is available. For instance, the vertices of the *Foursquare* community of Figure 7 correspond to 6 types of locations (out of the 261 available), all sport-related: Athletic-&Sport, Bike-Shop, Stadium, Sporting-Goods-Shop, Gym-/Fitness-Center, Motorcycle-Shop (most other communities exhibit more diverse labels). We investigate if the splitting of communities when  $k$  increases corresponds to more precise types of locations. For  $k = 5$ , the green community has the 6 type labels, but nodes of the blue and red ones only have 3 labels: Sporting-Goods-Shop, Gym-/Fitness-Center and Bike-Shop. Also, for  $k = 7$ , the green community splits into two parts, one of which focuses on two-wheeled sports: Motorcycle, Bike, Stadium, which highlights that the decomposition allows to derive the shared interests of the users.

## 5 Conclusion and perspectives

In this paper, we address the detection of dynamic communities in temporal networks, using the link stream formalism. Using the literature of the field, we introduced the notion of maximal  $k$ -clique of a link stream, with an algorithm to enumerate them. This leads to a new adaptation of the Clique Percolation Method to dynamical networks, called LSCPM, which pursues the work initiated by [19]. We provided a theoretical analysis of the complexity of our algorithm, as well as an open source implementation in Python. Then, we experimented with the algorithm, comparing it to the state of the art, and showed that it allows to obtain possibly relevant information on real-world examples.

We believe that the community detection with LSCPM can scale to even more massive networks and larger values of  $k$ . For instance, as memory consumption can be a limiting factor on massive streams because of the clique storage, the work done in [2] to reduce the RAM cost of the CPM method on graphs could be adapted to link streams. Also, it could be better in some cases to percolate maximal cliques instead of  $k$ -cliques, as done in OCPM [4], using efficient maximal clique enumeration methods in link streams such as [3]. In particular, we have seen that this may be an efficient alternative for larger  $k$  values.

Moreover, it would be interesting to develop the analysis of the effect of link duration  $\Delta$  and its practical implications. Indeed, when  $\Delta$  increases, the  $k$ -cliques grow longer, resulting in more aggregated LSCPM communities. This is the opposite effect of what happens when we increase  $k$ . We believe that the experimental study can be extended by testing the simultaneous tuning of these two parameters, to see how the communities are structured, and if this allows targeting relevant interaction cores in particular.

Finally, we believe that it is possible to adapt the algorithm for enumerating  $k$ -cliques in link streams, into a more general framework of temporal motif listing. Indeed, the call of the algorithm to enumerate graph cliques could be replaced in principle by any other motif mining algorithm in a graph, resulting in a related temporal motif. This paves the way to novel and efficient pattern mining algorithms in temporal networks.

---

## References

- 1 Thomas Aynaud and Jean-Loup Guillaume. Static community detection algorithms for evolving networks. In *8th international symposium on modeling and optimization in mobile, ad hoc, and wireless networks*, pages 513–519. IEEE, 2010.
- 2 Alexis Baudin, Maximilien Danisch, Sergey Kirgizov, Clémence Magnien, and Marwan Ghanem. Clique percolation method: memory efficient almost exact communities. In *Advanced Data Mining and Applications: 17th International Conference, ADMA 2021, Sydney, NSW, Australia, February 2–4, 2022, Proceedings, Part II*, pages 113–127. Springer, 2022.
- 3 Alexis Baudin, Clémence Magnien, and Lionel Tabourier. Faster maximal clique enumeration in large real-world link streams. *arXiv preprint arXiv:2302.00360*, 2023.
- 4 Souâad Boudebza, Rémy Cazabet, Faïçal Azouaou, and Omar Nouali. OLCPM: An online framework for detecting overlapping communities in dynamic social networks. *Computer Communications*, 123:36–51, 2018. Code available at: [https://figshare.com/articles/software/OLCPM\\_algorithm/5914846](https://figshare.com/articles/software/OLCPM_algorithm/5914846).
- 5 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.
- 6 Rémy Cazabet, Giulio Rossetti, and Frédéric Amblard. Dynamic community detection, 2017.
- 7 Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. Online search of overlapping communities. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*, pages 277–288, 2013.
- 8 Maximilien Danisch, Oana Balalau, and Mauro Sozio. Listing  $k$ -cliques in sparse real-world graphs. In *Proceedings of the 2018 World Wide Web Conference*, pages 589–598, 2018.
- 9 Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.
- 10 Julie Fournet and Alain Barrat. Contact patterns among high school students. *PloS one*, 9(9):e107878, 2014. Data available at: <http://www.sociopatterns.org/datasets/high-school-dynamic-contact-networks>.
- 11 Petter Holme and Jari Saramäki. Temporal networks. *Physics reports*, 519(3):97–125, 2012.
- 12 Lorenzo Isella, Juliette Stehlé, Alain Barrat, Ciro Cattuto, Jean-François Pinton, and Wouter Van den Broeck. What’s in a crowd? analysis of face-to-face behavioral networks. *Journal of Theoretical Biology*, 271(1):166–180, 2011. Data available at: <http://www.sociopatterns.org/datasets/infectious-sociopatterns/>.
- 13 Moses C Kiti, Michele Tizzoni, Timothy M Kinyanjui, Dorothy C Koech, Patrick K Munywoki, Milosch Meriac, Luca Cappa, André Panisson, Alain Barrat, Ciro Cattuto, et al. Quantifying social contacts in a household setting of rural kenya using wearable proximity sensors. *EPJ data science*, 5:1–21, 2016.
- 14 Jussi M Kumpula, Mikko Kivelä, Kimmo Kaski, and Jari Saramäki. Sequential algorithm for fast clique percolation. *Physical review E*, 78(2):026109, 2008.

- 15 Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8:1–29, 2018.
- 16 Matteo Magnani, Obaida Hanteer, Roberto Interdonato, Luca Rossi, and Andrea Tagarelli. Community detection in multiplex networks. *ACM Computing Surveys (CSUR)*, 54(3):1–35, 2021.
- 17 Alan Mislove. *Online Social Networks: Measurement, Analysis, and Applications to Distributed Information Systems*. PhD thesis, Rice University, Department of Computer Science, May 2009. Data available at: <https://socialnetworks.mpi-sws.org/data-wosn2008.html>.
- 18 Nam P Nguyen, Thang N Dinh, Sindhura Tokala, and My T Thai. Overlapping communities in dynamic networks: their detection and mobile applications. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 85–96, 2011.
- 19 Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 2007.
- 20 Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *nature*, 435(7043):814–818, 2005.
- 21 Gang Pan, Wangsheng Zhang, Zhaohui Wu, and Shijian Li. Online community detection for large complex networks. *PloS one*, 9(7):e102799, 2014.
- 22 Fergal Reid, Aaron McDaid, and Neil Hurley. Percolation computation in complex networks. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 274–281. IEEE, 2012.
- 23 Giulio Rossetti, Luca Pappalardo, Dino Pedreschi, and Fosca Giannotti. Tiles: an online algorithm for community discovery in dynamic social networks. *Machine Learning*, 106:1213–1241, 2017.
- 24 Michael T Schaub, Jean-Charles Delvenne, Martin Rosvall, and Renaud Lambiotte. The many facets of community detection in complex networks. *Applied network science*, 2(1):1–13, 2017.
- 25 Yifu Tang, Jianxin Li, Nur Al Hasan Haldar, Ziyu Guan, Jiajie Xu, and Chengfei Liu. Reliable community search in dynamic networks. *arXiv preprint arXiv:2202.01525*, 2022.
- 26 Robert E. Tarjan and Jan van Leeuwen. Worst-case analysis of set union algorithms. *J. ACM*, 31(2):245–281, March 1984.
- 27 Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.
- 28 Tiphaine Viard, Clémence Magnien, and Matthieu Latapy. Enumerating maximal cliques in link streams with durations. *Information Processing Letters*, 133:44–48, 2018.
- 29 Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2014. Data available at: <https://sites.google.com/site/yangdingqi/home/foursquare-dataset>.

## A Proofs for Section 3 (Algorithms)

► **Theorem 3** (*k*-clique enumeration time complexity). *Algorithm 1 enumerates all k-cliques of the input link stream in time  $\mathcal{O}\left(m \cdot k^3 \cdot \left(\frac{d}{2}\right)^{k-2} + m \cdot d^2 + m \cdot \log(m)\right)$ , where  $m$  is the number of links and  $d$  the maximal degree of a node, that is its maximal number of neighbors at any given time.*

**Proof.** First, we show that the update of  $G$  from Lines 4 to 6 is done in  $\mathcal{O}(m \cdot \log(m))$  in the whole loop of Line 3.  $G$  is stored as an associative array, associating to each vertex the set of its neighbors, so the addition of an edge at Line 4 is done in constant time. Line 5 is also done in constant time. Finding all edges that have an ending time smaller than  $b$  (Line 6) can be done by maintaining a sorted list of the end times of links in  $G$ , which can be done in  $\mathcal{O}(\log(m))$  for each new link. Then, the deletion itself is done in constant time by going through this list. The global complexity is therefore in  $\mathcal{O}(m \cdot \log(m) + m) = \mathcal{O}(m \cdot \log(m))$ .

Consider an iteration of the loop starting at Line 3 and  $(b, e, u, v)$  its associated link. We need to compute all  $k$ -cliques of  $G$  containing  $u$  and  $v$ . This is equivalent to computing the  $(k-2)$ -cliques of the graph induced by the common neighbors of  $u$  and  $v$ , that we note  $G(N(u) \cap N(v))$ . Computing this induced subgraph is done in  $\mathcal{O}(d^2)$ . Then, the overall complexity of these operations over all iterations is in  $\mathcal{O}(m \cdot d^2)$ .

Enumerating the  $(k-2)$ -cliques of  $G(N(u) \cap N(v))$  depends on the value of  $k$ . If  $k=3$  it consists in enumerating vertices, which is in  $\mathcal{O}(d)$ . If  $k=4$ , it is enumerating the edges, in  $\mathcal{O}(d^2)$ . If  $k \geq 4$ , then we use the  $k$ -clique enumeration algorithm in graphs described in [8]. In that paper, Theorem 5.7 gives the complexity of enumeration in  $\mathcal{O}\left(k \cdot m \cdot \left(\frac{d}{2}\right)^{k-2}\right)$ . Thus, the  $(k-2)$ -clique enumeration is in  $\mathcal{O}\left((k-2) \cdot d^2 \cdot \left(\frac{d}{2}\right)^{k-4}\right)$  and the overall complexity of Line 7 is in  $\mathcal{O}\left(k \cdot \left(\frac{d}{2}\right)^{k-2}\right)$ , for any value of  $k$ . Note that this value sets an upper bound on the number of  $k$ -cliques enumerated by the loop iteration. Each of these  $k$ -cliques is then processed by the loop at Line 8, in  $\mathcal{O}(k \cdot (k-1)) = \mathcal{O}(k^2)$ . Then, the total complexity of these operations in the iteration is in  $\mathcal{O}\left(k^3 \cdot \left(\frac{d}{2}\right)^{k-2}\right)$ , it is thus in  $\mathcal{O}\left(m \cdot k^3 \cdot \left(\frac{d}{2}\right)^{k-2}\right)$  for the entire loop.

Combining the cost of the above operations finally gives the result. Note that this result can be slightly refined by keeping  $k \cdot (k-1) \cdot (k-2)$  instead of  $k^3$ , which may have a significant impact, since  $k$  values are usually small (typically  $\leq 10$ ). ◀

► **Theorem 4** (LSCPM time complexity). *The time complexity of Algorithm 2 is in  $\mathcal{O}\left((k + \alpha(n_k)) \cdot k \cdot n_k + c(k)\right)$ , where  $\alpha$  is the inverse Ackermann function,  $n_k$  the number of  $k$ -cliques of the link stream, and  $c(k)$  the complexity of enumerating  $k$ -cliques, given in Theorem 3. It is thus in  $\mathcal{O}\left((k + \alpha(n_k)) \cdot m \cdot k^2 \cdot \left(\frac{d}{2}\right)^{k-2} + m \cdot d^2 + m \cdot \log(m)\right)$ .*

**Proof.** With a Union-Find data structure, it is known that the amortized cost of **Union** and **Find** functions is in  $\mathcal{O}(\alpha(N))$ , if  $N$  is the number of elements in the structure (see for example [26]). Here, the Union-Find structure contains at most 1 element per maximal  $k$ -clique, since there cannot be more than one call to **MakeSet** (Line 16) in each iteration of the loop starting at Line 5. Indeed, if a **MakeSet** is performed, then  $p$  is no longer equal to  $-1$  and no other is performed until the end of this loop. So the complexity of each call to **Union** and **Find** functions in the procedure is in  $\mathcal{O}(\alpha(n_k))$ .

Now, consider a maximal  $k$ -clique  $(C_k, [t_0, t_1])$  corresponding to an iteration of the loop starting at Line 5. Line 7 performs one iteration per vertex of  $C_k$ , that is  $k$  iterations. The operation at Line 9 to find the last element of **TimeUF** $[C_{k-1}]$  is in  $\mathcal{O}(k-1)$ . During this



loop, there is also at most one call to `Union` and `Find`, and other operations in constant time. So, it runs in  $\mathcal{O}(k + \alpha(n_k))$ . Thus, in total, the loop starting at Line 5 runs in  $\mathcal{O}(n_k \cdot k \cdot (k + \alpha(n_k)))$ .

In addition, we have to take into account the complexity of the enumeration of  $k$ -cliques and  $(k-1)$ -cliques given as input to the algorithm. However, Theorem 3 indicates that the complexity of enumerating  $(k-1)$ -cliques is included in the one of enumerating  $k$ -cliques, denoted  $c(k)$ . We thus obtain an overall complexity of Algorithm 2 in  $\mathcal{O}((k + \alpha(n_k)) \cdot k \cdot n_k + c(k))$ .

Finally, we saw in the proof of Theorem 3 above that at each iteration of the loop at Line 3 of Algorithm 1, the number of  $k$ -cliques listed is in  $\mathcal{O}\left(k \cdot \left(\frac{d}{2}\right)^{k-2}\right)$ . Since there are  $m$  iterations, we get that the number of  $k$ -cliques  $n_k$  is in  $\mathcal{O}\left(m \cdot k \cdot \left(\frac{d}{2}\right)^{k-2}\right)$ . Hence the second part of the theorem by combining the above bound on  $n_k$  and Theorem 3. ◀

Note that the output of Algorithm 2 is the Union-Find structure and thus a post-processing is required to produce the actual communities. This post-processing is done by going through the set of elements  $(C_{k-1}, [t_0, t_1])$  in `TimeUF`, that are at most  $k \cdot n_k$  (at most  $k$  per maximal  $k$ -clique). Then, it performs a `Find` operation on them and adds each of its  $k-1$  associated vertices to its community during the corresponding time interval. Adding a vertex to its community can be done in constant time if the nodes of the Union-Find are browsed in chronological order, which is possible by storing their creation time order at Line 16. So the complexity of the post-processing is also in  $\mathcal{O}(k \cdot n_k \cdot (k + \alpha(n_k)))$ .



# Discovering Predictive Dependencies on Multi-Temporal Relations

Beatrice Amico   

Department of Computer Science, University of Verona, Italy

Carlo Combi   

Department of Computer Science, University of Verona, Italy

Romeo Rizzi   

Department of Computer Science, University of Verona, Italy

Pietro Sala   

Department of Computer Science, University of Verona, Italy

---

## Abstract

In this paper, we propose a methodology for deriving a new kind of approximate temporal functional dependencies, called Approximate Predictive Functional Dependencies (APFDs), based on a three-window framework and on a multi-temporal relational model. Different features are proposed for the Observation Window (OW), where we observe predictive data, for the Waiting Window (WW), and for the Prediction Window (PW), where the predicted event occurs. We then discuss the concept of approximation for such APFDs, introduce two new error measures. We prove that the problem of deriving APFDs is intractable. Moreover, we discuss some preliminary results in deriving APFDs from real clinical data using MIMIC III dataset, related to patients from Intensive Care Units.

**2012 ACM Subject Classification** Information systems → Relational database model; Information systems → Data mining

**Keywords and phrases** temporal databases, temporal data mining, functional dependencies

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.4

## 1 Introduction

Knowledge from databases may be expressed by discovering patterns and data dependencies. Database dependencies express relevant characteristics of datasets, thereby enabling various critical analyses of data. Functional dependencies (FDs) have been proposed as a way of mining data, i.e., by discovering those FDs that hold on most data. The considered approximation may be heterogeneous and deal with both null values, quantitative data, data deletion/updates, and so on [7, 4, 18, 7, 12, 19].

Temporal Functional Dependencies (TFDs) received some interest since the nineties, initially as a way for specifying constraints on temporal data [32, 9, 5], and, more recently, as a mining approach in their approximate version, looking for hidden temporal patterns inside data [8, 25, 10].

To the best of our knowledge, TFDs have not yet been considered for the prediction task. Such decision-support task is mainly devoted to the prediction of some (future) event based on a (past) data history. Thus, as time is an inherent feature of this task, TFDs are interesting candidates as a formal tool, for discovering the predictivity of the stored data. Within this context, in this paper we propose and discuss an original temporally-oriented data mining framework to support the prediction of future events through the identification of recurring past temporal data patterns, expressed as *Approximate Temporal Predictive Functional Dependencies* (APFDs), according to a 3-window -based temporal framework. New kinds of



© Beatrice Amico, Carlo Combi, Romeo Rizzi, and Pietro Sala;  
licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 4; pp. 4:1–4:19



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

error and related thresholds are introduced, to deal with the required approximation. The main novelty can be summarized in the formalization of a new framework to exploit the predictive aspect of the APFDs, according to the following specific aspects:

- We introduce a new temporal framework based on three temporal windows: observation window (OW), waiting window (WW), and prediction window (PW). The waiting window is explicitly introduced to create a time span before the prediction for being able to (possibly) manage the predicted event.
- We define and exemplify the entire framework for the approximate predictive functional dependencies (APFDs) in a formal way by introducing and characterizing *multi-temporal relations*. It allows the representation of temporal patterns (made by attribute values) related to a set of observed entities (e.g., patients) and characterizes their predictivity, with respect to a target attribute (e.g., a disease).
- We discuss different kinds of error measures, named  $G_3$ ,  $H_3$ , and  $J_3$ , to be evaluated when deriving APFDs;
- We discuss the (data) complexity of the problem of checking for APFDs and prove that is exponential. We then propose a new algorithm for checking APFDs.
- We provide some experimental results on real clinical data from patients in Intensive Care Units, using data from MIMIC III [16], to obtain different APFDs.

With respect to the preliminary proposal of APFDs sketched in [3], as specific novelties, here we first characterize a new temporal data model, based on relations having multiple valid times; we introduce the three-window framework and the related APFDs for such model; we extensively consider the related data complexity; we propose a new algorithm for checking APFDs; we discuss further experimental evaluation.

Our paper unfolds as follows. Section 2 contains the related work; in Section 3 we introduce and motivate the 3-window-based framework for prediction, the formalization of APFDs and their approximation; in Section 4 we discuss the data complexity of deriving APFDs, and provide a deterministic algorithm that could stop the analysis of a relation, as soon as it verifies that the relation cannot satisfy the given APFD; in Section 5, we introduce and discuss some experimental results and finally in Section 6 we draw some conclusions. The Appendix A completes the description of our approach through the proof of the NP-hardness of the APFDs-checking problem.

## 2 Related work

FDs were originally proposed to specify data constraints in the relational setting and then to derive normalized relational schemata [2].

Let us briefly recall the concept of FD in the context of relational databases [2]. Let  $r$  be a relation over the relational schema  $R(U)$  and let  $X, Y \subseteq U$ .  $r$  fulfills the functional dependency  $X \rightarrow Y$  (written as  $r \models X \rightarrow Y$ ) if  $\forall t, t' \in r (t[X] = t'[X], \rightarrow t[Y] = t'[Y])$ .

In more recent years FDs have been extended in many different directions and with different goals. Here we mainly consider three different research directions: the first one deals with the representation of constraints on temporal data through *temporal functional dependencies* (TFDs), the second one focuses on the discovery of *approximate functional dependencies* (AFDs), and the third one deals with the use of *FDs to support prediction and classification tasks*.

TFDs add a temporal dimension to classical FDs to deal with temporal data. In literature, several kinds of TFDs have been proposed and various representation formalisms have been developed [5, 15, 29, 30, 31, 9]. In [9] Combi et al. propose a new formalism for the

representation of TFDs, involving multiple time granularities. They identify four relevant classes, named *pure temporally grouping*, *pure temporally evolving*, *temporally mixed*, and *temporally hybrid* TFDs, respectively.

In [22], the authors face another temporal aspect, which stems from the observation that frequent constraint violations in a database may be related to the fact that the considered (mini) world is changing, while the specified constraints remain static. FDs violated by current data are then identified and some approaches are proposed to suitably modify the given FD according to the new reality represented through the current data. In [26], the authors deal with the problem of continuously discovering FDs on dynamic datasets in an efficient way, and propose an incremental approach to solve it.

AFDs derive from the concept of plain FD. Given a relation  $r$  where an FD holds for most of the tuples in  $r$ , we may identify some tuples for which that FD does not hold. In [18], Kivinen and Mannila introduce three measures, known as  $G_1$ ,  $G_2$  and  $G_3$  considering, respectively, the number of violating couples of tuples, the number of tuples that violate the functional dependency, and finally the minimum number of tuples in  $r$  to be deleted for the FD to hold. Discovering AFDs is a computationally expensive task, and different algorithms have been proposed to perform the discovery in an efficient way [19]. More recently, AFDs have been included in the wider scenario of *relaxed FDs* (RFDs), where not only exceptions, i.e., violating tuples, are considered, but also similarities among attribute values and conditional constraints [7, 6].

Temporal data mining techniques merging AFDs with TFDs have been proposed in [8], where the authors propose approximate temporal functional dependencies (ATFDs), which are defined and measured either on temporal granules or on sliding windows, and apply them to mine data from psychiatry and pharmacovigilance domains. They introduce a new error measure  $G_4$ , which considers the minimum number of tuples in  $r$  which must be modified for the plain TFD to hold on all the tuples of  $r$ . In [1], the authors present AETAS, a system for the discovery of approximate temporal functional dependencies. The discovered TFDs are mainly pure temporally grouping TFDs with moving windows, according to the classification proposed in [9]. Also conditional TFDs are considered, where the moving window may have different values according to specific values of atemporal attributes. As an interesting aspect of AETAS, the authors deal with the discovery of TFDs from dirty web data, as well as with the discovery of the “optimal” duration for the moving window.

Moving to contributions dealing with the use of *FDs to support prediction and classification tasks*, in [20] the authors show that if there is a functional dependency between features, it is likely to affect the classifier negatively. In [21], the authors address the notion of trusting ML models by using also functional dependencies, discussing on the relationships between supervised classification and functional dependencies. They consider the issue of estimating the feasibility of classification over a given dataset using functional dependencies. As far as we know, few studies till now considered functional dependencies in this context, where, given a set of features  $(A_1, \dots, A_n, C)$  where  $C$  values represent the class to be classified, the problem is to understand whether functional dependencies such as  $A_1, \dots, A_k \rightarrow A_j$  influence the classification performances.

### 3 The predictive aspects of functional dependencies

In this section, firstly we delineate the problem at hand, and introduce a 3-window model for the interpretation of predictive temporal data; then we illustrate the definitions needed to obtain a Predictive Functional Dependency, and finally, we analyze the concept of approximation for the Predictive Functional Dependencies.

■ **Table 1** The multi-temporal relation `PatientHistory`, with a single atemporal attribute and one attribute for each valid time.

#	Patient	$\overline{HR}^1$	$\overline{VT}^1$	$\overline{SpO_2}^2$	$\overline{VT}^2$	$\overline{Drug}^3$	$\overline{VT}^3$	AKI	$\dot{VT}$
1	Daisy	High	19	High	21	Aspirin	23	False	28
2	Daisy	Low	2	High	4	Aspirin	6	False	18
3	Daisy	Low	2	Medium	4	Aspirin	6	False	12
4	Daisy	Medium	5	Medium	7	Indapamide	9	False	18
5	Luke	Low	7	High	8	Ibuprofen	12	True	17
6	Luke	Low	7	High	8	Ibuprofen	12	True	21
7	Luke	Medium	9	High	13	Sulindac	14	True	18
8	Luke	Medium	9	High	13	Sulindac	14	True	21
9	Stevie	Medium	4	Medium	7	Metolazone	8	True	13
10	Stevie	High	1	Low	2	Aspirin	5	False	8
11	Stevie	High	1	Low	2	Indapamide	7	False	8
..	...	...	...	...	...	...	...	...	...
36	Stevie	High	1	Low	2	Aspirin	5	False	25
..	...	...	...	...	...	...	...	...	...

### 3.1 A motivating scenario from Clinical Medicine

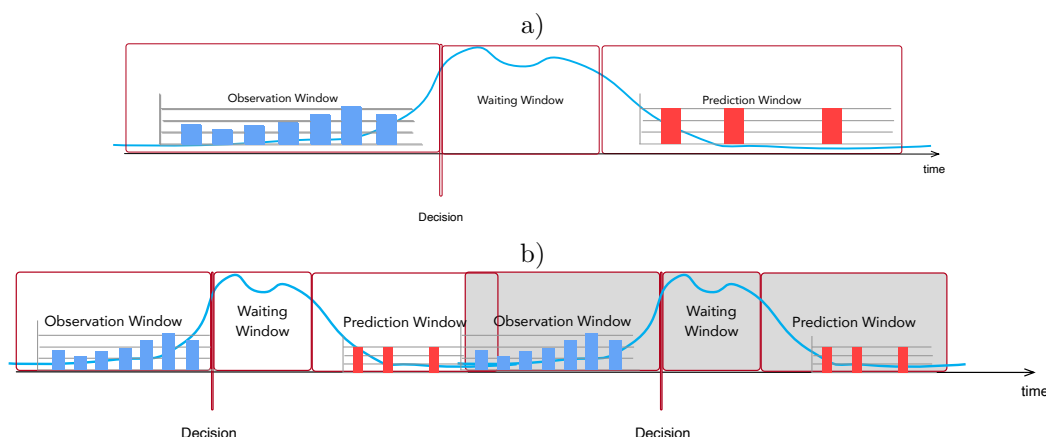
To illustrate the relevance and the potential meaning of our approach, we consider a real-world example from the domain of Intensive Care Unit (ICU) focusing on patients suffering from Acute Kidney Injury (AKI) [28], used as reference throughout the paper. In ICU, Acute Kidney Injury is a frequent clinical problem, characterized by sudden loss of the ability of the kidneys to excrete wastes, concentrate urine, store electrolytes, and maintain fluid balance [27].

In 2012, KDIGO (Kidney Disease: Improving Global Outcomes) published specific guidelines [17] for the definition of AKI, where a patient receives the diagnosis if one of the following criteria is satisfied: (i) an increase in serum creatinine by  $\geq 0.3$  mg/dl ( $\geq 26.5$   $\mu\text{mol/l}$ ) within 48 h, (ii) an increase in serum creatinine to  $\geq 1.5$  times baseline within the previous 7 days and (iii) a urine volume  $\leq 0.5$  ml/kg/h for 6 hours.

As we are interested in discovering whether some clinical data features allow the early identification of AKI patients, let us assume that we derive through a suitable query the (possibly materialized) view `PatientHistory`. It represents different ordered states of patients, we would like to associate to a final state, specifying whether the patient has AKI. Each state is represented by some attribute values and is associated to a *valid time* (VT), representing the timepoint when the state information is true in the modeled world [14]. Table 1 (partially) shows a possible instance of `PatientHistory` describing a clinical history of three patients, Daisy, Luke, and Stevie, who have some measured vital signs and undergo five different drugs, some of them specific for the AKI treatment. Such history can be derived from the data contained in a clinical database [16].

### 3.2 A 3-window framework for the interpretation of predictive temporal data

In general, the prediction models exploit the use of two-time windows, namely (i) a data collection (or observation) window, and (ii) a prediction window. Even though there are approaches [11, 24] which consider a third temporal window, to the best of our knowledge, a general and formal prediction framework considering three different time windows has not yet been considered in the data mining literature.



■ **Figure 1** The time windows of the proposed framework: (a) the anchored and (b) the unanchored–sliding window– case.

According to this view, depicted in Figure 1, we can observe:

1. Decisions are taken after gathering information for some time span (*Observation Window*: OW).
2. After the moment when the decision is taken, we have to execute all the related actions and (possibly) wait for a while (*Waiting Window*: WW). The WW is held to be the minimum time interval required to act in order to prevent the event in the prediction window. Indeed, not all the performed actions have an instantaneous effect.
3. The last temporal window refers to when the possible effects of the decision are observable and thus we can evaluate the suitability of the taken decision (*Observation Window*: OW).

It is worth noting that the span of such windows may be different and could be also composed of a single time-point. Moreover, the *Waiting Window* could be missing, i.e., of zero length, in case of decisions having an immediate observable effect.

In general, we may identify different orthogonal features for the introduced time windows.

The first distinction is between (i) *anchored* and (ii) *unanchored* time windows. Indeed, with anchored time windows, we are able to represent specific periods of the considered time axis. An example of anchored time windows for the motivating scenario could consist of specifying OW as the first 4 hours from the admission to the ICU, the following 2 hours as WW (i.e., the fifth and sixth hour after the ICU admission), ending with the PW from the seventh to the tenth hour after the ICU admission. Figure 1 a) depicts the three anchored windows, the time-point corresponding to the decision moment, and possible temporal evolution of some observed quantitative parameter, having some varying behavior. On the other side, unanchored time windows represent windows that “move” through the time axis, constraining only the distance between the considered data. An example of such kind of windows for our scenario could consist in specifying again 4 hours, 2 hours, and 4 hours for OW, WW, and PW, respectively, but not anchored to any point of the time axis. Figure 1 b) represents two partially overlapping views, representing unanchored time windows. In this case, we may consider a possibly infinite number of unanchored (sliding) windows, even by specifying the width of the step size of sliding.

► **Definition 1** (Unanchored Time-Frame). *An unanchored time-frame (uTF)  $\alpha$  is a triple  $\langle OW, WW, PW \rangle$  where OW, WW, and PW are expressed as durations, i.e., time distances. They allow the representation of three different unanchored windows, which we will use to observe temporal data.*

► **Definition 2** (Anchored Time-Frame). *An anchored time-frame (aTF)  $\alpha$  is a time-frame associated to an anchor time point and can be represented through the structure  $\langle atp, \langle OW, WW, PW \rangle \rangle$ , where  $atp$  is a (anchor) time point.*

A second subtle distinction, which may provide different results for prediction and is orthogonal with respect to the distinction between anchored and unanchored time windows, is between (i) *fixed-length* and (ii) *variable-length* time windows. Indeed, OW, and consequently the following WW and PW, could be either of fixed length, without any further constraint related to the temporal position of data inside it, or of variable length, and thus ending with the last time point associated with the data to consider in the window.

### 3.3 A multi-temporal relational model and its connection to the temporal framework

Let us introduce the concept of *multi-temporal relation*. Informally, a multi-temporal relation is characterized by multiple valid times. Each tuple of such relation represents a piece of history of a given entity, through the values of attributes holding at different (valid) times. A set of attributes of such relation allows the (optional) identification of the considered entities (e.g., a patient, an employee) and their characterization. Any other attribute of such relation is associated with a specific valid time.

► **Definition 3** (Multi-temporal relation (*mt-relation*)). *Given an overall set of attributes  $\mathcal{A}$  and a set of valid time attributes  $\mathcal{VT}$ , a multi-temporal relation  $mtr$  is a relation with schema  $WT$  where  $W \subseteq \mathcal{A}$  and  $T = \{VT_1, \dots, VT_i, \dots, VT_k, VT_{k+1}\} \subseteq \mathcal{VT}$  are  $k + 1$  valid time attributes.*

*For a multi-temporal relation schema, a mapping  $Vtime : T \rightarrow 2^W$  allows us to specify the attribute subset associated to a specific valid time. For such mapping, it holds*

$$Vtime(VT_i) \cap Vtime(VT_j) = \emptyset \text{ for any } i, j \text{ with } i \neq j$$

*The (possibly empty) set  $Z \subseteq W$ ,  $Z = W - \bigcup_{i=1}^{k+1} Vtime(VT_i)$  contains attributes not associated with any valid time attribute.*

*For any relation  $mtr$  it holds*

$$\forall t \in mtr(t[VT_i] < t[VT_j]) \text{ for } 1 \leq i < j \leq k + 1$$

As we will discuss in the following, the main idea here is to propose a general framework allowing the definition of “specialized” functional dependencies having the antecedent composed of a set of attributes, called *predictive attributes*, ordered according to the corresponding valid times and the consequent defined as the predicted attributes. In order to distinguish such roles for attributes, we introduce a suitable partition of attributes, according to the following definition.

► **Definition 4** (Prediction-oriented partition of mt-relation valid times). *Given a multi-temporal relation  $mtr$  with schema  $WT$ , where  $W \subseteq \mathcal{A}$  and  $T = \{VT_1, \dots, VT_i, \dots, VT_k, VT_{k+1}\}$ , attributes in  $T$  are partitioned in two sets  $\mathcal{O}$ , for observation-related valid times, and  $\mathcal{P}$ , for prediction-related valid times, where it holds*

$$\forall VT_o, VT_p ((VT_o \in \mathcal{O} \wedge VT_p \in \mathcal{P}) \implies \forall t \in mtr(t[VT_o] < t[VT_p]))$$

For the sake of simplicity and without losing generality, in the following, we assume that  $\mathcal{O} \equiv \{VT_1, VT_2, \dots, VT_k\}$ , while  $\mathcal{P} \equiv \{VT_{k+1}\}$ . According to this choice, we use an overline-based notation for (ordered) observation-related valid times and the associated attributes. We use a dot notation for the prediction-oriented valid time and the associated attributes.



► **Example 5.** The relation view depicted in Table 1 considers attributes according to the introduced notation. More precisely, in this case  $\mathcal{O} \equiv \{\overline{VT}^1, \overline{VT}^2, \overline{VT}^3\}$ ,  $\mathcal{P} \equiv \{VT\}$ , and  $Vtime(\overline{VT}^1) = \{\overline{HR}^1\}$ ,  $Vtime(\overline{VT}^2) = \{\overline{SpO}_2^2\}$ ,  $Vtime(\overline{VT}^3) = \{\overline{Drug}^3\}$ , and  $Vtime(VT) = \{\overline{AKI}\}$ .

Given a multi-temporal relation  $mtr$ , we are now interested in verifying which tuples are “fine” with or “contained” in a given time frame. More precisely, we are interested in eliciting those tuples having the (some of the)  $k$  observation-related valid times contained in the observation window  $OW$ , and the last valid time in the prediction window  $PW$ . We will call them *consistent* with the considered time frame.

In the following, we will introduce different kinds of *time-frame consistency*, mainly considering both the partial containment of some valid times in the observation window and different requirements for the observation window.

Indeed, as for the first aspect, we may be interested in verifying the partial/complete containment of the  $k$  observation-related valid times within the given  $OW$ , while for the second one, we may consider either fixed length  $OW$ s, or flexible observation windows, which end with the last valid time we have to consider in the given  $OW$ .

► **Definition 6** (Time-frame tuple consistency with range and modality). *Given a tuple  $t$  of a multi-temporal relation  $mtr$  with schema  $WT$ , where  $W \subseteq \mathcal{A}$  and  $T = \{VT_1, \dots, VT_i, \dots, VT_k, VT_{k+1}\} \subseteq \mathcal{VT}$ , and a (either anchored or unanchored) time frame  $\alpha$ , we say that  $t$  is time-frame consistent with  $\alpha$  according to modality  $m \in \{\textit{flex}', \textit{fixed}'\}$  in the range  $[i_1, i_2]$ , where  $1 \leq i_1 < i_2 \leq k$ , if formula  $\Theta(t, \alpha, m, [i_1, i_2])$  holds.*

*Formula  $\Theta(t, \alpha, m, [i_1, i_2])$  is defined according to the following cases:*

- $\Theta(t, \alpha, \textit{fixed}', [i_1, i_2]) \equiv t[\overline{VT}^{i_2}] - t[\overline{VT}^{i_1}] \leq OW \wedge t[VT] - t[\overline{VT}^{i_1}] > OW + WW \wedge t[VT] - t[\overline{VT}^{i_1}] < OW + WW + PW$   
-if the time-frame is unanchored-, or
- $\Theta(t, \alpha, \textit{fixed}', [i_1, i_2]) \equiv t[\overline{VT}^{i_1}] \geq atp \wedge t[\overline{VT}^{i_2}] \leq atp + OW \wedge t[VT] > atp + OW + WW \wedge t[VT] < atp + OW + WW + PW$   
-if the time-frame is anchored-, or
- $\Theta(t, \alpha, \textit{flex}', [i_1, i_2]) \equiv t[\overline{VT}^{i_2}] - t[\overline{VT}^{i_1}] \leq OW \wedge t[VT] - t[\overline{VT}^{i_2}] > WW \wedge t[VT] - t[\overline{VT}^{i_2}] < WW + PW$   
-if the time-frame is unanchored-, or
- $\Theta(t, \alpha, \textit{flex}', [i_1, i_2]) \equiv t[\overline{VT}^{i_1}] \geq atp \wedge t[\overline{VT}^{i_2}] \leq atp + OW \wedge t[VT] - t[\overline{VT}^{i_2}] > WW \wedge t[VT] - t[\overline{VT}^{i_2}] < WW + PW$   
-if the time-frame is anchored-

### 3.4 Defining Predictive FDs

The overall idea is now to temporally characterize functional dependencies  $X \rightarrow Y$  for the introduced multi-temporal relational model, by considering for the attribute set  $X$  those attributes related to “past” properties, while attributes  $Y$  would be those attributes related to “future” properties. “Past” and “future” values are evaluated according to a given time-frame consistency.

► **Definition 7** (Predictive Functional Dependency (PFD)). *Given an mt-relation schema  $MTR(Z\overline{U}^1\overline{U}^2 \dots \overline{U}^k \dot{U} \cup \{\overline{VT}^1, \overline{VT}^2, \dots, \overline{VT}^k, VT\})$ , a time frame, and a modality  $m \in \{\textit{flex}'', \textit{fixed}''\}$ , a Predictive Functional Dependency is expressed as:*

$$S\overline{P}^h\overline{Q}^i \dots \overline{R}^j \xrightarrow{\alpha, m} \dot{Y} \quad \text{with } 1 \leq h < i < \dots < j \leq k$$

where  $S \subseteq Z, \overline{P}^h \subseteq \overline{U}^h, \overline{Q}^i \subseteq \overline{U}^i, \overline{R}^j \subseteq \overline{U}^j$  and  $\dot{Y} \subseteq \dot{U}$  is the predicted attribute set.

A PFD holds on an *mt*-relation *mtr* with schema *MTR* in a timeframe *TF* with modality *m*, with an extended range semantics (denoted as  $mtr \models_{\alpha, m}^E \overline{S} \overline{P}^h \overline{Q}^i \dots \overline{R}^j \rightarrow \dot{Y}$ ) iff

$$\forall t, t' \in mtr((t[\overline{S} \overline{P}^h \overline{Q}^i \dots \overline{R}^j] = t'[\overline{S} \overline{P}^h \overline{Q}^i \dots \overline{R}^j] \wedge \Theta(t, \alpha, m, [1, k]) \wedge \Theta(t', \alpha, m, [1, k])) \rightarrow t[\dot{Y}] = t'[\dot{Y}])$$

A PFD holds on an *mt*-relation *mtr* with schema *MTR* in a timeframe *TF* with modality *m*, with a restricted range semantics (denoted as  $mtr \models_{\alpha, m}^R \overline{S} \overline{P}^h \overline{Q}^i \dots \overline{R}^j \rightarrow \dot{Y}$ ) iff

$$\forall t, t' \in mtr((t[\overline{S} \overline{P}^h \overline{Q}^i \dots \overline{R}^j] = t'[\overline{S} \overline{P}^h \overline{Q}^i \dots \overline{R}^j] \wedge \Theta(t, \alpha, m, [h, j]) \wedge \Theta(t', \alpha, m, [h, j])) \rightarrow t[\dot{Y}] = t'[\dot{Y}])$$

According to the previous definition, it is straightforward to observe that the given PFD has to hold, by considering only a subset of *mtr*, composed of tuples consistent with the considered time frame, the modality, and the range. Such subset is called *time-frame relation view* (*TF-view*). More formally, the *TF-view* *w* is defined as  $w = TFv(mtr, \alpha, m, [i_1, i_2]) \equiv \{t \mid t \in mtr \wedge \Theta(t, \alpha, m, [i_1, i_2])\}$ . Hereinafter, we will consider a time-frame  $\alpha = \langle 6, 2, 10 \rangle$ , *m* = 'fixed', and an extended semantics, i.e., considering the range  $[1, k]$ .

► **Example 8.** Let us consider the *mtr* depicted in Table 1. Tuples #10, #11, and #36 are out of the time frame  $\alpha$ . It is straightforward to observe that the PFD  $\overline{Drug} \xrightarrow{\alpha, m} \overline{AKI}$  holds. On the other side, PFDs  $\overline{HR}^1, \overline{SpO}_2^2 \xrightarrow{\alpha, m} \overline{AKI}$ ,  $\overline{HR}^1 \xrightarrow{\alpha, m} \overline{AKI}$  and  $\overline{SpO}_2^2 \xrightarrow{\alpha, m} \overline{AKI}$  do not hold.

### 3.5 Discovering Approximate PFDs

To mine PFDs in a generic multi-temporal relation we have first to isolate those tuples that fit, with respect to a given modality and to a given semantics, the considered temporal frame, composed of OW, WW, and PW. As a second step, we need to deal with some kind of approximation, as it could happen that some PFDs hold on a subset of tuples of the time-frame relation view, we consider. Thus, we have to evaluate whether considering such subset is acceptable with respect to the prediction task supported by the considered PFDs.

In other words, we require a PFD *f* to be satisfied by most tuples of the *TF-view* *w*. A very small portion of tuples of *w* is allowed to violate the dependency. In the context of predictive functional dependencies, we consider one of the measures proposed in [18] and introduce two other error measures, specifically tailored to the predictive purpose of approximate PFDs.

Given a *TF-view*  $w \subseteq mtr$ , the first error measure  $G_3$  considers the minimum number of tuples in *w* to be deleted to obtain a relation *s* where the given FD holds [18]. In our context, it is expressed according to the following definition.

► **Definition 9** (Error measure  $G_3$ ). *Given a TF-view*  $w = TFv(mtr, \alpha, m, [1, k])$  *of an mt-relation* *mtr* *with schema*  $Z \overline{U}^1 \overline{U}^2 \dots \overline{U}^k \dot{B} \cup \{\overline{V}T^1, \overline{V}T^2, \dots, \overline{V}T^k, VT\}$ , *and a PFD*  $\overline{S} \overline{P}^h \overline{Q}^i \dots \overline{R}^j \xrightarrow{\alpha, m} \dot{Y}$ , *where*  $S \subseteq Z, \overline{P}^h \subseteq \overline{U}^h, \overline{Q}^i \subseteq \overline{U}^i, \overline{R}^j \subseteq \overline{U}^j$  *and*  $\dot{Y} \subseteq \dot{U}$ , *and any relation*  $s \subseteq w$ , *such that*  $s \models_{\alpha, m}^E \overline{S} \overline{P}^h \overline{Q}^i \dots \overline{R}^j \rightarrow \dot{Y}$ , *the error measure*  $G_3$  *is expressed as:*  $G_3 = |w| - |s|$ . *The related scaled measurement*  $g_3$  *is defined as:*  $g_3 = \frac{G_3}{|w|}$ .

Let us now introduce some new kinds of error, which may be of interest in the context of prediction. The first issue is in considering another error, no longer focused on the number of tuples that we have to delete to satisfy the PFD, but focused on the number of entities



that we accept to discard for the sake of the PFD. The new error measure  $H_3$  permits, for example, to disregard data of entities with a very low number of tuples, which could create noise in our dataset.

► **Definition 10** (Error measure  $H_3$ ). *Given a TF-view  $w = TFv(mtr, \alpha, m, [1, k])$  of an mt-relation  $mtr$  with schema  $Z\bar{U}^1\bar{U}^2..\bar{U}^k\dot{B} \cup \{\bar{V}T^1, \bar{V}T^2, \dots, \bar{V}T^k, \dot{V}T\}$ , and a PFD  $S\bar{P}^h\bar{Q}^i \dots \bar{R}^j \xrightarrow{\alpha, m} \dot{Y}$ , where  $S \subseteq Z, \bar{P}^h \subseteq \bar{U}^h, \bar{Q}^i \subseteq \bar{U}^i, \bar{R}^j \subseteq \bar{U}^j$  and  $\dot{Y} \subseteq \dot{U}$ , and any relation  $s \subseteq w$ , such that  $s \models_{\alpha, m}^E S\bar{P}^h\bar{Q}^i \dots \bar{R}^j \rightarrow \dot{Y}$ , the error measure  $H_3$  is expressed as:  $H_3 = |\{t[Z] \mid \exists t \in w\}| - |\{t[Z] \mid \exists t \in s\}|$ . The related scaled measurement  $h_3$  is defined as:  $h_3 = \frac{H_3}{|\{t[Z] \mid \exists t \in w\}|}$ .*

Finally, considering the number of tuples for each entity we accept to discard to satisfy the PFD, we formalize a last error measure, namely  $J_3$ . It ensures to maintain enough “consistent” information for each entity.

► **Definition 11** (Error measure  $J_3$ ). *Given a TF-view  $w = TFv(mtr, \alpha, m, [1, k])$  of an mt-relation  $mtr$  with schema  $Z\bar{U}^1\bar{U}^2..\bar{U}^k\dot{B} \cup \{\bar{V}T^1, \bar{V}T^2, \dots, \bar{V}T^k, \dot{V}T\}$ , a PFD  $S\bar{P}^h\bar{Q}^i \dots \bar{R}^j \xrightarrow{\alpha, m} \dot{Y}$ , where  $S \subseteq Z, \bar{P}^h \subseteq \bar{U}^h, \bar{Q}^i \subseteq \bar{U}^i, \bar{R}^j \subseteq \bar{U}^j$  and  $\dot{Y} \subseteq \dot{U}$ , and any relation  $s \subseteq w$ , such that  $s \models_{\alpha, m}^E S\bar{P}^h\bar{Q}^i \dots \bar{R}^j \rightarrow \dot{Y}$ , the error measure  $J_3$  is expressed as in the following.*

Let  $w_{[v]} \equiv \{t[Z] \mid t \in w \wedge t[Z] = v\}$  and  $s_{[v]} \equiv \{t[Z] \mid t \in s \wedge t[Z] = v\}$ , then

$$J_3 = \max_{(v \in \{t[Z] \mid t \in s\})} \{|w_{[v]}| - |s_{[v]}|\}$$

The related scaled measurement  $j_3$  is defined as follows:

$$j_3 = \max_{(v \in \{t[Z] \mid t \in s\})} \left\{ \frac{|w_{[v]}| - |s_{[v]}|}{|w_{[v]}|} \right\}$$

According to the introduced error measures, we are now able to define an approximate predictive functional dependency as follows:

► **Definition 12** (Approximate Predictive Functional Dependency (APFD)). *Given a TF-view  $w = TFv(mtr, \alpha, m, [1, k])$  of an mt-relation  $mtr$  with schema  $Z\bar{U}^1\bar{U}^2..\bar{U}^k\dot{B} \cup \{\bar{V}T^1, \bar{V}T^2, \dots, \bar{V}T^k, \dot{V}T\}$ ,  $w$  fulfills the APFD*

$$S\bar{P}^h\bar{Q}^i \dots \bar{R}^j \xrightarrow{\alpha, m, \varepsilon} \dot{Y}$$

(written as  $w \models_{\alpha, m}^E S\bar{P}^h\bar{Q}^i \dots \bar{R}^j \xrightarrow{\varepsilon} \dot{Y}$ ), where  $\varepsilon = \langle \varepsilon_g, \varepsilon_h, \varepsilon_j \rangle$  and  $S \subseteq Z, \bar{P}^h \subseteq \bar{U}^h, \bar{Q}^i \subseteq \bar{U}^i, \bar{R}^j \subseteq \bar{U}^j, \dot{Y} \subseteq \dot{U}$ , if a relation  $s \subseteq w$  exists such that  $s \models_{\alpha, m}^E S\bar{P}^h\bar{Q}^i \dots \bar{R}^j \rightarrow \dot{Y}$  with  $g_3 \leq \varepsilon_g \wedge h_3 \leq \varepsilon_h \wedge j_3 \leq \varepsilon_j$ . In other words,  $\varepsilon_g, \varepsilon_h, \varepsilon_j$  are the maximum acceptable errors defined by the user for  $g_3, h_3$ , and  $j_3$ , respectively.

► **Example 13.** Suppose that our final goal is to preserve at least the 75% of the tuples ( $\varepsilon_g = 0.25$ ), the 80% of the patients ( $\varepsilon_h = 0.2$ ), and the 50% of the tuples for each patient ( $\varepsilon_j = 0.5$ ). In Table 1, the PFD  $\overline{HR}^1, \overline{SpO}_2^2 \xrightarrow{\alpha, m} \dot{AKI}$  is satisfied by considering a (sub)instance  $s$  by deleting tuples #2 and #9. Thus, in this case,  $g_3 = 2/9, h_3 = 1/3$ , as any tuples for patient Stevie disappear; and  $j_3 = 1/4$  as we delete a tuple of Daisy. It is easy to see that  $g_3 < \varepsilon_g, h_3 > \varepsilon_h$ , while  $j_3 < \varepsilon_j$ . On the other side, if we consider the instance

$s'$ , by deleting tuples #2 and #4, we would observe that the PFD is still satisfied, while  $g_3 = 2/9$ ,  $h_3 = 0/3$ , and  $j_3 = 2/4$ . In this case, all the errors are below or equal to the given thresholds. Thus, we can say that  $w \models_{\alpha,m}^E \overline{HR}^1, \overline{SpO}_2^2 \xrightarrow{\epsilon} \dot{AKI}$  with  $\epsilon \equiv \langle 0.35, 0.2, 0.5 \rangle$ .

If we set the error thresholds as  $\varepsilon_g = 0.25$ ,  $\varepsilon_h = 0.4$ , and  $\varepsilon_j = 0.3$  (mainly we accept to discard some more patients, but we increase the number of tuples per patient we want to preserve), we can observe that  $s \models_{\alpha,m}^E \overline{HR}^1, \overline{SpO}_2^2 \rightarrow \dot{AKI}$ , while  $s' \not\models_{\alpha,m}^E \overline{HR}^1, \overline{SpO}_2^2 \rightarrow \dot{AKI}$ . Thus,  $w \models_{\alpha,m}^E \overline{HR}^1, \overline{SpO}_2^2 \xrightarrow{\epsilon} \dot{AKI}$  also with  $\epsilon \equiv \langle 0.35, 0.4, 0.3 \rangle$ .

It is easy to prove that if  $w \models_{\alpha,m}^E \overline{SP}^h \overline{Q}^i \dots \overline{R}^j \xrightarrow{\epsilon} \dot{Y}$ , it will also hold  $w \models_{\alpha,m}^E \overline{SS}_1 \overline{P}^h \overline{P}_1^h \overline{Q}^i \overline{Q}_1^i \overline{V}^x \dots \overline{R}^j \overline{R}_1^j \xrightarrow{\epsilon} \dot{Y}$ , where  $S_1 \subseteq Z$ ,  $\overline{P}_1^h \subseteq \overline{U}^h$ ,  $\overline{Q}_1^i \subseteq \overline{U}^i$ ,  $\overline{R}_1^j \subseteq \overline{U}^j$ ,  $\overline{V}^x \subseteq \overline{U}^x$  with  $i < x < j$ .

As an example, as  $w \models_{\alpha,m}^E \overline{HR}^1, \overline{SpO}_2^2 \xrightarrow{\epsilon} \dot{AKI}$  for the *TF*-view  $w$  depicted in Table 1, it is also the case that  $w \models_{\alpha,m}^E \text{Patient}, \overline{HR}^1, \overline{SpO}_2^2 \xrightarrow{\epsilon} \dot{AKI}$ . After adding the new attribute *Patient* in the antecedent, nothing changes for mt-relation  $s \subseteq w$ , for which  $\overline{HR}^1, \overline{SpO}_2^2 \rightarrow \dot{AKI}$  holds, independently from the values of attribute *Patient*.

As we are interested in finding the minimum predictive attribute set, here we introduce the definition of minimal APFDs as follows:

► **Definition 14** (Minimal APFD). *An APFD  $\overline{SP}^h \overline{Q}^i \dots \overline{R}^j \xrightarrow{\epsilon}_{\alpha,m} \dot{Y}$  is minimal for  $w$ , if  $w \models_{\alpha,m}^E \overline{SP}^h \overline{Q}^i \dots \overline{R}^j \xrightarrow{\epsilon} \dot{Y}$  and  $\forall \overline{V} \subset \overline{SP}^h \overline{Q}^i \dots \overline{R}^j$  we have that  $w \not\models_{\alpha,m}^E \overline{V} \xrightarrow{\epsilon} \dot{Y}$ .*

Minimal APFDs provide the most compact representation of the existing dependencies.

► **Example 15.** Considering the *mt*-relation  $w$  depicted in Table 1, it is straightforward to observe that the following two APFDs hold for  $\epsilon \equiv \langle 0.25, 0.4, 0.4 \rangle$  and are minimal.

$$w \models_{\alpha,m}^E \overline{HR}^1, \overline{SpO}_2^2 \xrightarrow{\epsilon} \dot{AKI}, w \models_{\alpha,m}^E \overline{Drug}^3 \xrightarrow{\epsilon} \dot{AKI}$$

As for the minimality of the first APFD, both  $\overline{SpO}_2^2 \xrightarrow{\epsilon}_{\alpha,m} \dot{AKI}$  and  $\overline{HR}^1 \xrightarrow{\epsilon}_{\alpha,m} \dot{AKI}$  cannot satisfy the first threshold, i.e.,  $g_3 \leq 0.25$ .

## 4 The (data) complexity of deriving an APFD

As we said before, to obtain a set  $s \subseteq w$  which satisfies an APFD, we have to consider the three different thresholds.

We reduced the problem in hand to a general *3SAT* problem, showing that checking an APFD considering all the three thresholds belongs to the class *NP*.

Before starting with the theoretical analysis let us recall that an instance of *SAT* problem is a logical formula formed by a conjunction of disjunctive clauses. Namely, each clause is a disjunction of literals, and the general formula is a conjunction of disjunctive clauses. Therefore, an instance of *SAT* is a conjunction of clauses, each of them representable as a set of literals. In the specific case of *3SAT*, each clause has exactly 3 literals [23].

Let us now introduce a simple relation representing any *mt*-relation. To discuss the complexity of checking an APFD, it is enough to consider a relation having a single attribute ( $Z$ ) representing the entity attribute, a single attribute ( $A$ ) representing the antecedent, the predicted attribute ( $\dot{B}$ ). Moreover, let us assume that the domain of all attributes is  $\mathcal{N}$  or a subset of it (the predicted values for  $\dot{B}$  will be either 0 or 1, to represent boolean values). Thus, we will consider a relation  $w$  with schema  $W(A, \dot{B}, Z)$ . Before introducing the two

problems and then proving the NP-hardness of checking APFDs by a suitable reduction to an NP problem, let us introduce a simple reformulation of the satisfaction of error thresholds for  $G_3$  and  $H_3$  by a relation  $w$  in terms of conflict resolution (in the following we will make use of the standard projection operation  $\pi$  of relational algebra).

► **Definition 16.** Given a relation  $w \subset \mathbb{N}^3$ , a natural number  $0 \leq k < |w|$ , and a natural number  $0 \leq h < |\pi_Z(w)|$  we say that  $w$  admits a conflict resolution of order  $(k, h)$  if there exists a subset  $w^- \subseteq w$  such that:

1.  $|w^-| \leq k$
2. for every pair of triplets  $(a, \dot{b}, z), (a', \dot{b}', z') \in w \setminus w^-$  if  $a = a'$  then  $\dot{b} = \dot{b}'$ ;
3.  $|\pi_Z(w)| - |\pi_Z(w \setminus w^-)| \leq h$ .

According to the introduced simplified form of mt-relation and the previous definition of conflict resolution, we may now represent the problem of checking an APFD as in the following. It is worth noting that the order  $(k, h)$  of the conflict resolution represents the thresholds for errors  $G_3$  and  $H_3$ , respectively.

► **Problem 1.** Given a relation  $w \subset \mathbb{N}^3$ , a natural number  $0 \leq k < |w|$ , and a natural number  $0 \leq h < |\pi_Z(w)|$  determine whether or not  $w$  admits a *conflict resolution* of order  $(k, h)$ .

Now, we introduce the problem, well-known in the literature, we will use for the reduction.

► **Problem 2.** Given an instance  $C$  of 3SAT in which each clause features only *positive* literals,  $C = \{\{a_1^1, a_2^1, a_3^1\}, \dots, \{a_1^n, a_2^n, a_3^n\}\}$ , with variable set  $\mathcal{A} = \{a_j^i : 1 \leq i \leq n, 1 \leq j \leq 3\}$ , and a number  $0 \leq p < |C|$  determine whether or not there exists an assignment  $\sigma : \mathcal{A} \rightarrow \{0, 1\}$ <sup>1</sup> such that  $|\{i : \sigma(a_1^i) = \sigma(a_2^i) = \sigma(a_3^i)\}| \leq p$  and  $C$  is satisfied.

For the sake of brevity, given a clause  $\{a_1^i, a_2^i, a_3^i\}$  in  $C = \{\{a_1^1, a_2^1, a_3^1\}, \dots, \{a_1^n, a_2^n, a_3^n\}\}$  and an assignment  $\sigma : \mathcal{A} \rightarrow \{0, 1\}$  we say that  $\{a_1^i, a_2^i, a_3^i\}$  is homogeneous w.r.t  $\sigma$ , or simply *homogeneous* when  $\sigma$  is clear from the context, if and only if  $\sigma(a_1^i) = \sigma(a_2^i) = \sigma(a_3^i)$ . Then, Problem 2 may be equivalently redefined as: given a set of clauses  $C = \{\{a_1^1, a_2^1, a_3^1\}, \dots, \{a_1^n, a_2^n, a_3^n\}\}$  deciding whether or not there exists an assignment  $\sigma$  for the variables in  $C$  that makes  $C$  satisfied and at most  $p$  clauses of  $C$  homogeneous w.r.t  $\sigma$ .

The complexity of Problem 2 is well known, as in the following theorem.

► **Theorem 17.** *Problem 2 is NP-Complete [23].*

The following theorem proves that checking an APFD according to the introduced error thresholds is NP-hard.

► **Theorem 18.** *Problem 1 is NP-Hard.*

**Proof.** The proof is by reduction from Problem 2 and is reported in Appendix A. ◀

Proved that the Problem 1 is *NP-Hard*, it is now necessary to find a deterministic algorithm that could stop the analysis of a relation, as soon as it verifies that the relation cannot satisfy the given APFD. Algorithm 1 provides the pseudo-code of such algorithm. The general idea of this algorithm is searching for a solution considering one tuple at a time, until it is possible to generate a solution, which satisfies the selected thresholds. Throughout the

<sup>1</sup> here 0 and 1 represent the logical values false and true, respectively.

■ **Algorithm 1** DeterministicADC.

---

**Input:** an instance  $w$  of the relation  $W$ , and three real numbers  $\epsilon_{g_3}$ ,  $\epsilon_{h_3}$ , and  $\epsilon_{j_3}$  in  $[0, 1]$   
**Output:** a relation  $s \subseteq w$  s.t.  $s \models A \rightarrow B$ ,  $g_3(w, s) \geq 1 - \epsilon_{g_3}$ ,  $h_3(w, s) \geq 1 - \epsilon_{h_3}$ ,  
 $j_3(w, s) \geq 1 - \epsilon_{j_3}$

▷ Prepare data for initial call according to epsilons

```

1 begin
2    $del \leftarrow \lfloor \epsilon_{g_3} |w| \rfloor$ 
3    $count \leftarrow \epsilon_{h_3} \lfloor |\pi_Z(w)| \rfloor$ 
4   for  $z \in \pi_Z(w)$ : do
5      $thresholds[z] \leftarrow \lfloor \epsilon_{j_3} |\sigma_{Z=z}(w)| \rfloor$ 
6   return RecADC( $w, del, count, thresholds$ )
7 Function RecADC( $w, del, count, thresholds$ ):
8   ▷ This is the last recursive call before success
9   if  $w = \emptyset$  then
10    return  $\emptyset$ 
11  let  $a \in \pi_A(w)$ 
12  ▷ For each value of B
13  for  $boolean\_val \in \{0, 1\}$  do
14    ▷  $del\_tuples$ : tuples removed according to selection
15     $del\_tuples \leftarrow \sigma_{A=a \wedge B=boolean\_val}(w)$ 
16     $s \leftarrow \sigma_{A=a \wedge B=\neg boolean\_val}(w)$ 
17     $out \leftarrow \{\}$ 
18    for  $z \in \pi_Z(del\_tuples)$ : do
19       $thresholds'[z] \leftarrow thresholds[z] - |\sigma_{Z=z}(del\_tuples)|$ 
20      if  $thresholds'[z] < 0 \leq thresholds[z]$  then
21         $out \leftarrow out \cup \{z\}$ 
22    ▷  $out$ : the  $z$  groups that must disappear, since their tuples passed below
23    the threshold  $\epsilon_{j_3}$  in the current state
24    if  $count - |out| \geq 0$  then
25      ▷  $count'$ : represent the  $z$  groups still to be considered
26       $count' \leftarrow count - |out|$ 
27       $del\_tuples \leftarrow del\_tuples \cup \sigma_{Z=z:z \in out}(w)$ 
28      if  $del - |del\_tuples| \geq 0$  then
29        ▷ If the final test succeeds, we proceed with the recursive call on
30        the updated values
31         $del' \leftarrow del - |del\_tuples|$ 
32         $w' \leftarrow w \setminus (del\_tuples \cup s)$ 
33         $s' \leftarrow RecADC(w', del', count', thresholds')$ 
34        if  $s' \neq fail$  then
35          return  $s \cup s'$ 
36  return fail

```

---

code,  $w$  is the entire relation.  $del, count, thresholds$  represent the counters that control the errors.  $del$  counts the number of remaining tuples,  $count$  controls the number of remaining entities, and  $thresholds$  verifies the number of remaining tuples for each entity. After a trivial check about the (non) emptiness of relation  $w$ , for each value  $a \in \pi_A(w)$ , we try one boolean value and verify the dependency, if it fails, we try the second boolean value and verify the dependency. If both choices failed, then the algorithm fails. If one of the boolean values satisfies the thresholds, we update the counters, building at every step an intermediate relation  $s'$ , as long as the thresholds are satisfied.

## 5 Deriving APFDs: an experimental evaluation

Here, we provide some results from an experimental evaluation on real-world clinical data. We derived APFDs by using a simpler, even sub-optimal, mining algorithm.

### 5.1 Computing APFDs

As for the first experimental evaluations of the proposed approach, we adopted a sub-optimal solution, on top of the well-known TANE [13] algorithm, a popular approximate functional dependency detection algorithm, customizing it to mine only approximate functional dependencies with a fixed consequent, the predicted attribute  $\dot{Y}$ .

To find all minimal non-trivial dependencies, TANE works as follows. It starts the search from singleton sets of attributes and works its way to larger attribute sets through the set containment lattice level by level. When the algorithm is processing a set  $X$ , it tests dependencies of the form  $X \setminus A \rightarrow A$ , where  $A \in X$ . This guarantees that only non-trivial dependencies are considered. In our proposal, we compute all the Approximate Predictive Functional Dependencies, considering the three errors,  $g_3$ ,  $h_3$ ,  $j_3$ .

Given  $TF$ -view  $w$  and the predicted attribute  $\dot{Y}$ , our approach was mainly based on the following steps:

- Derive  $s$  by TANE, such that  $g_3 \leq \varepsilon_g$ ;
- Check on  $s$  that  $h_3 \leq \varepsilon_h$ ;
- If the previous check is fine, check that  $j_3 \leq \varepsilon_j$ .

It is easy to observe that this approach, while extracting APFDs that are satisfied by  $w$  according to the given thresholds, could exclude other APFDs that are associated to some  $s$ , which is not maximal, i.e., minimal with respect to  $g_3$ , but still satisfies  $g_3 \leq \varepsilon_g$ . And such  $s$  could satisfy also the other thresholds.

It is well known that the complexity of deriving AFDs is exponential in the number of attributes [13, 19], while the complexity of checking a single dependency is linear in the number of tuples (data complexity). In our experiments, even though the “maximality” of  $s$  is related to a composite error threshold  $\varepsilon = \langle \varepsilon_g, \varepsilon_h, \varepsilon_j \rangle$  and many possible relations  $s$  would be derived to evaluate a single APFD –making the data complexity higher as shown in the previous section–, the data complexity remains linear, as we rely on TANE, and check only further thresholds.

### 5.2 Dataset and data transformation

Our proposal has been applied to the clinical domain of the Intensive Care Unit (ICU) using the MIMIC III (Medical Information Mart for Intensive Care) [16] dataset, with the aim of finding significant APFDs for the AKI diagnosis. MIMIC III is a freely accessible relational database of de-identified patients, hospitalized in the intensive care units at Beth Israel Deaconess Medical Center between 2001 and 2012.

The data are associated with more than 46 000 patients and almost 60 000 admissions. The information contained in the database includes demographics, vital sign measures (such as heart rate, systolic and diastolic pressures, oxygen saturation, and body temperature) registered at the bedside, laboratory test results, administered drugs, medications and procedures.

From the original dataset, we used seven tables, transformed through an ETL (Extract, Transform, Load) process.  $D\_ITEMS$  and  $D\_LABITEMS$  were the reference tables needed to label every measure related to a patient.  $PATIENTS$  and  $ICUSTAYS$  were used to retrieve

information about the admission and discharge from the ICU and the age. *PRESCRIPTIONS* provided information about the administered medications. We mainly considered four categories: diuretics, Non-steroidal anti-inflammatory drugs (NSAID), radiocontrast agents, and angiotensin. *LABEVENTS* was used to extract information about serum creatinine and urine and *CHARTEVENTS* for heart rate, diastolic pressure and oxygen saturation. We categorized the numerical variables into “low, medium, high” according to clinical literature.

We considered two 3-window settings. The first one was characterized by an OW of 72 hours, a WW of 12 hours, and then a PW of 36 hours, where there is the (possible) onset of the illness according to one of the KDIGO criteria. The second one was characterized by an OW of 120 hours, a WW of 12 hours, and a PW of 36 hours. Starting from the literature [33], we considered six measures: creatinine, administered drugs, respiratory rate, oxygen saturation, and diastolic blood pressure. From a cohort of 50.711 patients, we considered three different *TF*-views:

- *TF*-view #1, with four states of the same measure (serum creatinine) to build a sequence of four values of a measure, where any value is the next of the preceding one (if any), within the first 3-window setting. In this case, we obtain 2546 subjects (1878 patients without AKI, 668 patients with AKI) with 3839 rows;
- *TF*-view #2, with four states of the same measure (administered drugs) to build a sequence of four values of a measure, where any value is the next of the preceding one (if any), within the second 3-window setting. In this case, we obtain 148 subjects (109 patients without AKI, 39 patients with AKI) with 1047 rows;
- *TF*-view #3 with four states, each one related to a different measure (administered drug, diastolic blood pressure, respiratory rate, oxygen saturation) with  $\overline{VT}^k = \overline{VT}^{k-1} + 1$  for  $k = 1, \dots, 3$  within the second 3-window setting. In this case, we have 413 subjects (305 patients without AKI, 108 patients with AKI) with 193.173 rows.

With the two 3-window settings, we achieved similar results. First of all, the error values were completely comparable between the two settings. Secondly, we recorded a similar trend in all the *TF*-views. Indeed, the temporal states kept dropping until the results of functional dependencies consisted of a single antecedent, with the increase of error  $\epsilon$ .

Regarding serum creatinine, our experiments suggested that creatinine needed a medium-long history to provide predictive patterns, so considering the 4 measures the difference in terms of error between functional dependencies that had more than one antecedent state, and those that had only one state, was very small. With six measures we were able to have temporal patterns formed by more than one state.

In Table 2, we reported some of the APFDs obtained through the algorithm, with the corresponding error thresholds. The algorithm took a few minutes for each *TF*-view to extract these APFDs.

During the experimental evaluation, we observed that data related to some patients are completely discarded when mining APFDs. Indeed, dealing with a large population, whatever the entity under study, it may be common to completely discard some (entity) outliers.

## 6 Conclusions

In this paper, we introduced a 3-window framework for the specification and evaluation of Approximate Predictive Functional Dependencies, dealing with the capability of exploiting data dependencies for the prediction task. The declarative framework, which we represented through relational calculus queries and formulas, allows one to consider different kinds of anchored and unanchored time windows.

■ **Table 2** APFDs from the three *TF*-views.

APFD	$\varepsilon_g$	$\varepsilon_h$	$\varepsilon_j$	<i>TF</i> -view
$\overline{Creat^1}, \overline{Creat^3} \rightarrow \dot{AKI}$	27.45%	27%	50%	#1
$\overline{Creat^1}, \overline{Creat^4} \rightarrow \dot{AKI}$	27.45%	27%	50%	#1
$\overline{Drug^1}, \overline{Drug^2}, \overline{Drug^4} \rightarrow \dot{AKI}$	21%	30%	50%	#2
$\overline{Drug^1}, \overline{Drug^2}, \overline{Drug^4} \rightarrow \dot{AKI}$	21%	30%	80%	#2
$\overline{Drug^1}, \overline{Drug^2}, \overline{Drug^3} \rightarrow \dot{AKI}$	21%	30%	80%	#2
$\overline{Drug^1}, \overline{Drug^3}, \overline{Drug^4} \rightarrow \dot{AKI}$	21%	30%	80%	#2
$\overline{Drug^1}, \overline{RespRate^3} \rightarrow \dot{AKI}$	10%	51%	75%	#3
$\overline{RespRate^3} \rightarrow \dot{AKI}$	30%	75%	75%	#3
$\overline{Drug^1} \rightarrow \dot{AKI}$	30%	75%	75%	#3
$\overline{Spo_2^4} \rightarrow \dot{AKI}$	30%	75%	75%	#3

Such dependencies have been specified with respect to three different kinds of error related to: the number of tuples to be deleted for having the corresponding PFD holding, the number of entities having all tuples deleted for having the corresponding PFD holding, and the number of tuples we admit to discard for any entity.

We also discussed the computational aspects related to the extraction of APFDs. We detailed a theoretical analysis of the complexity to derive a relation  $s \subseteq w$  considering the error thresholds  $G_3$  and  $H_3$ . We reduced the problem in hand to a general *3SAT* problem, showing that checking an APFD considering all the three thresholds belongs to the class *NP*.

We applied our approach to real clinical data, specifically to MIMIC III dataset, obtaining results that demonstrate the applicability of this new type of temporal pattern mining in medicine, but also in other contexts where the core of the problem is finding temporal patterns in the past associated, in a prediction-oriented approach, to following (future) events.

## References

- Ziawasch Abedjan, Cuneyt Gurcan Akcora, Mourad Ouzzani, Paolo Papotti, and Michael Stonebraker. Temporal rules discovery for web data cleaning. *Proc. VLDB Endow.*, 9(4):336–347, 2015. doi:10.14778/2856318.2856328.
- Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. URL: <http://webdam.inria.fr/Alice/>.
- Beatrice Amico and Carlo Combi. A 3-window framework for the discovery and interpretation of predictive temporal functional dependencies. In Martin Michalowski, Syed Sibte Raza Abidi, and Samina Abidi, editors, *Artificial Intelligence in Medicine - 20th International Conference on Artificial Intelligence in Medicine, AIME 2022, Halifax, NS, Canada, June 14-17, 2022, Proceedings*, volume 13263 of *Lecture Notes in Computer Science*, pages 299–309. Springer, 2022. doi:10.1007/978-3-031-09342-5\_29.
- Laure Berti-Équille, Hazar Harmouch, Felix Naumann, Noël Novelli, and Saravanan Thirumuranathan. Discovery of genuine functional dependencies from relational data with missing values. *Proc. VLDB Endow.*, 11(8):880–892, 2018. doi:10.14778/3204028.3204032.
- Claudio Bettini, Sushil Jajodia, and Sean Wang. *Time granularities in databases, data mining, and temporal reasoning*. Springer Science & Business Media, 2000.
- Loredana Caruccio, Vincenzo Deufemia, Felix Naumann, and Giuseppe Polese. Discovering relaxed functional dependencies based on multi-attribute dominance. *IEEE Trans. Knowl. Data Eng.*, 33(9):3212–3228, 2021. doi:10.1109/TKDE.2020.2967722.
- Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. Relaxed functional dependencies - A survey of approaches. *IEEE Trans. Knowl. Data Eng.*, 28(1):147–165, 2016. doi:10.1109/TKDE.2015.2472010.



- 8 Carlo Combi, Matteo Mantovani, Alberto Sabaini, Pietro Sala, Francesco Amaddeo, Ugo Moretti, and Giuseppe Pozzi. Mining approximate temporal functional dependencies with pure temporal grouping in clinical databases. *Comput. Biol. Medicine*, 62:306–324, 2015. doi:10.1016/j.combiomed.2014.08.004.
- 9 Carlo Combi, Angelo Montanari, and Pietro Sala. A uniform framework for temporal functional dependencies with multiple granularities. In *International Symposium on Spatial and Temporal Databases*, pages 404–421. Springer, 2011.
- 10 Carlo Combi and Pietro Sala. Mining approximate interval-based temporal dependencies. *Acta Informatica*, 53(6-8):547–585, 2016. doi:10.1007/s00236-015-0246-x.
- 11 Abdur Rahim Mohammad Forkan and Ibrahim Khalil. A clinical decision-making mechanism for context-aware and patient-specific remote monitoring systems using the correlations of multiple vital signs. *Computer methods and programs in biomedicine*, 139:1–16, 2017. doi:10.1016/j.cmpb.2016.10.018.
- 12 Chris Giannella and Edward Robertson. On approximation measures for functional dependencies. *Inf. Syst.*, 29(6):483–507, August 2004. doi:10.1016/j.is.2003.10.006.
- 13 Yka Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal*, 42(2):100–111, 1999. doi:10.1093/comjnl/42.2.100.
- 14 Christian S. Jensen and Richard T. Snodgrass. Valid time. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems, Second Edition*. Springer, 2018. doi:10.1007/978-1-4614-8265-9\_1066.
- 15 Christian S Jensen, Richard T Snodgrass, and Michael D Soo. Extending existing dependency theory to temporal databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):563–582, 1996. doi:10.1109/69.536250.
- 16 Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016. doi:10.1038/sdata.2016.35.
- 17 Arif Khwaja. Kdigo clinical practice guidelines for acute kidney injury. *Nephron Clinical Practice*, 120(4):c179–c184, 2012.
- 18 Jyrki Kivinen and Heikki Mannila. Approximate inference of functional dependencies from relations. *Theor. Comput. Sci.*, 149(1):129–149, 1995. doi:10.1016/0304-3975(95)00028-U.
- 19 Sebastian Kruse and Felix Naumann. Efficient discovery of approximate dependencies. *Proc. VLDB Endow.*, 11(7):759–772, 2018. doi:10.14778/3192965.3192968.
- 20 Ohbyung Kwon and Jae Mun Sim. Effects of data set features on the performances of classification algorithms. *Expert Systems with Applications*, 40(5):1847–1857, 2013. doi:10.1016/j.eswa.2012.09.017.
- 21 Marie Le Guilly, Jean-Marc Petit, and Vasile-Marian Scuturici. Evaluating classification feasibility using functional dependencies. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XLIV*, pages 132–159. Springer, 2020. doi:10.1007/978-3-662-62271-1\_5.
- 22 Mirjana Mazuran, Elisa Quintarelli, Letizia Tanca, and Stefania Ugolini. Semi-automatic support for evolving functional dependencies. In Evaggelia Pitoura, Sofian Maabout, Georgia Koutrika, Amélie Marian, Letizia Tanca, Ioana Manolescu, and Kostas Stefanidis, editors, *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016*, pages 293–304. OpenProceedings.org, 2016. doi:10.5441/002/edbt.2016.28.
- 23 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991. doi:10.1016/0022-0000(91)90023-X.
- 24 Parivash Pirasteh, Slawomir Nowaczyk, Sepideh Pashami, Magnus Löwenadler, Klas Thunberg, Henrik Ydreskog, and Peter Berck. Interactive feature extraction for diagnostic trouble codes



- in predictive maintenance: A case study from automotive domain. In *Proceedings of the Workshop on Interactive Data Mining*, pages 1–10, 2019. doi:10.1145/3304079.3310288.
- 25 Pietro Sala, Carlo Combi, Matteo Mantovani, and Romeo Rizzi. Discovering evolving temporal information: Theory and application to clinical databases. *SN Comput. Sci.*, 1(3):153, 2020. doi:10.1007/s42979-020-00160-9.
  - 26 Philipp Schirmer, Thorsten Papenbrock, Sebastian Kruse, Felix Naumann, Dennis Hempfing, Torben Mayer, and Daniel Neuschäfer-Rube. Dynfd: Functional dependency discovery in dynamic datasets. In Melanie Herschel, Helena Galhardas, Berthold Reinwald, Irimi Fundulaki, Carsten Binnig, and Zoi Kaoudi, editors, *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*, pages 253–264. OpenProceedings.org, 2019. doi:10.5441/002/edbt.2019.23.
  - 27 Robert W Schrier, Wei Wang, Brian Poole, Amit Mitra, et al. Acute renal failure: definitions, diagnosis, pathogenesis, and therapy. *The Journal of clinical investigation*, 114(1):5–14, 2004. doi:10.1172/JCI22353.
  - 28 Shigehiko Uchino, Rinaldo Bellomo, Donna Goldsmith, Samantha Bates, and Claudio Ronco. An assessment of the rifle criteria for acute renal failure in hospitalized patients. *Critical care medicine*, 34(7):1913–1917, 2006. doi:10.1097/01.CCM.0000224227.70642.4F.
  - 29 Victor Vianu. Dynamic functional dependencies and database aging. *Journal of the ACM (JACM)*, 34(1):28–59, 1987. doi:10.1145/7531.7918.
  - 30 Jef Wijsen. Design of temporal relational databases based on dynamic and temporal functional dependencies. In James Clifford and Alexander Tuzhilin, editors, *Recent Advances in Temporal Databases, Proceedings of the International Workshop on Temporal Databases, Zürich, Switzerland, 17-18 September 1995*, Workshops in Computing, pages 61–76. Springer, 1995. doi:10.1007/978-1-4471-3033-8\_4.
  - 31 Jef Wijsen. Temporal fds on complex objects. *ACM Trans. Database Syst.*, 24(1):127–176, 1999. doi:10.1145/310701.310715.
  - 32 Jef Wijsen. *Temporal Dependencies*, pages 3955–3961. Springer, 2018. doi:10.1007/978-1-4614-8265-9\_396.
  - 33 Zhenxing Xu, Jingyuan Chou, Xi Sheryl Zhang, Yuan Luo, Tamara Isakova, Prakash Ad-ekkanattu, Jessica S Ancker, Guoqian Jiang, Richard C Kiefer, Jennifer A Pacheco, et al. Identifying sub-phenotypes of acute kidney injury using structured and unstructured electronic health record data with memory networks. *Journal of biomedical informatics*, 102:103361, 2020. doi:10.1016/j.jbi.2019.103361.

## A Data Complexity

In this Appendix, we first provide the proof of Theorem 18 and then discuss some algorithmic issues.

**Proof of Theorem 18.** The proof is by reduction from Problem 2. Let  $C = \{\{a_1^1, a_2^1, a_3^1\}, \dots, \{a_1^n, a_2^n, a_3^n\}\}$  and  $p$  an instance of Problem 2. We introduce the following relation  $w_C = \{(a_j^i, 0, 2i) : 1 \leq i \leq n, 1 \leq j \leq 3\} \cup \{(a_j^i, 1, 2i + 1) : 1 \leq i \leq n, 1 \leq j \leq 3\}$ . It is easy to observe that  $|w_C| = 6|C|$  and  $w_C$  may be generated in polynomial space from  $C$ . Let us define a function  $clause : w_C \rightarrow \{1, \dots, n\}$  defined as:

$$clause(a_j^i, \dot{b}, z) = \begin{cases} \frac{z}{2} & \text{if } z \text{ is even} \\ \frac{(z-1)}{2} & \text{otherwise} \end{cases}.$$

Let us observe that function  $clause$  is well-defined and maps each element  $(a_j^i, \dot{b}, z) \in w_C$  to the index of the clause which corresponds to it in the above construction. Now we prove that  $(C, p)$  is a positive instance of Problem 2 if and only if  $(w_C, |w_C|, p)$  is a positive instance of Problem 1.

For the left-to-right direction, let us assume that  $C = \{\{a_1^1, a_2^1, a_3^1\}, \dots, \{a_1^n, a_2^n, a_3^n\}\}$  and  $p$  is a positive instance of Problem 2. Let  $\mathcal{A}$  be the set of all and only variables which appear in  $C$ . Thus, there exists an assignment  $\sigma : \mathcal{A} \rightarrow \{0, 1\}$  and at most  $p$  distinct indexes  $i_1, \dots, i_p$  such that  $\sigma(a_1^{i_k}) = \sigma(a_2^{i_k}) = \sigma(a_3^{i_k})$  for each  $1 \leq k \leq p$ . Let  $w_{\bar{C}} = \{(a_j^i, 1, 2i) : \sigma(a_j^i) = 0\} \cup \{(a_j^i, 0, 2i+1) : \sigma(a_j^i) = 1\}$ . Let us observe that  $w_{\bar{C}} \subseteq w_C$ . For proving that  $w_{\bar{C}}$  satisfies the three conditions of Definition 16 for the pair  $(|w_C|, p)$  we need to prove the following useful property:

**(OddEvenProperty)** for each  $1 \leq i \leq n$  we have that  $\{2i, 2i+1\} \cap \pi_Z(w_C \setminus w_{\bar{C}}) \neq \emptyset$ .

Informally speaking property (*OddEvenProperty*) states that for every possible value  $2i \in \pi_Z(w_C \setminus w_{\bar{C}})$  it is not the case that both  $2i$  and  $2i+1$  do not belong to  $\pi_Z(w_C \setminus w_{\bar{C}})$ . Let us assume by contradiction that there exists an index  $i$  with  $1 \leq i \leq n$  for which  $2i \notin \pi_Z(w_C \setminus w_{\bar{C}})$  and  $2i+1 \notin \pi_Z(w_C \setminus w_{\bar{C}})$ . Thus, for each  $j$  with  $1 \leq j \leq 3$  all the tuples of the form  $(a_j^i, 1, 2i)$  and  $(a_j^i, 0, 2i+1)$  belong to  $w_{\bar{C}}$ . Let us take any index  $j$  with  $1 \leq j \leq 3$ . We have  $(a_j^i, 1, 2i), (a_j^i, 0, 2i+1) \in w_{\bar{C}}$ . By definition of  $w_{\bar{C}}$  from  $(a_j^i, 1, 2i) \in w_{\bar{C}}$  we have that  $\sigma(a_j^i) = 0$ , and from  $(a_j^i, 0, 2i+1) \in w_{\bar{C}}$  we have that  $\sigma(a_j^i) = 1$  (contradiction).

Now we are ready to prove that conditions 1., 2., and 3. of Definition 16 are satisfied by the pair  $(w_C, |w_C|, p)$  and thus  $(w_C, |w_C|, p)$  is a positive instance of Problem 1. Condition 1. of Definition 16 imposes that  $|w_{\bar{C}}| \leq |w_C|$  which is trivially satisfied since  $w_{\bar{C}} \subseteq w_C$ . Condition 2. of Definition 16 imposes that for every pair of triplets  $(a_j^i, \dot{b}, z), (a_{j'}^{i'}, \dot{b}', z') \in w_C \setminus w_{\bar{C}}$  if  $a_j^i = a_{j'}^{i'}$ , i.e., they represent the occurrence of the same variable possibly in two distinct clauses we have  $\dot{b} = \dot{b}'$ . Let us assume by contradiction that this is not the case, then there exists  $(a_j^i, 0, z), (a_{j'}^{i'}, 1, z') \in w_C \setminus w_{\bar{C}}$  for some  $z, z' \in \{2, \dots, 2n+1\}$  with  $a_j^i = a_{j'}^{i'}$ . By definition of  $w_{\bar{C}}$  the fact that  $(a_j^i, 0, z) \in w_C \setminus w_{\bar{C}}$  means that  $\sigma(a_j^i) = 0$  while  $(a_{j'}^{i'}, 1, z') \in w_C \setminus w_{\bar{C}}$  means that  $\sigma(a_{j'}^{i'}) = 1$  since  $a_j^i = a_{j'}^{i'}$ , we have a contradiction.

Condition 3. of Definition 16 imposes that  $|\pi_Z(w_C)| - |\pi_Z(w_C \setminus w_{\bar{C}})| \leq p$ . Let us assume by contradiction that there exist  $p+1$  distinct indexes  $2 \leq i_1 < \dots < i_{p+1} \leq 2n+1$  such that  $i_j \notin \pi_Z(w_C \setminus w_{\bar{C}})$  for every  $1 \leq j \leq p+1$ . This means that for every  $1 \leq j \leq p+1$  if  $i_j$  is even (resp., odd) then  $(a_q^{i_j}, 1, i_j) \in w_{\bar{C}}$  (resp.,  $(a_q^{i_j}, 0, i_j) \in w_{\bar{C}}$ ) for each  $1 \leq q \leq 3$  and thus by definition of  $w_{\bar{C}}$  we have  $\sigma(a_q^{i_j}) = 0$  for each  $1 \leq q \leq 3$ , thus the clause  $i_j/2$  (resp.,  $(i_j-1)/2$ ) is homogeneous w.r.t to  $\sigma$ .

Since,  $\sigma$  is a “witness” that  $(C, p)$  is a positive instance of Problem1 we have that is the number of clauses homogeneous w.r.t  $\sigma$  is at most  $p$ . Since we just proved that  $2 \leq i_1 < \dots < i_{p+1} \leq 2n+1$  may be associated to  $p+1$  homogeneous clauses then there exist  $1 \leq j' < p+1$  such that  $i_{j'}$  is even and  $i_{j'+1} = i_{j'} + 1$  because at least two distinct indexes among  $i_1, \dots, i_{p+1}$  must be mapped to the same clause. However, by applying the (*OddEvenProperty*) on  $i_{j'}, i_{j'+1}$  we have that at least one among  $i_{j'}$  and  $i_{j'+1}$  must belong to  $\pi_Z(w_C \setminus w_{\bar{C}})$  and thus we have a contradiction.

For the right-to-left direction, let us assume that  $w_C$  and  $(|w_C|, p)$  is a positive instance of Problem 1. Thus, there exists  $w_{\bar{C}} \subseteq w_C$  and a function  $f : \mathcal{A}' \rightarrow \{0, 1\}$  with  $\mathcal{A}' \subseteq \mathcal{A}$  such that:

- for all  $(a, \dot{b}) \in \pi_{A\dot{B}}(w_C \setminus w_{\bar{C}})$  we have  $\dot{b} = f(a)$ ;
- $|\pi_Z(w_C)| - |\pi_Z(w_C \setminus w_{\bar{C}})| \leq p$ .

Let us assume w.l.o.g. that  $w_{\bar{C}}$  is minimal, that is for every  $(a, \dot{b}) \in \pi_{A\dot{B}}(w_{\bar{C}})$  we have that there exists  $(a, \dot{b}') \in \pi_{A\dot{B}}(w_C \setminus w_{\bar{C}})$  with  $\dot{b} \neq \dot{b}'$ . In other words, any tuple in  $\pi_{A\dot{B}}(w_{\bar{C}})$  “conflicts” with at least one tuple in  $\pi_{A\dot{B}}(w_C \setminus w_{\bar{C}})$ . Under this assumption, we may easily prove that  $\mathcal{A}' = \mathcal{A}$ . Let us assume by contradiction that  $\mathcal{A}' \subset \mathcal{A}$ . Thus, there exists  $a \in \mathcal{A} \setminus \mathcal{A}'$  such that  $(a, 0), (a, 1) \in \pi_{A\dot{B}}(w_{\bar{C}})$ . If we take  $w_{\bar{C}} = w_{\bar{C}} \setminus \{(a, 0, z) : (a, 0, z) \in w_{\bar{C}}\}$  we have

that  $w_C \setminus w_C^-$  admits a  $(|w_C|, p')$  conflict resolution with  $p' \leq p$  since, informally speaking, we are possibly “reducing” the size of  $w_C^-$ . By construction, we have that  $\{(a, 0, z) : (a, 0, z) \in w_C^-\} \neq \emptyset$  because since  $a \in \mathcal{A}$  we have that there exists at least one clause  $\{a_1^i, a_2^i, a_3^i\}$  in  $C$  for which  $a_j^i = a$  for some  $j \in \{1, 2, 3\}$  and thus  $(a, 0, 2i + 1) \in w_C$ . Thus, we can conclude that  $w_C^-$  is not minimal (contradiction). By having  $\mathcal{A}' = \mathcal{A}$  we can now claim that  $f$  is also a completely defined assignment for  $C$ . Let us prove that  $f$  is an assignment that makes at most  $p$  clauses in  $C$  homogeneous. Let us assume by contradiction that  $f$  makes at least  $p + 1$  distinct clauses homogeneous and let  $i_1 < \dots < i_{p+1}$  be the indexes of such clauses. By construction and by minimality of  $w_C^-$ , let us assume that for every  $1 \leq h \leq p + 1$  either  $(a_1^{i_h}, 0, 2i_h + 1) \in w_C \setminus w_C^-$  for every  $j \in \{1, 2, 3\}$  – in such a case  $f(a_1^{i_h}) = f(a_2^{i_h}) = f(a_3^{i_h}) = 0$ –, or  $(a_1^{i_h}, 0, 2i_h) \in w_C \setminus w_C^-$  for every  $j \in \{1, 2, 3\}$  – in such a case  $f(a_1^{i_h}) = f(a_2^{i_h}) = f(a_3^{i_h}) = 1$ . This means that for each  $1 \leq h \leq p + 1$ , if  $f(a_1^{i_h}) = f(a_2^{i_h}) = f(a_3^{i_h}) = 1$ , we have  $2i_h \in \pi_Z(w_C \setminus w_C^-)$  and  $2i_h + 1 \notin \pi_Z(w_C \setminus w_C^-)$ . Symmetrically, for each  $1 \leq h \leq p + 1$  if  $f(a_1^{i_h}) = f(a_2^{i_h}) = f(a_3^{i_h}) = 0$  we have  $2i_h \notin \pi_Z(w_C \setminus w_C^-)$  and  $2i_h + 1 \in \pi_Z(w_C \setminus w_C^-)$ . Let  $U = \{2i_1, 2i_2 + 1, \dots, 2i_{p+1}, 2i_{p+1} + 1\}$ . We can conclude that  $\pi_Z(w_C \setminus w_C^-) \cap U$  and  $\pi_Z(w_C^-) \cap U$  is a bi-partition of  $U$  with  $|\pi_Z(w_C \setminus w_C^-) \cap U| = |\pi_Z(w_C^-) \cap U| = p + 1$ . Since we have  $(\pi_Z(w_C^-) \cap U) \cap \pi_Z(w_C \setminus w_C^-) = \emptyset$  and trivially  $\pi_Z(w_C^-) \cap U \subseteq \pi_Z(w_C)$ , we have that  $(\pi_Z(w_C^-) \cap U) \subseteq (\pi_Z(w_C) \setminus \pi_Z(w_C \setminus w_C^-))$  and, thus,  $|\pi_Z(w_C^-) \cap U| = p + 1 \leq |\pi_Z(w_C)| - |\pi_Z(w_C \setminus w_C^-)|$ . Thus  $|\pi_Z(w_C)| - |\pi_Z(w_C \setminus w_C^-)| \geq p + 1$  (contradiction). ◀

As we just proved, the problem of verifying any APFD even only considering  $H_3$  is NP-Hard. Algorithm 2 represents a guess and check non-deterministic algorithm to solve the general problem, namely to verify all three errors. This algorithm shows that the verification of the three errors is an NP-complete problem. In the following algorithms, the symbol  $\triangleright$  precedes comments.

■ **Algorithm 2** ApproximateDependencyCheck.

**Input:** an instance  $w$  of relation  $W$ , and three real numbers  $\epsilon_{g_3}$ ,  $\epsilon_{h_3}$ , and  $\epsilon_{j_3}$  in  $[0, 1]$

**Output:** a relation  $s \subseteq w$  s.t.  $s \models A \rightarrow B$ ,  $g_3(w, s) \geq 1 - \epsilon_{g_3}$ ,  $h_3(w, s) \geq 1 - \epsilon_{h_3}$ ,  
 $j_3(w, s) \geq 1 - \epsilon_{j_3}$

```

1 begin
2   guess  $s \subseteq w$ 
3    $\triangleright$  Check if  $s \models A \rightarrow B$ 
4   for  $v \in \pi_A(s)$  do
5     if  $|\pi_B(\sigma_{A=v}(s))| \geq 2$  then
6       fail
7      $\triangleright$  Check  $g_3(w, s)$ 
8     if  $\frac{|s|}{|w|} < 1 - \epsilon_{g_3}$  then
9       fail
10     $\triangleright$  Check  $h_3(w, s)$ 
11    if  $\frac{|\pi_Z(s)|}{|\pi_Z(w)|} < 1 - \epsilon_{h_3}$  then
12      fail
13     $\triangleright$  Check  $j_3(w, s)$ 
14    for  $z \in \pi_Z(s)$ : do
15      if  $\frac{|\sigma_{Z=z}(s)|}{|\sigma_{Z=z}(w)|} < 1 - \epsilon_{j_3}$  then
16        fail
17    return  $s$ 

```



# Prime Scenarios in Qualitative Spatial and Temporal Reasoning

Yakoub Salhi   

CRIL UMR 8188, Université d'Artois & CNRS, France

Michael Sioutis   

LIRMM UMR 5506, Université de Montpellier & CNRS, France

---

## Abstract

The concept of prime implicant is a fundamental tool in Boolean algebra, which is used in Boolean circuit design and, recently, in explainable AI. This study investigates an analogous concept in qualitative spatial and temporal reasoning, called prime scenario. Specifically, we define a prime scenario of a qualitative constraint network (QCN) as a minimal set of decisions that can uniquely determine solutions of this QCN. We propose in this paper a collection of algorithms designed to address various problems related to prime scenarios. The first three algorithms aim to generate a prime scenario from a scenario of a QCN. The main idea consists in using path consistency to identify the constraints that can be ignored to generate a prime scenario. The next two algorithms focus on generating a set of prime scenarios that cover all the scenarios of the original QCN: The first algorithm examines every branch of the search tree, while the second is based on the use of a SAT encoding. Our last algorithm is concerned with computing a minimum-size prime scenario by using a MaxSAT encoding built from countermodels of the original QCN. We show that this algorithm is particularly useful for measuring the robustness of a QCN. Finally, a preliminary experimental evaluation is performed with instances of Allen's Interval Algebra to assess the efficiency of our algorithms and, hence, also the difficulty of the newly introduced problems here.

**2012 ACM Subject Classification** Theory of computation → Constraint and logic programming; Computing methodologies → Temporal reasoning; Computing methodologies → Spatial and physical reasoning

**Keywords and phrases** Spatial and Temporal Reasoning, Qualitative Constraints, Prime Scenario, Prime Implicant, Robustness Measurement

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.5

**Supplementary Material** *Software:* <https://seafire.lirmm.fr/d/9c0cbd2cd0954252ab96/>

**Funding** *Michael Sioutis:* The work was partially funded by the Agence Nationale de la Recherche (ANR) for the “Hybrid AI” project that is tied to the chair of Dr. Sioutis, and the I-SITE program of excellence of Université de Montpellier that complements the ANR funding.

## 1 Introduction

The role of prime implicants is pivotal in various domains, including knowledge compilation [2, 5], Boolean circuit simplification [21, 22, 17], and diagnosis [7, 28]. Additionally, many recent research works have employed prime implicants to explain decisions by compiling machine learning classifiers into Boolean circuits [30, 9, 10, 11, 4].

Qualitative Spatial and Temporal Reasoning (QSTR) focuses on reasoning about space and time using qualitative human-like descriptions, e.g.,  $x$  {is north of}  $y$ , as opposed to quantitative ones [15]. QSTR is a rich symbolic AI framework concerned with studying various types of spatial and temporal relationships, such as the relative position of objects [14], the ordering and duration of events [1], and the mereotopology of regions [23]. By employing qualitative representations, QSTR allows modeling and reasoning about complex entities and phenomena in a more flexible and intuitive way without resorting to, often prohibitively expensive, numerical precision.



© Yakoub Salhi and Michael Sioutis;  
licensed under Creative Commons License CC-BY 4.0

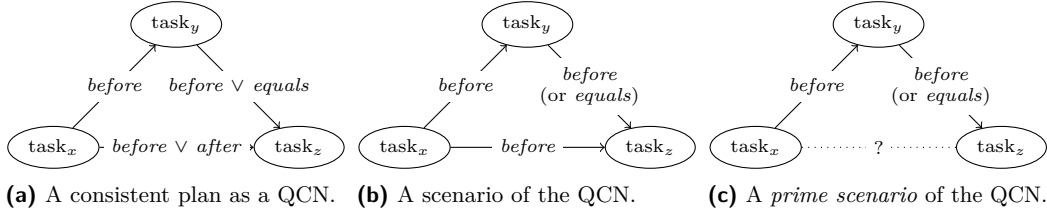
30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 5; pp. 5:1–5:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



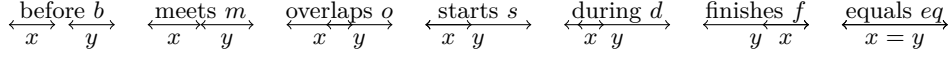
■ **Figure 1** An illustration of the knowledge compilation notion of *prime scenario* of a qualitative constraint network (QCN) (see also Definition 4); a set of prime scenarios can form a *prime scenario cover* of a QCN, for such a cover, here, we only need to additionally consider the prime scenario in Figure 1c with  $\text{task}_y \{equals\} \text{task}_z$  instead of  $\text{task}_y \{before\} \text{task}_z$ .

In this study, we introduce a novel notion, called prime scenario, that serves as the QSTR analogue of the notion of prime implicant. A prime scenario is defined as a minimal set of decisions that can only lead to solutions of the original qualitative constraint network (QCN); see Figure 1. While the notion of prime implicant shares similarities with that of prime scenarios, there are significant distinctions that hinder the direct application of prime implicant computation approaches to our context. Notably, prime scenarios are based on binary relations between variables, while prime implicants rely on truth values of variables. For instance, any literal entailed by a prime implicant belongs to that implicant; in contrast, singleton constraints entailed by prime scenarios do not have this property. To better grasp this point, consider the following constraints:  $x \{before\} y$ ,  $y \{before, equals\} z$ , and  $x \{before, after\} z$  (Figure 1a); although the two first constraints entail  $x \{before\} z$ , this constraint does not belong to the prime scenario  $\{x \{before\} y, y \{before\} z\}$  (Figure 1c): it is redundant.

It is worth mentioning that our notion of prime scenario has some relation to that of prime sub-QCN introduced in [13]. Specifically, the constraints that are not included in the prime scenario are redundant when we require the instantiated part within the prime scenario. In particular, for every *atomic* QCN, the prime scenarios are the prime sub-QCNs. Intuitively, the difference between prime scenarios and prime sub-QCNs bears a resemblance to the difference between prime implicants and the formulas resulting from the elimination of redundant clauses in propositional formulas expressed in conjunctive normal form.

To illustrate the *motivation* behind our novel work here, consider the example of machine learning classifiers that can be compiled into QCNs, much like as in the ongoing research involving Boolean circuits that we mentioned in the beginning. In this case, the solutions correspond to positive decisions, while the remaining interpretations correspond to negative ones. To explain the decisions made by these classifiers, prime scenarios can be used in a similar way as prime implicants are used to explain decisions of classifiers compiled into Boolean circuits. In particular, a prime scenario that covers a solution can be seen as a sufficient reason behind the decision associated with this solution. What is more, the notions of prime scenario and prime scenario cover that we introduce here (Figure 1), form a step towards compiling QCNs and open new avenues for research in this field: Prime scenarios can be used in the context of compilation of spatio-temporal knowledge bases, and prime scenario covers would be a classical way to perform such compilations.

With regard to the discussion above, our main *contributions* are fivefold: (i) We define the notion of prime scenario of a QCN and propose three algorithms for computing it (Section 3); (ii) we introduce and study the related problem of prime scenario cover of a QCN and present two distinct algorithms for solving it, a constraint- and a SAT-based one (Section 4); (iii) we focus on obtaining a minimum-size prime scenario of a QCN and devise a countermodel-



■ **Figure 2** A representation of the 13 base relations  $b$  of IA, each one relating two potential intervals  $x$  and  $y$  as in  $x b y$ ; the converse of  $b$ , i.e.,  $b^{-1}$ , can be denoted by  $bi$  and is omitted in the figure.

based MaxSAT encoding to tackle this task, and (iv) we show how the minimum-size prime scenarios are useful for measuring the robustness of a QCN (Section 5); and finally (v) we experimentally evaluate all our algorithms and make our code available for any interested researcher to use (Section 6).

## 2 Preliminaries

A qualitative spatial or temporal constraint language is based on a finite set  $B$  of *jointly exhaustive and pairwise disjoint* relations, called *base relations*, and defined over an infinite domain  $D$  [15] (e.g.,  $\mathbb{R}$ ). The base relations of such a language can be used to represent the definite knowledge between any two of its entities (e.g.,  $x$  contains  $y$ ). 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  $\diamond$  [15]. For all  $r \in 2^B$ , we have that  $r^{-1} = \bigcup\{b^{-1} \mid b \in r\}$ . The weak composition ( $\diamond$ ) of two base relations  $b, b' \in B$  is defined as the smallest (i.e., most restrictive) relation  $r \in 2^B$  that includes  $b \circ b'$ , or, formally,  $b \circ b' = \{b'' \in B \mid b'' \cap (b \circ b') \neq \emptyset\}$ , where  $b \circ b' = \{(x, y) \in D \times D \mid \exists z \in D \text{ such that } (x, z) \in b \wedge (z, y) \in b'\}$  is the (true) composition of  $b$  and  $b'$ . For all  $r, r' \in 2^B$ , 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) [1]. IA considers time intervals (as temporal entities) and the set of base relations  $B = \{eq (= Id), b, bi, m, mi, o, oi, s, si, d, di, f, fi\}$  to encode knowledge about the temporal relations between intervals on the real line, as described in Figure 2.

Finally, representing and reasoning about qualitative spatio-temporal information can be facilitated by a *qualitative constraint network (QCN)*; we recall the following definition:

- **Definition 1.** A qualitative constraint network (QCN) is a tuple  $(V, C)$  where:
- $V = \{v_1, \dots, v_n\}$  is a finite set of variables over some infinite domain  $D$  (e.g.,  $\mathbb{R}$ );
  - and  $C$  is a mapping  $C : V \times V \rightarrow 2^B$  associating a relation with each pair of variables s.t.  $C(v, v) = \{Id\}$  for all  $v \in V$ , and  $C(v, v') = (C(v', v))^{-1}$  for all  $v, v' \in V$ .

For convenience, we often consider that the set of variables of a QCN consists of integers, and we use  $[[\mathcal{N}]]$  to denote the set  $\{(i, j) \in V \times V : i < j\}$ .

A QCN  $\mathcal{N} = (V, C)$  is said to be *trivially inconsistent* iff  $\exists v, v' \in V$  such that  $C(v, v') = \emptyset$ .

A *solution* of a QCN  $\mathcal{N} = (V, C)$  is a mapping  $\sigma : V \rightarrow D$  such that  $\forall v, v' \in V$ ,  $\exists b \in C(v, v')$  such that  $(\sigma(v), \sigma(v')) \in b$ ;  $\mathcal{N}$  is said to be *consistent* iff it admits a solution.

A *sub-QCN*  $\mathcal{N}'$  of  $\mathcal{N}$ , denoted by  $\mathcal{N}' \subseteq \mathcal{N}$ , is a QCN  $(V, C')$  such that,  $\forall u, v \in V$ ,  $C'(u, v) \subseteq C(u, v)$ . (This term is also known as a *refined QCN* in the literature.)

A *scenario* of  $\mathcal{N}$  is a consistent atomic sub-QCN  $\mathcal{S}$  of  $\mathcal{N}$ , where a QCN  $\mathcal{S} = (V, C')$  is *atomic* iff  $\forall v, v' \in V$ ,  $|C'(v, v')| = 1$ . To refer to the set of scenarios of  $\mathcal{N}$ , we employ the notation  $\text{Scenarios}(\mathcal{N})$ .



Throughout the paper, we use the following notational conventions for a QCN  $\mathcal{N} = (V, C)$ :

- For two variables  $v, v' \in V$ , we use  $\mathcal{N}[v, v']$  to denote the relation  $C(v, v')$ .
- For two variables  $v, v' \in V$  and a relation  $r \in 2^{\mathbf{B}}$ , we use  $v r v'$  to denote that  $C(v, v') = r$  when there is no ambiguity about the considered QCN.
- For two variables  $v, v' \in V$  and a relation  $r \in 2^{\mathbf{B}}$ , we use  $\mathcal{N}_{[v, v']/r}$  to denote the result of substituting  $C(v, v')$  with  $r$  in  $\mathcal{N}$ , i.e.,  $\mathcal{N}_{[v, v']/r}$  is the QCN  $(V, C')$  defined by  $C'(v, v') = r$ ,  $C'(v', v) = r^{-1}$  and,  $\forall (u, u') \in (V \times V) \setminus \{(v, v'), (v', v)\}$ ,  $C'(u, u') = C(u, u')$ .

A *counter-scenario* of a QCN  $\mathcal{N} = (V, C)$  is a consistent atomic QCN  $\mathcal{S}$  over  $V$  that is *not* a scenario of  $\mathcal{N}$ , i.e., there exist  $i, j \in V$  such that  $\mathcal{S}[i, j] \not\subseteq \mathcal{N}[i, j]$ . We denote the set of counter-scenarios of  $\mathcal{N}$  as  $\text{CounterS}(\mathcal{N})$ .

In general, there exists only one type of QCNs that do not admit any counter-scenario: those in which every constraint is *universal*, i.e., it contains all base relations. In such cases, we use  $\mathcal{N}_{\top}$  to denote the universal QCN when the set of variables is assumed to be known, or to refer to this type of QCNs.

Given a set of variables  $V$ , we define a *q-assignment* over  $V$  as a partial function  $f$  from  $\{(i, j) : i, j \in V \text{ and } i < j\}$  to  $\mathbf{B}$ . We use  $\mathcal{N}_V^f$  to denote the QCN  $(V, C)$  defined as follows:

- for each  $(i, j) \in \text{dom}(f)$ ,  $C(i, j) = \{f(i, j)\}$ ; and
- for each  $i, j \in V$  with  $i < j$  and  $(i, j) \notin \text{dom}(f)$ ,  $C(i, j) = \mathbf{B}$ .

Given a QCN  $\mathcal{N}$ , we use  $\text{min}(\mathcal{N})$  to denote the equivalent *minimal* sub-QCN of  $\mathcal{N}$  [26], i.e., the sub-QCN that contains only the feasible base relations of the original one.

It is important to note that in this paper, we focus on calculi with the following property:

- **Note 2.** For any q-assignment  $f$  over  $V$ , the closure of  $\mathcal{N}_V^f$  under path consistency (with weak composition, or, equivalently, under *algebraic closure* [25]) yields  $\text{min}(\mathcal{N}_V^f)$ .

This property holds for many widely adopted qualitative calculi, such as IA [1] (mentioned earlier) and RCC8 [23]; a fuller listing is provided in the proof of Theorem 2 in [16].

As a direct consequence of the aforementioned property, we also have that, for any q-assignment  $f$  over  $V$ , path consistency decides the consistency of  $\mathcal{N}_V^f$ .

Given a consistent atomic QCN  $\mathcal{S} = (V, C)$ , we say that a q-assignment  $f$  over  $V$  *covers*  $\mathcal{S}$  if  $\mathcal{S}$  is a scenario of  $\mathcal{N}_V^f$ .

In the sequel, we also represent a q-assignment as a set of expressions of the form  $(i, j) \mapsto b$ :  $f$  corresponds to the set  $\{(i, j) \mapsto f(i, j) : (i, j) \in \text{dom}(f)\}$ .

### 3 Prime Scenarios

In this section, we introduce the concept of prime scenario, which can be thought of as analogous to that of prime implicant in propositional logic.

- **Definition 3** (Convergent Q-Assignment). A convergent q-assignment (CQA) of a QCN  $\mathcal{N} = (V, C)$  is a q-assignment  $\pi$  over  $V$  where (1)  $\mathcal{N}_V^\pi$  is consistent, and (2) every scenario of  $\mathcal{N}_V^\pi$  is a scenario of  $\mathcal{N}$ .

Convergent q-assignments are similar in concept to implicants in propositional logic. Property 1 states that a CQA maintains consistency, and Property 2 says that a CQA cannot lead to a scenario that does not satisfy the original QCN. By virtue of this second property,  $\pi(i, j) \in C(i, j)$  holds for every  $(i, j) \in \text{dom}(\pi)$ .

- **Definition 4** (Prime Scenario). A prime scenario of a QCN  $\mathcal{N}$  is a convergent q-assignment  $\pi$  of  $\mathcal{N}$  where for every  $D \subsetneq \text{dom}(\pi)$ ,  $\pi|_D$  is not a convergent q-assignment.



---

**Algorithm 1** FINDONEPS\_1( $\mathcal{N}, \mathcal{S}$ ).

---

```

in      : A QCN  $\mathcal{N} = (V, C)$  and a complete scenario  $\mathcal{S}$  of  $\mathcal{N}$ 
out     : A prime scenario  $\pi$  that covers  $\mathcal{S}$ 
1  $\pi \leftarrow \{(i, j) \mapsto b : (i, j) \in \llbracket \mathcal{N} \rrbracket, b \in \mathcal{S}[i, j], \mathcal{N}[i, j] \neq \mathbf{B}\}$ ;
2 for  $(i, j) \in \llbracket \mathcal{N} \rrbracket$  do
3    $\mathcal{N}' \leftarrow \text{PATHCONSISTENCY}(\mathcal{N}_{V_{[i, j]}/\mathbf{B}}^\pi)$ ;
4   if  $\mathcal{N}' \subseteq \mathcal{N}$  then
5      $\pi \leftarrow \pi|_{\text{dom}(\pi) \setminus \{(i, j)\}}$ ;
6 return  $\pi$ 

```

---

In other words, a prime scenario is a CQA that has a minimal domain (w.r.t. set inclusion). We use  $\text{PSes}(\mathcal{N})$  to denote the set of prime scenarios of  $\mathcal{N}$ .

To distinguish between prime scenarios and standard scenarios more clearly, we will refer to the latter as complete scenarios.

► **Proposition 5.** *The problem of determining whether a q-assignment is a prime scenario of a QCN is tractable.*

**Proof.** We show that we can determine whether a q-assignment is a prime scenario by linearly applying the polytime procedure of path consistency. Let  $\mathcal{N} = (V, C)$  be a QCN and  $\pi$  a q-assignment of  $\mathcal{N}$ . To determine whether  $\pi$  is a prime scenario, we first need to check that  $\mathcal{N}_V^\pi$  is consistent, which can be done using path consistency (see Note 2 and the discussion after). Using, again, path consistency, we can determine whether every complete scenario of  $\mathcal{N}_V^\pi$  is a complete scenario of  $\mathcal{N}$  (see Note 2). Indeed, we only have to show  $\mathcal{N}' \subseteq \mathcal{N}$ , where  $\mathcal{N}'$  is the result of applying path consistency on  $\mathcal{N}_V^\pi$ . Similarly, to show that  $\pi$  is minimal w.r.t. set inclusion, we can use path consistency to show that, for every  $(i, j) \in \text{dom}(\pi)$ ,  $\mathcal{N}_{ij} \not\subseteq \mathcal{N}$ , where  $\mathcal{N}_{ij}$  is the result of applying path consistency on  $\mathcal{N}_V^{\pi|_{\text{dom}(\pi) \setminus \{(i, j)\}}}$ . ◀

Let us recall that a prime implicant of a propositional formula is a minimal consistent conjunction of literals whose Boolean models are models of this formula. This definition clearly shows that prime implicants and prime scenarios are similar in concept. However, a closer examination reveals that there are significant differences between them, making the study of prime scenarios highly compelling and of great interest. First, prime scenarios are more complex structures by involving constraints and qualitative relations. Secondly, universal constraints, which are analogous to tautologies in the case of propositional logic, can be involved in prime scenarios, whereas tautologies can be simply ignored in prime implicant computation. Consider, for instance, the QCN  $\mathcal{N}$  in Point Algebra PA [33] ( $\mathbf{B} = \{<, =, >\}$ ) that corresponds to the following constraints:  $i\{<, =, >\}j$ ,  $j\{<, =, >\}k$  and  $i\{<\}k$ ; we obtain that  $\pi = \{(i, j) \mapsto <, (j, k) \mapsto <\}$  is a prime scenario of  $\mathcal{N}$  even though the two involved constraints in  $\pi$  are universal in  $\mathcal{N}$ . Thirdly, unlike entailed literals in the case of prime implicants, the singleton constraints entailed from a prime scenario do not belong to it. The prime implicants benefit significantly from this advantage, as it enables the use of unit propagation to efficiently compute them.

## Computing One Prime Scenario

The focus here is on the computation of a prime scenario that covers a given complete scenario. We propose three different algorithms that are centered around the idea of computing a prime scenario from a precomputed CQA.

■ **Algorithm 2** FINDONEPS\_2( $\mathcal{N}, \mathcal{S}$ ).

---

```

in      : A QCN  $\mathcal{N} = (V, C)$  and a complete scenario  $\mathcal{S}$  of  $\mathcal{N}$ 
out     : A prime scenario  $\pi$  that covers  $\mathcal{S}$ 
1  $\mathcal{N}' \leftarrow \mathcal{N}_\top$ ;
2  $P \leftarrow \llbracket \mathcal{N} \rrbracket$ ;
3 while  $\mathcal{N}' \not\subseteq \mathcal{N}$  do
4   | Let  $(i, j) \in P$  s.t.  $|\mathcal{N}'[i, j]| > 1$  and  $\mathcal{N}[i, j] \neq \mathbf{B}$ ;
5   |  $\mathcal{N}' \leftarrow \text{PATHCONSISTENCY}(\mathcal{N}'_{[i, j]/\mathcal{S}[i, j]})$ ;
6   |  $P \leftarrow P \setminus \{(i, j)\}$ 
7  $\pi \leftarrow \{(i, j) \mapsto b : (i, j) \in \llbracket \mathcal{N} \rrbracket \setminus P, b \in \mathcal{S}[i, j]\}$ ;
8 for  $(i, j) \in \llbracket \mathcal{N} \rrbracket \setminus P$  do
9   |  $\mathcal{N}' \leftarrow \text{PATHCONSISTENCY}(\mathcal{N}'_{V/[i, j]/\mathbf{B}})$ ;
10  | if  $\mathcal{N}' \subseteq \mathcal{N}$  then
11  | |  $\pi \leftarrow \pi|_{\text{dom}(\pi) \setminus \{(i, j)\}}$ ;
12 return  $\pi$ 

```

---

■ **Algorithm 3** FINDONEPS\_3( $\mathcal{N}, \mathcal{S}$ ).

---

```

in      : A QCN  $\mathcal{N} = (V, C)$  and a complete scenario  $\mathcal{S}$  of  $\mathcal{N}$ 
out     : A prime scenario  $\pi$  that covers  $\mathcal{S}$ 
1  $\mathcal{N}' \leftarrow \mathcal{N}$ ;
2  $min \leftarrow 1$ ;
3  $max \leftarrow n$ ;
4 while  $min \neq max$  do
5   |  $v \leftarrow (max + min)/2$ ;
6   |  $\mathcal{N}'' \leftarrow \text{PATHCONSISTENCY}(\mathcal{N}''_{[i_1, j_1]/\mathcal{S}[i_1, j_1], \dots, [i_v, j_v]/\mathcal{S}[i_v, j_v]})$ ;
7   | if  $\mathcal{N}'' \subseteq \mathcal{N}$  then
8   | |  $max \leftarrow v$ ;
9   | else
10  | |  $min \leftarrow v + 1$ ;
11  | |  $\mathcal{N}' \leftarrow \mathcal{N}''$ ;
12  $\pi \leftarrow \{(i_k, j_k) \mapsto b_k : 1 \leq k \leq min, b \in \mathcal{S}[i, j]\}$ ;
13 for  $k \in 1, \dots, min$  do
14  | if  $\text{PATHCONSISTENCY}(\mathcal{N}'_{V/[i_k, j_k]/\mathbf{B}}) \subseteq \mathcal{N}$  then
15  | |  $\pi \leftarrow \pi|_{\text{dom}(\pi) \setminus \{(i, j)\}}$ ;
16 return  $\pi$ 

```

---

Algorithm 1 starts by obtaining a CQA from a given complete scenario: its domain corresponds to the set of non-universal constraints in the original QCN. It then iterates over this CQA, applying path consistency to determine if the domain can be reduced.

Algorithm 2 begins by constructing a more compact CQA compared to Algorithm 1. It achieves this by using a while loop, which adds a constraint at each iteration using the given complete scenario until it reaches a CQA. Then, similarly to algorithm 1, it uses a for loop to compute a prime scenario from the obtained CQA.

Algorithm 3 is described by fixing  $\{(i, j) \in \llbracket \mathcal{N} \rrbracket : \mathcal{N}[i, j] \neq \mathbf{B}\} = \{(i_1, j_1), \dots, (i_n, j_n)\}$ . Similar to Algorithm 2, it starts by computing a CQA and then utilizes a for loop to obtain a prime scenario from the computed CQA. However, unlike Algorithm 2, Algorithm 3 incorporates a dichotomic search to compute a CQA, which might enable it to perform the search more efficiently.

■ **Algorithm 4** COMPUTEPSCOVER( $\mathcal{N}, \mathcal{N}', \pi$ ).

---

```

in      : Two QCNs  $\mathcal{N} = (V, C)$  and  $\mathcal{N}'(V, C')$ , and q-assignment  $\pi$  over  $V$ 
out     : A PS cover of  $\mathcal{N}$  by assigning  $\mathcal{N}_\top$  to  $\mathcal{N}'$  and  $\emptyset$  to  $\pi$ 
1  $\mathcal{N}'' \leftarrow \text{PATHCONSISTENCY}(\mathcal{N}')$ ;
2 if  $\exists (i, j) \in \llbracket \mathcal{N} \rrbracket \setminus \text{dom}(\pi), \mathcal{N}''[i, j] \cap \mathcal{N}[i, j] = \emptyset$  then
3   | return  $\emptyset$ ;
4 if  $\mathcal{N}'' \subseteq \mathcal{N}$  then
5   | return  $\{\text{FINDONEPS}(\mathcal{N}, \pi)\}$ ;
6 Let  $(i, j) \in \llbracket \mathcal{N} \rrbracket \setminus \text{dom}(\pi)$  s.t.  $\mathcal{N}''[i, j] \not\subseteq \mathcal{N}[i, j]$ ;
7  $R \leftarrow \emptyset$ ;
8 for  $b \in \mathcal{N}''[i, j] \cap \mathcal{N}[i, j]$  do
9   |  $R \leftarrow R \cup \{\text{FINDPSCOVER}(\mathcal{N}, \mathcal{N}''_{[i, j]/b}, \pi \cup \{(i, j) \mapsto b\})\}$ ;
10 return  $R$ 

```

---

By employing three distinct algorithms, we can benefit from the advantages and the strength of each approach. Our experiments have revealed that these algorithms exhibit varying levels of accuracy and efficiency for specific instances. Note that the considered approaches are similar to some approaches used in propositional logic for computing prime implicants, prime implicates, and minimal unsatisfiable cores (e.g., see [29, 18, 8]).

#### 4 Prime Scenario Cover

Prime implicant cover is a key knowledge compilation concept in the realm of Boolean circuit design, as it allows us to simplify complex Boolean functions: a function is represented as a disjunction of prime implicants that cover all its models. In this section, we investigate a similar concept in QSTR, called prime scenario cover.

We define a *prime scenario cover* of a QCN  $\mathcal{N}$  as any set  $\mathcal{C}$  of prime scenarios of  $\mathcal{N}$  such that each complete scenario of  $\mathcal{N}$  is covered by at least one element of  $\mathcal{C}$ .

A prime scenario cover provides a simplified representation of the original QCN. It can also be regarded as a compact representation of all complete scenarios of the initial QCN.

#### Computing A Prime Scenario Cover

We propose two distinct approaches for computing a prime scenario cover of a given QCN. The first approach considers every branch of the search tree to cover all scenarios, while the second is based on an encoding in the SAT problem.

#### Constraint-based Approach

Algorithm 4 generates a prime scenario cover by recursively exploring the search tree and including a prime scenario for each found CQA. To obtain a prime scenario cover, we need to invoke COMPUTEPSCOVER by assigning  $\mathcal{N}_\top$  to  $\mathcal{N}'$  and  $\emptyset$  to  $\pi$ . The code in Lines 2–3 ensures that search-subtrees without any CQA are not considered. The code in Lines 4–5 generates a prime scenario from a found CQA using one of the approaches described previously. Finally, the code in Lines 6–9 selects a constraint in the current QCN to continue exploring the search tree by making new decisions.

■ **Algorithm 5** COMPUTEPSCOVER( $\mathcal{N}$ ).

---

```

in      : A QCN  $\mathcal{N} = (V, C)$ 
out     : A PS cover  $\mathcal{C}$  of  $\mathcal{N}$ 
1  $\mathcal{C} \leftarrow \emptyset$ ;
2  $\Phi \leftarrow \text{SATEnc}(\mathcal{N})$ ;
3 while  $\text{SAT}(\Phi)$  do
4    $\pi \leftarrow \text{FINDONEPS}(\mathcal{N}, \mathcal{S}_\omega)$ ;
5    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\pi\}$ ;
6    $\Phi \leftarrow \Phi \wedge \bigvee_{(i,j) \in \text{dom}(\pi)} \neg p_{ij}^{\pi(i,j)}$ 
7 return  $\mathcal{C}$ 

```

---

### SAT-based Approach

To define our second algorithm, we use a SAT encoding of the consistency problem [19, 35]. For every  $(i, j) \in \llbracket \mathcal{N} \rrbracket$  and every  $b \in \mathbf{B}$ , we associate a distinct propositional variable  $p_{ij}^b$ . Then, we define the encoding  $\text{SATEnc}(\mathcal{N})$  as follows: (1)  $\sum_{b \in C(i,j)} p_{ij}^b = 1$  for each  $(i, j) \in \llbracket \mathcal{N} \rrbracket$ ; and (2)  $\bigwedge_{\substack{b_1 \in C(i,j) \\ b_2 \in C(j,k)}} (p_{ij}^{b_1} \wedge p_{jk}^{b_2} \rightarrow \bigvee_{b_3 \in (b_1 \diamond b_2) \cap C(i,k)} p_{ik}^{b_3})$  for every  $(i, j), (j, k) \in \llbracket \mathcal{N} \rrbracket$ .

Note that the sum constraints in Formula (1) can be linearly encoded as CNF formulas in several ways (e.g., see [31]).

For every model  $\omega$  of  $\text{SATEnc}(\mathcal{N})$ , the associated complete scenario of  $\mathcal{N}$ , denoted  $\mathcal{S}_\omega$ , is defined as follows: for every  $(i, j) \in \llbracket \mathcal{N} \rrbracket$ ,  $\mathcal{S}_\omega[i, j] = \{b : \omega(p_{ij}^b) = 1\}$ .

Algorithm 5 allows us to compute a prime scenario cover by ensuring that each newly found prime scenario covers at least one complete scenario that is not covered by the previously obtained prime scenarios. Indeed, in each iteration of the while loop, the computed complete scenario is not covered by the prime scenarios found in the previous iterations, thanks to the addition of blocking clauses in Line 6.

## 5 Minimum-Size Prime Scenarios

The minimum-size prime scenarios are those that have the smallest possible domains. We think that, like minimum-size prime implicants, minimum-size prime scenarios can be applied in various contexts. In this section, after describing our algorithm for computing minimum-size prime scenarios, we introduce a novel application by showing that these prime scenarios can be useful for analyzing and reasoning about robustness. Specifically, they can help us to define a robustness measure that provides insights into the number of critical constraints.

### Computing a Minimum-Size Prime Scenario: PMaxSAT-based Approach

Given two QCNs  $\mathcal{N}$  and  $\mathcal{N}'$  over the same set of variables  $V$ , we use  $\text{comp}(\mathcal{N}, \mathcal{N}')$  to denote the set  $\{(i, j) \mapsto b : (i, j) \in \llbracket \mathcal{N} \rrbracket \text{ and } b \in \mathcal{N}[i, j] \setminus \mathcal{N}'[i, j]\}$ .

A *hitting set* is a subset of a collection of sets that intersects with every element in the collection. A hitting set is said to be *minimal* if it cannot be reduced in size without ceasing to be a hitting set.

The following theorem shows that all prime scenarios can be obtained from the minimal hitting sets of collections of sets built from the counter-scenarios.

► **Theorem 6.** *A  $q$ -assignment  $\pi$  is a prime scenario of  $\mathcal{N}$  iff  $\pi$  is a minimal hitting set of  $\mathcal{H} = \{\text{comp}(\mathcal{N}, \mathcal{N}') : \mathcal{N}' \in \text{CounterS}(\mathcal{N})\}$  and  $\mathcal{N}_V^\pi$  is consistent.*

■ **Algorithm 6** MINIMUMSIZEPS( $\mathcal{N}$ ).

---

```

in      : A QCN  $\mathcal{N} = (V, C)$ 
out     : A minimum-size prime scenario of  $\mathcal{N}$ 
1 Let  $\mathcal{S}_0$  an arbitrary counter-scenario of  $\mathcal{N}$ ;
2  $\mathcal{H} \leftarrow \{\text{comp}(\mathcal{N}, \mathcal{S}_0)\}$ ;
3 while true do
4    $\pi \leftarrow \text{GETHS}(\text{MaxSATMH}(\mathcal{H}, \mathcal{N}))$ ;
5    $\mathcal{N}' \leftarrow \text{PATHCONSISTENCY}(\mathcal{N}_V^\pi)$ ;
6   if  $\mathcal{N}' \subseteq \mathcal{N}$  then
7     | return  $\pi$ 
8   Let  $\mathcal{S}$  be an arbitrary scenario of  $\mathcal{N}'$  where  $\mathcal{S}[i, j] \not\subseteq \mathcal{N}[i, j]$  for some  $(i, j) \in \llbracket \mathcal{N} \rrbracket$ ;
9    $\mathcal{H} \leftarrow \mathcal{H} \cup \{\text{comp}(\mathcal{N}, \mathcal{S})\}$ ;

```

---

**Proof.** First, we prove the “if” part. Let  $\pi$  be a q-assignment such that  $\mathcal{N}_V^\pi$  is consistent and  $\pi$  is a minimal hitting set of  $\mathcal{H}$ . We assume for the sake of contradiction that  $\mathcal{N}_V^\pi$  is satisfied by a counter-scenario  $\mathcal{N}'$  of  $\mathcal{N}$ . This implies that  $\pi \cap \text{comp}(\mathcal{N}, \mathcal{N}') = \emptyset$ . However, this contradicts the assumption that  $\pi$  is a hitting set of  $\mathcal{H}$ . Therefore,  $\pi$  must be a CQA of  $\mathcal{N}$ . To prove that  $\pi$  is a prime scenario, we must show that its domain is minimal w.r.t. set inclusion. This follows directly from the fact that  $\pi$  is a minimal hitting set of  $\mathcal{H}$ . Indeed, any proper subset  $\pi'$  of  $\pi$  does not hit at least one element of  $\mathcal{H}$ , which means that  $\mathcal{N}_V^{\pi'}$  is satisfied by at least one counter-scenario of  $\mathcal{N}$ . Consequently,  $\pi$  is a prime scenario of  $\mathcal{N}$ .

Now, we move to the “only if” part. Let  $\pi$  be a prime scenario of  $\mathcal{N}$ . Suppose that there is counter-scenario  $\mathcal{N}'$  of  $\mathcal{N}$  s.t.  $\pi \cap \text{comp}(\mathcal{N}, \mathcal{N}') = \emptyset$ . Thus  $\mathcal{N}'$  is a complete scenario of  $\mathcal{N}_V^\pi$ , which leads to a contradiction. Therefore,  $\pi$  is a hitting set of  $\mathcal{H}$ . Just as in the “if” part, the minimality of  $\pi$  as a hitting set is implied by its minimality as a CQA. ◀

To some extent, Theorem 6 is similar to the minimal hitting set duality between prime implicants and prime implicates in the case of propositional logic [24, 27, 20].

Our algorithm generates candidate solutions by utilizing a Partial MaxSAT encoding to compute specific minimal hitting sets. We denote this encoding by  $\text{MaxSATMH}(\mathcal{H}', \mathcal{N})$ , where  $\mathcal{N} = (V, C)$  is a QCN and  $\mathcal{H}' \subseteq \{\text{comp}(\mathcal{N}, \mathcal{N}') : \mathcal{N}' \in \text{CounterS}(\mathcal{N})\}$ . In addition to the variables used to define the  $\text{SATEnc}(\mathcal{N})$  encoding, described in Section 4, we associate a distinct propositional variable  $q_{ij}^b$  with every  $(i, j) \mapsto b \in \bigcup \mathcal{H}'$ . The hard part of  $\text{MaxSATMH}(\mathcal{H}', \mathcal{N})$  corresponds to the conjunction of  $\text{SATEnc}(\mathcal{N})$  and the following formulas: (1)  $\bigvee_{(i,j) \mapsto b \in e} q_{ij}^b$  for each  $e \in \mathcal{H}$ ; and (2)  $q_{ij}^b \rightarrow p_{ij}^b$  for each  $(i, j) \mapsto b \in \bigcup \mathcal{H}'$ .

Formula (1) guarantees that each solution of the encoding hits all elements of  $\mathcal{H}'$ , and Formula (2) forces the truth values of the variables representing a complete scenario of  $\mathcal{N}$  to match those of the variables of the form  $q_{ij}^b$ .

The soft part of  $\text{MaxSATMH}(\mathcal{H}', \mathcal{N})$  corresponds to the set of unit clauses  $\{\neg q_{ij}^b : (i, j) \mapsto b \in \bigcup \mathcal{H}'\}$ . This allows us to minimize the size of the hitting set.

Given a solution  $\omega$  of  $\text{MaxSATMH}(\mathcal{H}', \mathcal{N})$ , its associated q-assignment is  $\pi_\omega = \{(i, j) \mapsto b \in \bigcup \mathcal{H}' : \omega(q_{ij}^b) = 1\}$ . Clearly,  $\pi_\omega$  is one of the smallest hitting sets of  $\mathcal{H}'$  such that  $\mathcal{N}_V^{\pi_\omega}$  is consistent and covers a scenario of  $\mathcal{N}$ .

Theorem 6 shows that every minimum-size prime scenario  $\pi$  of  $\mathcal{N}$  is a minimum-size hitting set of  $\mathcal{H} = \{\text{comp}(\mathcal{N}, \mathcal{N}') : \mathcal{N}' \in \text{CounterS}(\mathcal{N})\}$  where (1)  $\mathcal{N}_V^\pi$  is consistent, and (2) every complete scenario of  $\mathcal{N}_V^\pi$  is a complete scenario of  $\mathcal{N}$ . Consequently, if  $\pi$  is one of the smallest hitting sets of a subset  $\mathcal{H}' \subseteq \mathcal{H}$  that satisfies Properties 1 and 2, then  $\pi$  is a minimum-size prime scenario of  $\mathcal{N}$ . This is because every hitting set of  $\mathcal{H}$  is also a hitting set of  $\mathcal{H}'$ . Algorithm 6 uses this property to generate a minimum-size prime scenario. In

each iteration of the while loop, Algorithm 6 employs the encoding  $\text{MaxSATMH}(\mathcal{H}', \mathcal{N})$  to compute  $\pi$ , one of the smallest hitting sets that satisfies Property 1 (Line 4). It then uses path consistency to check whether  $\pi$  satisfies also Property 2 (Lines 5–6). If  $\pi$  satisfies both properties, then  $\pi$  is a minimum-size prime scenario and is returned; otherwise, the algorithm adds an element obtained from a new counter-scenario of  $\mathcal{N}$  to the collection of sets  $\mathcal{H}$ . In the worst case, all counter-scenarios of  $\mathcal{N}$  will be considered in  $\mathcal{H}$ , and this necessarily allows the algorithm to obtain a minimum-size prime scenario.

Algorithm 6 shares some similarities with the approach used in [6] for solving the MaxSAT problem. This approach leverages the duality between minimal correction subsets and minimal unsatisfiable subsets.

### An Application of Minimum-Size Prime Scenarios: Robustness Measure

Now, we demonstrate one possible use of minimum-size prime scenarios in reasoning about robustness in QCNs, cf. [32] and [34]. With respect to our terminology here, QCN robustness refers to the ability of a QCN to withstand *perturbations*, i.e., eliminations of base relations, without needing to transform counter-scenarios into scenarios: the scenarios that result after perturbation are also scenarios of the original QCN. In other words, a robust QCN can maintain its consistency when facing perturbations. Although certain robustness notions have been studied in [32] and [34], robustness measures that can be used to compare different QCNs with one another have not been formalized or introduced; in fact, those notions only compare the different scenarios (or refined QCNs) with one another of a single QCN.

We define a robustness measure as a function from the set of QCNs to positive real numbers. Our robustness measure, denoted  $R_{PS}$ , is defined as follows:

$$R_{PS}(\mathcal{N}) = \max\{|\llbracket \mathcal{N} \rrbracket| - |\text{dom}(\pi)| : \pi \in \text{PSes}(\mathcal{N})\}$$

where  $\max \emptyset = 0$ . For consistent QCNs, we clearly have  $R_{PS}(\mathcal{N}) = |\llbracket \mathcal{N} \rrbracket| - \min\{|\text{dom}(\pi)| : \pi \in \text{PSes}(\mathcal{N})\}$ ; It follows that  $R_{PS}$  can be computed from any minimum-size prime scenario.

Our measure captures the fact that the robustness increases by decreasing the number of the constraints that we need to instantiate to get a complete scenario of the given QCN.

To formally establish the suitability of our robustness measure, we present a result that lists interesting properties that can be considered as necessary for any robustness measure.

► **Proposition 7.** *The following properties are satisfied:*

1. for any inconsistent QCN  $\mathcal{N}$ ,  $R_{PS}(\mathcal{N}) = 0$ ;
2.  $R_{PS}(\mathcal{N}_\top) = |\llbracket \mathcal{N}_\top \rrbracket|$ ;
3. for all two QCNs  $\mathcal{N}$  and  $\mathcal{N}'$  with  $\text{Scenarios}(\mathcal{N}) = \text{Scenarios}(\mathcal{N}')$ ,  $R_{PS}(\mathcal{N}) = R_{PS}(\mathcal{N}')$ ;
4. for all two QCNs  $\mathcal{N}$  and  $\mathcal{N}'$  with  $\text{Scenarios}(\mathcal{N}) \subseteq \text{Scenarios}(\mathcal{N}')$ ,  $R_{PS}(\mathcal{N}) \leq R_{PS}(\mathcal{N}')$ .

**Proof.** Property 1 holds since every inconsistent QCN does not admit any prime scenario. Property 2 follows from the fact that  $\pi = \emptyset$  is a prime scenario of  $\mathcal{N}_\top$ . The fact that the QCNs having the same complete scenarios have also the same prime scenarios leads to Property 3. Property 4 stems from the observation that  $\text{PSes}(\mathcal{N}) \subseteq \text{PSes}(\mathcal{N}')$  holds when  $\text{Scenarios}(\mathcal{N}) \subseteq \text{Scenarios}(\mathcal{N}')$ . ◀

The first two properties state that the minimum robustness value is associated with inconsistent QCNs, while the maximum value corresponds to QCNs where all relations are trivial, viz.,  $\mathcal{N}_\top$ . The third property ensures that identical complete scenarios lead to the same robustness value. The last property guarantees that the robustness value does not decrease as more complete scenarios are considered.

■ **Table 1** Assessing the performance of obtaining (minimum) prime scenarios, the format being  $\frac{\text{min} \mid \text{avg.}(\mu) \mid \text{max prime index}}{\text{min} \mid \text{avg.}(\mu) \mid \text{max \# of oracle calls}}$  (# of timeouts); a timeout occurs after 1200s, and it is important to note that the oracle calls for the FINDONEPS variants concern the application of path consistency, whereas the ones for MINIMUMSIZEPS the solving of a Partial MaxSAT instance.

$d$	FINDONEPS_1	FINDONEPS_2	FINDONEPS_3	MINIMUMSIZEPS
9	$\frac{0.2 \mid 0.3 \mid 0.4}{45 \mid 45.0 \mid 45}$	$\frac{0.2 \mid \mathbf{0.29} \mid 0.36}{26 \mid \mathbf{38.24} \mid 52}$	$\frac{0.2 \mid 0.3 \mid 0.38}{34 \mid 45.11 \mid 50}$	$\frac{0.2 \mid \mathbf{0.26} \mid 0.31}{0.9k \mid 2.2k \mid 4.2k}^{(34)}$
8	$\frac{0.23 \mid 0.34 \mid 0.45}{40 \mid 40.0 \mid 40}$	$\frac{0.23 \mid \mathbf{0.33} \mid 0.43}{28 \mid \mathbf{39.02} \mid 54}$	$\frac{0.23 \mid 0.34 \mid 0.45}{23 \mid 41.03 \mid 45}$	$\frac{0.23 \mid \mathbf{0.29} \mid 0.35}{1.5k \mid 2.9k \mid 5.7k}^{(45)}$
7	$\frac{0.29 \mid 0.4 \mid 0.66}{35 \mid \mathbf{35.0} \mid 35}$	$\frac{0.29 \mid \mathbf{0.39} \mid 0.66}{26 \mid 39.98 \mid 60}$	$\frac{0.29 \mid 0.4 \mid 0.57}{27 \mid 37.39 \mid 40}$	$\frac{0.26 \mid \mathbf{0.33} \mid 0.46}{1.7k \mid 3.4k \mid 5.3k}^{(64)}$
6	$\frac{0.3 \mid 0.47 \mid 0.6}{30 \mid \mathbf{30.0} \mid 30}$	$\frac{0.3 \mid 0.46 \mid 0.6}{26 \mid 39.60 \mid 54}$	$\frac{0.33 \mid \mathbf{0.46} \mid 0.63}{21 \mid 32.89 \mid 34}$	$\frac{0.3 \mid \mathbf{0.38} \mid 0.47}{2.7k \mid 4.1k \mid 5.5k}^{(85)}$
5	$\frac{0.4 \mid 0.57 \mid 0.76}{25 \mid \mathbf{25.0} \mid 25}$	$\frac{0.4 \mid 0.57 \mid 0.76}{28 \mid 37.92 \mid 46}$	$\frac{0.4 \mid \mathbf{0.57} \mid 0.8}{23 \mid 28.3 \mid 29}$	$\frac{0.36 \mid \mathbf{0.45} \mid 0.56}{2.2k \mid 4.3 \mid 6.2k}^{(88)}$
4	$\frac{0.5 \mid 0.69 \mid 0.85}{20 \mid \mathbf{20.0} \mid 20}$	$\frac{0.5 \mid \mathbf{0.69} \mid 0.85}{24 \mid 34.1 \mid 40}$	$\frac{0.5 \mid 0.7 \mid 0.9}{21 \mid 23.57 \mid 24}$	$\frac{0.45 \mid \mathbf{0.52} \mid 0.55}{3.6k \mid 5.6k \mid 7.0k}^{(97)}$
3	$\frac{0.67 \mid \mathbf{0.83} \mid 1.0}{15 \mid \mathbf{15.0} \mid 15}$	$\frac{0.67 \mid \mathbf{0.83} \mid 1.0}{22 \mid 28.14 \mid 30}$	$\frac{0.67 \mid 0.84 \mid 1.0}{16 \mid 17.96 \mid 18}$	$\frac{0.6 \mid \mathbf{0.63} \mid 0.67}{5.0k \mid 5.0k \mid 5.0k}^{(98)}$

■ **Table 2** Assessing the performance of obtaining prime scenario covers, the format being avg. # of oracle calls; it is important to note that the oracle calls for COMPUTEPSCOVER concern the application of path consistency, whereas the ones for COMPUTEPSCOVER(SAT) the solving of a SAT instance, and that avg. cover size = avg. # of oracle calls of COMPUTEPSCOVER(SAT) – 1 (each oracle call in line 3 of Algorithm 5 computes a prime scenario in the cover, minus the last one).

	$d = 9$	8	7	6	5	4	3
COMPUTEPCOVER	0.2k	0.3k	0.5k	1.0k	2.3k	3.0k	3.5k
COMPUTEPCOVER(SAT)	<b>16.05</b>	<b>25.04</b>	<b>56.21</b>	<b>0.1k</b>	<b>0.4k</b>	<b>0.7k</b>	<b>1.0k</b>

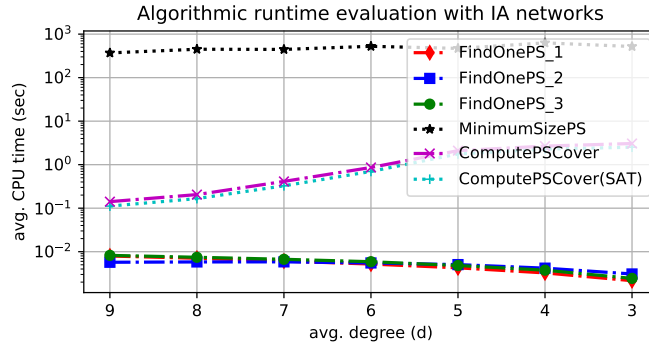
## 6 Experimentation

In this section, we perform a *preliminary* evaluation to assess the efficiency of our algorithms and, hence, also the difficulty of the introduced problems that they tackle. Our expectation is that: the FINDONEPS variants should run really fast as they involve a number of path consistency applications that is linear to the number of constraints of a QCN, the COMPUTEPSCOVER variants should run comparatively quite slower as they explore the search space of a QCN and mirror model counting algorithms, and the MINIMUMSIZEPS algorithm should be the slowest of all as it is not only dealing with finding a prime scenario for each of the exponentially many scenarios of a QCN, but one that is minimal too (there are many possibilities for a single scenario).

### Dataset, Measures, & Setup

To be able to have results that are comparable between fast polytime methods (the FINDONEPS variants) and methods for hard optimization problems (the MINIMUMSIZEPS algorithm), we consider QCNs of  $\text{IA}$  of 10 variables with a maximum of 2 base relations per non-universal constraint, for every avg. degree  $d \in (9, 8, \dots, 3)$  of their constraint graphs





■ **Figure 3** Assessing the runtime of our algorithms for the problems pertaining to prime scenarios.

(i.e., going from complete graphs to sparse ones). Specifically, we generate two arbitrary IA scenarios that we then proceed to unify; then, we create all the QCNs that result by considering one sub-graph of the initially complete constraint graph for every degree  $d$  in the aforementioned range, each with an avg. degree  $d$ . We consider 100 QCNs with an initially complete constraint graph, each yielding 6 more (sparser ones), hence a total of 700 QCNs. The size of the networks is relatively consistent with what has been used in the literature for similar optimization problems in order to present results that are as complete as possible (e.g., [3]), see also Table 1; in addition, a QCN of IA of  $n$  variables enumerates  $O(2^{n \cdot \log n})$  scenarios (qualitative solutions) [12], which translates to roughly 10 billion scenarios in our case.

All of the used measures are clear and intuitive, with the exception of *prime index*: this is the ratio of the # of non-universal constraints in a prime scenario to the # of non-universal constraints in the original QCN and, thus, takes values in  $(0, 1]$ . Clearly, the denser the network, the more opportunities there are to obtain a low measure of this type.

For the experiments we used an Intel®Core®CPU i7-12700H @ 4.70GHz, 16 GB of RAM, and the Ubuntu Linux 22.04 LTS OS. All coding/running was done in Python 3.10.6; the code is available at: <https://seafiler.lirmm.fr/d/9c0cbd2cd0954252ab96/>.

## Results & Remarks

The results are shown in Tables 1 and 2 and Figure 3, and confirm our expectations; we detail as follows. Regarding (minimum) prime scenario computation, the polytime FINDONEPS variants are extremely fast, and among those variants the simpler FINDONEPS\_1 has the best performance overall; in the case of computing a prime scenario that is also minimal, we can see that MINIMUMSIZEPS can reduce the min, avg., and max prime index values, but at a huge cost as the number of scenarios that this algorithm has to consider becomes detrimental to its runtime performance (see # of timeouts in Table 1 and runtime in Figure 3 in particular). Regarding prime scenario cover computation, the constraint-based and the SAT-based COMPUTEPSCOVER algorithms perform very similarly, with the SAT variant, viz., COMPUTEPSCOVER(SAT), performing better overall with respect to runtime performance (see Figure 3 in particular); here, we must note that we did not find any notable differences in the size of the covers that these algorithms computed (the same result applies to both, see the caption of Table 2), even though such differences may exist in general.



## 7 Conclusion and Perspectives

We introduced the novel notion of *prime scenario* to QSTR, which is analogous to that of prime implicant in the case of classical logic. In sum, we made five major contributions: first, we described three methods for computing one prime scenario; secondly, we presented two methods for computing a prime scenario cover, which is a set of prime scenarios that cover all the scenarios of a given QCN; thirdly, we proposed a method for computing a minimum-size prime scenario and, fourthly, demonstrated how this notion can be used to reason about robustness; and, fifthly, we experimentally evaluated all our algorithms and made our code available for any interested researcher to use. Our study opens up new perspectives by revealing previously unexplored ways to extend the notion of prime implicants to QSTR. Specifically, it sheds light on the possible use of prime scenarios to explain the decisions made by classifiers compiled into QCNs, in the same way as prime implicants [30, 9, 10, 11, 4], and opens new avenues for research in the field of knowledge compilation in the context of QSTR.

---

### References

- 1 James F. Allen. Maintaining Knowledge about Temporal Intervals. *Commun. ACM*, 26:832–843, 1983.
- 2 Marco Cadoli and Francesco M. Donini. A Survey on Knowledge Compilation. *AI Commun.*, 10:137–150, 1997.
- 3 Jean-François Condotta, Issam Nouaouri, and Michael Sioutis. A SAT Approach for Maximizing Satisfiability in Qualitative Spatial and Temporal Constraint Networks. In *KR*, 2016.
- 4 Adnan Darwiche and Auguste Hirth. On the (Complete) Reasons Behind Decisions. *J. Log. Lang. Inf.*, 32:63–88, 2023.
- 5 Adnan Darwiche and Pierre Marquis. A Knowledge Compilation Map. *J. Artif. Intell. Res.*, 17:229–264, 2002.
- 6 Jessica Davies and Fahiem Bacchus. Solving MAXSAT by Solving a Sequence of Simpler SAT Instances. In *CP*, 2011.
- 7 Johan de Kleer, Alan K. Mackworth, and Raymond Reiter. Characterizing Diagnoses and Systems. *Artif. Intell.*, 56:197–222, 1992.
- 8 Fred Hemery, Christophe Lecoutre, Lakhdar Sais, and Frédéric Boussemart. Extracting MUCs from Constraint Networks. In *ECAI*, 2006.
- 9 Alexey Ignatiev, Nina Narodytska, and João Marques-Silva. Abduction-Based Explanations for Machine Learning Models. In *AAAI*, 2019.
- 10 Alexey Ignatiev, Nina Narodytska, and João Marques-Silva. On Relating Explanations and Adversarial Examples. In *NeurIPS*, 2019.
- 11 Yacine Izza and João Marques-Silva. On Explaining Random Forests with SAT. In *IJCAI*, 2021.
- 12 Peter Jonsson and Victor Lagerkvist. An initial study of time complexity in infinite-domain constraint satisfaction. *Artif. Intell.*, 245:115–133, 2017.
- 13 Sanjiang Li, Zhiguo Long, Weiming Liu, Matt Duckham, and Alan Both. On redundant topological constraints. *Artif. Intell.*, 225:51–76, 2015.
- 14 Gerard Ligozat. Reasoning about cardinal directions. *J. Vis. Lang. Comput.*, 9:23–44, 1998. doi:10.1006/jvlc.1997.9999.
- 15 Gérard Ligozat. *Qualitative Spatial and Temporal Reasoning*. Iste Series. Wiley, 2011.
- 16 Zhiguo Long and Sanjiang Li. On Distributive Subalgebras of Qualitative Spatial and Temporal Calculi. In *COSIT*, 2015.
- 17 E. J. McCluskey Jr. Minimization of Boolean Functions\*. *Bell System Technical Journal*, 35:1417–1444, 1956.
- 18 Luigi Palopoli, Fiora Pirri, and Clara Pizzuti. Algorithms for Selective Enumeration of Prime Implicants. *Artif. Intell.*, 111:41–72, 1999.

- 19 Duc Nghia Pham, John Thornton, and Abdul Sattar. Modelling and solving temporal reasoning as propositional satisfiability. *Artif. Intell.*, 172:1752–1782, 2008.
- 20 Alessandro Previti, Alexey Ignatiev, António Morgado, and João Marques-Silva. Prime Compilation of Non-Clausal Formulae. In *IJCAI*, 2015.
- 21 W. V. Quine. The Problem of Simplifying Truth Functions. *The American Mathematical Monthly*, 59:521–531, 1952.
- 22 W. V. Quine. A Way to Simplify Truth Functions. *The American Mathematical Monthly*, 62:627–631, 1955.
- 23 David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions and Connection. In *KR*, 1992.
- 24 Raymond Reiter. A Theory of Diagnosis from First Principles. *Artif. Intell.*, 32:57–95, 1987.
- 25 Jochen Renz and Gérard Ligozat. Weak Composition for Qualitative Spatial and Temporal Reasoning. In *CP*, 2005.
- 26 Jochen Renz and Bernhard Nebel. Qualitative Spatial Reasoning Using Constraint Calculi. In *Handbook of Spatial Logics*, pages 161–215. Springer, 2007.
- 27 Ron Rymon. An Se-Tree-Based Prime Implicant Generation Algorithm. *Ann. Math. Artif. Intell.*, 11:351–366, 1994.
- 28 Stefan Schlobach, Zhisheng Huang, Ronald Cornet, and Frank van Harmelen. Debugging Incoherent Terminologies. *J. Autom. Reason.*, 39:317–349, 2007.
- 29 Robert Schrag. Compilation for Critically Constrained Knowledge Bases. In *AAAI*, 1996.
- 30 Andy Shih, Arthur Choi, and Adnan Darwiche. A Symbolic Approach to Explaining Bayesian Network Classifiers. In *IJCAI*, 2018.
- 31 Carsten Sinz. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In *CP*, 2005.
- 32 Michael Sioutis, Zhiguo Long, and Tomi Janhunen. On Robustness in Qualitative Constraint Networks. In *IJCAI*, 2020.
- 33 Marc Vilain, Henry Kautz, and Peter van Beek. Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report. In *Readings in Qualitative Reasoning About Physical Systems*, pages 373–381. Morgan Kaufmann, 1990.
- 34 Jan Wehner, Michael Sioutis, and Diedrich Wolter. On Robust Vs Fast Solving of Qualitative Constraints. In *ICTAI*, 2021.
- 35 Matthias Westphal and Stefan Wöflf. Qualitative CSP, Finite CSP, and SAT: Comparing Methods for Qualitative Constraint-based Reasoning. In *IJCAI*, 2009.

# Bounded-Memory Runtime Enforcement of Timed Properties

Saumya Shankar ✉ 

Indian Institute of Technology Bhubaneswar, India

Srinivas Pinisetty ✉ 

Indian Institute of Technology Bhubaneswar, India

Thierry Jéron ✉ 

Univ Rennes, Inria, IRISA, France

---

## Abstract

---

Runtime Enforcement (RE) is a monitoring technique aimed at correcting possibly incorrect executions w.r.t. a set of formal requirements (properties) of a system. In this paper, we consider enforcement monitoring of real-time properties. Thus, executions are modelled as timed words and specifications as timed automata. Moreover, we consider that the enforcer has the ability to delay events by storing or buffering them into its internal memory (and releasing them when the property is finally satisfied) and suppressing events when no delaying is appropriate. Practically, in an implementation, the internal memory of the enforcer is finite.

In this paper, we propose a new RE paradigm for timed properties, where the memory of the enforcer is bounded/finite, to address practical applications with memory constraints and timed specifications. Bounding the memory presents a number of difficulties, e.g., how to accommodate a timed event into the memory when the memory is full, s.t., regardless of the course of action we choose to handle this situation, the behaviour of the bounded enforcer should not significantly differ from that of the unbounded enforcer. The problem of how to optimally discard events when the buffer is full is significantly more difficult in a timed environment where the progress of time affects the satisfaction or violation of a property. We define the bounded-memory RE problem for timed properties and develop a framework for regular timed properties specified as timed automata. The proposed framework is implemented in Python, and its performance is evaluated. From experiments, we discovered that the enforcer has a reasonable execution time overhead.

**2012 ACM Subject Classification** Theory of computation → Logic and verification; Theory of computation → Automata over infinite objects; Software and its engineering → Software verification and validation

**Keywords and phrases** Formal methods, Runtime enforcement, Bounded-memory, Timed automata

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.6

**Funding** This work has been partially supported by The Ministry of Human Resource Development, Government of India (SPARC P#701), IIT Bhubaneswar Seed Grant (SP093).

## 1 Introduction

Runtime Enforcement (RE) [8, 11, 2, 6, 19] is a monitoring technique to ensure that a system conforms to a set of formal requirements (properties) at runtime. It does so by employing an enforcement monitor (enforcer), which modifies an untrustworthy (not satisfying the property) sequence of input events into a trustworthy (satisfying the property) sequence of output events. This transformation of an input sequence into an output sequence should be constrained by the so-called *Soundness* (the enforcer should only output correct sequences), *Transparency* (the enforcer should not modify correct input sequences), *Optimality* (actions should be released as soon as possible as output), etc. properties.



© Saumya Shankar, Srinivas Pinisetty, and Thierry Jéron;  
licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 6; pp. 6:1–6:22

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We focus on the enforcement of regular timed properties, where the enforcement of a property  $\varphi$  is done on the fly (online). The input sequence of events is made up of actions and delays between them. The general schema is as follows: an enforcer is placed between an event emitter (which emits a sequence of events  $\sigma$  as an input to the enforcer) and an event receiver (which receives the corrected sequence of events  $o$  from the enforcer). The enforcer can increase the delays between the events to satisfy the desired timed property. Thus, when a sequence of events is received by the enforcer that is currently not satisfying the property, the enforcer stores or delays those events until a time/events come that result in the satisfaction of the property. Now, for that purpose, the enforcer is equipped with an internal memory (which we refer to as a *buffer* throughout the paper). Moreover, we consider that the enforcer can also suppress input events when it determines that the input events cannot be corrected by delaying, whatever their continuation.

In usual RE mechanisms such as [8, 11], the buffer of the enforcer is regarded as infinite/unbounded. But in the case of a real implementation of the enforcer, this assumption is obviously not realistic: an internal buffer is inevitably bounded [9, 24]. Thus, a situation may arise, where an event needs to be buffered (since it is not currently satisfying the property but can satisfy the property in the future with the arrival of more events), and the buffer is full. Now, to make room for this incoming event, one can arbitrarily remove some events from the buffer, or just discard the received event. However, these naive approaches can result in deviation in behaviour from the unbounded enforcer.

[23] studies RE for untimed properties with a bounded buffer, i.e., it gives a framework where the enforcer tackles the situation when the buffer is finite in an optimal way (minimal dropping of events from the buffer (“cleaning”), minimal deviation from the unbounded enforcer). The framework in [23] (referred to as *Bounded-Memory Runtime Enforcement*) synthesised an enforcer, for a given regular property  $\varphi$ , with the maximum size of buffer  $k$ , that takes as input a word  $\sigma$  and outputs a word  $o$  that (i) satisfies  $\varphi$  (soundness), (ii) is a prefix or subword of input  $\sigma$  (transparency), (iii) the output is as long as possible (optimality) and equivalent to one produced by an unbounded-memory enforcer ( $\infty$ -compatible).

For various domains such as safety-critical systems, cyber-physical systems, and communication protocols, the correct functioning of their software systems depends crucially on real-time considerations (timing constraints and deadlines) where the time between events matters. Thus, we have timed properties to specify and verify models of real-time systems. They allow expressing constraints on the time that should elapse between (sequences of) events. Timed properties can be formally expressed using models such as *timed automata* [1]. By enforcing timed properties on these systems, we can ensure that the system behaviour adheres to these timing requirements, thereby guaranteeing its correctness and reliability.

RE allows for dynamic monitoring during execution. It provides a mechanism to detect violations of the timed properties at runtime (during execution of the system) and respond accordingly, by taking corrective actions. This enables proactive management of timing-related issues and helps prevent potential failures. Thus, various formal RE monitor synthesis approaches for timed properties, where properties are expressed as timed automata (or its variant) have been proposed [7, 15, 17, 18, 16].

**Motivation.** A framework for RE of “timed” properties with memory constraints on the buffer of the enforcer, however, has not been investigated yet, which is required since in a real-time safety-critical system, the properties are timed properties and the buffer is constrained by memory limitations.

Let us understand the usefulness of the bounded-memory enforcer for timed properties by an example. Consider a wireless sensor network in which the sensor nodes move throughout the environment working to gather and process information (e.g., temperature, humidity,

air quality, etc.) about their surroundings at specific intervals. These nodes typically consist of a low power microcontroller capable of limited information processing, sensors to capture specific data from the environment, memory to store collected data, and a radio to transmit data between nodes. Data buffers, used to handle large flows of network data, often consume large amounts of memory and therefore must be carefully allocated. So, if there are memory constraints on the whole sensor system, then the limited memory in sensor systems necessitates rethinking the data buffering model. Thus, whenever these systems are safeguarded by an enforcer which has constraints on the buffer size and we do not want to lose any random event stored in the buffer, then our work outshines others as it works on the principle of suppressing idempotent events (data) which makes insignificant differences in many cases.

Also, often the timing of the data captured by the sensor nodes is critical for maintaining temporal relevance. Consequently, this highlights the requirement for an enforcer that ensures adherence to timed properties.

**Contributions.** We introduce the first formal framework for bounded-memory runtime enforcement of timed properties modelled as timed automata<sup>1</sup>. We tackle the problem of obtaining an enforcer with memory constraints on the buffer, given a timed property. The enforcer intervenes when an execution is about to violate the property by its following abilities: delaying events in a buffer (increase the absolute dates of events of the observed input while allowing to shorten the delay between some events), releasing them when the property is finally satisfied, and suppressing (a minimal number of) events if there is no other way to avoid a property violation or a buffer overflow. The notions of soundness, monotonicity, and optimality are similar to the ones commonly used in the other RE frameworks [7]. Transparency is, however, simplified into two parts to account for cases where input events are corrected by just delaying or when suppression is also required. Moreover, we propose the notion of optimal suppression, which discards events when there is no possibility of finding a correct delay for an event.

The contributions can be summarized as follows:

1. We define an enforcer as one dedicated to a desired timed property  $\varphi$ , that transforms words.
2. We define the constraints that should be satisfied by the enforcer.
3. We present algorithms describing how the proposed enforcer can be implemented.
4. We implemented the proposed algorithms in Python and evaluated them using example properties. All our results are formalised and proven.

Due to space constraints, the algorithms and performance evaluation are provided at Appendix B, and sketches of proofs are provided at our github repository `BMRE_timed`.

## 2 Related Work

**Runtime enforcement.** *RE* is introduced in [21] by Schneider, where security policies are specified by security automata, a variant of the Büchi automaton. Later, many works extended the work of Schneider, e.g., Ligatti et al. introduced edit automata [10, 11] which

---

<sup>1</sup> Timed automata provide an explicit modeling of timing constraints and behaviour, including the ability to represent clock variables, timeouts, and synchronization mechanisms. This explicit representation makes timed automata well-suited for systems where precise timing constraints are crucial, such as real-time or embedded systems. Note that a timed temporal logic such as MTL can also be considered as an alternative for specifying properties, which can be transformed into timed automata [12].

not only recognise (truncate) the incorrect sequence of events but also correct those using edit functions. These edit functions perform suppression and insertion of input sequences (along with truncation). He also pioneered mandatory results automata [5], which have to provide a result to the target application before it can see the further action it wants to take.

**Runtime enforcement of timed properties.** The RE approaches are used to protect safety-critical systems. These are real-time systems, so the properties that need to be enforced are generally timed properties (properties with timing constraints on the sequence of events). Alur et al. in [1] pioneered *timed automata* to simulate the behaviour of real-time systems. Since then, different formal RE monitor synthesis approaches have been proposed for timed properties modelled by timed automata, e.g., [7, 15, 17, 18, 22].

**Tools for runtime monitoring of timed properties.** Several tools have been proposed to monitor the correctness of real-time systems. For example, the tool RT-MaC [20] verifies timeliness and reliability correctness properties at runtime. The Analog Monitoring Tool (AMT) [13] monitors the temporal properties of continuous signals. LARVA [4] can be used for the runtime verification of real-time properties of Java programs. The tool in [3] (implemented in Larva) presents dynamic communicating automata with timers and events to describe the properties of systems to monitor the temporal and contextual properties of Java programs. Tool TiPEX [14], as proposed in [7], implements the enforcement monitoring algorithms for timed properties.

**Runtime enforcement with memory limitations.** The approaches mentioned above do not consider any memory restrictions on the enforcer. There are a few works that take into consideration the memory limitations of the enforcer, e.g., the work by Fong in [9] and Talhi et al. in [24]. However, these approaches primarily concentrate on characterising the set of enforceable properties in a memory constrained environment. [23] introduces a framework for enforcement with bounded memory, restricted to untimed properties.

We extend the work in [23] and [7] to develop a framework for the bounded-memory RE of timed properties. To the best of our knowledge, our framework is the first to define how to synthesise an enforcer for a timed property that offers a solution when the memory of the enforcer is full at runtime.

### 3 Preliminaries and notations

We first recall some basic notions about untimed languages in Sect. 3.1. We then recall timed words and languages in Sect. 3.2 and talk about timed properties as timed automata in Sect. 3.3. At last, we introduce some preliminaries to RE of timed properties in Sect. 3.4.

#### 3.1 Untimed languages

**Languages.** A (finite) word  $w$  is a finite sequence of elements of finite alphabet  $\Sigma$ . The length of  $w$  is the number of elements in  $w$  and is denoted as  $|w|$ . The empty word over  $\Sigma$  is denoted by  $\epsilon$ . The sets of all words and non-empty words are denoted by  $\Sigma^*$  and  $\Sigma^+$  respectively. A language (property)  $\mathcal{L}$  over  $\Sigma$  is any subset of  $\Sigma^*$ .

The concatenation of two words  $w$  and  $w'$  is denoted by  $w \cdot w'$ . A word  $w'$  is a prefix of the word  $w$ , denoted  $w' \preceq w$ , whenever there exists a word  $w''$  such that  $w = w' \cdot w''$ , and  $w' \prec w$ , if additionally  $w' \neq w$ ; conversely  $w$  is said to be an extension of  $w'$ .

The set  $\text{pref}(w)$  stands for the *set of prefixes* of  $w$ . Subsequently,  $\text{pref}(\mathcal{L}) \stackrel{\text{def}}{=} \bigcup_{w \in \mathcal{L}} \text{pref}(w)$  is the set of prefixes of words in  $\mathcal{L}$ .

Given two words  $u$  and  $v$ , we define  $w = v^{-1} \cdot u$  as the residual of  $u$  by  $v$ , s.t.  $v \cdot w = u$ , if this word exists i.e., if  $v$  is a prefix of  $u$ . Intuitively,  $v^{-1} \cdot u$  is the suffix of  $u$  after reading prefix  $v$ . By extension, for a language  $\mathcal{L} \subseteq \Sigma^*$  and a word  $v \in \Sigma^*$ , the residual of  $\mathcal{L}$  by  $v$  is the language  $v^{-1} \cdot \mathcal{L} \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid v \cdot w \in \mathcal{L}\}$ . It is the set of suffixes of words which, concatenated to  $v$ , belong to  $\mathcal{L}$ . In other words,  $v^{-1} \cdot \mathcal{L}$  is the set of suffixes of words in  $\mathcal{L}$  after reading the prefix  $v$ .

A word  $w' = a_1 \dots a_n$  is a subword/subsequence of  $w$ , denoted  $w' \triangleleft w$ , if  $w'$  can be obtained by deleting some letters from  $w$  or, equivalently,  $w = w_0 a_1 w_1 \dots a_n w_n$  for some  $w_0, \dots, w_n \in \Sigma^*$ . We use the terms subword and subsequence interchangeably.

For a word  $w$  and  $i \in [1, |w|]$ , the  $i$ -th letter of  $w$  is denoted by  $w_{[i]}$ . Given a word  $w$  and integers  $i, j$ , s.t.  $1 \leq i \leq j \leq |w|$ , the suffix of word  $w$  starting from index  $i$  is denoted by  $w_{[i \dots]}$  and the subword from index  $i$  to  $j$  by  $w_{[i \dots j]}$ .

Given a  $n$ -tuple of symbols  $e = (e_1, \dots, e_n)$ , for  $i \in [1, n]$ , we define  $\Pi_i(e)$  as the projection of  $e$  on its  $i$ -th element ( $\Pi_i(e) = e_i$ ).

### 3.2 Timed words and languages

Let  $\Sigma$  be a finite alphabet of actions and  $\mathbb{R}_{\geq 0}$  denotes the set of non-negative real numbers. An event is a tuple  $(t, a) \in (\mathbb{R}_{\geq 0} \times \Sigma)$ , where  $\text{date}(t, a) \stackrel{\text{def}}{=} t$  is the absolute time instant at which action  $\text{act}(t, a) \stackrel{\text{def}}{=} a$  occurs.

A timed word over  $\Sigma$  denoted  $\sigma = (t_1, a_1) \cdot (t_2, a_2) \cdots (t_n, a_n)$ , is a finite sequence of non-decreasing events ( $i \leq j \implies t_i \leq t_j$ ), ranging over  $(\mathbb{R}_{\geq 0} \times \Sigma)^*$ . We denote the starting date of  $\sigma$  by  $\text{start}(\sigma) \stackrel{\text{def}}{=} t_1$ , and its ending date by  $\text{end}(\sigma) \stackrel{\text{def}}{=} t_n$  (with the convention that the starting and ending dates for the empty timed word  $\epsilon$  are equal to 0).

The set of timed words over  $\Sigma$  is denoted by  $\text{tw}(\Sigma)$ . A timed language is any subset  $\mathcal{L} \subseteq \text{tw}(\Sigma)$ .

Note that, even though the alphabet  $(\mathbb{R}_{\geq 0} \times \Sigma)$  is infinite in this case, previous notions and notations defined in the untimed case (related to length, prefix, subword/subsequence, etc.) naturally extend to timed words.

Concatenating timed words, on the other hand, takes greater care because, when concatenating two timed words, one must make sure that the result is a timed word, i.e., dates must not be decreasing. When we observe that the ending date of the first timed word does not exceed the starting date of the second one, this is ensured. Formally, let  $\sigma = (t_1, a_1) \cdots (t_n, a_n)$  and  $\sigma' = (t'_1, a'_1) \cdots (t'_m, a'_m)$  be two timed words with  $\text{end}(\sigma) \leq \text{start}(\sigma')$ <sup>2</sup>, their concatenation is  $\sigma \cdot \sigma' \stackrel{\text{def}}{=} (t_1, a_1) \cdots (t_n, a_n) \cdot (t'_1, a'_1) \cdots (t'_m, a'_m)$ . By convention  $\sigma \cdot \epsilon = \epsilon \cdot \sigma = \sigma$ . Concatenation is undefined otherwise.

The untimed projection of  $\sigma$  is  $\Pi_{\Sigma}(\sigma) \stackrel{\text{def}}{=} a_1 \cdot a_2 \cdots a_n \in \Sigma^*$ , i.e., dates are ignored.

### 3.3 Timed properties as timed automata

A timed automaton is a model used to specify the properties of a series of events where the timing between the events matters. In this section, we introduce timed automata as a formalism for specifying timed properties.

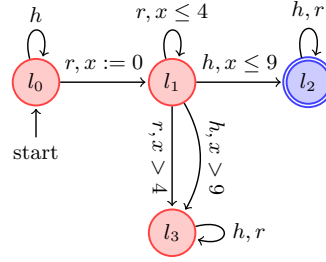
<sup>2</sup> Throughout the paper, when we consider concatenation of two timed words  $\sigma$  and  $\sigma'$  as  $\sigma \cdot \sigma'$ , we assume that  $\text{end}(\sigma) \leq \text{start}(\sigma')$ .



**Timed automata** . A Timed Automaton (TA) [1] is a finite automaton extended with a finite set of real valued clocks. A clock valuation for  $X$ , where  $X = \{x_1, \dots, x_k\}$  is a finite set of clocks, is an element of  $\mathbb{R}_{\geq 0}^X$ , i.e., a function from  $X$  to  $\mathbb{R}_{\geq 0}^X$ .  $v + \delta$  is the valuation assigning  $v(x) + \delta$  to each clock  $x$  of  $X$ , where  $v \in \mathbb{R}_{\geq 0}^X$  and  $\delta \in \mathbb{R}_{\geq 0}$  (delay since previous action). Given a set of clocks  $X' \subseteq X$ ,  $v[X' \leftarrow 0]$  is the clock valuation  $v$  where all clocks in  $X'$  are assigned to 0.  $\mathcal{G}(X)$  denotes the set of guards. These are clock constraints defined as Boolean combinations of simple constraints of the form  $x \bowtie c$  with  $x \in X$ ,  $c \in \mathbb{N}$  and  $\bowtie \in \{<, \leq, =, \geq, >\}$ . Given  $g \in \mathcal{G}(X)$  and  $v \in \mathbb{R}_{\geq 0}^X$ , we denote  $v \models g$  when  $g$  holds according to  $v$ .

► **Definition 1** (Timed automata). *TA is a tuple  $\mathcal{A} = (L, l_0, X, \Sigma, \Delta, F)$ , s.t.  $L$  is a finite set of locations.  $l_0 \in L$  is initial location.  $X$  is a finite set of clocks.  $\Sigma$  is a finite set of actions.  $\Delta \subseteq L \times \mathcal{G}(X) \times \Sigma \times 2^X \times L$  is the transition relation.  $F \subseteq L$  is the set of accepting locations.*

► **Example 2** (Timed automata). The automaton  $\mathcal{A}_P$  in Figure 1 denotes a timed automaton of a prototype property  $P$ , with  $L = \{l_0, l_1, l_2, l_3\}$  as the set of locations,  $l_0$  the initial location, and  $l_2$  the accepting location (denoted by double circles). The alphabet of events is  $\Sigma = \{h, r\}$ . The automaton has one clock  $x$ .



■ **Figure 1**  $\mathcal{A}_P$ .

The transitions can be understood as follows: from initial location  $l_0$  and on reception of input action  $h$ ,  $\mathcal{A}_P$  remains at the same location. It makes a transition to location  $l_1$  with the clock  $x$  being reset (to keep an eye on the reception of next  $r$  or  $h$  action) on reception of input action  $r$ . From location  $l_1$ , if action  $h$  is received within 9 t.u., then  $\mathcal{A}_P$  makes a transition to the accepting location  $l_2$ , otherwise goes to violating (non-accepting) location  $l_3$ . Moreover, from location  $l_1$ , if action  $r$  is received within 4 t.u.,  $\mathcal{A}_P$  remains at the same location  $l_1$ , otherwise goes to location  $l_3$ . From locations  $l_2$  and  $l_3$ , on input actions  $\{h, r\}$ ,  $\mathcal{A}_P$  remains at same respective locations.

► **Definition 3** (Semantics of TA). *The semantics of a TA is a timed transition system  $\llbracket \mathcal{A} \rrbracket = (Q, q_0, \Gamma, \rightarrow, Q_F)$  where the (infinite) set of states is given by  $Q = L \times \mathbb{R}_{\geq 0}^X$ .  $q_0 = (l_0, v_0)$  is the initial state where  $v_0$  is the valuation that maps each clock in  $X$  to 0. The set of accepting states is given by  $Q_F = F \times \mathbb{R}_{\geq 0}^X$ . The set of transition labels is given by  $\Gamma = \mathbb{R}_{\geq 0} \times \Sigma$ . A label is a pair consisting of a delay and an action. The transition relation  $\rightarrow \subseteq Q \times \Gamma \times Q$  is a set of transitions of the form  $t_r : (l, v) \xrightarrow{(\delta, a)} (l', v')$  with  $v' = (v + \delta)[Y \leftarrow 0]$  whenever there exists  $(l, g, a, Y, l') \in \Delta$  s.t.  $v + \delta \models g$  for  $\delta \in \mathbb{R}_{\geq 0}$ .*

**Deterministic and complete TA.**  $\mathcal{A}$  is *deterministic* whenever for any two distinct transitions  $(l, g_1, a, Y_1, l'_1)$  and  $(l, g_2, a, Y_2, l'_2) \in \Delta$ ,  $g_1 \wedge g_2$  is unsatisfiable.  $\mathcal{A}$  is *complete* whenever for any location  $l \in L$  and an action  $a \in \Sigma$ , the disjunction of the guards of the transitions leaving  $l$  and labelled by  $a$  evaluates to *true*.



In this work, we only consider deterministic and complete TA [1, 7]. So, wherever we say a TA, it refers to a deterministic and complete TA.

A run  $\rho$  from  $q \in Q$  is a sequence of moves in  $\llbracket \mathcal{A} \rrbracket : \rho = q \xrightarrow{(\delta_1, a_1)} q_1 \cdots q_{n-1} \xrightarrow{(\delta_n, a_n)} q_n$ , for some  $n \in N$ , where the trace of a run  $\rho$  is the timed word  $(t_1, a_1) \cdot (t_2, a_2) \cdots (t_n, a_n)$ , (where the date  $t_n$  of action  $a_n$  is the sum of all the delays i.e.,  $\sum_{i=1}^n \delta_i$ ). The set of runs from  $q_0 \in Q$  is denoted by  $Run(\mathcal{A})$ . The subset of runs accepted by  $\mathcal{A}$ , i.e., when  $q_n \in F_G$  is denoted by  $Run_{F_G}(\mathcal{A})$ .  $\mathcal{L}(\mathcal{A})$  denotes the language accepted by  $\mathcal{A}$  from its initial state  $q_0$ , whereas  $\mathcal{L}(\mathcal{A}, q)$  denotes the language accepted by  $\mathcal{A}$  from state  $q$ .

### 3.4 Preliminaries to runtime enforcement of timed properties

The following preliminaries will be useful for RE.

- **obs**( $\sigma, t$ ): Given  $t \in \mathbb{R}_{\geq 0}$ , and  $\sigma \in tw(\Sigma)$ , the observation of  $\sigma$  at date  $t$  (i.e.,  $obs(\sigma, t)$ ) is the maximal prefix of  $\sigma$  that can be observed at date  $t$ . Formally,

$$obs(\sigma, t) \stackrel{\text{def}}{=} \max_{\preceq} \{ \sigma' \in (\mathbb{R}_{\geq 0} \times \Sigma)^* \mid \sigma' \preceq \sigma \wedge \text{end}(\sigma') \leq t \}.$$

- **Delaying order**  $=_d$ : For  $\sigma, \sigma' \in tw(\Sigma)$ ,  $\sigma'$  delays  $\sigma$  (noted  $\sigma' =_d \sigma$ ) iff they have the same untimed projection but the dates of events in  $\sigma'$  is greater than or equal to the dates of corresponding events in  $\sigma$ . Formally:

$$(\sigma' =_d \sigma) \stackrel{\text{def}}{=} (\Pi_{\Sigma}(\sigma') = \Pi_{\Sigma}(\sigma)) \wedge \forall i \in [1, |\sigma|] : \text{date}(\sigma'_{[i]}) \geq \text{date}(\sigma_{[i]}).$$

Sequence  $\sigma'$  is obtained from  $\sigma$  by keeping all actions, but with a potential increase in dates. For example,  $(4, a) \cdot (7, b) \cdot (9, c) =_d (3, a) \cdot (5, b) \cdot (8, c)$ . Note that delays between events may be decreased (e.g., between  $b$  and  $c$ ), but absolute dates are increased.

- **Delayed prefix**  $\preceq_d$ : For  $\sigma, \sigma' \in tw(\Sigma)$ , we say that  $\sigma'$  is a delayed prefix of  $\sigma$  (noted  $\sigma' \preceq_d \sigma$ ) iff the untimed projection of  $\sigma'$  is a prefix of the untimed projection of  $\sigma$  and the dates of events in  $\sigma'$  is greater than or equal to the dates of corresponding events in  $\sigma$ . Formally:

$$(\sigma' \preceq_d \sigma) \stackrel{\text{def}}{=} (\Pi_{\Sigma}(\sigma') \preceq \Pi_{\Sigma}(\sigma)) \wedge \forall i \in [1, |\sigma'|] : \text{date}(\sigma'_{[i]}) \geq \text{date}(\sigma_{[i]}).$$

For example,  $(4, a) \cdot (7, b) \preceq_d (3, a) \cdot (5, b) \cdot (8, c)$ .

- **Delaying subsequence order**  $\triangleleft_d$ : For  $\sigma, \sigma' \in tw(\Sigma)$ , we say that  $\sigma'$  is a delayed subword/subsequence of  $\sigma$  (noted  $\sigma' \triangleleft_d \sigma$ ) iff there exists a subsequence  $\sigma''$  of  $\sigma$  such that  $\sigma'$  delays  $\sigma''$ . Formally:  $(\sigma' \triangleleft_d \sigma) \stackrel{\text{def}}{=} \{ \exists \sigma'' \in tw(\Sigma) : (\sigma' =_d \sigma'' \wedge \sigma'' \triangleleft \sigma) \}$ .

Sequence  $\sigma'$  is obtained from  $\sigma$  by first suppressing some actions and then increasing the dates of the actions that are kept. For example,  $(5, a) \cdot (10, c) \triangleleft_d (3, a) \cdot (5, b) \cdot (8, c)$  (where event  $(5, b)$  has been suppressed, while  $a$  and  $c$  are shifted in time).

- **Lexical order**  $\preceq_{lex}$ : For  $\sigma, \sigma' \in tw(\Sigma)$ , with same untimed projection (i.e.,  $\Pi_{\Sigma}(\sigma) = \Pi_{\Sigma}(\sigma')$ ), the order  $\preceq_{lex}$  is defined inductively as follows:  $\epsilon \preceq_{lex} \epsilon$ , and for two events with identical actions  $(\delta, a)$  and  $(\delta', a)$ ,  $(\delta, a) \cdot \sigma \preceq_{lex} (\delta', a) \cdot \sigma'$  if  $\delta < \delta' \vee (\delta = \delta' \wedge \sigma \preceq_{lex} \sigma')$ . This order is useful to select a unique timed word from a group of words with same untimed projection. For example  $(4, a) \cdot (5, b) \cdot (8, c) \cdot (11, d) \preceq_{lex} (4, a) \cdot (5, b) \cdot (9, c) \cdot (10, d)$ .

- **Choosing a unique timed word with minimal duration**  $min_{\preceq_{lex}, end}$ :

Given a set of timed words having the same untimed projection,  $min_{\preceq_{lex}, end}$  chooses a timed word among timed words with minimal ending date w.r.t. the lexical order: first the set of timed words with minimal ending date are considered, and then, from these timed words, the (unique) minimal one is selected w.r.t. the lexical order. Formally, for a set  $E \subseteq tw(\Sigma)$  such that  $\forall \sigma, \sigma' \in E : \Pi_{\Sigma}(\sigma) = \Pi_{\Sigma}(\sigma')$  (i.e., such that all words have the same untimed projection), we have  $min_{\preceq_{lex}, end}(E) = min_{\preceq_{lex}}(min_{\preceq_{end}}(E))$  where  $\sigma \preceq_{end} \sigma'$  if  $end(\sigma') \geq end(\sigma)$ , for  $\sigma, \sigma' \in tw(\Sigma)$ .

- **Maximal strict prefix**: The maximal strict prefix of  $\sigma \in tw(\Sigma)$  that belongs to  $\varphi \subseteq tw(\Sigma)$  is defined as  $max_{\preceq, \epsilon}^{\varphi}(\sigma) \stackrel{\text{def}}{=} max_{\preceq} \{ \sigma' \in (\mathbb{R}_{\geq 0} \times \Sigma)^* \mid \sigma' \prec \sigma \wedge \sigma' \in \varphi \} \cup \{ \epsilon \}$ .

The following notions of delayable (that checks whether a word is delayable w.r.t. the given property) are useful in defining constraints and respective enforcer.

- Function  $\text{delayable1}_\varphi(\sigma)$ : Given property  $\varphi \subseteq tw(\Sigma)$ , a timed word  $\sigma \in tw(\Sigma)$ , it returns the set of delayed words  $\sigma'$  of  $\sigma$ , s.t.  $\sigma'$  can be extended to satisfy  $\varphi$  in the future (i.e.,  $\sigma' \in \text{pref}(\varphi)$ ). Formally:  

$$\text{delayable1}_\varphi(\sigma) \stackrel{\text{def}}{=} \{\sigma' \in tw(\Sigma) : (\sigma' =_d \sigma) \wedge (\sigma' \in \text{pref}(\varphi))\}.$$
- Function  $\text{delayable2}_\varphi(\sigma_1, \sigma_2)$ : Given property  $\varphi \subseteq tw(\Sigma)$ , and two timed words  $\sigma_1, \sigma_2 \in tw(\Sigma)$ , it returns a set of delayed words  $\sigma'_2$  of  $\sigma_2$ , where each word in the set returned should start at or after the ending date of  $\sigma_2$ , which is the date  $t$  of the last event  $(t, a)$  of  $\sigma_2$ , s.t.  $\sigma_1 \cdot \sigma'_2$  can be extended to satisfy the property  $\varphi$  in the future. Formally,  

$$\text{delayable2}_\varphi(\sigma_1, \sigma_2) \stackrel{\text{def}}{=} \{\sigma'_2 \in tw(\Sigma) : (\sigma'_2 =_d \sigma_2) \wedge (\sigma_1 \cdot \sigma'_2 \in \text{pref}(\varphi)) \wedge (\text{start}(\sigma'_2) \geq \text{end}(\sigma_2))\}.$$
- Function  $k^\mathcal{L}(\sigma_1, \sigma_2)$ : Given  $\mathcal{L} \subseteq tw(\Sigma)$  and two timed words  $\sigma_1, \sigma_2 \in tw(\Sigma)$ , function  $k^\mathcal{L}(\sigma_1, \sigma_2)$  computes the set of timed words  $w$  that delay  $\sigma_2$ , start at or after the ending date of  $\sigma_2$ , s.t. when  $\sigma_1$  is extended with  $w$  (i.e.,  $\sigma_1 \cdot w$ ), the word should belong to  $\mathcal{L}$ . Formally,  

$$k^\mathcal{L}(\sigma_1, \sigma_2) \stackrel{\text{def}}{=} \{w \in \sigma_1^{-1} \cdot \mathcal{L} \mid (w =_d \sigma_2) \wedge (\text{start}(w) \geq \text{end}(\sigma_2))\}.$$

#### 4 Runtime enforcement in a timed context with unbounded memory

In this section, we define the enforcement monitoring framework with an unbounded buffer. We first define the expected constraints on the input/output behaviour of the enforcer in Sect. 4.1 and then define an enforcer dedicated to a desired timed property  $\varphi \in tw(\Sigma)$  in Sect. 4.2 ; this is a reformulation of paper [7]; the paper contains all the proofs.

##### 4.1 Constraints on an enforcer

We first define the constraints that should be satisfied by an enforcer before providing the actual definition of an enforcer in Sect. 4.2. The following constraints can serve as a specification of the expected behaviour of an enforcer for timed properties that has the ability to delay as well as suppress events.

► **Definition 4.** (*Constraints on an enforcer*). An enforcer for a timed property  $\varphi \subseteq tw(\Sigma)$  is a function  $E^\varphi : tw(\Sigma) \rightarrow tw(\Sigma)$ , satisfying the following constraints:

<b>Soundness</b>	<b>(Snd)</b>	$\forall \sigma \in tw(\Sigma) : E^\varphi(\sigma) \models \varphi \vee E^\varphi(\sigma) = \epsilon$
<b>Monotonicity</b>	<b>(Mo)</b>	$\forall \sigma, \sigma' \in tw(\Sigma) : \sigma \preceq \sigma' \implies E^\varphi(\sigma) \preceq E^\varphi(\sigma')$
<b>Transparency</b>	<b>(Tr1)</b>	$\forall \sigma \in tw(\Sigma), \text{delayable1}_\varphi(\sigma) = \emptyset \implies E^\varphi(\sigma) \triangleleft_d \sigma$
	<b>(Tr2)</b>	$\forall \sigma \in tw(\Sigma), \text{delayable1}_\varphi(\sigma) \neq \emptyset \implies E^\varphi(\sigma) \preceq_d \sigma$

*Soundness expresses that for any word  $\sigma \in tw(\Sigma)$ , the output produced by the enforcer (i.e.,  $E^\varphi(\sigma)$ ) should satisfy the property  $\varphi$ , as soon as it is non-empty. Monotonicity expresses that the output produced for the extension  $\sigma'$  of an input word  $\sigma$  (i.e.,  $E^\varphi(\sigma')$ ) extends the output produced for  $\sigma$  (i.e.,  $E^\varphi(\sigma)$ ), conveying that the output is a continuously growing timed word, i.e., for a given input timed word, what is output can only be changed by appending new events with greater dates. Transparency is defined using  $Tr_1$  and  $Tr_2$ . Here,  $Tr_1$  expresses that if no delayed word of  $\sigma \in tw(\Sigma)$  exists that can satisfy the property in the future, then*

the output of the enforcer for  $\sigma$  will be a delayed subword of  $\sigma$  (i.e., some events may be suppressed). The constraint  $Tr_2$  expresses that if any delayed word of  $\sigma \in tw(\Sigma)$  exists that can satisfy the property in the future, then the output of the enforcer for  $\sigma$  will be a delayed prefix of  $\sigma$  (i.e., none of the events are suppressed).

## 4.2 Definition of enforcement function

We now define an enforcement function / enforcer dedicated to a desired property  $\varphi$ . At an abstract level, it serves as a delayer with suppression, where suppression only occurs upon the reception of an event that inhibits any satisfaction of  $\varphi$  in the future.

► **Definition 5** (Enforcement function/ Enforcer). *The enforcer for a property  $\varphi \subseteq tw(\Sigma)$  is the function  $E^\varphi : tw(\Sigma) \rightarrow tw(\Sigma)$  defined as:*

$\forall \sigma \in tw(\Sigma), \forall t \in \mathbb{R}_{\geq 0}, \forall a \in \Sigma,$

$E^\varphi(\sigma) = \Pi_1(\text{store}^\varphi(\sigma)),$  where

$\text{store}^\varphi : tw(\Sigma) \rightarrow tw(\Sigma) \times tw(\Sigma)$  is defined as:

■  $\text{store}^\varphi(\epsilon) = (\epsilon, \epsilon)$

$$\text{store}^\varphi(\sigma \cdot (t, a)) = \begin{cases} (\sigma_s \cdot \min_{\leq_{lex}, end}(k^\varphi(\sigma_s, \sigma_{ca})), \epsilon) & \text{if } k^\varphi(\sigma_s, \sigma_{ca}) \neq \emptyset, \\ (\sigma_s, \sigma_c) & \text{if } k^{pref}(\varphi)(\sigma_s, \sigma_{ca}) = \emptyset, \\ (\sigma_s, \sigma_{ca}) & \text{otherwise.} \end{cases}$$

with:

- $(\sigma_s, \sigma_c) = \text{store}^\varphi(\sigma)$
- $\sigma_{ca} = \sigma_c \cdot (t, a)$

The unbounded enforcer  $E^\varphi$  takes a timed word over  $\Sigma$  as input, and produces a timed word over  $\Sigma$  as output.

For a given input  $\sigma$ , function  $\text{store}^\varphi$  computes a pair  $(\sigma_s, \sigma_c)$  of timed words: i)  $\sigma_s$  is the prefix of maximal length for which the absolute dates have been computed to satisfy property  $\varphi$ ; it is a delayed subsequence of the input  $\sigma$ ; and it is extracted by the projection function  $\Pi_1$  to produce output  $E^\varphi(\sigma)$ , ii)  $\sigma_c$  is a subsequence<sup>3</sup> of the remaining suffix of  $\sigma$  for which the releasing dates of events are still required to be computed. It is a temporary memory.

Enforcer  $E^\varphi$  incrementally computes a timed word according to the input timed word, and, inductively, is defined as follows: when the empty word  $\epsilon$  is input, it produces  $(\epsilon, \epsilon)$ . Otherwise, suppose that for the input  $\sigma$ , the result of  $\text{store}^\varphi(\sigma)$  is  $(\sigma_s, \sigma_c)$  and consider a newly received event  $(t, a)$ . Now, the new timed word to be corrected is  $\sigma_{ca} = \sigma_c \cdot (t, a)$ . There are three possible cases, according to the vacuity of the two sets  $k^\varphi(\sigma_s, \sigma_{ca})$  and  $k^{pref}(\varphi)(\sigma_s, \sigma_{ca})$ . For any  $\mathcal{L} \subseteq tw(\Sigma)$ , definition of  $k^\mathcal{L}(\sigma_s, \sigma_{ca})$  is given in Sect. 3.4.

$k^\varphi(\sigma_s, \sigma_{ca})$  is the set of timed words  $w$  that delay  $\sigma_{ca}$ , such that  $\sigma_s \cdot w$  satisfies  $\varphi$ ; and  $k^{pref}(\varphi)(\sigma_s, \sigma_{ca})$  is the set of timed words  $w$  that delay  $\sigma_{ca}$ , such that some additional continuation  $w'$  may satisfy  $\varphi$ , i.e.,  $\sigma_s \cdot w \cdot w' \models \varphi$ . Finally the three possible cases are:

- If  $k^\varphi(\sigma_s, \sigma_{ca}) \neq \emptyset$  (and thus  $k^{pref}(\varphi)(\sigma_s, \sigma_{ca}) \neq \emptyset$ ), it indicates that, for the timed word  $\sigma_{ca} = \sigma_c \cdot (t, a)$ , it is possible to choose its appropriate dates to satisfy  $\varphi$ . The minimal timed word in  $k^\varphi(\sigma_s, \sigma_{ca})$  w.r.t. the lexicographic order is selected among those with a minimal ending date and appended to  $\sigma_s$ . Since all events memorised in  $\sigma_c \cdot (t, a)$  are corrected and appended to  $\sigma_s$ ,  $\sigma_c$  is set to  $\epsilon$ .

<sup>3</sup> and not the complete suffix, since, some events may have been suppressed when no delaying allowed to satisfy  $\varphi$ , whatever is the continuation of  $\sigma$ , if any

## 6:10 Bounded-Memory Runtime Enforcement of Timed Properties

- If  $k^{pref(\varphi)}(\sigma_s, \sigma_{ca}) = \emptyset$  (and thus  $k^\varphi(\sigma_s, \sigma_{ca}) = \emptyset$ ), it means that, for the current input  $\sigma \cdot (t, a)$ , whatever is its continuation, there is no chance to find a suitable delaying for  $(t, a)$ . Thus, event  $(t, a)$  should be suppressed, leaving  $\sigma_c$  and  $\sigma_s$  unchanged.
- Otherwise, i.e., when  $k^\varphi(\sigma_s, \sigma_{ca}) = \emptyset$  but  $k^{pref(\varphi)}(\sigma_s, \sigma_{ca}) \neq \emptyset$ , it means that for  $\sigma_{ca} = \sigma_c \cdot (t, a)$ , it is not yet possible to select its appropriate dates to satisfy  $\varphi$ , however, if the input is continued, if at all, there is still a chance to do it in the future. Thus,  $\sigma_c$  is modified into  $\sigma_{ca} = \sigma_c \cdot (t, a)$  in memory, but  $\sigma_s$  is left unchanged.

► **Proposition 6 (Snd, Mo, Tr1, Tr2).** *Given some timed property  $\varphi \subseteq tw(\Sigma)$ , its enforcer  $E^\varphi$  as per Definition 5 satisfies Snd, Mo, Tr1, and Tr2 constraints as per Definition 4.*

► **Proposition 7 (Optimal Suppression).** *Given some timed property  $\varphi \subseteq tw(\Sigma)$ , its enforcer  $E^\varphi$  as per Definition 5 satisfies the following constraint:*

$$\begin{aligned} \forall \sigma \in tw(\Sigma), \exists \sigma_s, \sigma_c \in tw(\Sigma) : \text{store}^\varphi(\sigma) = (\sigma_s, \sigma_c) \wedge \forall (t, a) \in (\mathbb{R}_{\geq 0} \times \Sigma), t \geq \text{end}(\sigma_c) : \\ (\text{delayable}_{2\varphi}(\sigma_s, \sigma_c \cdot (t, a)) = \emptyset \implies \forall \sigma_{\text{con}} \in tw(\Sigma) : \text{start}(\sigma_{\text{con}}) \geq t, \\ E^\varphi(\sigma \cdot (t, a) \cdot \sigma_{\text{con}}) = E^\varphi(\sigma \cdot \sigma_{\text{con}})) \end{aligned}$$

(Opts)

For any input  $\sigma$ , for which the output of the  $\text{store}^\varphi(\sigma)$  is  $(\sigma_s, \sigma_c)$ , the optimal suppression constraint expresses that, when  $\sigma_c$  is extended with a timed event  $(t, a)$  and if no delayed word of  $\sigma_c \cdot (t, a)$  exists s.t.  $\sigma_s \cdot \sigma_c \cdot (t, a)$  can be extended to satisfy the property  $\varphi$  in future (i.e.,  $\text{delayable}_{2\varphi}(\sigma_s, \sigma_c \cdot (t, a)) = \emptyset$ ) then, the action  $a$  should be suppressed by the enforcer<sup>4</sup>.

► **Remark 8.** Snd, Mo, Tr1, Tr2 constraints outline an enforcer's expected input-output behaviour throughout the whole input sequence. However, it should be noted that it does not strongly constrain the output. These restrictions are specifically met by an enforcer that never produces any output. But, to be practical, a real enforcer should also offer some guarantees on the output sequence it produces in terms of delay, w.r.t. the input sequence. Such assurances are specified by the optimality property, which is provided in Appendix A.1.

► **Example 9 (Unbounded enforcer).** We illustrate how Definition 5 is applied to enforce a prototype property  $P$  of example 2, recognised by the automaton depicted in Fig. 1 with  $\Sigma = \{h, r\}$ , and the input timed word  $\sigma = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r) \cdot (8, r) \cdot (9, h)$ .

Table 1 shows evolution of the observed input timed word  $\text{obs}(\sigma, t)$  and the output of function  $\text{store}^P$  when the input timed word is  $\text{obs}(\sigma, t)$ . The first element of the output of function  $\text{store}^P$  can be extracted by projection function  $\Pi_1$  to produce output  $E^P(\sigma)$ . Variable  $t$  keeps track of physical time, i.e., it contains current date.

From the table, we can see that, at time  $t = 8$ , event  $(8, r)$  is suppressed, as it is not possible to correct the input sequences (since, reception of this event is leading to a violating location  $l_3$  in the timed automata 1). Till  $t < 9$ , the observed output is empty (since  $\Pi_1(\text{store}^P(\text{obs}(\sigma, t))) = \epsilon$ ). When  $t \geq 9$ , the observed output is  $(9, h) \cdot (10, h) \cdot (11, h) \cdot (12, h) \cdot (13, r) \cdot (17, h)$  (since  $\Pi_1(\text{store}^P(\text{obs}(\sigma, t))) = (9, h) \cdot (10, h) \cdot (11, h) \cdot (12, h) \cdot (13, r) \cdot (17, h)$ ).

## 5 Runtime enforcement in a timed context with bounded-memory

In this section, we introduce the enforcement monitoring framework with a bounded buffer. We describe how expected constraints must be adapted by the enforcer in Sect. 5.1 and then define an enforcer dedicated to a desired timed property  $\varphi \in tw(\Sigma)$  in Sect. 5.2.

<sup>4</sup> As stated in Tr1, suppression should only be done when necessary.

■ **Table 1** Evolution of the unbounded enforcer for property  $P$ .

$t \in [0, 1)$	$\text{obs}(\sigma, t) = \epsilon$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, \epsilon)$
$t \in [1, 2)$	$\text{obs}(\sigma, t) = (1, h)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h))$
$t \in [2, 3)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h))$
$t \in [3, 4)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h))$
$t \in [4, 5)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h))$
$t \in [5, 8)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r))$
$t \in [8, 9)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r) \cdot (8, r)$ $\text{store}^P(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r))$
$t \in [9, \infty)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r) \cdot (8, r) \cdot (9, h)$ $\text{store}^P(\text{obs}(\sigma, t)) = ((9, h) \cdot (10, h) \cdot (11, h) \cdot (12, h) \cdot (13, r) \cdot (17, h), \epsilon)$

A bounded-memory enforcer, denoted by  $E^{\varphi, k}$ , for a given timed property  $\varphi \in tw(\Sigma)$  is equipped with a buffer of size  $k$  and should be able to transform an input timed word  $\sigma$  which is possibly incorrect w.r.t.  $\varphi$  into an output timed word that is correct w.r.t.  $\varphi$ .

Enforcer  $E^{\varphi, k} : tw(\Sigma) \rightarrow tw(\Sigma) \times \{\perp, \top, \text{stop}\}$ , outputs a tuple consisting of an output word, referred by  $E_{\text{out}}^{\varphi, k}(\sigma)$ , (element of  $tw(\Sigma)$ ) and mode information, referred by  $E_{\text{mode}}^{\varphi, k}(\sigma)$ , (which is an element of  $\{\top, \perp, \text{stop}\}$ ) permitting to warn the user, where  $\top$ ,  $\perp$ , and  $\text{stop}$  respectively represent nominal mode indicating none of the events from the buffer were suppressed, degraded mode indicating some of the events from the buffer were suppressed, and  $\text{stop}$  mode indicating the enforcer cannot continue operating<sup>5</sup>.  $\text{buff}(E^{\varphi, k}(\sigma))$  refers to the buffer content of enforcer  $E^{\varphi, k}$ , after reading  $\sigma$ .

## 5.1 Constraints on an enforcer

We here define the constraints that should be satisfied by an enforcer.

► **Definition 10** (Constraints on an enforcer). *A bounded enforcer for a timed property  $\varphi \subseteq tw(\Sigma)$ , equipped with a buffer of size  $k$ , is a function  $E^{\varphi, k}$*

$$E^{\varphi, k} : tw(\Sigma) \rightarrow tw(\Sigma) \times \{\perp, \top, \text{stop}\}$$

satisfying the following constraints:

<b>Soundness</b>	<b>(SndB)</b>	$\forall \sigma \in tw(\Sigma) : E_{\text{out}}^{\varphi, k}(\sigma) \models \varphi \vee E_{\text{out}}^{\varphi, k}(\sigma) = \epsilon$
<b>Monotonicity</b>	<b>(Mo1B)</b>	$\forall \sigma, \sigma' \in tw(\Sigma) : \sigma \preceq \sigma' \implies E_{\text{out}}^{\varphi, k}(\sigma) \preceq E_{\text{out}}^{\varphi, k}(\sigma')$
	<b>(Mo2B)</b>	$\forall \sigma, \sigma' \in tw(\Sigma) : \sigma \preceq \sigma', (E_{\text{mode}}^{\varphi, k}(\sigma) = \perp \implies E_{\text{mode}}^{\varphi, k}(\sigma') = \perp)$
<b>Transparency</b>	<b>(Tr1B)</b>	$\forall \sigma \in tw(\Sigma), \text{delayable}1_{\varphi}(\sigma) = \emptyset \vee E_{\text{mode}}^{\varphi, k}(\sigma) = \perp \implies E_{\text{out}}^{\varphi, k}(\sigma) \triangleleft_d \sigma$
	<b>(Tr2B)</b>	$\forall \sigma \in tw(\Sigma), \text{delayable}1_{\varphi}(\sigma) \neq \emptyset \wedge E_{\text{mode}}^{\varphi, k}(\sigma) = \top \implies E_{\text{out}}^{\varphi, k}(\sigma) \preceq_d \sigma$

The notion of (SndB), (Mo1B), (Tr1B)<sup>6</sup> and (Tr2B)<sup>7</sup> are the same as in the unbounded case in Def. 4, with some notational variations and mode information being included as a consideration. (Mo2B) expresses that when the mode is degraded, it cannot return to nominal.

<sup>5</sup> Note that, intially the mode is considered to be nominal, i.e.,  $E_{\text{mode}}^{\varphi, k}(\epsilon) = \top$ .

<sup>6</sup> Tr1B can alternatively be expressed as:  $\forall \sigma \in tw(\Sigma), \forall (t, a) \in (\mathbb{R}_{\geq 0} \times \Sigma), E_{\text{mode}}^{\varphi, k}(\sigma) = \perp \vee \text{delayable}1_{\varphi}(\sigma \cdot a) = \emptyset \implies E_{\text{out}}^{\varphi, k}(\sigma \cdot a) \triangleleft_d \sigma \cdot a$

<sup>7</sup> Tr2B can alternatively be expressed as:  $\forall \sigma \in tw(\Sigma), \forall (t, a) \in (\mathbb{R}_{\geq 0} \times \Sigma), E_{\text{mode}}^{\varphi, k}(\sigma) = \top \wedge \text{delayable}1_{\varphi}(\sigma \cdot a) \neq \emptyset \implies E_{\text{out}}^{\varphi, k}(\sigma \cdot a) \preceq_d \sigma \cdot a$

## 5.2 Definition of enforcement functions

We now define a bounded-memory enforcer  $E^{\varphi,k}$  dedicated to a desired property  $\varphi$ . The enforcer works as a delayer with suppression, where suppression happens upon reception of an event that prevents any satisfaction of  $\varphi$  in the future or when the buffer is full.

**Preliminary to the enforcer.** During enforcement of an input timed word  $\sigma$ , when an incoming event needs to be buffered and the buffer is full, then the buffer is cleaned (i.e., some events stored in the buffer are suppressed to make room for the incoming events). The cleaning has to be done in such a way that, the treatment of future events by the enforcer is not affected (i.e., the enforcer handles the future events the same way with or without cleaning of the buffer). This is guaranteed if the state reached in the property automaton remains the same, even if some events are removed from the buffer. The notion of property equivalence can be understood as follows:

**Equivalence of two timed words.** Two timed words  $\sigma \in tw(\Sigma)$  and  $\sigma' \in tw(\Sigma)$  are  $\varphi$ -equivalent, noted  $\sigma \sim_{\varphi} \sigma'$  if runs  $\rho$  and  $\rho'$  from  $q_0$  of  $\mathcal{A}_{\varphi}$  (i.e.,  $(l_0, v_0)$ ) upon  $\sigma$  and  $\sigma'$  respectively i.e.,

$$\rho = q_0 \xrightarrow{(\delta_1, a_1)} q_1 \cdots q_{n-1} \xrightarrow{(\delta_n, a_n)} q_n \text{ and } \rho' = q_0 \xrightarrow{(\delta'_1, a'_1)} q'_1 \cdots q'_{m-1} \xrightarrow{(\delta'_m, a'_m)} q_m$$

(where, the trace of a run  $\rho$  and  $\rho'$  are timed words  $\sigma = (t_1, a_1) \cdot (t_2, a_2) \cdots (t_n, a_n)$  and  $\sigma' = (t'_1, a'_1) \cdot (t'_2, a'_2) \cdots (t'_m, a'_m)$  respectively (with  $t_n = \sum_{i=1}^n \delta_i$  and  $t'_m = \sum_{i=1}^m \delta'_i$ ), of different lengths) end on  $q_n$  and  $q_m$  respectively s.t.  $\mathcal{L}(\mathcal{A}_{\varphi}, q_n) = \mathcal{L}(\mathcal{A}_{\varphi}, q_m)$ .

It means that if the respective runs from the initial state of the property automaton upon two timed words end on two states, from where the language accepted are identical, then we say that the words are property equivalent.

► **Definition 11.** (*Bounded enforcement function / enforcer.*) A bounded enforcer for a property  $\varphi \subseteq tw(\Sigma)$  is the function  $E^{\varphi,k} : tw(\Sigma) \rightarrow tw(\Sigma) \times \{\top, \perp, stop\}$ , and is defined as:

$$\forall \sigma \in tw(\Sigma), \forall t \in \mathbb{R}_{\geq 0}, \forall a \in \Sigma,$$

$$E^{\varphi,k}(\sigma) = (\Pi_1(\text{store}^{\varphi,k}(\sigma)), \Pi_3(\text{store}^{\varphi,k}(\sigma))), \text{ where:}$$

$\text{store}^{\varphi,k} : tw(\Sigma) \rightarrow tw(\Sigma) \times tw(\Sigma) \times \{\top, \perp, stop\}$  is defined as:

- $\text{store}^{\varphi,k}(\epsilon) = (\epsilon, \epsilon, \top)$
- $\text{store}^{\varphi,k}(\sigma \cdot (t, a)) =$

$$\left\{ \begin{array}{ll} (\sigma_s \cdot \min_{\leq_{lex}} \text{end}(k^{\varphi}(\sigma_s, \sigma_{ca})), \epsilon, \{\top, \perp\}) & \text{if } k^{\varphi}(\sigma_s, \sigma_{ca}) \neq \emptyset, \\ (\sigma_s, \sigma_c, \perp) & \text{if } k^{\text{pref}(\varphi)}(\sigma_s, \sigma_{ca}) = \emptyset, \\ (\sigma_s, \sigma_{ca}, \{\top, \perp\}) & \text{if } k^{\text{pref}(\varphi)}(\sigma_s, \sigma_{ca}) \neq \emptyset \wedge |\sigma_{ca}| \leq k \\ (\sigma_s, \sigma_c, stop) & \text{if } k^{\text{pref}(\varphi)}(\sigma_s, \sigma_{ca}) \neq \emptyset \wedge |\sigma_{ca}| > k \\ & \wedge \text{Get\_SW}^{\varphi,k}(\sigma_s, \sigma_{ca}) = \emptyset \\ (\sigma_s, \text{Clean}^{\varphi,k}(\sigma_s, \sigma_{ca}), \perp) & \text{if } k^{\text{pref}(\varphi)}(\sigma_s, \sigma_{ca}) \neq \emptyset \wedge |\sigma_{ca}| > k \\ & \wedge \text{Get\_SW}^{\varphi,k}(\sigma_s, \sigma_{ca}) \neq \emptyset \end{array} \right.$$

with:

- $(\sigma_s, \sigma_c, \{\top, \perp\}) = \text{store}^{\varphi,k}(\sigma)$ ,
- $\sigma_{ca} = \sigma_c \cdot (t, a)$
- $E_{\text{out}}^{\varphi,k}(\sigma) = \Pi_1(E^{\varphi,k}(\sigma))$
- $E_{\text{mode}}^{\varphi,k}(\sigma) = \Pi_3(E^{\varphi,k}(\sigma))$
- $\text{buff}(E^{\varphi,k}(\sigma)) = \Pi_2(E^{\varphi,k}(\sigma))$

- $\text{Clean}^{\varphi,k} : tw(\Sigma) \times tw(\Sigma) \rightarrow tw(\Sigma)$   
 $\text{Clean}^{\varphi,k}(\sigma_s, \sigma_{ca}) = \sigma' \in \text{Get\_SW}^{\varphi,k}(\sigma_s, \sigma_{ca}) : \forall \sigma'' \in \text{Get\_SW}^{\varphi,k}(\sigma_s, \sigma_{ca}),$   
 $\sigma' \neq \sigma'' \wedge |\sigma'| > |\sigma''| \wedge (\text{index}(\sigma', \sigma_{ca}) \leq \text{index}(\sigma'', \sigma_{ca}))$
- $\text{index}(\sigma', \sigma_{ca}) = (i \in \mathbb{N} \mid i \in [1, |\sigma_{ca}|] : \sigma_{ca}[i] \neq \sigma'_{[i]})$
- $\text{Get\_SW}^{\varphi,k} : tw(\Sigma) \times tw(\Sigma) \rightarrow 2^{tw(\Sigma)}$   
 $\text{Get\_SW}^{\varphi,k}(\sigma_s, \sigma_{ca}) = \{\sigma'' \in tw(\Sigma) \mid \exists \sigma' \in \text{delayable}2^\varphi(\sigma_s, \sigma_{ca}) \wedge$   
 $\exists i, j, k \in \mathbb{N} \wedge 1 \leq i \leq j < k :$   
 $(\sigma'' = \sigma'_{[1\dots i-1]} \cdot \sigma'_{[j+1\dots k]}) \wedge (\sigma'_{[1\dots i-1]} \cdot \sigma'_{[i\dots j]} \cdot \sigma'_{[j+1\dots k]} \sim_\varphi \sigma'')\}$

The bounded enforcer  $E^{\varphi,k}$  takes a timed word over  $\Sigma$  as input, and produces a timed word over  $\Sigma$  and a sequence of modes (elements from the set  $\{\top, \perp, \text{stop}\}$ ) as output. For a given input  $\sigma$  over  $\Sigma$ , function  $\text{store}^{\varphi,k}$  computes a pair  $(\sigma_s, \sigma_c)$  of timed words over  $\Sigma$  (of which the first timed word is extracted by the projection function  $\Pi_1$  to produce the output  $E_{\text{out}}^{\varphi,k}(\sigma)$ ) and a sequence of mode as output (which is extracted by the projection function  $\Pi_3$  to be the mode  $E_{\text{mode}}^{\varphi,k}(\sigma)$ ). The pair  $(\sigma_s, \sigma_c)$  is same as in Def. 5.

Function  $E^{\varphi,k}$  incrementally computes a timed word according to the input timed word and is defined inductively as follows: When the empty word  $\epsilon$  is input, it produces  $(\epsilon, \epsilon, \top)$ . Otherwise, suppose that for input  $\sigma$ , the result of  $\text{store}^{\varphi,k}(\sigma)$  is  $(\sigma_s, \sigma_c, \{\top, \perp\})$  and consider a new received event  $(t, a)$ . Now, the new timed word to correct is  $\sigma_{ca} = \sigma_c \cdot (t, a)$ . There are five possible cases, according to the vacuity of the two sets  $k^\varphi(\sigma_s, \sigma_{ca})$  and  $k^{\text{pref}(\varphi)}(\sigma_s, \sigma_{ca})$ , whether the buffer is full, and if there exist any property equivalent word in the buffer:

- If  $k^\varphi(\sigma_s, \sigma_{ca}) \neq \emptyset$ . Since this case is similar to the first one in Def. 5, the same course of actions are followed. Mode remains unchanged.
- If  $k^{\text{pref}(\varphi)}(\sigma_s, \sigma_{ca}) = \emptyset$ . Since this case is similar to the second one in Def. 5, the same course of actions are followed. In addition, the mode changes to  $\perp$ .
- If  $k^{\text{pref}(\varphi)}(\sigma_s, \sigma_{ca}) \neq \emptyset$ , and  $|\sigma_{ca}| \leq k$ , i.e., it means that for  $\sigma_{ca} = \sigma_c \cdot (t, a)$ , it is not yet possible to select appropriate dates to satisfy  $\varphi$ , however, depending on the continuation of the input, if any, there is still a chance to do it in the future, and there is space in the buffer to accomodate this incoming event. Thus  $\sigma_c$  is modified into  $\sigma_{ca} = \sigma_c \cdot (t, a)$  in memory, but  $\sigma_s$  and mode are left unmodified.
- If  $k^{\text{pref}(\varphi)}(\sigma_s, \sigma_{ca}) \neq \emptyset$ ,  $|\sigma_{ca}| > k$ , and  $\text{Get\_SW}^{\varphi,k}(\sigma_s, \sigma_{ca}) = \emptyset$ . In this case, the buffer is full. Thus, cleaning of the buffer is required. However, there does not exist a property equivalent word (of length  $< k$ ) of the buffer contents to be able to replace the buffer contents. Thus, “safe” cleaning cannot be done and the enforcer stops operating (conveying this information by changing the mode to *stop*).
- If  $k^{\text{pref}(\varphi)}(\sigma_s, \sigma_{ca}) \neq \emptyset$ ,  $|\sigma_{ca}| > k$ , and  $\text{Get\_SW}^{\varphi,k}(\sigma_s, \sigma_{ca}) \neq \emptyset$ . In this case, there exists a property equivalent word (of length  $< k$ ) of the buffer contents to be able to replace the buffer contents. Thus, the function  $\text{Clean}^{\varphi,k}$  is called to clean the buffer (received event is also considered for cleaning) in order to accommodate the event.  $\sigma_s$  is left unmodified,  $\sigma_c$  is replaced by the output of function  $\text{Clean}^{\varphi,k}$ , and the mode changes to  $\perp$ .

Function  $\text{Clean}^{\varphi,k}$  takes two timed words over  $\Sigma$  as input. It produces a timed word as output, which should be a delayed subword of  $\sigma_{ca}$ . Also, the output word should be of maximal length, and the events discarded are the most *obsolete*<sup>8</sup> ones. For this purpose, function  $\text{Get\_SW}^{\varphi,k}$  provides all delayed subwords  $\sigma''$  of  $\sigma_{ca}$ . It does so by first finding all

<sup>8</sup> The earliest received events are considered for deletion in this approach; this is an implementation choice.



## 6:14 Bounded-Memory Runtime Enforcement of Timed Properties

delayed words  $\sigma'$  (of length  $k$ ) of  $\sigma_{ca}$  and then finding all the property equivalent words  $\sigma''$  (of length  $< k$ ) of every delayed word  $\sigma'$ , if any. Function  $\text{Clean}^{\varphi,k}$  selects the longest subwords among those and then chooses a unique subword, with the most *obsolete* action being discarded. This is done by comparing the indexes of  $\sigma_{ca}$  with the indexes of received subwords from function  $\text{Get\_SW}^{\varphi,k}$  using function  $\text{index}$ . The contents of the buffer are then substituted by the output of function  $\text{Clean}^{\varphi,k}$ .

► **Proposition 12 (SndB, Mo1B, Mo2B, Tr1B, Tr2B).** *Given some timed property  $\varphi \subseteq \text{tw}(\Sigma)$  and the maximum buffer size  $k$ , let  $n \in \mathbb{N}$  be the number of locations in  $\mathcal{A}_\varphi$ . If  $k \geq n$ , then the enforcer  $E^{\varphi,k}$  as per Definition 11 satisfies SndB, Mo1B, Mo2B, Tr1B, and Tr2B constraints as per Definition 10.*

The notion of optimal suppression in the case of a bounded enforcer is same as in case of unbounded enforcer (Opts), with some notational variations; we omit the definition due to space constraints. Also, similar to Sect. A.1, some additional constraints to provide some guarantees on the output sequence produced by the enforcer in terms of length and delay are provided in Appendix A.2.

► **Example 13 (Bounded enforcer).** We consider example 9, and see (in Table 2) how Def. 11 can be applied with  $k = 4$ . Other parameters are the same as in Table 1. From the table 2, we can see that, at time  $t = 5$ , the buffer is already full, thus function  $\text{Clean}^{P,4}$  invokes function  $\text{Get\_SW}^{P,4}$  to calculate the possible property equivalent subwords of the delayed word of  $(1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r)$ .

■ **Table 2** Evolution of the enforcer for property  $P$ .

$t \in [0, 1)$	$\text{obs}(\sigma, t) = \epsilon$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, \epsilon)$
$t \in [1, 2)$	$\text{obs}(\sigma, t) = (1, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h))$
$t \in [2, 3)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h))$
$t \in [3, 4)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h))$
$t \in [4, 5)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h))$
$t \in [5, 8)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (6, h) \cdot (7, h) \cdot (8, h) \cdot (9, r))$
$t \in [8, 9)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r) \cdot (8, r)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = (\epsilon, (6, h) \cdot (7, h) \cdot (8, h) \cdot (9, r))$
$t \in [9, \infty)$	$\text{obs}(\sigma, t) = (1, h) \cdot (2, h) \cdot (3, h) \cdot (4, h) \cdot (5, r) \cdot (8, r) \cdot (9, h)$ $\text{store}^{P,4}(\text{obs}(\sigma, t)) = ((9, h) \cdot (10, h) \cdot (11, h) \cdot (12, r) \cdot (16, h), \epsilon)$

For example, some of the words returned by function  $\text{Get\_SW}^{P,4}$  will be  $((6, h) \cdot (7, h) \cdot (8, h) \cdot (9, r))$ ,  $((5, h) \cdot (7, h) \cdot (8, h) \cdot (9, r))$ ,  $((7, h) \cdot (8, h) \cdot (9, r))$ , etc. Function  $\text{Clean}^{P,4}$  will first choose all the longest subwords among the set of delayed property equivalent subwords e.g.,  $((6, h) \cdot (7, h) \cdot (8, h) \cdot (9, r))$ ,  $((5, h) \cdot (7, h) \cdot (8, h) \cdot (9, r))$ , etc. Then it will choose a unique subword which is formed by deleting the most obsolete event, which is  $(6, h) \cdot (7, h) \cdot (8, h) \cdot (9, r)$  in this presented example with event  $(5, h)$  being suppressed while *cleaning*. All other cases are similar as in example 9. The final output is  $(9, h) \cdot (10, h) \cdot (11, h) \cdot (12, r) \cdot (16, h)$ .



► Remark 14 (Performance evaluation and analysis). We have provided an online algorithm and an experimentation framework in order to: i.) validate the viability of enforcement monitoring; and ii.) analyse the performance of the enforcer through experiments. Through performance analysis, we found that the enforcer had a reasonable execution time overhead (the average time taken in cleaning (per call) is found to be 0.019 s, which is low and reasonable<sup>9</sup>). The results are provided in Appendix B.

In the future, we plan to assess the framework within a more realistic scenario, taking into account various metrics. For instance, we aim to enhance the evaluation by increasing the complexity of the TAs, such as augmenting the number of locations within the TA. Additionally, we plan to explore the impact of varying buffer sizes, among other factors. This expanded evaluation will enable us to observe the growth patterns of time, memory, and other relevant aspects.

► Remark 15 (Stop mode of the enforcer and enforceable properties). Note that for any given property  $\varphi \subseteq tw(\Sigma)$ , for some input observation  $\sigma \in tw(\Sigma)$  if the mode of the enforcer changes to *stop* upon  $\sigma$ , then it is an indication that the enforcer should halt and cannot continue any further. We have,  $\forall \sigma \in tw(\Sigma), (E_{\text{mode}}^{\varphi,k}(\sigma) = \text{stop}) \implies \forall \sigma_{\text{con}}, E_{\text{out}}^{\varphi,k}(\sigma \cdot \sigma_{\text{con}}) = E_{\text{out}}^{\varphi,k}(\sigma)$ .

This happens when “safe” buffer cleaning is not attainable, (i.e., there does not exist a property equivalent word (of length  $< k$ ) of the buffer contents to be able to replace the buffer contents, when cleaning of the buffer is required), leading to halting of the enforcer and the mode getting changed to *stop*.

Ideally however, we expect that for a given property  $\varphi$ ,  $E^{\varphi,k}$  should continuously operate, and the mode of the enforcer should not change to *stop*. We call a given property  $\varphi$ , as continuously enforceable as per Def. 10 and  $E^{\varphi,k}$  an enforcer for  $\varphi$  as per Def. 11 if, for any input timed word  $\sigma \in tw(\Sigma)$ , mode is different from *stop*, i.e.,  $\forall \sigma \in tw(\Sigma), E_{\text{mode}}^{\varphi,k}(\sigma) \neq \text{stop}$ .

To achieve this, we introduce (in Appendix C) specific syntactic and semantic conditions on the TA, s.t. if a TA satisfies these conditions, it guarantees that the enforcer never halts.

## 6 Conclusion and future work

In conclusion, this paper presents a novel framework for enforcing timed properties with memory constraints on the enforcer. The proposed approach intervenes in executions by delaying or suppressing events to prevent property violations or buffer overflows. The paper defines the necessary constraints for an enforcer and proposes a dedicated function that transforms words to enforce a desired property. The presented algorithms provide implementation details for the proposed approach. Overall, this work contributes to the field of runtime enforcement of timed properties and provides a valuable framework for ensuring system correctness with real-time constraints. In the future, we also plan to develop other alternative implementations of the proposed enforcement framework using other TA frameworks such as TChecker<sup>10</sup> (to obtain the zone graphs and for reachability analysis).

---

### 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.

---

<sup>9</sup> Applications where this overhead is acceptable, the approach can be implemented.

<sup>10</sup> TChecker is a model-checking tool for real-timed systems. <https://www.labri.fr/perso/herbrete/tchecker/>

- 2 Roderick Bloem, Bettina Könighofer, Robert Könighofer, and Chao Wang. Shield synthesis: Runtime enforcement for reactive systems. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 533–548, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. doi:10.1007/978-3-662-46681-0\_51.
- 3 Christian Colombo, Gordon J. Pace, and Gerardo Schneider. Dynamic event-based runtime monitoring of real-time and contextual properties. In Darren Cofer and Alessandro Fantechi, editors, *Formal Methods for Industrial Critical Systems*, pages 135–149, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- 4 Christian Colombo, Gordon J. Pace, and Gerardo Schneider. Larva — safer monitoring of real-time java programs (tool paper). In *2009 Seventh IEEE International Conference on Software Engineering and Formal Methods*, pages 33–37, 2009. doi:10.1109/SEFM.2009.13.
- 5 Egor Dolzhenko, Jay Ligatti, and Srikar Reddy. Modeling runtime enforcement with mandatory results automata. *Int. J. Inf. Sec.*, 14(1):47–60, February 2015. doi:10.1007/s10207-014-0239-8.
- 6 Yliès Falcone, Jean-Claude Fernandez, and Laurent Mounier. What can you verify and enforce at runtime? *Int. J. Softw. Tools Technol. Transf.*, 14(3):349–382, 2012. doi:10.1007/s10009-011-0196-8.
- 7 Yliès Falcone, Thierry Jérón, Hervé Marchand, and Srinivas Pinisetty. Runtime enforcement of regular timed properties by suppressing and delaying events. *Systems & Control Letters*, 123:2–41, 2016. doi:10.1016/j.scl.2016.02.008.
- 8 Yliès Falcone, Laurent Mounier, Jean-Claude Fernandez, and Jean-Luc Richier. Runtime enforcement monitors: composition, synthesis, and enforcement abilities. *Formal Methods Syst. Des.*, 38(3):223–262, 2011. doi:10.1007/s10703-011-0114-4.
- 9 P. W. L. Fong. Access control by tracking shallow execution history. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 43–55, 2004. doi:10.1109/SECPRI.2004.1301314.
- 10 Jay Ligatti, Lujo Bauer, and David Walker. Edit automata: enforcement mechanisms for runtime security policies. *Int. J. Inf. Sec.*, 4(1-2):2–16, 2005. doi:10.1007/s10207-004-0046-8.
- 11 Jay Ligatti, Lujo Bauer, and David Walker. Run-time enforcement of nonsafety policies. *ACM Trans. Inf. Syst. Secur.*, 12(3), January 2009. doi:10.1145/1455526.1455532.
- 12 Oded Maler, Dejan Nickovic, and Amir Pnueli. From mitl to timed automata. In *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems, FORMATS'06*, pages 274–289, Berlin, Heidelberg, 2006. Springer-Verlag. doi:10.1007/11867340\_20.
- 13 Dejan Nickovic and Oded Maler. Amt: A property-based monitoring tool for analog systems. In Jean-François Raskin and P. S. Thiagarajan, editors, *Formal Modeling and Analysis of Timed Systems*, pages 304–319, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- 14 Srinivas Pinisetty, Yliès Falcone, Thierry Jérón, and Hervé Marchand. Tipex: A tool chain for timed property enforcement during execution. In Ezio Bartocci and Rupak Majumdar, editors, *Runtime Verification*, pages 306–320, Cham, 2015. Springer International Publishing.
- 15 Srinivas Pinisetty, Yliès Falcone, Thierry Jérón, Hervé Marchand, Antoine Rollet, and Omer Nguena-Timo. Runtime enforcement of timed properties revisited. *Formal Methods in System Design*, 45(3):381–422, 2014. doi:10.1007/s10703-014-0215-y.
- 16 Srinivas Pinisetty, Partha S. Roop, Steven Smyth, Nathan Allen, Stavros Tripakis, and Reinhard von Hanxleden. Runtime enforcement of cyber-physical systems. *ACM Trans. Embed. Comput. Syst.*, 16(5s):178:1–178:25, 2017. doi:10.1145/3126500.
- 17 Matthieu Renard, Yliès Falcone, Antoine Rollet, Srinivas Pinisetty, Thierry Jérón, and Hervé Marchand. Enforcement of (timed) properties with uncontrollable events. In *Theoretical Aspects of Computing - ICTAC 2015 - 12th International Colloquium Cali, Colombia, October 29-31, 2015, Proceedings*, pages 542–560, 2015. doi:10.1007/978-3-319-25150-9\_31.

- 18 Matthieu Renard, Antoine Rollet, and Yliès Falcone. Runtime enforcement of timed properties using games. *Formal Aspects of Computing*, 32(2):315–360, 2020. URL: <https://link.springer.com/article/10.1007/s00165-020-00515-2>.
- 19 G. Roc su. On safety properties and their monitoring. *Scientific Annals of Computer Science*, 22(2):327–365, 2012. doi:10.7561/SACS.2012.2.327.
- 20 U. Sammapun, Insup Lee, and O. Sokolsky. Rt-mac: runtime monitoring and checking of quantitative and probabilistic properties. In *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)*, pages 147–153, August 2005. doi:10.1109/RTCSA.2005.84.
- 21 Fred B. Schneider. Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50, February 2000. doi:10.1145/353323.353382.
- 22 Saumya Shankar and Srinivas Pinisetty. Serial compositional runtime enforcement of safety timed properties. In *Proceedings of the 16th Innovations in Software Engineering Conference, ISEC '23*, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3578527.3578529.
- 23 Saumya Shankar, Antoine Rollet, Srinivas Pinisetty, and Yliès Falcone. Bounded-memory runtime enforcement. In Owolabi Legunsen and Grigore Rosu, editors, *Model Checking Software*, pages 114–133, Cham, 2022. Springer International Publishing.
- 24 Chamseddine Talhi, Nadia Tawbi, and Mourad Debbabi. Execution monitoring enforcement under memory-limitation constraints. *Information and Computation*, 206(2):158–184, 2008. Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA '06). doi:10.1016/j.ic.2007.07.009.

## A Appendix: Additional constraints

### A.1 Additional constraints on an enforcer with unbounded-memory

We here provide some additional constraints to provide some guarantees on the output sequence produced by the enforcer in terms of delay.

#### ■ Optimality (minimum delay)

$$\begin{aligned}
 & \forall \sigma \in tw(\Sigma) : E^\varphi(\sigma) = \epsilon \vee \exists m, w \in tw(\Sigma) : E^\varphi(\sigma) = m \cdot w (\models \varphi) \text{ with} \\
 & m = \max_{\prec, \epsilon}^\varphi(E^\varphi(\sigma)) \text{ and} \\
 & w = \min_{\prec, lex, end} \{w' \in m^{-1} \cdot \varphi \mid \Pi_\Sigma(w') = \Pi_\Sigma(m^{-1} \cdot E^\varphi(\sigma)) \\
 & \quad \wedge m \cdot w' \triangleleft_d \sigma \wedge start(w') \geq end(\sigma)\} \tag{Opt}
 \end{aligned}$$

For any input  $\sigma$ , if the output  $E^\varphi(\sigma)$  is not empty, then it can be separated into: the maximal strict prefix  $m$  of  $E^\varphi(\sigma)$  satisfying property  $\varphi$ , and a suffix  $w$ . The optimality constraint expresses that, among those sequences  $w'$  that could have been chosen,  $w$  is the minimal one in terms of ending date, and lexical order. The “sequences that could have been chosen” are those such that  $m \cdot w'$  satisfies the property, have the same events, are delayed subsequences of the input  $\sigma$ , and have a starting date greater than or equal to  $end(\sigma)$ .

► **Proposition 16** (Optimality). *Given some property  $\varphi \subseteq tw(\Sigma)$ , its enforcer  $E^\varphi$  as per Definition 5 satisfies **Opt** property.*

### A.2 Additional constraints on an enforcer with bounded-memory

Similar to Sect. A.1, we here provide some additional constraints to provide some guarantees on the output sequence produced by the enforcer in terms of length and delay.

Let us first look at the below definition of  $\infty$ -compatibility used to define one of the optimality property: Consider the case when the buffer is full. Thus, as we have discussed, the enforcer would clean the buffer in such a way that, the treatment of the future events by the enforcer is not affected, implying that after cleaning, the behaviour of the bounded enforcer should be the same as that of the unbounded enforcer (bounded enforcer before/without cleaning). Thus, we define the notion of  $\infty$ -compatible as follows:

► **Definition 17** ( $\infty$ -compatible). *Enforcer  $E^{\varphi,k}$  is compatible with  $E^{\varphi,\infty}$  (enforcer with buffer size  $k = \infty$ , i.e., unbounded enforcer), noted  $\infty$ -compatible( $E^{\varphi,k}$ ), if  $\forall \sigma \in tw(\Sigma) : E^{\varphi,\infty}(\sigma) \cdot \text{buff}(E^{\varphi,\infty}(\sigma)) \sim_{\varphi} E_{\text{out}}^{\varphi,k}(\sigma) \cdot \text{buff}(E^{\varphi,k}(\sigma))$ .*

The above definition says that, for  $\varphi$ , a bounded enforcer  $E^{\varphi,k}$  is compatible with an unbounded enforcer  $E^{\varphi,\infty}$ , if for any input word  $\sigma$ , the concatenation of the output and the buffer content of  $E^{\varphi,\infty}$ , is  $\varphi$ -equivalent to the concatenation of the output and the buffer content of  $E^{\varphi,k}$ . Finally the optimality properties are:

■ **Optimality** (output is of maximum length):

Consider any bounded enforcer  $F^{\varphi,k}$  as per Def. 10. We have  
 $\exists \sigma \in tw(\Sigma), \forall (t, a) \in (\mathbb{R}_{\geq 0} \times \Sigma) :$

$$\begin{aligned} (E_{\text{out}}^{\varphi,k}(\sigma) \cdot \text{buff}(E^{\varphi,k}(\sigma)) = F_{\text{out}}^{\varphi,k}(\sigma) \cdot \text{buff}(F^{\varphi,k}(\sigma)) \quad \wedge \quad |E_{\text{out}}^{\varphi,k}(\sigma \cdot (t, a)) \cdot \text{buff}(E^{\varphi,k}(\sigma \cdot (t, a)))| \\ < |F_{\text{out}}^{\varphi,k}(\sigma \cdot (t, a)) \cdot \text{buff}(F^{\varphi,k}(\sigma \cdot (t, a)))|) \implies \neg(\infty\text{-compatible}(F^{\varphi,k})) \end{aligned} \quad (\text{Opt1B})$$

**Opt1B** expresses that an enforcer  $E^{\varphi,k}$  (as per Def. 11) is optimal; if, for some input, for any other enforcer  $F^{\varphi,k}$ , the length of the concatenation of its output and its buffer content is greater than the length of the concatenation of output and the buffer content of  $E^{\varphi,k}$ , then the output produced by  $F^{\varphi,k}$  is not  $\infty$ -compatible. Simply, it implies that, there does not exist an enforcer that can clean the buffer in a better way; by discarding less events and being  $\infty$ -compatible with the unbounded enforcer.

■ **Optimality** (minimum delay): The optimality property ensuring that each subsequence is output as soon as possible, with minimum delay for bounded-memory case is similar to **Opt** in Sect. A.1 for the unbounded-memory case, although with some notational variations.

$$\begin{aligned} \forall \sigma \in tw(\Sigma) : E_{\text{out}}^{\varphi,k}(\sigma) = \epsilon \vee \exists m, w \in tw(\Sigma) : E_{\text{out}}^{\varphi,k}(\sigma) = m \cdot w (\models \varphi) \text{ with} \\ m = \max_{\prec, \epsilon}^{\varphi}(E^{\varphi,k}(\sigma)) \text{ and} \\ w = \min_{\prec, \text{lex}, \text{end}} \{w' \in m^{-1} \cdot \varphi \mid \Pi_{\Sigma}(w') = \Pi_{\Sigma}(m^{-1} \cdot E^{\varphi,k}(\sigma)) \\ \wedge m \cdot w' \triangleleft_d \sigma \quad \wedge \quad \text{start}(w') \geq \text{end}(\sigma)\} \end{aligned} \quad (\text{Opt2B})$$

► **Proposition 18** (Optimality). *Given some property  $\varphi$  and the maximum buffer size  $k$ , its enforcer  $E^{\varphi,k}$  as per Definition 11 satisfies **Opt1B** and **Opt2B** properties.*

## B Appendix: Enforcement algorithm and performance evaluation

In Sect. 5.2, we provided an abstract view of our bounded-memory enforcer, defining it as a function that transforms words. In this section, we provide the overall enforcement algorithm. Also, we implemented the algorithm, to validate the feasibility of enforcement monitoring through experiments and analyse the performance of the enforcer.

## B.1 Enforcement algorithm

Let  $\mathcal{A}_\varphi = (L, l_0, X, \Sigma, \Delta, F)$  define  $\varphi$ . Consider that it has one clock<sup>11</sup>. The Algorithm 1, takes  $\mathcal{A}_\varphi$  and the buffer size  $k \in \mathbb{N}$  as input parameters.

In the algorithm,  $\sigma_s$  and  $\sigma_c$  refer to the output and the buffered events as in Def. 11. *currState* holds the current state. Function `await_event` is used to wait for a new input event. Function `check_reachability` computes all the reachable paths from the current state *currState* upon events in  $\sigma_c \cdot (\delta, a)$ . Function `get_acc_paths` takes as input all the paths returned by function `check_reachability` and returns only those that lead to a location in  $F$ . Function `get_od` computes the optimal delays for  $\sigma_c \cdot (\delta, a)$ . Also, it returns the state reached upon  $\sigma_c \cdot (\delta, a)$ . Function `append` is used to add the given event to the given word. Function `release` releases the given word as output. Function `check_reach_acc` takes all the paths returned by the function `check_reachability`, finds the last state of each paths and checks if an accepting location is reachable from that last state of the considered path. Function `len` is used to get the length of the word. Function `Get_SW` behaves as the same as function `Get_SW $\varphi, k$`  in Def. 11. It returns property equivalent subwords (of length  $< k$ ) of delayed  $\sigma_c \cdot (\delta, a)$ . Function `clean` optimally deletes events from the given word, similar<sup>12</sup> to the function `Clean $\varphi, k$`  in Def. 11. Function `sum_d` returns the sum of the delays of each of the events of the given word.

The algorithm proceeds as follows: Initially, buffers  $\sigma_c$  and  $\sigma_s$  are empty and *currState* is initialized with the initial state of  $\mathcal{A}_\varphi$  (i.e.,  $[l_0, 0]$ ). Variable  $c$  holds the sum of the delays of the events deleted by function `clean`<sup>13</sup>. It then enters into an infinite loop waiting for an input event. Upon receiving an event  $(\delta, a)$  (as in line no. 5 of Algorithm 1), where the delay  $\delta$  is relative to the delay of the previous received event, the enforcer adds extra delay  $c$  if returned by function `clean`, and progresses its clock (as shown in line 8). It then computes all the reachable paths from the current state *currState* upon events in  $\sigma_c \cdot (\delta, a)$  using function `check_reachability` (as shown in line 9). Function `get_acc_paths` then returns only those paths that lead to a state in  $F$  i.e., it returns all accepting paths, (as shown in line 10). If an accepting path exists (i.e.,  $accPaths \neq \emptyset$  as in line no. 11), then the enforcer computes optimal delays for  $\sigma_c \cdot (\delta, a)$ , (in  $\sigma_{ca}$ ), appends each event from  $\sigma_{ca}$  to  $\sigma_s$  to be released as output and sets the current state accordingly (as in line nos. 12-17). Otherwise, if an accepting path does not exist, it finds out if there is still a chance to reach an accepting location in the future using function `check_reach_acc`. If an accepting location is not reachable in the future, the enforcer drops/suppresses the received event and continues enforcement. Otherwise, if an accepting location is reachable, it appends the received event to the buffer, if the buffer is not full (as in line no. 24). Else, if the buffer is full, it optimally cleans the buffer to accommodate the received event, given that there exists any property equivalent subwords (of length  $< k$ ) delaying  $\sigma_c \cdot (\delta, a)$ ; otherwise the enforcer halts.

Function `clean` first computes the optimal delayed word  $\sigma_d$  of  $\sigma_{ca}$  using function `get_od`. It then enters a loop where in every iteration  $i$  ( $1 \leq i \leq k + 1$ ), it checks if a subword of length  $i$  from index  $j$  of  $\sigma_d$  can be read on a cycle s.t., the state remains same with or without

<sup>11</sup> This initial approach can be generalised and extended to any number of clocks.

<sup>12</sup> Function `clean` in this algorithm also returns the sum of the delays of the suppressed events in addition to the cleaned timed word; however Function `Clean $\varphi, k$`  of Def. 11 does not, since the enforcement mechanism considers the delays to be absolute.

<sup>13</sup> When function `clean` delete the very first or the intermediate events then  $c$  returned by it is 0 (because function `clean` internally has taken care of those delays by adding them to the delay of the next event in buffer, since the delays are relative; thus it need not be added to the incoming event  $(\delta, a)$ , thus  $c = 0$ ), otherwise  $c \neq 0$  (then,  $c$  is added to the delay of the incoming event).

■ **Algorithm 1** Algorithm Enforcer  $(\mathcal{A}_\varphi, k)$ .

---

```

1:  $\sigma_c, \sigma_s = []$ 
2:  $currState \leftarrow [l_0, 0]$ 
3:  $c = 0$ 
4: while true do
5:    $(\delta, a) \leftarrow \text{await\_event}()$ 
6:   if  $c \neq 0$  then
7:      $\delta = \delta + c$ 
8:    $currState[1] = currState[1] + \delta$ 
9:    $allPaths = \text{check\_reachability}(currState, \sigma_c \cdot (\delta, a))$ 
10:   $accPaths = \text{get\_acc\_paths}(allPaths)$ 
11:  if  $accPaths \neq \emptyset$  then
12:     $(\sigma_{ca}, state) = \text{get\_od}(currState, \sigma_c \cdot (\delta, a))$ 
13:    for  $event \in \sigma_{ca}$  do
14:       $\text{append}(\sigma_s, \sigma_{ca}[event])$ 
15:     $\sigma_c = []$ 
16:     $currState = state$ 
17:     $\text{release}(\sigma_s)$ 
18:  else
19:     $isReachable = \text{check\_reach\_acc}(allPaths)$ 
20:    if  $isReachable == \text{False}$  then
21:      continue
22:    else
23:      if  $\text{len}(\sigma_c) < k$  then
24:         $\text{append}(\sigma_c, (\delta, a))$ 
25:      else
26:        if  $\text{Get\_SW}(\sigma_s, \sigma_{ca}) \neq \emptyset$  then
27:           $(\sigma_c, c) = \text{clean}(currState, \sigma_c \cdot (\delta, a))$ 
28:        else
29:          exit


---


1: function  $\text{CLEAN}(curr\_state, \sigma_{ca})$ 
2:    $(\sigma_d, state_d) = \text{get\_od}(curr\_state, \sigma_{ca})$ 
3:   for  $i \in 1 \dots k + 1$  do
4:      $j = 1$ 
5:     while  $i + j \leq k + 2$  do
6:       if  $j == 1$  then
7:          $(DelayedW1, state1) = \text{get\_od}(curr\_state, \sigma_{d[j \dots j+i-1]})$ 
8:         if  $curr\_state == state1$  then
9:           return  $\sigma_{d[j+i \dots k+1]}, 0$ 
10:        else
11:           $(DelayedW1, state1) = \text{get\_od}(curr\_state, (\sigma_{d[1 \dots j-1]} \cdot \sigma_{d[j \dots j+i-1]})$ 
12:           $(DelayedW2, state2) = \text{get\_od}(curr\_state, \sigma_{d[1 \dots j-1]})$ 
13:          if  $state1 == state2$  then
14:            return  $\sigma_{d[1 \dots j-1]} \cdot \sigma_{d[j+i \dots k+1]}, \text{sum\_d}(\sigma_{d[j \dots j+i-1]})$ 
15:           $j++$ 


---



```

the subword. If yes, the subword is removed from  $\sigma_d$  and the resultant subword is returned by function `clean` along with the sum of delays of all the events of the subwords. The time complexity of function `clean` in Algorithm 1 is  $\mathcal{O}(k^3)$  with  $k$  as the buffer size.

## B.2 Implementation and performance evaluation

We implemented Algorithm 1 and developed an experimentation framework in order to i.) validate the feasibility of enforcement monitoring and ii.) analyse the performance of the enforcer through experiments.

**Our experimental framework.** There are tools for obtaining EMs, from the properties expressed using formalisms such as TA. We use the tool called TiPEX [14], as this is the RE monitor generation tool based on the theory of RE of timed properties proposed in [7], which we consider in this work.

We update the EMTA module to account for the bounded case. We add the definition of function `clean` as proposed in Algorithm 1 which will clean the buffer when required. In the implementation functions `checkReachability`, `getAccPaths`, `getOptimalDelays`, and `clean` straightaway maps to the functions `check_reachability`, `get_acc_paths`, `get_od`,

and `clean` in Algorithm 1. Function `check_reach_acc` in the algorithm is implemented to check if an action needs to be suppressed. It is implemented as follows: First it calls function `checkReachability` to give all the paths from the current state upon the given word; then, it finds the last state of each paths and checks if an accepting location is reachable from that last state of the considered path. It returns true in that case. Function `clean` is implemented in approx 50 LoC and Tipex is of approx 1200 LoC.

■ **Table 3** Effect on T(s) by bounded-memory enforcer by varying |Input|.

Input	T	clean	
		No.	$T_c$
100	1.513	0	0
200	5.843	2	0.0254
300	13.294	27	0.3564
400	23.306	52	0.8372
500	35.485	77	1.4245
600	51.615	102	2.1624
700	67.942	127	2.94386
800	89.137	152	3.88968
900	120.853	177	5.27106
1000	148.1558	202	6.62964

**Performance evaluation and analysis.** For measuring the performance, the timed property  $P$  in the prototype example 2 is used. The length of input sequence was varied from 100 to 1000 with an increment of 100 each time. To determine the worst case time taken, the input sequences were set so that function `clean` is invoked frequently by the bounded-memory enforcer. Considering buffer size<sup>14</sup>  $k = 50$ , we have the following observations from Tab. 3, where `Input` denotes the length of input sequences, `T` indicates the time taken by the bounded-memory enforcer to output the word, `No.` and  $T_c$  under `clean` indicates the number of times function `clean` is called and the total time taken by it respectively: i.) the time taken by the bounded-memory enforcer increases non-linearly<sup>15</sup> with the linear increase in trace length. ii.) the average time taken by the function `clean` (per call) is 0.019 s and is low/reasonable.

► **Remark 19.** Upon calculating the processing time for both the unbounded enforcer and the bounded-memory enforcer with the same set of input sequences of identical length, it is observed that the unbounded enforcer takes longer time. This is due to the extra workload of managing a buffer that can expand with incoming input sequences in the unbounded enforcer, as opposed to the bounded enforcer where maintaining a fixed-size buffer incurs less overhead.

## C Appendix: Enforceable properties

When we construct an enforcer for any given property  $\varphi \subseteq tw(\sigma)$  using the proposed approach, we ideally want the enforcer to never halt (i.e., the mode of the enforcer should never change to stop). While the fourth case ( $k^{pref(\varphi)}(\sigma_s, \sigma_{ca}) \neq \emptyset \wedge |\sigma_{ca}| > k \wedge Get\_SW^{\varphi, k}(\sigma_s, \sigma_{ca}) = \emptyset$ )

<sup>14</sup> Buffer allocation should be done carefully. If we use a small buffer, there will be a higher chance of buffer overflow, resulting in the deletion of more events. However, if we use a sufficiently sized buffer, there will be less overflow, which is crucial for retaining critical events and avoiding their loss.

<sup>15</sup> The behaviour is non-linear because of reasons such as, overhead in maintaining the list of uncorrected events (events in  $\sigma_c$ ), the more numbers of times function `clean` is called, etc.



## 6:22 Bounded-Memory Runtime Enforcement of Timed Properties

of Def. 11 in Sect. 5.2 may result in the enforcer halting, which is undesirable in practical scenarios, our objective is to ensure that the enforcer consistently operates. To achieve this, we propose (based on our preliminary observations) specific syntactic and semantic conditions on the TA, s.t. if the TA corresponding to any given property  $\varphi \subseteq tw(\sigma)$  satisfies these conditions, it guarantees that the enforcer never halts (i.e., the fourth case of Def. 11 never arise; or the function  $\text{Get\_SW}^{\varphi,k}$  always finds a word (of length  $< k$ ), which is property equivalent with the buffer contents).

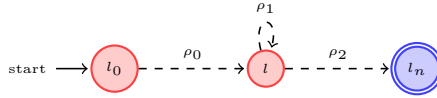
► **Definition 20** (Conditions on TA). We define the following conditions on TA  $\mathcal{A}_\varphi$ :

$\forall \rho \in \text{Run}_{FG}(\mathcal{A}_\varphi)$  and timed words  $\sigma_s, \sigma, \sigma_{con}$ ,  $\exists \rho_0, \rho_1, \rho_2 : \rho = \rho_0 \cdot \rho_1 \cdot \rho_2$ ,  
 $(\rho_0 = (l_0, v_0) \xrightarrow{\sigma_s} (l, v)) \wedge \rho_1 = (l, v) \xrightarrow{\sigma} (l, v') \wedge (\rho_2 = (l, v') \xrightarrow{\sigma_{con}} (l_n, v_n))$ ,  $\forall t_r \in \rho_1$ :

1.  $t_r = (l, g, a, \emptyset, l')$  where,  $g \in \mathcal{G}(X)$  with  $\bowtie \in \{<, \leq\}$ , or
2.  $t_r = (l, \epsilon, a, \emptyset, l')$

The above definition gives (not necessary but sufficient) syntactic and semantic conditions for a TA. The condition expresses that, for any run accepted by  $\mathcal{A}_\varphi$ , if it can be broken down (as shown in figure 2) into three consecutive runs  $\rho_0$ ,  $\rho_1$ , and  $\rho_2$  as follows:

- $\rho_0$ : from initial state  $(l_0, v_0)$  upon  $\sigma_s$ ,  $\mathcal{A}_\varphi$  makes the last transition to state  $(l, v)$ ,
- $\rho_1$ : from  $(l, v)$  upon  $\sigma$ ,  $\mathcal{A}_\varphi$  makes last transition to state  $(l, v')$  with no change in location,
- $\rho_2$ : from  $(l, v')$  upon  $\sigma_{con}$ ,  $\mathcal{A}_\varphi$  makes the last transition to state  $(l_n, v_n)$  (an accepting state),



■ **Figure 2** Runs of a TA, showing only the locations reached.



then all the transitions of the run  $\rho_1$  should be s.t., i) the guards are Boolean combinations of simple constraints where the operators are restricted to  $\{\leq, <\}$ , or, ii) the transitions should not have guards and resets acting on them.

► **Example 21.** (TA satisfying conditions of Def. 20). TA in Figure 1 of Example 2 satisfies the syntactic conditions of Def. 20, since the accepting run from initial state  $(l_0, v_0)$  can be broken down into runs  $\rho_0$  (from  $(l_0, v_0)$  to  $(l_1, v_1)$ ),  $\rho_1$  (from  $(l_1, v_1)$  to  $(l_1, v'_1)$ ), and  $\rho_2$  (from  $(l_1, v'_1)$  to  $(l_2, v_2)$ , where  $(l_2, v_2)$  is an accepting state) and the transition in  $\rho_1$  (e.g.,  $t_r = (l_1, \{x \leq 4\}, r, \epsilon, l_1)$ ) has guard with constraints where the operator is  $\leq$  (i.e.,  $x \leq 4$ )

► **Remark 22.** (Condition for enforceability.) If  $k \geq n$  (where  $n \in \mathbb{N}$  is the number of locations in  $\mathcal{A}_\varphi$ ), and the TA satisfies the conditions given in Definition 20, then  $E^{\varphi,k}$  defined in Def. 11 never halts.



# More Than 0s and 1s: Metric Quantifiers and Counting over Timed Words

Hsi-Ming Ho  

Department of Informatics, University of Sussex, UK

Khushraj Madnani  

Max Planck Institute for Software Systems, Kaiserslautern, Germany

---

## Abstract

We study the expressiveness of the *pointwise* interpretations (i.e. over timed words) of some predicate and temporal logics with metric and counting features. We show that counting in the unit interval  $(0, 1)$  is strictly weaker than counting in  $(0, b)$  with arbitrary  $b \geq 0$ ; moreover, allowing the latter indeed leads to expressive completeness for the metric predicate logic Q2MLO, recovering the corresponding result for the continuous interpretations (i.e. over signals). Exploiting this connection, we show that in contrast to the continuous case, adding “punctual” predicates into Q2MLO is still insufficient for the full expressive power of the Monadic First-Order Logic of Order and Metric (FO[<, +1]). Finally, we propose a generalisation of the recently proposed Pnueli automata modalities and show that the resulting metric temporal logic is expressively complete for FO[<, +1].

**2012 ACM Subject Classification** Theory of computation → Logic and verification

**Keywords and phrases** Temporal Logic, Expressiveness, Automata

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.7

## 1 Introduction

**Timed logics.** *Metric Temporal Logic* (MTL) [22] is a natural extension of *Linear Temporal Logic* (LTL) [31] with the capability of expressing real-time constraints by allowing intervals  $I$  to be specified with the “until” (U) and “since” (S) modalities of LTL. Intuitively,  $p \mathbf{U}_I q$  holds at a position  $i$  if there is a position  $j$  in the future where  $q$  holds, the time difference between  $i$  and  $j$  is within  $I$ , and  $p$  holds at all the points between  $i$  and  $j$ . While MTL provides a convenient and intuitive syntax for timing constraints, the problem of whether a given MTL formula has a model (behaviour) that satisfies it is undecidable [3, 29] – this makes MTL infeasible as a specification formalism for practical verification tasks. To remedy this issue, Alur, Feder, and Henzinger proposed in a seminal work [1] a syntactic fragment of MTL called *Metric Interval Temporal Logic* (MITL) where intervals associated with modalities are “*non-punctual*”, i.e. non-singular. They showed that the satisfiability and model-checking problems for MITL are decidable with EXPSPACE-complete complexity. In other words, by sacrificing perfect timing precision, we obtain a fully decidable timed specification formalism capable of expressing many practical properties of interest (see, e.g., [35]).

**Expressiveness.** Pnueli conjectured in the early 1990s that the trivial property “ $p$  and then  $q$  will happen in the next time unit” is not expressible in timed temporal logics like MTL and MITL. The conjecture (in different forms) is proved in [5, 12, 13, 30] and has led to several decidable extensions of MITL; one of the most notable extensions amongst them is Hirshfeld and Rabinovich’s Q2MLO [12]. It is straightforward to express the *counting modalities* and *Pnueli modalities* (a more general form of the aforementioned conjecture) in Q2MLO, and it admits a very simple and natural metric temporal logic characterisation: the extension of MITL with counting modalities is *expressively complete* for Q2MLO [16]. However, most of these results only hold for the *continuous* interpretations (i.e. over signals) of these logics



© Hsi-Ming Ho and Khushraj Madnani;

licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 7; pp. 7:1–7:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$$\begin{aligned} C_{(0,1)}\text{MTL} &\stackrel{\textcircled{1}}{=} C_0\text{MTL} \stackrel{\textcircled{2}}{=} \text{CMTL} \stackrel{\textcircled{3}}{=} \text{PQ2MLO} \stackrel{\textcircled{4}}{=} \text{FO}[\langle, +1]. \\ C_{(0,1)}\text{MITL} &\stackrel{\textcircled{5}}{=} C_0\text{MITL} \stackrel{\textcircled{6}}{=} \text{CMITL} \stackrel{\textcircled{7}}{=} \text{Q2MLO}. \end{aligned}$$

■ **Figure 1** The relevant expressiveness results in the continuous semantics.  $\textcircled{2}$  is trivial, e.g.,  $\mathbf{C}_{(1,2)}^2 p \iff \mathbf{F}_{=1} \mathbf{C}_{(0,1)}^2 p$ .  $\textcircled{1}$  can similarly be seen to hold by an easy case analysis, e.g.,  $\mathbf{C}_{(0,2)}^2 p \iff \mathbf{C}_{(0,1)}^2 p \vee \mathbf{F}_{=1}(\mathbf{C}_{(0,1)}^2 p) \vee (\mathbf{C}_{(0,1)}^1 p \wedge \mathbf{F}_{=1}(p \vee \mathbf{C}_{(0,1)}^1 p))$ .  $\textcircled{3}$  and  $\textcircled{4}$  are proved in [20].  $\textcircled{5}$ ,  $\textcircled{6}$ , and  $\textcircled{7}$  follow from [14, 16] and [19].

$$\begin{aligned} C_{(0,1)}\text{MTL}_{\text{fut}} &\stackrel{\textcircled{1}}{\subsetneq} C_0\text{MTL}_{\text{fut}} \stackrel{\textcircled{2}}{\subsetneq} \text{CMTL}_{\text{fut}} \stackrel{\textcircled{3}}{\subseteq} \text{PQ2MLO} \subseteq \text{PGQMLO} \subseteq \text{FO}[\langle, +1]. \\ C_{(0,1)}\text{MITL}_{\text{fut}} &\stackrel{\textcircled{4}}{\subsetneq} C_0\text{MITL}_{\text{fut}} \stackrel{\textcircled{5}}{\subsetneq} \text{CMITL}_{\text{fut}} \stackrel{\textcircled{6}}{\subseteq} \text{Q2MLO}. \end{aligned}$$

■ **Figure 2** The relevant known expressiveness results in the pointwise semantics (where the subscript “fut” stands for the future-only fragments).  $\textcircled{1}$ ,  $\textcircled{2}$ ,  $\textcircled{4}$ , and  $\textcircled{5}$  are proved in [24].  $\textcircled{3}$  and  $\textcircled{6}$  follow from [19]. The rest are syntactic inclusions.

and do not hold for the *pointwise* interpretations (i.e. over timed words). This is unfortunate from a practical point of view, as the latter is usually more amenable to automata-based implementations (e.g., UPPAAL [27]).

**Contributions.** The present work focusses on the expressiveness of these logics. We show that, as opposed to the situation in the continuous semantics, counting in  $(0, b)$  is strictly more expressive than counting in  $(0, 1)$ , and by allowing this modest generalisation we can actually recover the expressive completeness result for Q2MLO; this is also in stark contrast with the future-only fragments of these logics in the pointwise semantics, where counting in  $(0, b)$  is still insufficient for the expressiveness of (future) Q2MLO [24]. Similarly, we show that Q2MSO (the second-order version of Q2MLO) is characterised by MITL with counting modalities and *untimed* automata modalities. Finally, we show that Q2MLO with punctual predicates is still strictly less expressive than  $\text{FO}[\langle, +1]$  (once again in stark contrast with the continuous case), and we propose an extension to achieve the full expressiveness of  $\text{FO}[\langle, +1]$ .

**Related work.** Compared to the situation in the continuous semantics, there are very few expressive completeness results regarding timed temporal logics like MTL and MITL in the pointwise semantics in the literature. D’Souza and Tabareau [8] showed that “vanilla” MITL is expressively complete for a restricted fragment of the *Monadic First-Order Logic of Order and Metric* ( $\text{FO}[\langle, +1]$ ) in the pointwise semantics. It is shown in [17] that MTL with counting modalities is still strictly less expressive than  $\text{FO}[\langle, +1]$  in the pointwise semantics. On the practical side, counting modalities appear to be amenable to implementations, e.g., Bersani, Rossi, and San Pietro [4] proposed an SMT-based tool for deciding the satisfiability of MITL with counting modalities.

## 2 Preliminaries

We give a brief introduction to (linear-time) timed logics and some technical tools and notations used in the paper. For more detailed reviews and comparisons of relevant results, we refer the readers to [6, 15].

$$\begin{aligned}
C_{(0,1)}\text{MTL} &\stackrel{\textcircled{1}}{\subsetneq} C_0\text{MTL} \equiv \text{CMTL} \stackrel{\textcircled{2}}{\equiv} \text{PQ2MLO} \stackrel{\textcircled{3}}{\subsetneq} \text{PGQMLO} \subseteq \text{PGQMLO}^{\text{frac}} \stackrel{\textcircled{4}}{\equiv} \text{FO}[\langle, +1]. \\
C_{(0,1)}\text{MITL} &\stackrel{\textcircled{5}}{\subsetneq} C_0\text{MITL} \equiv \text{CMITL} \stackrel{\textcircled{6}}{\equiv} \text{Q2MLO}.
\end{aligned}$$

■ **Figure 3** The results of this paper (in the pointwise semantics). ① and ⑤ follow from Theorem 9. ② and ⑥ follow from Theorem 13. ③ is Corollary 16, and ④ is Theorem 17.

**Timed languages.** A *timed word* over a finite alphabet  $\Sigma$  is an  $\omega$ -sequence of *events*  $(\sigma_i, \tau_i)_{i \geq 1}$  over  $\Sigma \times \mathbb{R}_{\geq 0}$  with  $(\tau_i)_{i \geq 1}$  an increasing sequence of non-negative real numbers (“*timestamps*”) such that for any  $r \in \mathbb{R}_{\geq 0}$ , there is some *position*  $j \geq 1$  with  $\tau_j \geq r$  (i.e. we consider *strictly monotonic* timed words and require them to be “*non-Zeno*”).<sup>1</sup> We denote by  $\rho[i, j]$  the *finite* timed word formed by the sequence of events  $(\sigma_\ell, \tau_\ell)_{i \leq \ell \leq j}$ . We denote by  $T\Sigma^\omega$  the set of all timed words over  $\Sigma$ . A *timed language* is a subset of  $T\Sigma^\omega$ .

**Metric predicate logics.** We start by defining *Monadic Second-Order Logic of Order and Metric* ( $\text{MSO}[\langle, +1]$ ), which encompasses all the timed logics discussed in this paper.

► **Definition 1** ( $\text{MSO}[\langle, +1]$  [3, 33]). *Monadic Second-Order Logic of Order and Metric* ( $\text{MSO}[\langle, +1]$ ) *formulae are generated by*

$$\vartheta ::= \top \mid X(x) \mid x < x' \mid d(x, x') \in I \mid \vartheta_1 \wedge \vartheta_2 \mid \neg \vartheta \mid \exists x \vartheta \mid \exists X \vartheta$$

where  $X$  is an atomic proposition,  $x, x'$  are first-order variables,  $d$  is the distance predicate,  $I \subseteq \mathbb{R}_{\geq 0}$  is an interval with endpoints in  $\mathbb{N}_{\geq 0} \cup \{\infty\}$ , and  $\exists x, \exists X$  are first- and second-order quantifiers, respectively.<sup>2</sup>

As a convention we write, e.g.,  $(0, b)$ , to refer to  $(0, b)$  or  $(0, b]$ . The fragment of  $\text{MSO}[\langle, +1]$  without second-order quantifiers is the *Monadic First-Order Logic of Order and Metric* ( $\text{FO}[\langle, +1]$ ). The fragment of  $\text{MSO}[\langle, +1]$  without the distance predicate is the *Monadic Second Logic of Order* ( $\text{MSO}[\langle]$ ). The fragment of  $\text{FO}[\langle, +1]$  without the distance predicate is the *Monadic First-Order Logic of Order* ( $\text{FO}[\langle]$ ).

► **Definition 2** (Q2MLO [12]). *Q2MLO is the smallest fragment of  $\text{FO}[\langle, +1]$  obtained from  $\text{FO}[\langle]$  by the following rules:*

- All  $\text{FO}[\langle]$  formulae with a single free variable are Q2MLO formulae (note that they may use Q2MLO formulae as atomic propositions).
- If  $\vartheta(x_0, x)$  is an  $\text{FO}[\langle]$  formula where  $x_0$  and  $x$  are the only free first-order variables, then  $\exists x (x_0 < x \wedge d(x_0, x) \in I \wedge \vartheta(x_0, x))$  and  $\exists x (x < x_0 \wedge d(x_0, x) \in I \wedge \vartheta(x_0, x))$ , where  $I$  is non-singular, are also Q2MLO formulae (with free first-order variable  $x_0$ ).

We denote by  $\text{Q2MLO}_{0, \infty}$  the fragment of Q2MLO with only intervals of the forms  $(0, b)$  or  $\langle a, \infty$ , and  $\text{Q2MLO}_0$  is the even more restricted fragment where only intervals of the form  $(0, b)$  are allowed.<sup>3</sup> We also define Q2MSO [26], the smallest fragment of  $\text{MSO}[\langle, +1]$  obtained from  $\text{MSO}[\langle]$  by the rules in the previous definition (replacing  $\text{FO}[\langle]$  by  $\text{MSO}[\langle]$ ).

<sup>1</sup> We restrict ourselves to strictly monotonic timed words to simplify the definitions of metric predicate logics; all the results carry over to the case of non-strictly monotonic timed words as well.

<sup>2</sup> Following [33], we use  $d(x, x')$  in place of a “+1” function symbol.

<sup>3</sup> Note that non-metric  $\text{FO}[\langle]$  formulae are still allowed in these fragments.

## 7:4 More Than 0s and 1s: Metric Quantifiers and Counting over Timed Words

► **Definition 3** (PQ2MLO [20]). PQ2MLO (where “P” stands for “punctual”) is obtained from Q2MLO by adding the rule:

$$\blacksquare \exists x (x_0 < x \wedge d(x_0, x) \in I \wedge \vartheta(x))$$

where  $I$  is a singular interval and  $\vartheta(x)$  is a Q2MLO formula with a single free variable  $x$ .

**Metric temporal logics.** We start by defining *Extended Metric Temporal Logic* (EMTL) [33] where all operators are defined by *non-deterministic finite automata* (NFAs). An NFA over  $\Sigma$  is a tuple  $\mathcal{A} = \langle \Sigma, S, s_0, \Delta, F \rangle$  where  $S$  is a finite set of locations,  $s_0 \in S$  is the initial location,  $\Delta \subseteq S \times \Sigma \times S$  is the transition relation, and  $F$  is the set of final locations. We say that  $\mathcal{A}$  is *deterministic* (a DFA) iff for each  $s \in S$  and  $\sigma \in \Sigma$ ,  $|\{(s, \sigma, s') \mid (s, \sigma, s') \in \Delta\}| \leq 1$ . A *run* of  $\mathcal{A}$  on  $\sigma_1 \dots \sigma_n \in \Sigma^+$  is a sequence of locations  $s_0 s_1 \dots s_n$  where there is a transition  $(s_i, \sigma_{i+1}, s_{i+1}) \in \Delta$  for each  $i$ ,  $0 \leq i < n$ . A run of  $\mathcal{A}$  is *accepting* iff it ends in a final location. A finite word is *accepted* by  $\mathcal{A}$  iff  $\mathcal{A}$  has an accepting run on it.

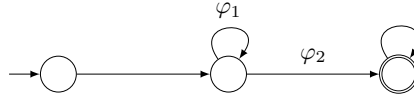
► **Definition 4** (EMTL [33]). Extended Metric Temporal Logic (EMTL) formulae over a finite set of atomic propositions AP are generated by

$$\varphi ::= \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg \varphi \mid \mathcal{A}_I(\varphi_1, \dots, \varphi_n) \mid \overleftarrow{\mathcal{A}}_I(\varphi_1, \dots, \varphi_n)$$

where  $p \in \text{AP}$ ,  $\mathcal{A}$  is an NFA over the  $n$ -ary alphabet  $\{1, \dots, n\}$ <sup>4</sup>, and  $I \subseteq \mathbb{R}_{\geq 0}$  is an interval with endpoints in  $\mathbb{N}_{\geq 0} \cup \{\infty\}$

As a convention, modalities with left arrows above them denote their “past” versions [2, 33]. We omit the subscript  $I$  when  $I = (0, \infty)$  and write pseudo-arithmetic expressions for lower or upper bounds, e.g., “ $< 3$ ” for  $(0, 3)$ . We also omit the arguments  $\varphi_1, \dots, \varphi_n$  and simply write  $\mathcal{A}_I$  or  $\overleftarrow{\mathcal{A}}_I$ , if clear from the context. EMITL [33] is the fragment of EMTL with only non-singular intervals. EMITL<sub>0,∞</sub> is the fragment of EMITL with only intervals of the forms  $(0, b)$  or  $\langle a, \infty \rangle$ .

► **Definition 5** (MTL [22]). Metric Temporal Logic (MTL) is the fragment of EMTL with only the “until” and “since” modalities defined by the NFA  $\mathcal{A}^U$  below:



MTL formulae are usually written in infix notation as  $\varphi_1 \mathbf{U}_I \varphi_2$  and  $\varphi_1 \mathbf{S}_I \varphi_2$ . We also use the usual shortcuts like  $\mathbf{F}_I \varphi \equiv \top \mathbf{U}_I \varphi$  and  $\mathbf{G}_I \varphi \equiv \neg \mathbf{F}_I \neg \varphi$ . *Metric Interval Temporal Logic* (MITL) [1] is the fragment of MTL with only non-singular intervals (or, equivalently, the fragment of EMITL with only the “until” and “since” modalities). MITL<sub>0,∞</sub> is the fragment of MITL with only intervals of the forms  $(0, b)$  or  $\langle a, \infty \rangle$  (or, equivalently, the fragment of EMITL<sub>0,∞</sub> with only the “until” and “since” modalities). *Linear Temporal Logic* (LTL) [31] is the fragment of MITL<sub>0,∞</sub> where all operators are labelled by  $(0, \infty)$ .<sup>5</sup>

► **Definition 6** (CMTL [14, 16]). CMTL is obtained from MTL by adding the counting modalities  $\mathbf{C}_I^k$  defined by the MSO[ $<, +1$ ] formula

$$\vartheta_I^{\mathbf{C},k}(x, X) = \exists x_1 \dots \exists x_k (x < x_1 < \dots < x_k \wedge d(x, x_1) \in I \wedge d(x, x_k) \in I \wedge \bigwedge_{1 \leq i \leq k} X(x_i))$$

<sup>4</sup> For clarity, we use  $\varphi_1, \dots, \varphi_n$  directly as transition labels (instead of  $1, \dots, n$ ) in the figures.

<sup>5</sup> We adopt the *strict* semantics for  $\mathbf{U}$  and  $\mathbf{S}$ , which subsumes the usual “next” and “previous” operators.

as well as  $\overleftarrow{\mathbf{C}}_I^k$  defined by the past counterpart of  $\vartheta_I^{\mathbf{C},k}(x, X)$ .<sup>6</sup>  $\mathbf{C}_0\text{MTL}$  is the fragment of CMTL where the counting modalities use only intervals of the form  $(0, b)$  where  $b \in \mathbb{N}_{>0} \cup \{\infty\}$ .

$\mathbf{C}_{(0,1)}\text{MTL}$  is the fragment of  $\mathbf{C}_0\text{MTL}$  where the counting modalities use only  $(0, 1)$ . We will freely combine notations to refer to various fragments of metric temporal logics, e.g.,  $\mathbf{C}_{(0,1)}\text{MITL}$  is obtained from MITL by adding  $\mathbf{C}_I^k$  and  $\overleftarrow{\mathbf{C}}_I^k$  with  $I = (0, 1)$ .

**Semantics of  $\text{MSO}[\langle, +1]$ .** With each timed word  $\rho = (\sigma_i, \tau_i)_{i \geq 1}$  over  $\Sigma_{\text{AP}} = 2^{\text{AP}}$  we associate a structure  $M_\rho$  whose universe  $U_\rho$  is  $\{i \mid i \geq 1\}$ . The order relation  $<$  and atomic propositions in AP are interpreted in the expected way, e.g.,  $P(i)$  holds in  $M_\rho$  iff  $P \in \sigma_i$ . The distance predicate  $d(x, x') \in I$  holds iff  $|\tau_x - \tau_{x'}| \in I$ . The satisfaction relation for  $\text{MSO}[\langle, +1]$  is defined inductively in the usual way. We write  $\rho, j_1, \dots, j_m, J_1, \dots, J_n \models \vartheta(x_1, \dots, x_m, X_1, \dots, X_n)$  if  $j_1, \dots, j_m \in U_\rho$ ,  $J_1, \dots, J_n \subseteq U_\rho$ , and  $\vartheta(j_1, \dots, j_m, J_1, \dots, J_n)$  holds in  $M_\rho$ . We say that two  $\text{MSO}[\langle, +1]$  formulae  $\vartheta_1(x)$  and  $\vartheta_2(x)$  are *equivalent* if for all timed words  $\rho = (\sigma_i, \tau_i)_{i \geq 1}$  and  $j \in U_\rho$ ,

$$\rho, j \models \vartheta_1(x) \iff \rho, j \models \vartheta_2(x).$$

**Semantics of EMTL.** EMTL can be embedded into  $\text{MSO}[\langle, +1]$  through Büchi-Elgot-Trakhtenbrot theorem [25], but we can also define the satisfaction relation directly. Given an EMTL formula  $\varphi$  over AP, a timed word  $\rho = (\sigma_i, \tau_i)_{i \geq 1}$  over  $\Sigma_{\text{AP}}$  and  $i \geq 1$ , define  $\rho, i \models \varphi$  as follows:

- $\rho, i \models \top$ ;
- $\rho, i \models p$  iff  $p \in \sigma_i$ ;
- $\rho, i \models \varphi_1 \wedge \varphi_2$  iff  $\rho, i \models \varphi_1$  and  $\rho, i \models \varphi_2$ ;
- $\rho, i \models \neg\varphi$  iff  $\rho, i \not\models \varphi$ ;
- $\rho, i \models \mathcal{A}_I(\varphi_1, \dots, \varphi_n)$  iff there exists  $j \geq i$  such that (i)  $\tau_j - \tau_i \in I$  and (ii) there is an accepting run of  $\mathcal{A}$  on  $a_i \dots a_j$  where  $\rho, \ell \models \varphi_{a_\ell}$  ( $a_\ell \in \{1, \dots, n\}$ ) for each  $\ell$ ,  $i \leq \ell \leq j$ .
- $\rho, i \models \overleftarrow{\mathcal{A}}_I(\varphi_1, \dots, \varphi_n)$  is defined symmetrically.

We say that  $\rho$  *satisfies*  $\varphi$  (written  $\rho \models \varphi$ ) iff  $\rho, 1 \models \varphi$ .

**Ehrenfeucht-Fraïssé games for CMTL.** An  $m$ -round CMTL Ehrenfeucht-Fraïssé (EF) game starts with round 0 and ends with round  $m$ . The game is played by two players (*Spoiler* and *Duplicator*) on a pair of timed words  $\rho = (\sigma_i, \tau_i)_{i \geq 1}$  and  $\rho' = (\sigma'_i, \tau'_i)_{i \geq 1}$ . A *configuration* is a pair of positions  $(i, j)$ , respectively in  $\rho$  and  $\rho'$ . In each round  $r$  ( $0 \leq r \leq m$ ), the game proceeds as follows. *Spoiler* first checks whether the two events that correspond to the current configuration  $(i_r, j_r)$  in  $\rho$  and  $\rho'$  satisfy the same atomic propositions. If this is not the case then *Spoiler* wins the game. Otherwise if  $r < m$ , *Spoiler* chooses  $I \subseteq \mathbb{R}_{\geq 0}$  with endpoints in  $\mathbb{N}_{\geq 0} \cup \{\infty\}$  and plays either of the following moves:

- **U<sub>I</sub>-move:** *Spoiler* chooses one of the two timed words (say  $\rho$ ) and picks  $i'_r$  such that  $i_r < i'_r$  and  $\tau_{i'_r} - \tau_{i_r} \in I$  (if there is no such  $i'_r$ , then *Duplicator* wins the game). *Duplicator* must choose  $j'_r$  such that  $\tau'_{j'_r} - \tau'_{j_r} \in I$  – if this is not possible then *Spoiler* wins the game. Otherwise, *Spoiler* plays either of the following “parts”:
- **F-part:** The game proceeds to the next round with  $(i_{r+1}, j_{r+1}) = (i'_r, j'_r)$ .

<sup>6</sup> Note that  $\mathbf{C}_I^k$  and  $\overleftarrow{\mathbf{C}}_I^k$  are subsumed by EMTL even when  $\inf I \neq 0$  [19].

- **U-part:** If  $j'_r = j_r + 1$  the game proceeds to the next round with  $(i_{r+1}, j_{r+1}) = (i'_r, j'_r)$ . If  $i'_r = i_r + 1$  but  $j'_r \neq j_r + 1$  then *Spoiler* wins the game. Otherwise, *Spoiler* picks  $j''_r$  such that  $j_r < j''_r < j'_r$ ; *Duplicator* has to choose  $i''_r$  such that  $i_r < i''_r < i'_r$  in response – if this is not possible then *Spoiler* wins the game. Otherwise, the game proceeds to the next round with  $(i_{r+1}, j_{r+1}) = (i''_r, j''_r)$ .
- **S<sub>I</sub>-move:** Defined symmetrically.
- **C<sub>I</sub><sup>k</sup>-move:** *Spoiler* chooses one of the two timed words (say  $\rho$ ) and picks  $i_r^1, \dots, i_r^k$  such that  $i_r < i_r^1 < \dots < i_r^k$  and  $\tau_{i_r^\ell} - \tau_{i_r} \in I$  for all  $\ell, 1 \leq \ell \leq k$  (if there are no such  $i_r^1, \dots, i_r^k$  then *Duplicator* wins the game); *Duplicator* must choose  $j_r^1, \dots, j_r^k$  such that  $\tau_{j_r^\ell} - \tau_{j_r} \in I$  for all  $\ell, 1 \leq \ell \leq k$  – if this is not possible then *Spoiler* wins the game. *Spoiler* then picks  $j''_r = j_r^\ell$  for some  $\ell, 1 \leq \ell \leq k$ , *Duplicator* chooses  $i''_r = i_r^\ell$  for some  $\ell, 1 \leq \ell \leq k$ , and the game proceeds to the next round with  $(i_{r+1}, j_{r+1}) = (i''_r, j''_r)$ .
- **C<sub>I</sub>-move:** Defined symmetrically.

We say that *Duplicator* has a *winning strategy* for the  $m$ -round CMTL EF game on  $\rho$  and  $\rho'$  that starts from configuration  $(i, j)$  if and only if, no matter how *Spoiler* plays, *Duplicator* can always win the  $m$ -round CMTL EF game on  $\rho$  and  $\rho'$  with  $(i_0, j_0) = (i, j)$ . If this is not the case then we say that *Spoiler* has a winning strategy. The following theorem relates the number of rounds of CMTL EF games to the modal depth (i.e., the maximal depth of nesting of modalities) of CMTL formulae.

► **Theorem 7** ([24, 30]). *For timed words  $\rho, \rho'$  and a CMTL formula  $\varphi$  of modal depth  $\leq m$ , if *Duplicator* has a winning strategy for the  $m$ -round CMTL EF game on  $\rho, \rho'$  with  $(i_0, j_0) = (1, 1)$ , then*

$$\rho \models \varphi \iff \rho' \models \varphi.$$

Note that the theorem above can also be specialised to sublogics of CMTL; for example, the corresponding theorem for  $C_{(0,1)}$ MITL is obtained by forcing  $I = (0, 1)$  in  $C_I^k$ -moves.

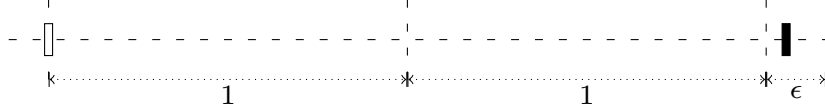
**Expressiveness.** We say that a metric logic  $L'$  is *expressively complete* for a metric logic  $L$  iff for any formula  $\vartheta(x) \in L$ , there is an equivalent formula  $\varphi(x) \in L'$ .<sup>7</sup> We say that  $L'$  is *at least as expressive as* (or *more expressive than*)  $L$  (written  $L \subseteq L'$ ) iff for any formula  $\vartheta(x) \in L$ , there is an *initially equivalent* formula  $\varphi(x) \in L'$  (i.e.,  $\vartheta(1)$  and  $\varphi(1)$  evaluate to the same truth value for any timed word). We say that  $L'$  and  $L$  are *equally expressive* (written  $L' \equiv L$ ) iff  $L \subseteq L'$  and  $L' \subseteq L$ . If  $L \subseteq L'$  but  $L' \not\subseteq L$  then we say that  $L'$  is *strictly more expressive than*  $L$  (or  $L$  is *strictly less expressive than*  $L'$ ).

### 3 Expressive completeness for Q2MLO

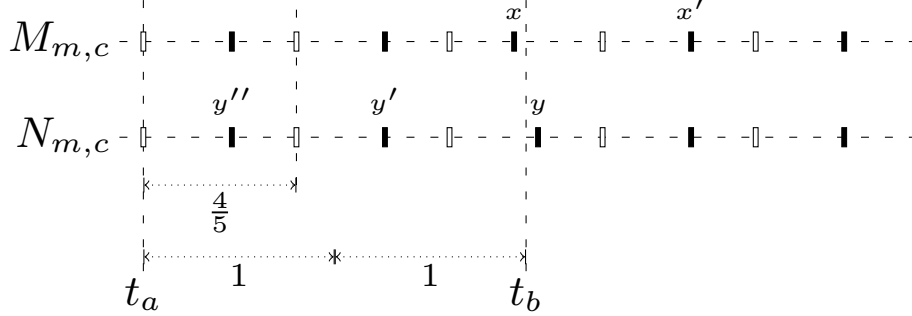
**Counting in (0, 1).** We argue that counting in  $(0, 1)$  is not sufficiently expressive in the pointwise semantics; in particular, counting in  $(0, b)$  cannot be expressed in MTL extended with  $C_{(0,1)}^k$  and  $\overleftarrow{C}_{(0,1)}^k$ , and it turns out to be essential for achieving the full expressiveness of Q2MLO. This is in stark contrast with the situation in the continuous semantics, where LTL extended with  $C_{(0,1)}^k$  and  $\overleftarrow{C}_{(0,1)}^k$  is expressively complete for Q2MLO [14, 16]. We show this by constructing two families of timed words  $(M_{m,c})$  and  $(N_{m,c})$  over  $\Sigma_{\{p,q\}}$  (inspired by [30]) that can be told apart easily by a  $C_0$ MTL formula using  $C_{(0,b)}^k$ , yet they are indistinguishable by all  $C_{(0,1)}$ MTL formulae of modal depth  $\leq m$ , all constants  $\leq c$ , and where all occurrences of counting modalities  $C_I^{k'}$  and  $\overleftarrow{C}_I^{k'}$  have  $k' \leq k$ .

<sup>7</sup> Formulae of metric temporal logics are  $\text{MSO}[<, +1]$  formulae with a single free first-order variable.

We start by describing  $N_{m,c}$  for some fixed  $m, c \in \mathbb{N}_{\geq 0}$ . Let  $c'$  be the least integer greater than  $\frac{5}{4} \cdot (c+3) + 1$  and  $\epsilon = \frac{1}{6}$ . We put an  $\emptyset$ -event at time 0, and then a number of overlapping segments start at time  $(c+1)$  where each segment consists of a  $\{p\}$ -event and a  $\{q\}$ -event (note that each  $\{p\}$ -event or  $\{q\}$ -event uniquely identifies a segment). If the  $\{p\}$ -event in the  $i^{\text{th}}$  segment is at, say,  $t$ , then its  $\{q\}$ -event is at  $t + 2 + \frac{i}{3 \cdot m \cdot c' + 3} \cdot \epsilon$  (see Figure 4). We put a total of  $2 \cdot m \cdot c' + 1$  segments where  $\{p\}$ -events in neighbouring segments are separated by  $\frac{4}{5}$ . Finally, we put an infinite sequence of  $\emptyset$ -events, equally separated by  $(c+1)$  and starting at  $(c+1)$  after the  $\{q\}$ -event in the last segment.  $M_{m,c}$  is almost identical to  $N_{m,c}$ , except for the middle (i.e.,  $(m \cdot c' + 1)^{\text{th}}$  segment – say this segment starts at  $t$ , then in  $M_{m,c}$  we shift the corresponding  $\{q\}$ -event to  $t + 2 - \frac{m \cdot c' + 1}{3 \cdot m \cdot c' + 3} \cdot \epsilon$  instead. For convenience, we write  $t_a$  for the timestamp of the  $\{p\}$ -event in the middle segment (i.e.  $t_a = (c+1) + \frac{4}{5} \cdot m \cdot c'$ ),  $t_b = t_a + 2$ , and denote the corresponding  $\{q\}$ -events in  $M_{m,c}$  and  $N_{m,c}$  by  $x$  and  $y$  respectively with timestamps  $t_x$  and  $t_y$  (see Figure 5). It is easy to see that no  $\{q\}$ -event is at an integer distance to some other  $\{p\}$ -event or  $\{q\}$ -event. This completes the description of  $M_{m,c}$  and  $N_{m,c}$ . We say a configuration  $(i, j)$  is *identical* if  $i = j$ . For a position  $i \geq 1$  in  $M_{m,c}$  or  $N_{m,c}$ , we write  $\text{seg}(i)$  for the segment to which the  $i^{\text{th}}$  event belongs. For convenience we define  $\text{seg}(i) = 0$  if the  $i^{\text{th}}$  event is an  $\emptyset$ -event.



■ **Figure 4** A segment in  $N_{m,c}$ . The white box is the  $\{p\}$ -event and the black box is the  $\{q\}$ -event.



■ **Figure 5** The events near the middle segments of  $M_{m,c}$  and  $N_{m,c}$ . White boxes are  $\{p\}$ -events and black boxes are  $\{q\}$ -events.

We are now ready to state the main technical lemma, which intuitively says that *Duplicator* can either keep the configuration identical or far enough from the beginnings and the ends of both  $M_{m,c}$  and  $N_{m,c}$  (where *Spoiler* can easily win the EF game).

- **Lemma 8.** *In the  $m$ -round  $C_{(0,1)}$ MTL EF game on  $M_{m,c}$ ,  $N_{m,c}$  starting from  $(1, 1)$ , Duplicator has a winning strategy such that for each round  $0 \leq r \leq n$ , the  $i_r^{\text{th}}$ -event in  $M_{m,c}$  and the  $j_r^{\text{th}}$ -event in  $N_{m,c}$  satisfy the same atomic propositions and*
- *if  $\text{seg}(i_r) \neq \text{seg}(j_r)$ , then  $r \geq 1$  and  $\text{seg}(i_r), \text{seg}(j_r) \in [(m-r+1) \cdot c' - 1, (m+r-1) \cdot c' + 3]$ .*

**Proof.** We describe a winning strategy for *Duplicator* by induction on  $r$ . The basic idea is to make the resulting configuration identical whenever possible (and thus the induction hypothesis trivially holds); otherwise we use a copy-cat strategy (i.e. try to make  $\text{seg}(i_{r+1}) -$



$\text{seg}(i_r) = \text{seg}(j_{r+1}) - \text{seg}(j_r)$ ). If that is also not possible, we must choose another event that satisfies the same atomic propositions. In the following, we refer to the timed word that *Spoiler* first chooses as  $\rho^s = (\sigma_i^s, \tau_i^s)_{i \geq 0}$  ( $\rho^d = (\sigma_i^d, \tau_i^d)_{i \geq 0}$  for that of *Duplicator*).

- *Base step.* The induction hypothesis holds trivially for  $(i_0, j_0) = (1, 1)$ .
- *Induction step.* Suppose the claim holds for  $r < m$ . We prove it also holds for  $r + 1$ .
  - $(i_r, j_r) = (1, 1)$ :  
Since all segments happen at time  $> c$ , *Duplicator* can always make  $(i_{r+1}, j_{r+1})$  an identical configuration, if necessary.
  - $(i_r, j_r) \neq (1, 1)$  is identical:  
We may assume  $r > 0$ . Observe from Figure 5 that any two  $\{p\}$ -events that are  $5n$  segments away are separated by  $4n$ . More specifically, since  $t_b - t_a = 2$ ,  $\{p\}$ -events whose distances to  $t_a$  are integers will also have integer distances to  $t_b$ . We consider the following cases:
    - \*  $(i_r, j_r)$  both correspond to  $\emptyset$ -events: since they are separated from any other events by  $> c$ , *Duplicator* can always make  $(i_{r+1}, j_{r+1})$  identical if necessary.
    - \*  $(i_r, j_r)$  both correspond to  $\{p\}$ -events and *Spoiler* plays an  $\mathbf{U}_I$ -move or  $\mathbf{S}_I$ -move and picks (say)  $i'_r = x$ . *Duplicator* may either choose  $j'_r = y$  (then *Duplicator* can surely make  $(i_{r+1}, j_{r+1})$  identical later) or if that is not possible, choose event  $j'_r = y'$ . In the latter case, if *Spoiler* plays the  $\mathbf{F}$ -part, it is obvious that the resulting configuration  $(i_{r+1}, j_{r+1})$  would satisfy the claim. If *Spoiler* plays  $\mathbf{U}$ -part, *Duplicator* may either make  $(i_{r+1}, j_{r+1})$  identical or  $\text{seg}(j_{r+1}) - \text{seg}(i_{r+1}) = -1$ . In this latter case it is clear that the claim still holds ( $\text{seg}(i_{r+1}) = m \cdot c' + 2$  or  $\text{seg}(i_{r+1}) = m \cdot c' + 4$ ). If *Spoiler* plays a  $\mathbf{C}_I^k$ -move or  $\overleftarrow{\mathbf{C}}_I^k$ -move, as  $I = (0, 1)$ , *Duplicator* can always make  $(i_{r+1}, j_{r+1})$  identical if necessary.
    - \*  $(i_r, j_r)$  corresponds to  $\{q\}$ -events except  $x$  and  $y$ , and *Spoiler* chooses, say, event  $i'_r = x$ . The reasoning is exactly similar to the case above.
    - \*  $(i_r, j_r)$  corresponds to events  $x$  and  $y$ . If *Spoiler* plays an  $\mathbf{U}_I$ -move or  $\mathbf{S}_I$ -move, chooses some event  $z$ , and forces *Duplicator* not to choose the corresponding event but another one in a neighbouring segment, then that event  $z$  must be less than  $(c + 1)$  away from  $t_b$ . If it happens before  $t_b$ , then  $t_a$  would have distance  $< (c - 1)$  to it. If it happens after  $t_b$ , then  $t_a$  would be  $< (c + 3)$  away from it. Assume that  $z$  happens before  $t_b$ . If  $z$  is a  $\{p\}$ -event, we divide  $(c - 1)$  by  $\frac{4}{5}$  to obtain  $\frac{5}{4} \cdot (c - 1) > |\text{seg}(z) - \text{seg}(i_r)|$  where  $\text{seg}(i_r) = m \cdot c' + 1$ . Observe that the  $\{p\}$ -event  $z'$  that *Duplicator* chooses as the response will be at most one more segment away. Then the claim holds regardless of *Spoiler* plays  $\mathbf{F}$ -part or  $\mathbf{U}$ -part (may cause a drift of two more segments) later. If  $z$  is a  $\{q\}$ -event, observe that its corresponding  $\{p\}$ -event in the same segment must be less than  $2 + \frac{1}{5} < 3 \cdot \frac{4}{5}$  away from  $z$ . Add this to  $(c - 1)$  and divide the result by  $\frac{4}{5}$  gives  $\frac{5}{4} \cdot (c - 1) + 3 < \frac{5}{4} \cdot (c + 2)$ . Again, the  $\{q\}$ -event  $z'$  that *Duplicator* chooses will be at most one more segment away. The case for  $z$  happens after  $t_b$  is similar. If *Spoiler* plays a  $\mathbf{C}_I^k$ -move or  $\overleftarrow{\mathbf{C}}_I^k$ -move, as  $I = (0, 1)$ , *Duplicator* can always make  $(i_{r+1}, j_{r+1})$  identical if necessary.
  - $(i_r, j_r)$  is not identical:  
We claim that no matter how *Spoiler* plays, *Duplicator* can always either make  $(i_{r+1}, j_{r+1})$  identical or, ensure that  $(i_{r+1}, j_{r+1})$  has not moved towards the nearest end by  $\geq c'$  segments. In the latter case the claim holds by the induction hypothesis. If *Spoiler* plays an  $\mathbf{C}_I^k$ -move or  $\overleftarrow{\mathbf{C}}_I^k$ -move, it is once again clear that *Duplicator* can follow a copy-cat strategy if necessary, but this is not always the case for  $\mathbf{U}_I$ -moves and  $\mathbf{S}_I$ -moves. In the following, we focus on  $\mathbf{U}_I$ -moves and  $\mathbf{S}_I$ -moves and assume that

*Spoiler* always chooses some event that is more than two events away from the current event, e.g.,  $j'_r > j_r + 2$ . If  $j'_r \leq j_r + 2$ , it is easy to see that *Duplicator* can simply choose  $i'_r = i_r + (j'_r - j_r)$  (unless  $(i_r, j_r)$  are very close to one of the ends, which will not happen).

Assume that  $(i_r, j_r)$  corresponds to a pair of  $\{p\}$ -events and (without loss of generality) assume that *Spoiler* chooses a position  $j'_r$  such that  $j'_r > j_r$ . If *Duplicator* can choose  $i'_r$  such that  $i'_r = j'_r$ , *Duplicator* chooses  $i'_r = j'_r$ . Then, if *Spoiler* plays **F**-part, it is immediate that  $i_{r+1} = j_{r+1}$ . If *Spoiler* plays **U**-part, then *Duplicator* makes  $i_{r+1} = j_{r+1}$  whenever possible. Otherwise, for example, if  $i_r < j_r$  and *Spoiler* chooses some  $\{p\}$ -event in  $(\tau_{i_r}^d, \tau_{j_r}^d)$  as  $i_{r+1}$ , then *Duplicator* chooses  $j_{r+1} = j_r + 2$ . Observe that  $i_{r+1}$  has moved towards  $j_r$  (and away from the nearest end). The claim holds by the induction hypothesis. If *Duplicator* cannot choose  $i'_r$  such that  $i'_r = j'_r$ , consider the following cases:

- \* *Duplicator* can choose  $i'_r$  such that  $i'_r = i_r + (j'_r - j_r)$ : If *Duplicator* cannot choose  $i'_r = j'_r$ , then *Duplicator* chooses  $i'_r = i_r + (j'_r - j_r)$ . As before, we know that  $\tau_{j'_r}^s < \tau_{j_r}^s + (c + 1)$ . It is easy to see that  $\text{seg}(i_{r+1}) - \text{seg}(i_r) < c'$  and  $\text{seg}(j_{r+1}) - \text{seg}(j_r) < c'$ , and hence the claim holds by the induction hypothesis.
- \* *Duplicator* cannot choose  $i'_r$  such that  $i'_r = i_r + (j'_r - j_r)$ : This can only happen when  $j'_r$  corresponds to a  $\{q\}$ -event. Observe that all  $\{p\}$ -events in neighbouring segments are separated by  $\frac{4}{5}$ . These imply that there exists  $t$  such that  $t - \tau_{j_r}^s = n = n' \cdot \frac{1}{5}$  for some  $n, n' \in \mathbb{N}_{>0}$ , and there exists  $|k_1|, |k_2| < 1, k_1, k_2 \neq 0$  such that  $t - \tau_{j_r}^s$  lies between

$$\begin{aligned} & \tau_{j'_r}^s - \tau_{j_r}^s = n_1 \cdot \frac{1}{5} + k_1 \cdot \epsilon, n_1 \in \mathbb{N}_{>0} \text{ and} \\ & \tau_{i_r + (j'_r - j_r)}^d - \tau_{i_r}^d = n_2 \cdot \frac{1}{5} + k_2 \cdot \epsilon, n_2 \in \mathbb{N}_{>0}. \end{aligned}$$

It is obvious that  $n_1 = n_2$ . If  $k_1 \cdot k_2 > 0$ , since there is no integer multiple of  $\frac{1}{5}$  that lies between, e.g.,  $n_1 \cdot \frac{1}{5}$  and  $n_1 \cdot \frac{1}{5} + \epsilon$ , this is a contradiction. If  $k_1 \cdot k_2 < 0$ , we must have  $n' = n_1 = n_2$ . This only happens when  $i_r + (j'_r - j_r)$  in  $\rho^d$  corresponds to event  $x$ . In this case, *Duplicator* chooses the corresponding event in a neighbouring segment. For example, if  $(i_r, j_r)$  corresponds to a pair of  $\{p\}$ -events,  $\text{seg}(i_r) = m \cdot c' + 1, \text{seg}(j_r) = m \cdot c', I = (2, 3)$  and  $j'_r = y'$ , then *Duplicator* chooses  $i'_r = x'$ . Now if *Spoiler* plays **F**-part, since we know that  $\tau_{j'_r}^s < \tau_{j_r}^s + (c + 1)$ , the claim holds. If *Spoiler* plays **U**-part, e.g., in the aforementioned example, *Spoiler* chooses  $i_{r+1} = x$ , then *Duplicator* chooses  $j_{r+1} = y''$  – the claim also holds.

Now assume that  $(i_r, j_r)$  corresponds to a pair of  $\{q\}$ -events and assume that the *Spoiler* chooses a position  $j'_r$  such that  $j'_r < j_r$ . Most cases can be argued in very similar ways. We consider the situation when *Duplicator* cannot choose  $i'_r$  such that  $i'_r = i_r + (j'_r - j_r)$ . If  $j'_r$  corresponds to a  $\{p\}$ -event then the argument is exactly similar to above. Otherwise if  $j'_r$  corresponds to a  $\{q\}$ -event, observe the fact that all  $\{q\}$ -events in neighbouring segments, except  $x$ , are separated by  $\frac{4}{5} + \frac{1}{3 \cdot m \cdot c' + 3} \cdot \epsilon$ . By a similar argument, if  $k_1 \cdot k_2 < 0$ , *Duplicator* chooses the corresponding event in a neighbouring segment. It can be argued in the same way that the claim holds regardless of *Spoiler* plays **F**-part or **U**-part later.  $\blacktriangleleft$

Lemma 8 implies that any  $C_{(0,1)}$ MTL formula of modal depth  $\leq m$  and largest constant  $\leq c$  cannot distinguish  $M_{m,c}$  and  $N_{m,c}$ . However, from Figure 5 it is obvious that

$$M_{m,c} \models \mathbf{F}(p \wedge \mathbf{C}_{(0,2)}^3 q) \wedge N_{m,c} \not\models \mathbf{F}(p \wedge \mathbf{C}_{(0,2)}^3 q),$$

as each interval like  $(t_a, t_b)$  in  $N_{m,c}$  contains at most two  $\{q\}$ -events. We thus have the theorem below, which can be seen as a strengthened version of a corresponding result in [24] (which holds for the future-only fragments).

► **Theorem 9.**  $C_0\text{MTL} \subsetneq C_{(0,1)}\text{MTL}$ .

**Counting in  $(0, b)$ .** We now show that once we bridge the expressiveness gap indicated by Theorem 9, we can derive a corresponding expressive completeness result for Q2MLO in the pointwise semantics. Before we give the main proof, let us state a crucial observation.

► **Theorem 10.**  $\text{Q2MLO}_0 \equiv \text{Q2MLO}$ .

**Proof.** We first note that  $\text{Q2MLO}_{0,\infty}$  is equally expressive as Q2MLO; this can be obtained as a simple corollary of the main result of [18] ( $\text{EMITL}_{0,\infty}$  is already as expressive as full EMITL), since all the automata modalities involved in the proof are counter free (aperiodic) and thus equivalent to  $\text{FO}[<]$  formulae of the form  $\vartheta(x_0, x)$ . To see that  $\text{Q2MLO}_0 \equiv \text{Q2MLO}_{0,\infty}$ , note that, e.g., the Q2MLO formula

$$\exists x (x_0 < x \wedge d(x_0, x) \in (a, \infty) \wedge \vartheta(x_0, x))$$

is equivalent to an EMITL formula  $\mathcal{A}_{(a,\infty)}$  where  $\mathcal{A}$  is the automaton equivalent of  $\vartheta(x_0, x)$ ; we assume (without loss of generality [34]) that  $\mathcal{A} = \langle \Sigma, S, s_0, \Delta, F \rangle$  is a DFA and in particular, at most one of the arguments holds at any position. Let  $\mathcal{B}^{s,\varphi}$  be the automaton obtained from  $\mathcal{A}$  by adding a new location  $s_F$ , declaring it as the only final location, and adding new transitions  $s' \xrightarrow{\varphi_a \wedge \varphi} s_F$  for every  $s' \xrightarrow{\varphi_a} s$  in  $\mathcal{A}$ . Let  $\mathcal{C}^s$  be the automaton obtained from  $\mathcal{A}$  by adding new non-final locations  $s'_0$  and  $s'_1$ , adding new transitions  $s'_0 \rightarrow s'_1$  (i.e. labelled with  $\top$ ) and  $s'_1 \xrightarrow{\varphi_a} s''$  for every  $s \xrightarrow{\varphi_a} s''$  in  $\mathcal{A}$ , and setting the initial location to  $s'_0$ . Intuitively,  $\mathcal{B}^{s,\varphi}$  enforces  $\varphi$  at the point when  $s$  is reached in  $\mathcal{A}$  and  $\mathcal{C}^s$  “runs”  $\mathcal{A}$  from  $s$ . We can argue that  $\mathcal{A}_{(a,\infty)}$  is equivalent to

$$\mathcal{A}_{(0,\infty)} \wedge \neg \bigvee_{s \in S} \mathcal{B}_{(0,a]}^{s,\varphi}$$

where  $\varphi = \neg \mathcal{C}^s$ . This can be translated into a Q2MLO<sub>0</sub> formula. ◀

We have thus reduced the problem to expressing Q2MLO<sub>0</sub> formulae in C<sub>0</sub>MITL. The proof below essentially follows [14, 16] with the exception that instead of the composition method [32] we use Myhill-Nerode congruence, which appears to be more natural in a pointwise setting. It suffices to show that we can use a C<sub>0</sub>MITL formula to express a Q2MLO<sub>0</sub> formula of the form

$$\exists x (x_0 < x \wedge d(x_0, x) \in (0, b) \wedge \vartheta(x_0, x)) \quad (1)$$

where  $\vartheta(x_0, x)$  is an  $\text{FO}[<]$  formula, as we can repeatedly apply the equivalence on the minimal subformula until the whole formula is turned into a C<sub>0</sub>MITL formula.

We say an  $\text{FO}[<]$  formula  $\vartheta(x_0, x)$  is *functional* if for any given timed word  $\rho$  and positions  $i_0, i$ , if we have  $\rho, i_0, i \models \vartheta(x_0, x)$  then  $i_0 < i$  and  $i$  is unique for  $i_0$ : if  $\rho, i_0, i' \models \vartheta(x_0, x)$  then it must be the case that  $i' = i$ . It is not hard to see that (1) remains equivalent if we replace  $\vartheta(x_0, x)$  by its “functional” counterpart

$$\vartheta'(x_0, x) = x_0 < x \wedge \vartheta(x_0, x) \wedge \forall x' (x_0 < x' < x \implies \neg \vartheta(x_0, x')) .$$

We recall some facts about functional formulae before stating the main theorem. Intuitively, once we restrict ourselves to the case of functional  $\vartheta(x_0, x)$ , then for any given position  $i_0$ , there can be only a bounded number of pairs of positions  $(i, j)$  such that  $i < i_0 < j$  and  $\rho, i, j \models \vartheta(x_0, x)$ . In particular if  $\rho, i_0, i \models \vartheta(x_0, x)$ , we can make use of counting modalities to enforce that  $\tau_i - \tau_{i_0} \in (0, b)$ .

► **Lemma 11.** *If  $\vartheta(x_0, x)$  is functional and  $i_0$  is a position in the timed word  $\rho$ , then  $|\{j \mid \rho, i, j \models \vartheta(x_0, x) \text{ and } i < i_0 < j\}| \leq r$  where  $r$  is the number of locations of the minimal DFA equivalent to  $\vartheta(x_0, x)$ .*

**Proof.** Suppose to the contrary that there exists a set  $\{(i_1, j_1), \dots, (i_{r+1}, j_{r+1})\}$  of  $r + 1$  distinct pairs of positions  $(i, j)$  (where  $j_1, \dots, j_{r+1}$  are all distinct) that satisfy the condition;  $i_1, \dots, i_{r+1}$  must also be all distinct as  $\vartheta(x_0, x)$  is functional. Let  $\mathcal{D}$  be the minimal DFA equivalent to  $\vartheta(x_0, x)$ . As there are only  $r$  locations in  $\mathcal{D}$ , it must be the case that  $\mathcal{D}$  reaches some specific location  $s$  after reading  $\rho[i_u, i_0]$  and  $\rho[i_v, i_0]$  for some  $u \neq v$ , and it follows that  $\rho, i_u, j_u \models \vartheta(x_0, x)$  and  $\rho, i_u, j_v \models \vartheta(x_0, x)$ . This contradicts the fact that  $\vartheta(x_0, x)$  is functional. ◀

If  $\vartheta(x_0, x)$  is functional, we say that a pair of positions  $(i_1, j_1)$  such that  $\rho, i_1, j_1 \models \vartheta(x_0, x)$  is of  $\vartheta$ -nesting depth at least  $m$  in  $\rho$  if there exist positions  $i_1 < \dots < i_m < j_m < \dots < j_1$  such that  $\rho, i_\ell, j_\ell \models \vartheta(x_0, x)$  for all  $\ell \in \{1, \dots, m\}$ . We say  $(i_1, j_1)$  is of  $\vartheta$ -nesting depth  $m$  in  $\rho$  if it is of  $\vartheta$ -nesting depth at least  $m$  but not  $m + 1$  in  $\rho$ . Let

$$R_{\vartheta}^{\geq m}(y_1) = \exists x_1, x_2, \dots, x_m, y_2, \dots, y_m (x_1 < x_2 < \dots < x_m < y_m < \dots < y_2 < y_1 \\ \wedge \vartheta(x_1, y_1) \wedge \vartheta(x_2, y_2) \wedge \dots \wedge \vartheta(x_m, y_m))$$

and  $R_{\vartheta}^m(y_1) = R_{\vartheta}^{\geq m}(y_1) \wedge \neg R_{\vartheta}^{\geq m+1}(y_1)$ . Intuitively,  $\rho, j_1 \models R_{\vartheta}^m(y_1)$  iff there exists  $i_1$  such that  $(i_1, j_1)$  is of  $\vartheta$ -nesting depth at least  $m$  in  $\rho$ .

► **Lemma 12.** *If  $\vartheta(x_0, x)$  is functional and  $(i, j)$  is of  $\vartheta$ -nesting depth  $m$  in the timed word  $\rho$ , then if  $(i', j')$  where  $j' < j$  is also of  $\vartheta$ -nesting depth  $m$  in  $\rho$  (i.e.  $\rho, j' \models R_{\vartheta}^m(y_1)$ ), we necessarily have  $i' < i$ .*

**Proof.**  $i' > i$  contradicts the fact that  $(i, j)$  is of  $\vartheta$ -nesting depth  $m$  in  $\rho$ , and  $i' = i$  contradicts the fact that  $\vartheta(x_0, x)$  is functional. ◀

► **Theorem 13.**  $C_0\text{MITL} \equiv Q2\text{MLO}$ .

**Proof.** Fix a functional formula  $\vartheta(x_0, x)$  and a timed word  $\rho$ . Let  $R_{\vartheta}^{m,\ell}(x_0)$  be the formula that says  $x_\ell$ , the  $\ell$ -th point  $> x_0$  satisfying  $R_{\vartheta}^m$ , also happens to satisfy  $\vartheta(x_0, x_\ell)$ , i.e.

$$R_{\vartheta}^{m,\ell}(x_0) = \exists x_1, \dots, x_\ell (x_0 < x_1 < \dots < x_\ell \wedge \vartheta(x_0, x_\ell) \\ \wedge \forall x (x \in (x_0, x_\ell] \implies (R_{\vartheta}^m(x) \iff \bigvee_{i \in \{1, \dots, \ell\}} x = x_i))) .$$

By Lemma 11 and Lemma 12, we know that  $\ell$  can at most be  $r + 1$  (where  $r$  is the number of locations of the minimal DFA equivalent to  $\vartheta(x_0, x)$ ). If  $(i_0, i)$  satisfies  $\vartheta(x_0, x)$ , then  $(i_0, i)$  must be of  $\vartheta$ -nesting depth  $m$  in  $\rho$  for some  $m \leq r$ . To express

$$\exists x (x_0 < x \wedge d(x_0, x) \in (0, b) \wedge \vartheta(x_0, x)) ,$$

we take the disjunction over all the possible choices of  $m$ 's and  $\ell$ 's:

$$\bigvee_{m \in \{1, \dots, r\}} \left( \bigvee_{\ell \in \{1, \dots, r+1\}} \left( \exists x_1, \dots, x_\ell (x_0 < x_1 < \dots < x_\ell \wedge d(x_0, x_\ell) \in (0, b) \right. \right. \\ \left. \left. \wedge \bigwedge_{i \in \{1, \dots, \ell\}} R_{\vartheta}^m(x_i) \wedge R_{\vartheta}^{m,\ell}(x_0) \right) \right) .$$

The formula above is equivalent to

$$\bigvee_{m \in \{1, \dots, r\}} \left( \bigvee_{\ell \in \{1, \dots, r+1\}} ((\mathbf{C}_{(0,b)}^\ell R_\vartheta^m) \wedge R_\vartheta^{m,\ell}) \right)$$

where  $R_\vartheta^m$ ,  $R_\vartheta^{m,\ell}$  are the LTL equivalents of  $R_\vartheta^m(y_1)$  and  $R_\vartheta^{m,\ell}(x_0)$ , respectively.  $\blacktriangleleft$

► **Corollary 14.**  $\mathbf{C}_0\text{MITL}$  with untimed automata modalities is expressively complete for Q2MSO.

#### 4 Expressive completeness for $\text{FO}[\langle, +1]$

**Generalising EMITL.** We know that in the continuous semantics PQ2MLO [20] is expressively complete for  $\text{FO}[\langle, +1]$ ; in other words, the only expressiveness gap between (decidable) Q2MLO and (undecidable)  $\text{FO}[\langle, +1]$  is the capability to express punctualities. Unfortunately, this pleasant result does not hold in the pointwise semantics.

► **Theorem 15.** PQ2MLO is strictly less expressive than  $\text{FO}[\langle, +1]$ .

**Proof.** Thanks to Theorem 13, it suffices to show that  $\mathbf{C}_0\text{MTL}$  is strictly less expressive than  $\text{FO}[\langle, +1]$ . In fact, we can prove the stronger result that MTL with arbitrary rational endpoints (which subsumes  $\mathbf{C}_I^k$ ) is still insufficient for expressing the property below (“ $X$  holds at the first event in  $I$  from now’):

$$\mathbf{B}_I^\rightarrow(x, X) = \exists x' (x < x' \wedge d(x, x') \in I \wedge X(x') \wedge \neg \exists x'' (x < x'' < x' \wedge d(x, x'') \in I)) \quad (2)$$

The detailed proof can be found in the full version of this paper.  $\blacktriangleleft$

The theorem above suggests that we need more involved extensions to make Q2MLO as expressive as  $\text{FO}[\langle, +1]$  in the pointwise semantics; at least we must be able to specify (2). PnEMTL [23] is a generalisation of EMTL where instead of just between the current point and a single witness point, one can use “Pnueli automata’ modalities to specify behaviours between multiple witness points as well. More precisely, the semantics of Pnueli automata modalities are defined as follows:

- $\rho, i \models \mathcal{F}_{I_1, \dots, I_k}(\mathcal{A}_1, \dots, \mathcal{A}_k)$  iff there exists  $j_1, \dots, j_k$  such that
  1.  $i < j_1 < \dots < j_k$ .
  2. For each  $\ell \in \{1, \dots, k\}$ ,  $\tau_{j_\ell} - \tau_i \in I_\ell$ .
  3. For each  $\ell \in \{1, \dots, k\}$ , there is an accepting run of  $\mathcal{A}_\ell$  on  $a_{j_{\ell-1}} \dots a_{j_\ell}$  ( $\ell > 1$ ) or  $a_i \dots a_{j_\ell}$  ( $\ell = 1$ ) such that for each  $m$ ,  $j_{\ell-1} \leq m \leq j_\ell$  (or  $i \leq m \leq j_\ell$ ),  $\rho, m \models \varphi_{a_m}$  ( $a_m \in \{1, \dots, n_\ell\}$  where  $n_\ell$  is the arity of the alphabet of  $\mathcal{A}_\ell$ ).

■  $\rho, i \models \mathcal{P}_{I_1, \dots, I_k}(\mathcal{A}_1, \dots, \mathcal{A}_k)$  (the past counterpart) is defined symmetrically.

In [23], it is also shown that PnEMTL is expressively equivalent to PGQMSO, a generalisation of PQ2MSO with the following rule:

- if  $\vartheta_1(x_0, x_1), \dots, \vartheta_k(x_0, x_k)$  are MSO[ $\langle$ ] formulae where for each  $\vartheta_\ell(x_0, x_\ell)$  ( $\ell \in \{1, \dots, k\}$ ),  $x_0$  and  $x_\ell$  are the only free first-order variables, then  $\exists x_1 \dots \exists x_k (x_0 < x_1 < \dots < x_k \wedge d(x_0, x_1) \in I_1 \wedge \dots \wedge d(x_0, x_k) \in I_k \wedge \vartheta(x_0, x_1) \wedge \dots \wedge \vartheta(x_0, x_k))$  and the past counterpart, where  $I_1, \dots, I_k$  are (possibly singular) intervals with endpoints in  $\mathbb{N}_{\geq 0} \cup \{\infty\}$ , are also PGQMSO formulae (with free first-order variable  $x_0$ ).

As we can easily express (2) in PGQMLO (the first-order fragment of PGQMSO) [23, Theorem 6.4], we have the following corollary.

► **Corollary 16.** PQ2MLO  $\subsetneq$  PGQMLO.

**Order of fractional parts.** While we have not been able to prove or disprove whether  $\text{PGQMLO} \equiv \text{FO}[\langle, +1]$ , we can show that a simple extension of  $\text{PnEMTL}$ , where one is allowed to specify orders of fractional parts of witnesses, can capture the full expressiveness of  $\text{FO}[\langle, +1]$ . Let  $\mathcal{F}_{I_1, \dots, I_k}^{\text{frac}, N}(\mathcal{A}_1, \dots, \mathcal{A}_k)$  be the new modalities where

- $\mathcal{A}_1, \dots, \mathcal{A}_k$  are all counter free (aperiodic).
- Each of  $I_1, \dots, I_k$  is a left-closed, right-open subinterval of  $[-N, N]$  with integer endpoints and length 1 (e.g.,  $[3, 4)$  or  $[-7, -6)$ ).

The intended semantics when evaluated at position  $i_0$  is as follows:

- There exists  $k$  ‘witness’ points  $i_1, \dots, i_k$  such that  $i_\ell \in I_\ell$  for all  $\ell \in \{1, \dots, k\}$ .
- The fractional parts of the witnesses are in this order, i.e.  $\text{frac}(\tau_{i_1}) < \dots < \text{frac}(\tau_{i_k})$ .
- For each  $\ell \in \{1, \dots, k\}$ ,  $\mathcal{A}_\ell$  has an accepting run on the ‘stacked’ word [17] formed by all events in  $\tau_{i_0} + [-N, N]$  with the fractional parts in  $[\tau_{i_{\ell-1}}, \tau_{i_\ell})$ . More precisely, the transitions of  $\mathcal{A}$  are partitioned into  $2N$  sets, where each set is only enabled for events in the corresponding unit subinterval of  $\tau_{i_0} + [-N, N]$ .

In the same way we define the past counterpart  $\mathcal{P}^{\text{frac}}$  and its semantics, and denote by  $\text{PGQMLO}^{\text{frac}}$  the extension of  $\text{PGQMLO}$  with these modalities.

► **Theorem 17.**  $\text{PGQMLO}^{\text{frac}} \equiv \text{FO}[\langle, +1]$ .

**Proof (sketch).** Following [21], the main challenge is to express formulae of the form

$$\begin{aligned} \exists z_0 \dots \exists z_{n-1} \left( x = z_0 < \dots < z_{n-1} \wedge d(x, z_{n-1}) < 1 \right. \\ \wedge \bigwedge \{ \Phi_i(z_i) : 0 \leq i < n \} \\ \wedge \bigwedge \{ \forall u (z_i < u < z_{i+1} \implies \Psi_i(u)) : 0 \leq i < n-1 \} \\ \left. \wedge \forall u (z_{n-1} < u \wedge d(x, u) < 1 \implies \Psi_{n-1}(u)) \right) \end{aligned}$$

where  $\Phi_i$  and  $\Psi_i$  are Boolean combinations of atomic formulae. This is readily possible with  $\mathcal{F}^{\text{frac}}$  and subformulae of the forms  $\mathbf{F}_{=1} p$  and  $\mathbf{\bar{F}}_{=1} p$ . ◀

## 5 Conclusion and future work

The general consensus in the real-time verification community is that the continuous interpretations of timed logics are more well behaved and admit more robust characterisations. The present paper showed that by allowing a mild generalisation of the counting modalities, we can recover the pleasant expressive completeness result for  $\text{Q2MLO}$  – one of the most expressive decidable fragments of  $\text{FO}[\langle, +1]$  – in the pointwise semantics as well. On the other hand, we also showed that as opposed to the situation in the continuous semantics, the full expressiveness of  $\text{FO}[\langle, +1]$  cannot be achieved by simply adding punctual predicates – we remedy this by proposing a more involved variant of  $\text{PnEMTL}$ , which we showed to be expressively complete for  $\text{FO}[\langle, +1]$ . We list some possible future directions below.

- The expressive completeness for  $\text{FO}[\langle, +1]$  is achieved with a family of modalities that enable one to specify the relative orders of the fractional parts of the points involved. This begs the question of whether this feature is really necessary; in other words, is  $\text{PGQMLO}$  strictly less expressive than  $\text{FO}[\langle, +1]$ ?
- Is it possible to add (or perhaps restricted versions of) the modalities  $\mathcal{F}^{\text{frac}}$  and  $\mathcal{P}^{\text{frac}}$  to  $\text{GQMLO}$  while retaining the decidability of the satisfiability problem?



- It is known that the pointwise and continuous interpretations of  $\text{FO}[\langle, +1]$  are actually equally expressive [9], if one considers a special “timed word” form of signals [5, 7, 28]. Does a similar result hold for Q2MLO as well?
- There are some existing SMT-based tools for checking the satisfiability of CMITL in the continuous semantics (e.g., [4]), although they require a predetermined bound  $k$  on the variability of signals. In light of the recent developments in back-end algorithms [10, 11], it would be interesting to see how a timed-automata-based implementation compares in terms of practical performance.

---

## References

- 1 Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.
- 2 Rajeev Alur and Thomas A. Henzinger. Back to the future: Towards a theory of timed regular languages. In *FOCS*, pages 177–186. IEEE Computer Society Press, 1992.
- 3 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.
- 4 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.
- 5 Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. *Information and Computation*, 208(2):97–116, 2010.
- 6 Patricia Bouyer, François Laroussinie, Nicolas Markey, Joël Ouaknine, and James Worrell. Timed temporal logics. In *Models, Algorithms, Logics and Tools - Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday*, volume 10460 of *LNCS*, pages 211–230. Springer, 2017.
- 7 Deepak D’Souza and Pavithra Prabhakar. On the expressiveness of mtl in the pointwise and continuous semantics. *International Journal on Software Tools for Technology*, 9(1):1–4, 2007.
- 8 Deepak D’Souza and Nicolas Tabareau. On timed automata with input-determined guards. In Yassine Lakhnech and Sergio Yovine, editors, *FORMATS/FTRTFT*, volume 3253 of *LNCS*, pages 68–83. Springer, 2004.
- 9 Deepak D’Souza and Raveendra Holla. Equivalence of pointwise and continuous semantics of fo with linear constraints. In *ICLA*, pages 40–45, 2021.
- 10 R Govind, Frédéric Herbretreau, and Igor Walukiewicz. Abstractions for the local-time semantics of timed automata: a foundation for partial-order methods. In *LICS*, pages 1–14. ACM/IEEE, 2022.
- 11 Frédéric Herbretreau, B Srivathsan, and Igor Walukiewicz. Checking timed büchi automata emptiness using the local-time semantics. In *CONCUR*, volume 243 of *LIPICs*, pages 12:1–12:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 12 Yoram Hirshfeld and Alexander Rabinovich. A framework for decidable metrical logics. In *ICALP*, volume 1644 of *LNCS*, pages 422–432. Springer, 1999.
- 13 Yoram Hirshfeld and Alexander Rabinovich. Expressiveness of metric modalities for continuous time. *Logical Methods in Computer Science*, 3(1):1–11, 2007.
- 14 Yoram Hirshfeld and Alexander Rabinovich. Continuous time temporal logic with counting. *Information and Computation*, 214:1–9, 2012.
- 15 Yoram Hirshfeld and Alexander Moshe Rabinovich. Logics for real time: Decidability and complexity. *Fundamenta Informaticae*, 62(1):1–28, 2004.
- 16 Yoram Hirshfeld and Alexander Moshe Rabinovich. An expressive temporal logic for real time. In *MFCS*, volume 4162 of *LNCS*, pages 492–504. Springer, 2006.
- 17 Hsi-Ming Ho. On the expressiveness of metric temporal logic over bounded timed words. In *RP*, volume 8762 of *LNCS*, pages 138–150. Springer, 2014.




- 18 Hsi-Ming Ho. Revisiting timed logics with automata modalities. In *HSCC*, pages 67–76. ACM, 2019.
- 19 Hsi-Ming Ho and Khushraj Madnani. When do you start counting? revisiting counting and pnueli modalities in timed logics. In *DCM. EPTCS*, 2023. To appear.
- 20 Paul Hunter. When is metric temporal logic expressively complete? In *CSL*, volume 23 of *LIPICs*, pages 380–394. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
- 21 Paul Hunter, Joël Ouaknine, and James Worrell. Expressive completeness for metric temporal logic. In *LICS*, pages 349–357. IEEE Computer Society Press, 2013.
- 22 Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
- 23 Shankara Narayanan Krishna, Khushraj Madnani, Manuel Mazo Jr., and Paritosh Pandya. From non-punctuality to non-adjacency: A quest for decidability of timed temporal logics with quantifiers. *Formal Aspects of Computing*, 2022.
- 24 Shankara Narayanan Krishna, Khushraj Madnani, and Paritosh K. Pandya. Metric temporal logic with counting. In *FoSSaCS*, volume 9634 of *LNCS*, pages 335–352. Springer, 2016.
- 25 Shankara Narayanan Krishna, Khushraj Madnani, and Paritosh K. Pandya. Making metric temporal logic rational. In *MFCS*, volume 83 of *LIPICs*, pages 77:1–77:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
- 26 Shankara Narayanan Krishna, Khushraj Madnani, and Paritosh K. Pandya. Logics meet 1-clock alternating timed automata. In *CONCUR*, volume 118 of *LIPICs*, pages 39:1–39:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- 27 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.
- 28 Joël Ouaknine, Alexander Rabinovich, and James Worrell. Time-bounded verification. In *CONCUR*, volume 5710 of *LNCS*, pages 496–510. Springer, 2009.
- 29 Joël Ouaknine and James Worrell. On metric temporal logic and faulty turing machines. In *FoSSaCS*, volume 3921 of *LNCS*, pages 217–230. Springer, 2006.
- 30 Paritosh K. Pandya and Simoni S. Shah. On expressive powers of timed logics: Comparing boundedness, non-punctuality, and deterministic freezing. In *CONCUR*, volume 6901 of *LNCS*, pages 60–75. Springer, 2011.
- 31 Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.
- 32 Saharon Shelah. The monadic theory of order. *The Annals of Mathematics*, 102(3):379, November 1975.
- 33 Thomas Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *FTRTFT*, volume 863 of *LNCS*, pages 694–715. Springer, 1994.
- 34 Pierre Wolper and Moshe Y. Vardi. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.
- 35 Yuchen Zhou, Dipankar Maity, and John S. Baras. Timed automata approach for motion planning using metric interval temporal logic. In *ECC*, pages 690–695. IEEE, 2016.



# Analyzing Complex Systems with Cascades Using Continuous-Time Bayesian Networks

Alessandro Bregoli ✉ 

Department of Informatics, Systems and Communication, University of Milano-Bicocca, Italy

Karin Rathsman ✉ 

European Spallation Source ERIC, Lund, Sweden

Marco Scutari ✉ 

Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Lugano, Switzerland

Fabio Stella ✉ 

Department of Informatics, Systems and Communication, University of Milano-Bicocca, Italy

Søren Wengel Mogensen ✉ 

Department of Automatic Control, Lund University, Sweden

---

## Abstract

---

Interacting systems of events may exhibit *cascading* behavior where events tend to be temporally clustered. While the cascades themselves may be obvious from the data, it is important to understand which states of the system trigger them. For this purpose, we propose a modeling framework based on *continuous-time Bayesian networks* (CTBNs) to analyze cascading behavior in complex systems. This framework allows us to describe how events propagate through the system and to identify likely *sentry states*, that is, system states that may lead to imminent cascading behavior. Moreover, CTBNs have a simple graphical representation and provide interpretable outputs, both of which are important when communicating with domain experts. We also develop new methods for knowledge extraction from CTBNs and we apply the proposed methodology to a data set of alarms in a large industrial system.

**2012 ACM Subject Classification** Mathematics of computing → Markov processes; Mathematics of computing → Bayesian networks

**Keywords and phrases** event model, continuous-time Bayesian network, alarm network, graphical models, event cascade

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.8

**Funding** *Søren Wengel Mogensen*: The work of SWM was funded by a DFF-International Postdoctoral Grant (0164-00023B) from Independent Research Fund Denmark. SWM is a member of the ELLIIT Strategic Research Area at Lund University.

**Acknowledgements** The authors would like to thank Per Nilsson for sharing his knowledge about the cryogenics plant and for providing valuable feedback on the work presented in this paper.

## 1 Introduction

Many real-world phenomena can be modeled as interacting sequences of events of different types. This includes social networks where user activity influences the activity of other users [11]. In healthcare, patient history may be modeled as a sequence of events [46]. In this paper, we focus on an industrial application in which the events are alarm signals of a complex engineered system. As an illustration, consider Figure 1. Three different alarms ( $A$ ,  $B$ , and  $C$ ) monitor a process each within an industrial system. These processes may, for instance, represent measured temperatures or pressures. An alarm transitions to *on* when its the process it monitors leaves a prespecified range of values and transitions to *off* when the



© Alessandro Bregoli, Karin Rathsman, Marco Scutari, Fabio Stella, and Søren Wengel Mogensen; licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 8; pp. 8:1–8:21

Leibniz International Proceedings in Informatics



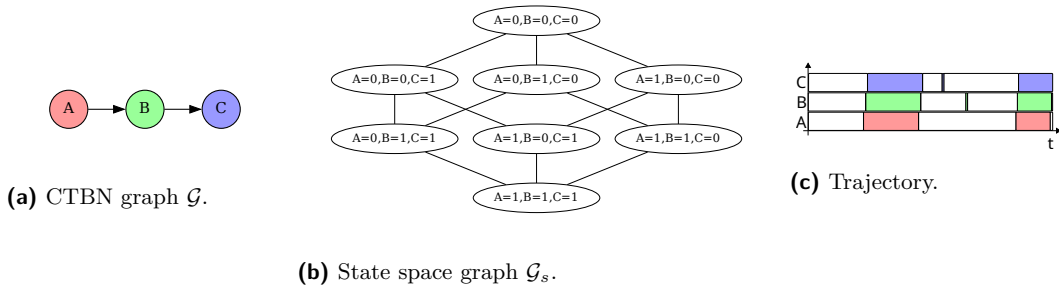
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

process is again within the range corresponding to “normal operation”. In Figure 1c, the colored segments correspond to the alarm being *on*. When an alarm changes state, we say that an *event* occurs.

In particular, we are interested in systems that exhibit a *cascading* behavior in the sense that events are strongly clustered in time. Therefore, we should use a model class which is capable of expressing cascades. However, important information may also be contained in non-cascading parts of the observed data. This is explained by the fact that our goal is twofold: We wish to understand which states of the system lead to cascades, and we also wish to understand how different components of the system interact. These goals are connected as understanding the inner workings of the system will also help us understand the cascading behavior. We can achieve both by using *continuous-time Bayesian networks* (CTBNs) [32], a class of parsimonious Markov processes with a discrete state space. Moreover, CTBNs are equipped with a graphical representation of how different components interact. In our application, data is sampled at a very high frequency and using a continuous-time modeling framework makes for a conceptually and computationally simple approach. In CTBNs, we define the concept of a *sentry state* which is a state that may lead to an imminent cascade. In an industrial setting, identifying such states may give operators an early warning, which in turn facilitates the mitigation of underlying issues before an actual alarm cascade occurs. Moreover, sentry states can be used to apply state-based alarm techniques [18], which otherwise require a known alarm structure.

In applications to complex systems, analysts often need to communicate findings to domain experts. CTBNs also allow easy communication using their graphical representation which is crucial for their applicability to real-world problems. We describe a useful interpretation of the graph of a CTBN and provide some new results in this direction. We apply the methods developed in this paper to a challenging data set from the European Spallation Source (ESS), a research facility in Lund, Sweden. This data set describes how alarms propagate through a subsystem of the neutron source at ESS.

The paper is structured as follows. We start with a description of the ESS data as this motivates the following developments (Section 2). In Section 3, we describe related work and compare CTBNs with other approaches. Section 4 describes continuous-time Bayesian networks. Sections 5 and 6 contain the main theoretical contributions of the paper: Section 5 describes the concept of a sentry state while Section 6 explains how the graphical



■ **Figure 1** a) Graph  $\mathcal{G}$  of a CTBN consisting of three nodes ( $A$ ,  $B$ ,  $C$ ). b) State space graph  $\mathcal{G}_s$  of the CTBN, i.e., nodes represent states of the CTBN and two states are adjacent if a transition from one to the other, and vice versa, is possible. c) A trajectory for the CTBN. A white segment indicates that the corresponding process is in state 0 (off), while a colored segment indicates that the corresponding process is in state 1 (on).

representation of a CTBN may assist interpretation and communication. Section 7 presents numerical experiments, including a description of the ESS data analysis. A discussion concludes the main paper and the appendices contain auxiliary results.

## 2 Data set

The European Spallation Source ERIC (ESS) is a large research facility which is being built in Lund, Sweden. Its main components include a linear proton accelerator, a tungsten target, and a collection of neutron instruments [16]. It comprises a large number of systems, including an integrated control system [15]. The facility has a goal of 95% availability and state-of-the-art alarm handling may contribute to reaching this goal.

Operators of large facilities are often facing large quantities of data in real time and good tools may help system understanding and support decision making. Operators rely on alarm systems to warn them about unexpected behavior. However, alarm problems are common [18]. One example is that of *cascading alarms*. In large facilities, different alarms monitor different processes and when an issue occurs this may result in a large number of alarms that occur within a short time frame due to the interconnectedness of the different processes. Operators will often find it difficult to respond to such cascades as hundreds or thousands of alarms may sound, making it difficult to identify the underlying issue.

The alarm system has two purposes. One, it should help operators foresee and mitigate fault situations. Two, it should help operators understand a fault situation. In this paper, we illustrate how the methods we propose can help achieve these goals using data from the accelerator cryogenics plant at ESS, which has been in operation since 2019.

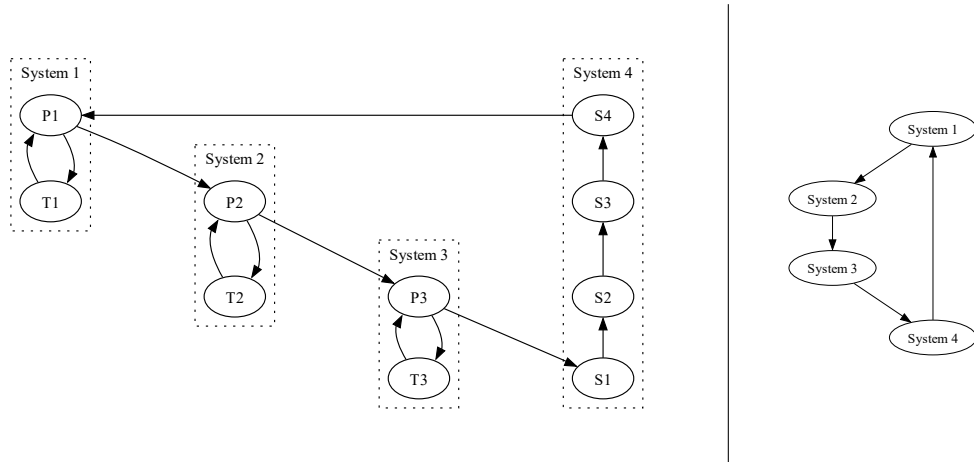
► **Example 1** (Simplified ESS alarm network). *In this example, we will look at a simplified version of the alarm system at ESS. Each of the alarm processes  $P1, T1, P2, T2, P3, T3, S1, S2, S3,$  and  $S4$  in Figure 2 (left) monitors a physical process, e.g., a temperature or a pressure. At each time point, each alarm process is either 1 (alarm is on) or 0 (alarm is off). An edge,  $\rightarrow$ , in the graph implies a dependence in transition rates, e.g., the rate with which  $P1$  changes its state depends only on the current states of  $P1, T1,$  and  $S4$ . Alarm cascades occur when a certain state triggers a fast progression of alarm onsets. In this formalism, this is modeled by the dependence of transition rates on the current state: therefore, cascades tend to unfold along the directed edges of the graph.*

*Large engineered systems can often be divided into subsystems (in Figure 2, Systems 1, 2, 3, and 4). In alarm networks, a group of alarms may monitor processes that are known to be correlated as they are measured from the same subsystem. Moreover, many systems of a realistic size comprise so many alarms or variables that processes must be grouped into subsystems to enable a system-level understanding of their dependencies. In Section 6, we show that graphical representations using these subsystems (e.g., Figure 2 (right)), instead of the processes themselves, are also useful and have a clear interpretation.*

## 3 Related work

Our work has connections to several major directions in the literature of dynamical models. We will focus on methods that are relevant for modeling cascades of events.

Continuous-time Markov processes with a discrete state space are often used as models of cascading failures in complex networks such as power grids [34, 28] and as models of burst behavior in biological cells [6]. They are also used in chemistry [4], reliability modeling [20], queueing theory [26], and genetics [7]. Graphs may be used as representations of networks



■ **Figure 2** Graphs from Example 1. Left: Graph such that each node represents an alarm process. Edges,  $\rightarrow$ , represent sparsity in transition rate dependence. The transition rate of each process,  $A$ , only depends on its own state and the states of its *parent* processes, i.e., processes  $B$  such that the edge  $B \rightarrow A$  is in the graph. Right: Graph representing transition rate dependencies between entire subsystems rather than between individual alarm processes. This graph is computed from the larger graph on the left and can be given a mathematical interpretation (Section 6).

and several methods use graphs to represent cascading structures [30, 29]. Some of these methods use simple models of contagion and study influential nodes in a graph [22]. Among these we find the *independent cascade model* and the *linear threshold model* [21, 12]. These models are targeted at applications in which cascades or “epidemics” are the salient feature.

*Change-point detection* methods aim to recover the time points at which distributional changes occur. There is a large literature on change-point detection in various classes of stochastic processes and application fields such as neuroscience [37], DNA sequence segmentation [9], speech recognition [39], and climate change [36]. [3] provides a survey on change-point methods in discrete-time models. [42] also provides a survey and discusses subsampling of continuous-time processes. [45] provides a method for change-point detection in a class of multivariate point processes. There are subtle, but important, differences between the task at hand and change-point detection. The alarm network that motivates our work is thought to operate in the same mode throughout the observation period. Moreover, the detection of cascades is trivial in this application as they are evident from simply visualizing the data. Instead, we focus on identifying the states that are likely to lead to a cascade. In addition, our modeling approach should allow for a qualitative understanding of the interactions between system components.

As a result, we would like to explicitly model how events propagate through the system. A traditional approach is to use *point processes* [13]. In our setting, we can think of a point process as a sequence of pairs  $(t_1, e_1), (t_2, e_2), (t_3, e_3), \dots$  such that  $(t_i)$  is an increasing sequence of time points and  $(e_i)$  denotes the type of event. Point process models have been used for modeling cascades of failures [23]. [35] describes self-exciting point processes that model cascading behavior using explicit exogenous influences on the system. These methods can in principle also be applied in the setting of this paper. The CTBN-based method proposed in this paper models the state of the system directly and this facilitates the notion of *senary states* which is central to this paper. Moreover, as the alarm status takes values in a discrete set, the CTBN provides a natural representation of the alarm data.

While the above describes general approaches to the modeling of dynamical systems with cascading behavior, there are also more specialized methods for handling or analyzing alarm cascades in large industrial systems. We now summarize the connections to this work; see also [2], which is a recent review of methods for alarm cascade problems (in that paper known as alarm *floods*). In short, so-called *knowledge-based* methods use expert knowledge of the cause-effect relations in the system to find root causes and explain alarm cascades. Among these are *multilevel flow modeling* [24] and *signed directed graphs* [44]. In contrast, there is also a large number of *data-based* methods. Data-based methods can be subdivided into classes of methods depending on their purpose. Some approaches aim to classify the fault type from an input sequence of alarms or to simply reduce the number of alarms, e.g., using data mining, clustering, or machine learning methods such as *artificial neural networks* [47, 38, 5]. This goal is somewhat different from ours and our method is more closely related to methods using *probabilistic graphical models*, in particular dynamic Bayesian networks [19]. These methods produce an easily interpretable output represented by a graph. Our contribution in relation to this prior work is twofold. 1) We introduce the CTBN framework as a natural further development of alarm modeling. This allows continuous-time modeling which is useful for our data as it is collected using a very high sampling rate. Discretization of time will therefore either lead to a prohibitively large number of time intervals or to a loss of temporal information if using fewer, longer time intervals. 2) We define the novel concept of a *sentry state* which identifies a set of states that are of interest when analyzing cascading behavior. The sentry state concept may also be used in other alarm propagation models and is not specific to CTBNs.

## 4 Models

Continuous-time Bayesian networks (CTBNs; [31]) are a class of continuous-time Markov processes (CTMPs) with a factored state space and a certain sparsity in how transition rates depend on the current state. This sparsity can be represented by a directed graph. In this sense, they are similar to classical Bayesian networks [33] but their directed graphs are allowed to contain cycles. CTBNs have proved to be both effective and efficient representations of discrete-state continuous-time dynamical systems [1, 43, 25]. We first define a CTMP.

### 4.1 Continuous-Time Markov Processes

A *continuous-time Markov process* (CTMP) [41] is a continuous-time stochastic process  $\mathbf{X} = \{X(t) : t \in [0, \infty)\}$  which satisfies the following Markov property:

$$X(t_1) \perp\!\!\!\perp X(t_3) | X(t_2), \quad \forall t_1 < t_2 < t_3, \quad (1)$$

where  $\cdot \perp\!\!\!\perp \cdot | \cdot$  denotes conditional independence. The state of the process  $\mathbf{X}$  changes in continuous-time and takes values in the domain  $S$  which we assume to be a finite set. In our application, each state  $s \in S$  can be represented by an  $n$ -vector with binary entries indicating whether each alarm is *on* or *off*. A CTMP can be parameterized by the *initial distribution*  $P_0$  and the *intensity matrix*  $Q_{\mathbf{X}}$ . The initial distribution  $P_0$  is any distribution on the state space. The intensity matrix  $Q_{\mathbf{X}}$  models the evolution of the stochastic process  $\mathbf{X}$ . Each row of  $Q_{\mathbf{X}}$  sums to 0 and models two distinct processes:

1. The time when  $\mathbf{X}$  abandons the current state  $x$ , which follows an *exponential distribution* with parameter  $q_x \in \mathbb{R}^+$ .
2. The state to which  $\mathbf{X}$  transitions when abandoning the state  $x$ . This follows a *multinomial distribution* with parameters  $\theta_{xy} = \frac{q_{xy}}{q_x}$ ,  $x, y \in S$ ,  $x \neq y$ .



An instance of the intensity matrix  $Q_X$ , when  $S$  has three states, is as follows

$$Q_{\mathbf{X}} = \begin{bmatrix} -q_{x_1} & q_{x_1x_2} & q_{x_1x_3} \\ q_{x_2x_1} & -q_{x_2} & q_{x_2x_3} \\ q_{x_3x_1} & q_{x_3x_2} & -q_{x_3} \end{bmatrix} \quad q_{x_i} > 0, q_{x_ix_j} \geq 0 \forall i, j. \quad (2)$$

We say that a realization of a CTMP,  $\sigma$ , is a *trajectory*. This is a right-continuous, piecewise constant function of time. It can be represented as a sequence of time-indexed events,

$$\sigma = \{\langle t_0, X(t_0) \rangle, \langle t_1, X(t_1) \rangle, \dots, \langle t_I, X(t_I) \rangle\}, \quad t_0 < t_1 < \dots < t_I. \quad (3)$$

## 4.2 Continuous-Time Bayesian Networks

General CTMPs do not assume any sparsity. CTBNs impose structure on a CTMP by assuming a factored state space  $S = \{S_1 \times S_2 \times \dots \times S_L\}$  such that  $X(t) = (X_1(t), \dots, X_L(t)) \in S$  where each  $S_j$ ,  $j = 1, \dots, L$ , represents the domain of a distinct component of the process.<sup>1</sup> In the alarm data,  $S_j = \{0, 1\}$  for each  $j$  and  $X_j(t)$  indicates if the  $j$ 'th alarm is *on* or *off* at time  $t$ . The structure imposed by the CTBN is useful when interpreting a learned model. In essence, the CTBN framework allows us to learn which components of the system act independently, or conditionally independently, and this can be communicated to experts.

A CTBN is a tuple  $\mathcal{N} = \langle P_0, \mathbf{X}, \mathcal{G}, \mathbf{Q}_X \rangle$  where  $\mathbf{X} = \{\mathbf{X}_1, \dots, \mathbf{X}_L\}$  is a set of stochastic processes. A CTBN is specified by:

- An initial probability distribution  $P_0$  on the factored state space  $S$ .
- A continuous-time transition model, specified as:
  - a directed (possibly cyclic) graph  $\mathcal{G}$  with node set  $\mathbf{X}$ ;
  - a set of conditional intensity matrices  $\mathbf{Q}_{\mathbf{X}_j | \text{Pa}(\mathbf{X}_j)}$  for each process  $\mathbf{X}_j \in \mathbf{X}$ .

Given the graph  $\mathcal{G}$ , each node/process  $\mathbf{X}_j$  has a *parent set*  $\text{Pa}(\mathbf{X}_j)$  consisting of all nodes/processes,  $\mathbf{X}_i$ , with an edge directed from  $\mathbf{X}_i$  to  $\mathbf{X}_j$  in  $\mathcal{G}$ ,  $\mathbf{X}_i \rightarrow \mathbf{X}_j$ . A *conditional intensity matrix* (CIM)  $\mathbf{Q}_{\mathbf{X}_j | \text{Pa}(\mathbf{X}_j)}$  consists of a set of intensity matrices, one for each possible configuration of the states of the parent set  $\text{Pa}(\mathbf{X}_j)$  of the node/process  $\mathbf{X}_j$ , that is, one for each element of  $\times_{\mathbf{X}_i \in \text{Pa}(\mathbf{X}_j)} S_i$ . The CIM describes how the transition intensity of process  $j$  depends on the state of the system. However, it does not necessarily depend on the state of every other process, but only on the states of the processes that are parents of  $j$ .

In a CTBN only one process can transition at any given time. This assumption is reasonable for the alarm data as it is sampled at a high rate. When we apply this model to the alarm data, the interpretation is straightforward. The CIM of  $\mathbf{X}_j$  describes how likely this alarm is to change its state (from *on* to *off*, or from *off* to *on*), and this only depends on the current state of the alarms in its parent set.

► **Example 2.** Assume we observe three alarm processes ( $A$ ,  $B$ , and  $C$ ) each monitoring a measured process and that we represent this by a stochastic process  $\mathbf{X}$  that takes values in  $\{0, 1\} \times \{0, 1\} \times \{0, 1\}$  indicating the status of each of the three alarms. If  $\mathbf{X}$  is a general CTMP, then the transition rates of each alarm process may depend on the entire state of the system. On the other hand, if  $\mathbf{X}$  is a CTBN represented by the graph  $\mathcal{G}$  in Figure 1a there is a certain sparsity in the way the transition rates depend on the current state. It follows

<sup>1</sup> A CTBN is specialization of a CTMP. It is possible to reformulate a CTBN as a CTMP by applying the so-called amalgamation procedure [41].

directly that the transition rate of alarm process  $C$  only depends on the states of processes  $B$  and  $C$ . Similarly, the transition rate of alarm process  $B$  only depends on the states of processes  $A$  and  $B$  while the transition rate of process  $A$  only depends on its own state. When we learn a CTBN from data, we can therefore use its graph as a qualitative summary of the interconnections of the alarms. An edge from  $A$  to  $B$  in this graph means that a change in the state of  $A$  may change the transition rates of  $B$  and therefore cascades are expected to occur along the directed edges of the graph.

Another type of graphical representation is also useful: Figure 1b shows the associated factored state space graph  $\mathcal{G}_s$ . In this graph, each node represents a state (in contrast, in  $\mathcal{G}$  each node represents a process/alarm) and edges represent the possible transitions between states. We will say that  $\mathcal{G}$  is the graph of the CTBN and refer to  $\mathcal{G}_s$  as the state space graph.

As we will see in the numerical experiments, CTBNs are capable of producing “cascading” behavior. However, they also model the non-cascading behavior: This is important for our application because we would also like to use the information contained outside periods with cascades. The CTBN framework has the following advantages that are critical to our application: 1) It exploits the factorization of the multivariate alarm process to make it feasible to learn a CTBN model from data. 2) It takes into account the duration of an event as well as its occurrence. Moreover, it uses both the cascading and non-cascading data. 3) It has a graphical representation which is easy to interpret, thus facilitating communication.

### 4.3 Reward function

A *reward function* is a function that maps the states of one or more processes onto a real number. We use a reward function to compute the discounted, expected number of transitions (that is, alarms changing their states) when starting the process in some initial state,  $x$ . A reward function consists of two quantities,

- $\mathcal{R}(x) : S \rightarrow \mathbb{R}$ , the *instantaneous reward* of state  $X = x$ , and
  - $\mathcal{C}(x, x') : S \times S \rightarrow \mathbb{R}$ , the *lump sum reward* when  $X$  transitions from state  $x$  to state  $x'$ .
- We use the lump sum reward which is an indicator of transitions,

$$\mathcal{C}(x, x') = 1, \quad \text{for all } x, x' \in S, \quad (4)$$

and we let the instantaneous reward be zero. We will use the *infinite-horizon expected discounted reward* [17],

$$V_\alpha(x) = \mathbb{E}_x \left[ \sum_{i=0}^{\infty} e^{-\alpha t_{i+1}} \mathcal{C}(X(t_i), X(t_{i+1})) + \int_{t_i}^{t_{i+1}} e^{-\alpha t} \mathcal{R}(X(t)) dt \right] : t_i < t_{i+1} \quad (5)$$

where  $\alpha > 0$  is referred to as the *discounting factor*,  $\mathbb{E}_x(\cdot)$  is the expectation when conditioning on  $X(0) = x$ , and the  $t_i$ 's are the transition times. We use  $\mathcal{R}(x) = 0$  for all  $x$  and therefore

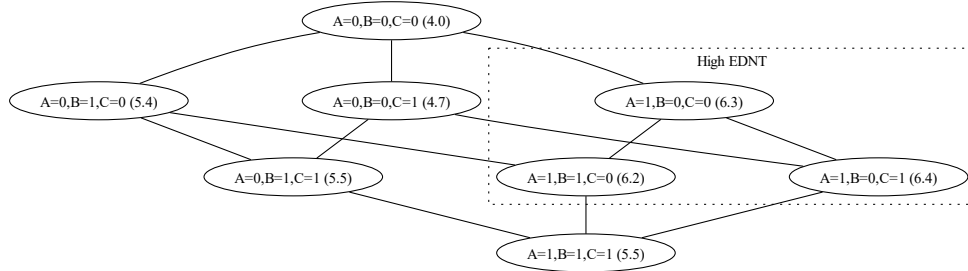
$$V_\alpha(x) = \mathbb{E}_x \left[ \sum_{i=0}^{\infty} e^{-\alpha t_{i+1}} \mathcal{C}(X(t_i), X(t_{i+1})) \right] = \mathbb{E} \left[ \sum_{i=0}^{\infty} e^{-\alpha t_{i+1}} \right] : t_i < t_{i+1}. \quad (6)$$

This simply counts the number of transitions including a discounting factor. Clearly, other reward functions can be chosen to analyze other or more specialized types of behavior. If, for instance, we are only interested in a subset of transitions we can modify the lump sum reward accordingly. A value of the parameter  $\alpha$  can be chosen using, e.g., prior information on the length of typical cascades. This concludes the introduction of the modeling framework and the following sections describe the contributions of this paper.

## 5 Sentry state

Given a CTBN  $\mathcal{N} = \langle P_0, \mathbf{X}, \mathcal{G}, \mathbf{Q}_X \rangle$ , we are interested in understanding its cascading behavior. We are in particular interested in identifying what we will call *sentry states*. A sentry state is a state which may trigger a *ripple effect*, that is, a sequence of fast transitions.

► **Example 3.** An example of a ripple effect can be found in the CTBN in Figure 1a which consists of three processes  $\mathbf{X}_A$ ,  $\mathbf{X}_B$  and  $\mathbf{X}_C$ , forming a chain  $A \rightarrow B \rightarrow C$ , and each taking values in  $\{0, 1\}$ ,  $S_A = S_B = S_C = \{0, 1\}$ . The trajectory in Figure 1c shows that every time  $A$  transitions,  $B$  and  $C$  quickly transition as well. In other words, when  $A$  changes its state, a ripple effect occurs such that  $B$  and  $C$  also change states to match the state of their parent. The starting point of these cascades of events is a sentry state as defined in this section.



■ **Figure 3** Visualization of Example 3: The state space graph (same graph as in Figure 1b, but presented slightly differently) is annotated with EDNTs for each node (in parentheses) and a subset of nodes with relatively high EDNT values is highlighted. Sentry states are high EDNT states to which transitions from low EDNT states are possible and such a transition increases the risk of an imminent cascade. Some states with high EDNT tend to occur in the middle of a cascade which motivates using the REDNT metric. In this example, transitioning from  $(A = 0, B = 0, C = 0)$  to  $(A = 1, B = 0, C = 0)$  may trigger an alarm in  $B$  which in turn may trigger an alarm in  $C$ , resulting in a cascade (note that only transitions along the edges in the state space graph are possible).

It is important to stress that we are interested in states that *start* a cascade of events. Intuitively, this means that we are assuming the existence of at least one state in the state space graph which is directly connected to the sentry state and which has a much smaller expected number of transitions than the sentry state itself. We can observe this in the example in Figure 1c: Before starting the sequence of transitions of processes  $\mathbf{X}_A$ ,  $\mathbf{X}_B$ , and  $\mathbf{X}_C$  from 0 to 1, the CTBN remained for a long time in state  $\{A = 0, B = 0, C = 0\} \in S$ . Similarly, before changing the state of all processes  $A$ ,  $B$  and  $C$  from 1 to 0, the CTBN remained for a long time in state  $\{A = 1, B = 1, C = 1\} \in S$ .

### 5.1 Sentry state identification

In order to identify a sentry state, we need to take a further step from the heuristic definition of a sentry state we have just given and formalize the concept. For this purpose, we first compute the expected (discounted) number of transitions for each state of the CTBN. This can be achieved by using the lump sum reward in (4) to obtain the *Expected Discounted Number of Transitions* (EDNT) of each state  $x \in S$ ,

$$EDNT_\alpha(x) = E \left[ \sum_{i=0}^{\infty} e^{-\alpha t_{i+1}} C(X(t_i), X(t_{i+1})) \right] : C(x, x') = 1. \quad (7)$$

There is no guarantee that a state with high EDNT is often the starting point of a cascade. States that tend to occur in the middle of a cascade may easily have a high EDNT if the cascade tends to continue after reaching that state. We are interested in early detection of cascades and the solution we propose is to take into account the number of transitions in the neighborhood  $\text{Ne}_{\mathcal{G}_s}(x)$  of the state  $x \in S$ . For this purpose, we define a new quantity called *Relative Expected Discounted Number of Transitions* (REDNT),

$$REDNT_\alpha(x) = \max_{x' \in \text{Ne}_{\mathcal{G}_s}(x)} \frac{EDNT_\alpha(x)}{EDNT_\alpha(x')} \quad (8)$$

where  $\alpha$  in (7) and (8) is the discounting factor as in (5), and the neighborhood in (8) refers to the undirected state space graph (see Figure 1b). The central idea is that a large ratio between two adjacent states implies that transition from one to the other leads to a significant change in the expected discounted number of transitions. We will use REDNT to identify potential sentry states (states with high values of REDNT are likely sentry states).

One could propose other ways to aggregate EDNT across different states. We focus on REDNT as defined above in the interest of brevity. We let  $\bar{s}$  denote the number of alarms that are *on* in the state  $s = (s_1, s_2, \dots, s_L)$ ,  $\bar{s} = \sum_{i=1}^L s_i$ . In the alarm data application, we are mostly interested in sentry states such that  $\bar{s}$  is fairly small. States with large  $\bar{s}$  may also have large REDNT values; however, these are states that occur when a cascade is already happening. As we want early detection, we should focus on sentry states such that  $\bar{s}$  is small.

## 5.2 Monte Carlo algorithm

We are now left with the problem of estimating the EDNT of each state from which we can compute the REDNT. We propose a Monte Carlo approach based on Algorithm 1 from [32]. This sampling algorithm starts from an initial state  $X(0)$  and generates a single trajectory  $\sigma$  ending at time  $t_{end}$ . After the *initialization* phase the algorithm enters into a loop. At each iteration, the algorithm samples a time to transition for each of the variables, identifies the next *transitioning variable*, generates the *next state*, and *resets the time to transition* for the transitioned variable and all its children. We combine Algorithm 1 with (7) to compute

$$\widehat{EDNT}_\alpha(x) = \frac{1}{|\sigma|} \sum_{\sigma \in \Sigma} \sum_{i=0}^{|\sigma|} e^{-\alpha t_i} C(x(t_i), x(t_{i+1})) \quad (9)$$

where  $|\sigma|$  is the number of trajectories generated by Algorithm 1 and  $|\sigma|$  represents the number of events in the trajectory  $\sigma$ .

In order to compute  $\widehat{EDNT}_\alpha$ , we need to set the values of the following hyperparameters:

1.  $\alpha$ , the discounting factor;
2.  $t_{end}$ , the ending time for each trajectory;
3.  $|\sigma|$ , the number of trajectories to be generated.

Choosing the discounting factor  $\alpha$  and the ending time  $t_{end}$ , we decide the importance of the *distant future* and appropriate values depend on the application. On the other hand,  $|\sigma|$  controls the trade-off between the quality of the approximation and its computational cost: We can choose its value using a stopping-rule approach based on variance as proposed in [8].

■ **Algorithm 1** Forward Sampling for CTBN.

---

```

procedure CTBN-SAMPLE( $X(0), t_{end}$ )
   $t \leftarrow 0, \sigma \leftarrow \{\langle 0, X(0) \rangle\}$  ▷ Initialization
  loop
    for all  $\mathbf{X}_i \in \mathbf{X}$  s.t.  $Time(\mathbf{X}_i)$  is undefined do ▷ Time to transition sampling
       $\Delta t \leftarrow$  draw a sample from an exponential with rate  $q_{\mathbf{X}_i(t)|Pa(\mathbf{X}_i(t))}$ 
       $Time(\mathbf{X}_i) \leftarrow t + \Delta t$ 
    end for
     $j = \arg \min_{\mathbf{X}_i \in \mathbf{X}} [Time(\mathbf{X}_i)]$  ▷ Transitioning variable
    if  $Time(\mathbf{X}_j) > t_{end}$  then
      return  $Tr$ 
    end if
     $x_j \leftarrow$  draw a sample from a multinomial with  $\theta_{\mathbf{X}_j(t)|Pa(\mathbf{X}_j(t))}$  ▷ Next state
     $t \leftarrow Time(\mathbf{X}_j)$ 
    Add  $\langle t, X(t) \rangle$  to  $\sigma$ 
    Undefine  $Time(\mathbf{X}_j)$  and  $Time(\mathbf{X}_i) \forall \mathbf{X}_i \in Pa(\mathbf{X}_j)$  ▷ Reset the time to transition
  end loop
return  $\sigma$ 
end procedure

```

---

## 6 Graphical information

This paper proposes the CTBN as a modeling tool for systems with cascades. However, CTBNs have other useful properties: The interplay between the graph and the probabilistic model facilitates both communication with subject matter experts and easy computation of various statistics that summarize the learned system.

We define the *parent set* of  $A$ ,  $Pa(A) = (\bigcup_{\mathbf{X}_i \in A} Pa(\mathbf{X}_i)) \setminus A$ . Note that  $Pa(A) \cap \{A\} = \emptyset$  for all  $A$ . We define the *closure* of  $A$ ,  $Cl(A) = Pa(A) \cup \{A\}$ . A *walk* is a sequence of adjacent edges and a *path* is a sequence of adjacent edges such that no node is repeated. We say that  $\mathbf{X}_i$  is an *ancestor* of  $\mathbf{X}_j$  if there exists a *directed* path from  $\mathbf{X}_i$  to  $\mathbf{X}_j$ ,  $\mathbf{X}_i \rightarrow \dots \rightarrow \mathbf{X}_j$ , such that all the edges point toward  $\mathbf{X}_j$ . We define  $An(A)$  to be the set of nodes that are in  $A$  or are ancestors of a node in  $A$ . Therefore,  $A \subseteq An(A)$  for all  $A$ . We say that the node set  $A$  is *ancestral* if  $An(A) = A$ , that is, if  $A$  contains all ancestors of every node in  $A$ . In Figure 4a, the set  $\{A, B\}$  is ancestral while the set  $\{B, C\}$  is not ancestral. We let  $\mathcal{G}_A$  denote the graph with node set  $A$  such that for  $\mathbf{X}_i, \mathbf{X}_j \in A$ , the edge  $\mathbf{X}_i \rightarrow \mathbf{X}_j$  is in  $\mathcal{G}_A$  if  $\mathbf{X}_i \rightarrow \mathbf{X}_j$  is in  $\mathcal{G}$ . We construct the *moral graph*,  $\mathcal{G}^m$ , of  $\mathcal{G}$  by replacing all edges with *undirected* edges,  $-$ , and adding an undirected edge between two nodes if there exists a node of which they are both parents. In an undirected graph and for disjoint  $A, B$ , and  $C$ , we say that  $A$  and  $B$  are *separated* by  $C$  if every path between  $A$  and  $B$  is intersected by  $C$ .

### 6.1 Decomposition properties

The graph of a CTBN has a clear interpretation, as the transition rate of  $\mathbf{X}_i$  only depends on the current value of  $Cl(\mathbf{X}_i)$ . The following results provide another interpretation of the graph in terms of conditional independence. We let  $\bar{\mathbf{X}}_A(t)$  denote the process  $A$  until time point  $t$ ,  $\bar{\mathbf{X}}_A(t) = \{X_i(s) : i \in A, s \leq t\}$ . The next result follows from Proposition 5 in [14].

► **Proposition 1.** *Let  $\mathcal{G} = (\mathbf{X}, E)$  be the graph of a CTBN, and let  $A, B, C \subseteq V$  be disjoint. If  $A$  and  $B$  are separated by  $C$  in  $(G_{An(A \cup B \cup C)})^m$ , then  $\bar{\mathbf{X}}_A(t) \perp\!\!\!\perp \bar{\mathbf{X}}_B(t) \mid \bar{\mathbf{X}}_C(t)$  for all  $t$ .*

The above result allows us to decompose the learned system into components  $A$  and  $B$  that operate independently conditionally on  $C$ . That is, all dependence between  $A$  and  $B$  is “explained” by  $C$ . However, the graph may not be very informative to experts if it contains too many nodes. We address this issue now and further results are in Appendix A.

## 6.2 Hierarchical analysis

The graph of a CTBN may be too large to be easily examined visually. If the number of components in a system is large experts mostly reason about groups of components. For instance, in the alarm data, the system is known to comprise different subsystems, which form a natural partition of the components  $\mathbf{X}$ . We show that Proposition 1 still applies in an aggregated version of the graph. We define a *graph partition* to formalize this.

► **Definition 1** (Graph partition). *Let  $\mathcal{G} = (\mathbf{X}, E)$  be a directed graph and let  $D = \{D_1, \dots, D_m\}$  be a partition of  $\mathbf{X}$ . The graph partition of  $\mathcal{G}$  induced by  $D$ ,  $\mathcal{D}$ , is the directed graph  $(D, E_D)$  with node set  $D$  such that  $D_k \rightarrow D_l$ ,  $k \neq l$ , is in  $E_D$  if and only if there exists  $\mathbf{X}_i \in D_k$  and  $\mathbf{X}_j \in D_l$  such that  $\mathbf{X}_i \rightarrow \mathbf{X}_j$  in  $\mathcal{G}$ .*

In a graph partition, each node,  $D_k \in D$ , corresponds to a subset of the node set in the original graph. Underlined symbols, e.g.,  $\underline{A}$ , represent subsets of  $D = \{D_1, \dots, D_m\}$ . When  $\underline{A} \subseteq D$ , we let  $A$  denote the corresponding nodes in the original graph,  $A = \bigcup_{D_i \in \underline{A}} D_i$ . The following is an extension of the classical separation property to a graph partition and it means that separation in a graph partition can be translated into separation in the underlying graph. This in turn implies a conditional independence in the CTBN.

► **Proposition 2.** *Let  $D$  be a partition of  $V$  and let  $\mathcal{D}$  be the graph partition induced by  $D$ . Let  $\underline{A}, \underline{B}, \underline{C} \subseteq D$  be disjoint. If  $\underline{A}$  and  $\underline{B}$  are separated by  $\underline{C}$  in  $(\mathcal{D}_{An(\underline{A} \cup \underline{B} \cup \underline{C})})^m$ , then  $A$  and  $B$  are separated by  $C$  in  $(\mathcal{G}_{An(A \cup B \cup C)})^m$ .*

► **Example 4** (Simplified ESS alarm network). *We revisit the example in Figure 2 and denote the graph on the left by  $\mathcal{G}$ . This graph represents a CTBN as introduced in Section 4. System 1, System 2, System 3, and System 4 constitute a partition,  $D$ , of the node set of  $\mathcal{G}$  and we let  $\mathcal{D}$  denote the corresponding graph partition (Figure 2 (right)). If we let  $\underline{A} = \{\text{System 1}\}$ ,  $\underline{B} = \{\text{System 3}\}$ , and  $\underline{C} = \{\text{System 2, System 4}\}$ , then  $\underline{A}$  and  $\underline{B}$  are separated by  $\underline{C}$  in  $(\mathcal{D}_{An(\underline{A} \cup \underline{B} \cup \underline{C})})^m$  (in this case  $An(\underline{A} \cup \underline{B} \cup \underline{C}) = \{\text{System 1, System 2, System 3, System 4}\}$  and  $(\mathcal{D}_{An(\underline{A} \cup \underline{B} \cup \underline{C})})^m$  is simply the undirected version of  $\mathcal{D}$  as every node only has a single parent). The set  $\underline{A}$  corresponds to processes  $A = \{P1, T1\}$ ,  $\underline{B}$  corresponds to processes  $\{P3, T3\}$ , and  $\underline{C}$  corresponds to  $\{P2, T2, S1, S2, S3, S4\}$ . Proposition 2 gives that  $A$  and  $B$  are separated by  $C$  in  $(\mathcal{G}_{An(A \cup B \cup C)})^m$ . It follows from Proposition 1 that the processes in  $A$  and the processes in  $B$  are independent when conditioning on the processes in  $C$ . This means that the state of System 1 is irrelevant when reasoning about the state of System 3 if we already account for Systems 2 and 4. Using this procedure, conditional independences can be found using both the original graph and a graph partition. Furthermore, in both  $\mathcal{G}$  and  $\mathcal{D}$ , the edges have a simple interpretation: The transition rates of the processes corresponding to a node only depend on the processes corresponding to the parent nodes.*

### 6.2.1 Condensation

The graph  $\mathcal{G}$  of a CTBN may be cyclic. A possible simplification is to collapse each cyclic component into a node to form the *condensation* of  $\mathcal{G}$  which is a *directed acyclic graph*. We say that  $A \subseteq V$  is a *strongly connected component* if for every  $\mathbf{X}_i \in A$  and every  $\mathbf{X}_j \in A$

there exists a directed path from  $\mathbf{X}_i$  to  $\mathbf{X}_j$ . The strongly connected components constitute a partition of  $V$  and the *condensation* is the graph partition they induce. The condensation has some properties that do not hold for general graph partitions (see Appendix A).

► **Definition 2 (Condensation).** *Let  $\mathcal{G}$  be a directed graph and let  $D = \{D_1, \dots, D_m\}$  be its strongly connected components. We say that the graph partition of  $\mathcal{G}$  induced by  $D$  is the condensation of  $\mathcal{G}$ .*

## 7 Numerical experiments and examples

We now study the performance of the proposed approach. We generate synthetic data from CTBNs such that the sentry states are known. Data is generated from different CTBNs (additional experiments are in Appendix C). In all of them,

- each process,  $\mathbf{X}_j \in \mathbf{X}$ , has a binary state space.
- the CTBN consists of *slow processes* and *fast processes*.
- each process,  $\mathbf{X}_j \in \mathbf{X}$ , replicates the state of its parent processes,  $\text{Pa}(\mathbf{X}_j)$ .
- if a process,  $\mathbf{X}_j \in \mathbf{X}$ , has more than one parent, it stays in state 0 with high probability if at least one of its parents is in state 0.

### Experiment 1

The first synthetic experiment is based on a CTBN model whose graph  $\mathcal{G}$  is a chain consisting of three nodes  $A$ ,  $B$ , and  $C$  (Figure 4a). The corresponding CIMs for the processes  $A$ ,  $B$ , and  $C$  are shown in Table A1 in Appendix C. This CTBN describes a structured stochastic process such that the root process,  $A$ , changes slowly from the state no-alarm (0) to the state alarm (1) and vice versa. This can be seen from the CIM corresponding to process  $A$ . The CIMs associated with processes  $B$  and  $C$  make these two processes replicate the state of their parent process and this happens at a faster rate. Therefore, starting from  $(0, 0, 0)$ , if process  $A$  changes its state, process  $B$  quickly changes its state to match that of its parent  $A$ . The same holds true for the process  $C$ . For this reason, we expect  $\{A = 1, B = 0, C = 0\}$  to be a sentry state because as soon as the process  $A$  transitions from state 0 to state 1, a fast sequence of transitions (a cascade of events) makes the processes  $B$  and  $C$  transition from state 0 to state 1. This behavior is shown in Figure 4b. Estimates of the REDNT quantity are shown in Table A2 and they confirm that  $\{A = 1, B = 0, C = 0\}$  is a sentry state.

### Experiment 2

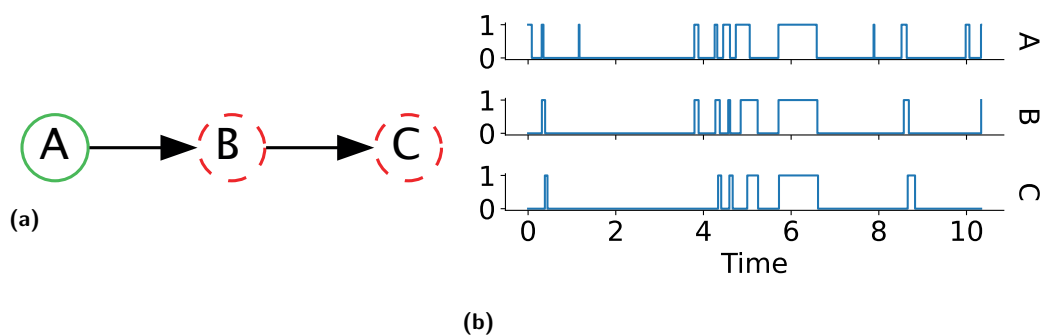
The second synthetic experiment is based on the CTBN shown in Figure 5a which consists of a slow cycle  $(A, B, C)$  and a fast chain  $(D, E, F)$ . In this CTBN, the sentry state is expected to be  $\{A = 0, B = 0, C = 1, D = 0, E = 0, F = 0\}$ . Figure 5b shows that this state triggers a fast sequence of alarms in the chain  $(D, E, F)$  and a slow sequence of alarms in the cycle  $(A, B, C)$ . Estimates of the REDNT quantity are shown in Table A3 and they confirm that  $\{A = 0, B = 0, C = 1, D = 0, E = 0, F = 0\}$  is a sentry state.

### Comparison

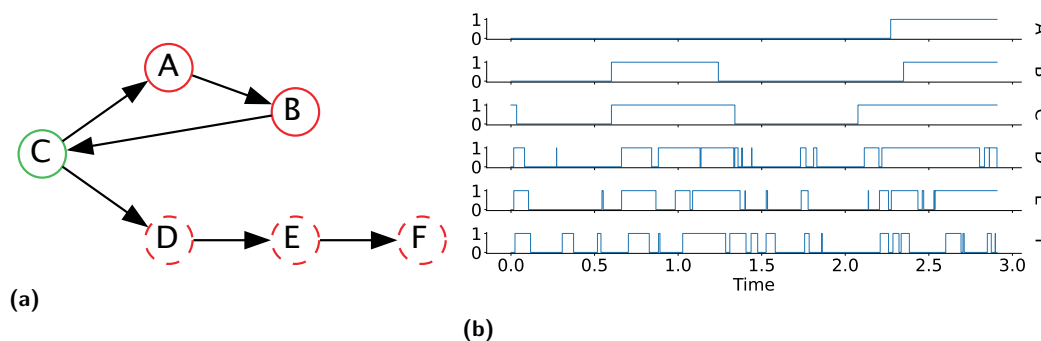
We compare the REDNT method to the naive approach proposed in Appendix B. In synthetic data it is easier to identify the two parameters of the naive approach. Each synthetic experiment has only two transition rates and we can let the parameter  $\lambda_{ft}$ <sup>2</sup> be

<sup>2</sup> Threshold between a slow and a fast transition (Appendix B).





■ **Figure 4** (a) depicts the graph  $\mathcal{G}$  of a CTBN. Its CIMs are in Table A1 in the appendix. Slow processes are represented by solid line nodes, while fast processes are represented by dashed line nodes. Colors describe the most likely sentry state,  $s = (s_1, s_2, s_3)$ , in this system: If a node is green, the corresponding alarm is 1 (*on*) in  $s$ . If a node is red, the corresponding alarm is 0 (*off*) in  $s$ . (b) shows an example trajectory from the CTBN represented in Figure 4a. Each function in the plot represents the evolution of one of the three binary processes,  $A$ ,  $B$ , and  $C$ .



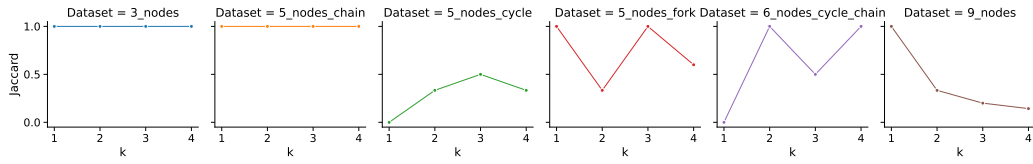
■ **Figure 5** (a) Graph  $\mathcal{G}$  of a CTBN model consisting of six processes. The graph  $\mathcal{G}$  contains the cycle  $(A, B, C)$  as well as the chain  $(D, E, F)$ . See the caption of Figure 4 for an explanation of the node colors.

the median elapsed time between two consecutive events when combining events of all types. The parameter  $\lambda_{mcl}$ <sup>3</sup> can be determined based on the structure of the network. For instance, in the example in Figure 4a we expect a cascade to have at least two transitions,

$$\{A = 1, B = 0, C = 0\} \rightarrow \{A = 1, B = 1, C = 0\} \rightarrow \{A = 1, B = 1, C = 1\}.$$

To identify sentry states using the naive approach we should simply identify the cascades of events and compute the fraction of times that observing a specific state coincides with the start of a cascade. As already mentioned in Section 5, we are interested in sentry states with a low number of active alarms. For this reason, we consider only states such that the number of active alarms is less than or equal to the size of the largest parent set in the true graph. The naive approach and the REDNT both produce a list where states are ordered from the most likely sentry state to the least likely. We compare the two approaches with the *Jaccard similarity* [27] using the  $K$  most likely sentry states. We tested our approach

<sup>3</sup> It determines the minimum number of fast consecutive events to be considered a cascade See Appendix B.



■ **Figure 6** This figure reports the Jaccard similarity@k between the REDNT and the naive approach. The x-axis represents the number of states taken into account from the ordered lists generated by the two methods. The structures of the networks used for the experiments are depicted in Figures 4a, A1, A2, A3, 5a, and A4.

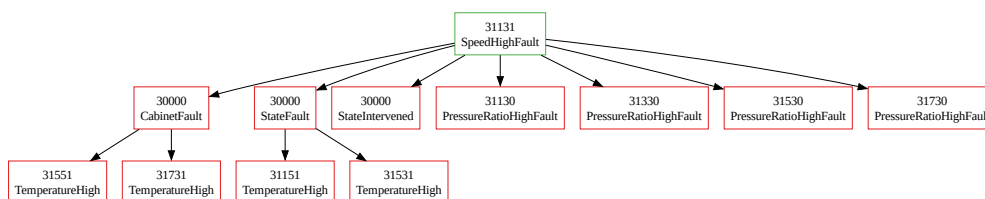
on 6 different structures with different numbers of nodes. Results are reported in Figure 6. In every experiment, the two methods share at least one state in their top-two lists. It is important to emphasize that the parameters of the naive method have been set knowing the length of cascades. Conversely, the REDNT method does not require this knowledge in order to identify sentry states.

### ESS data set

The last experiment is performed on a real data set provided by the European Spallation Source ERIC (ESS) as described in Section 2. The data set consists of observations of 138 alarm processes from January 2020 to March 2023. No structure was provided, thus we use the score-based structure learning algorithm presented in [32]. We chose not to use the constraint-based algorithm [10] because, in the case of binary variables, it has been shown to be outperformed by the score-based algorithm. The score-based algorithm penalizes the size of the parent sets, leading to sparsity in the graphical structure. For this data, the learned graph is composed of disconnected components. We only present the results of applying the REDNT method for one of them. The most likely sentry state has the alarm *SpeedHighFault* set to *on* and everything else set to *off* (see Figure 7). We observe that the connected component which contains *SpeedHighFault* is a rooted directed tree, and that *SpeedHighFault* is the root. This means that an alarm in *SpeedHighFault* propagates along the directed paths in the tree. A CTBN assumes that at most one event occurs at any point in time. This is reasonable in this application because of the high sampling rate.

The four *PressureRatioHighFault* alarms in Figure 7 could be verified as consequences of the root cause *SpeedHighFault*, both from documentation and by an experienced operator. On the other hand, the alarms *StateIntervened*, *CabinetFault*, and *StateFault* were not evident from the documentation and were not expected to be related to the root alarm *SpeedHighFault*. If these connections are real, this information is relevant to operators as they may look for reasons for this connection and enhance their process understanding. Moreover, the identified root alarm *SpeedHighFault* can be given a high priority to ensure that the operators will be made aware of a potential cascade starting from this alarm.

The CTBN was learned with no structural input, but using a prior distribution one can provide engineering knowledge to a Bayesian learning method. We believe that the graphical output can easily be shown to and interpreted by engineers; however, user studies are needed to validate this. Results like Propositions 1 and 2 allow a formal interpretation and help us find those components of the system that function independently when accounting for other parts of the machine. Moreover, the sentry states that are identified can be presented to the operators. In the example shown in figure 7 the sentry state is covered by an already existing alarm, but if a more complex sentry state would have been identified then an additional



■ **Figure 7** Graph  $\mathcal{G}$  of the CTBN learned from the ESS data set. Figure 4 explains the node colors.

alarm would be needed. Using domain expertise, one can assign appropriate instructions to sentry states and these can be presented to operators when the machine reaches a fault state. When we have learned a CTBN from data, we may also compute the risk of reaching each of the sentry states from the current state. This can be provided to operators in real time to facilitate early mitigation. This should be studied in detail in future work.

► **Example 5** (Simplified ESS alarm network). *We now return to our running example. This is an example system and it does not correspond to the learned network. Imagine that we start from the state with all alarms off. Assume we have learned from data, using the CTBN framework, that the P1 alarm is very likely to go on if both S4 and T1 alarms are on. Furthermore, assume that P2 is very likely to go on when P1 is on and that P3 is very likely to go on when P2 is on. In this case, the state such that  $S_4 = 1$ ,  $T_1 = 1$ , and such that every other node is zero is likely to be a sentry state. This knowledge can be useful in two ways. First, this can be presented to experts so that they can map common cascades, and their sentry state starting points, to underlying causes using domain knowledge. Second, during operation a warning can be issued when reaching a sentry state, along with the recommended course of action. It is also possible to compute, given the current state, the risk of reaching each of the sentry states within some time interval to facilitate an earlier warning if the expected time from reaching the sentry state to the actual cascade does not suffice for mitigation.*

## 8 Discussion

In this paper we defined the concept of sentry states in CTBNs and we presented a naive approach and a heuristic (REDNT) for identifying such sentry states. The synthetic experiments showed that REDNT can identify the configuration of the network from which a fast sequence of events starts. A key limitation is the fact that the REDNT heuristic is computed for each state and the number of states is exponential in the number of nodes. However, the simplicity of its implementation and the effectiveness showed in the synthetic experiments make the REDNT heuristic attractive. Moreover, only states with few active alarms may be of interest and this reduces the computational cost. The proposed heuristic assigns a score to each state in the state space of a CTBN; a possible extension of this work is the identification of the contribution of each process to the REDNT.

This paper laid the theoretical groundwork for the implementation of online early warning systems based on the identification of sentry states. In practical implementations, a list of sentry states can be provided to domain experts for them to formulate appropriate actions in order to mitigate alarm cascades. This is left for future work. Moreover, the graph representing a learned CTBN indicates how the behavior of each alarm process depends on the states of the other alarm processes. As illustrated in this paper, this graph also represents

conditional independences in the system. In future work, we hope to demonstrate that the intended end users, engineers and system operators, also find this graphical tool useful.

---

## References

- 1 E. Acerbi, E. Viganò, M. Poidinger, A. Mortellaro, T. Zelante, and F. Stella. Continuous Time Bayesian Networks Identify Prdm1 as a Negative Regulator of TH17 Cell Differentiation in Humans. *Scientific Reports*, 6:23128, 2016.
- 2 Haniyeh Seyed Alinezhad, Mohammad Hossein Roohi, and Tongwen Chen. A review of alarm root cause analysis in process industries: Common methods, recent research status and challenges. *Chemical Engineering Research and Design*, 2022.
- 3 Samaneh Aminikhanghahi and Diane J Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.
- 4 David F Anderson and Thomas G Kurtz. Continuous time markov chain models for chemical reaction networks. In *Design and analysis of biomolecular circuits: engineering approaches to systems and synthetic biology*, pages 3–42. Springer, 2011.
- 5 Rajeevan Arunthavanathan, Faisal Khan, Salim Ahmed, and Syed Imtiaz. Autonomous fault diagnosis and root cause analysis for the processing system using one-class svm and nn permutation algorithm. *Industrial & Engineering Chemistry Research*, 61(3):1408–1422, 2022.
- 6 F Ball, RK Milne, and GF Yeo. Multivariate semi-markov analysis of burst properties of multiconductance single ion channels. *Journal of Applied Probability*, 39(1):179–196, 2002.
- 7 Juraj Bergman, Dominik Schrempf, Carolin Kosiol, and Claus Vogl. Inference in population genetics using forward and backward, discrete and continuous time processes. *Journal of Theoretical Biology*, 439:166–180, 2018.
- 8 Martin Bicher, Matthias Wastian, Dominik Brunmeir, and Niki Popper. Review on monte carlo simulation stopping rules: How many samples are really enough? *Simul. Notes Eur.*, 32(1):1–8, 2022.
- 9 Jerome V Braun and Hans-Georg Muller. Statistical methods for DNA sequence segmentation. *Statistical Science*, pages 142–162, 1998.
- 10 Alessandro Bregoli, Marco Scutari, and Fabio Stella. A constraint-based algorithm for the structural learning of continuous-time bayesian networks. *International Journal of Approximate Reasoning*, 138:105–122, 2021. doi:10.1016/j.ijar.2021.08.005.
- 11 Giulia Cencetti, Federico Battiston, Bruno Lepri, and Márton Karsai. Temporal properties of higher-order interactions in social networks. *Scientific reports*, 11(1):7028, 2021.
- 12 Deepayan Chakrabarti, Yang Wang, Chenxi Wang, Jurij Leskovec, and Christos Faloutsos. Epidemic thresholds in real networks. *ACM Transactions on Information and System Security (TISSEC)*, 10(4):1–26, 2008.
- 13 Daryl J Daley, David Vere-Jones, et al. *An introduction to the theory of point processes: volume I: elementary theory and methods*. Springer, 2003.
- 14 Vanessa Didelez. Graphical models for composable finite markov processes. *Scandinavian Journal of Statistics*, 34(1):169–185, 2007.
- 15 EPICS. The experimental physics and industrial control system. Last accessed 2023-04-25. URL: <https://epics-controls.org/about-epics/>.
- 16 ESS. European spallation source. Last accessed 2023-04-25. URL: <https://europenspallationsource.se/about>.
- 17 Xianping Guo and Onésimo Hernández-Lerma. Continuous-time markov decision processes. In *Continuous-Time Markov Decision Processes*, pages 9–18. Springer, 2009.
- 18 B.R. Hollifield and E. Habibi. *Alarm Management: A Comprehensive Guide : Practical and Proven Methods to Optimize the Performance of Alarm Management Systems*. International Society of Automation, 2011. URL: <https://books.google.se/books?id=UuSMswEACAAJ>.

- 19 Jinqiu Hu, Laibin Zhang, Zhansheng Cai, Yu Wang, and Anqi Wang. Fault propagation behavior study and root cause reasoning with dynamic Bayesian network based framework. *Process Safety and Environmental Protection*, 97:25–36, 2015.
- 20 Srinivasan M Iyer, Marvin K Nakayama, and Alexandros V Gerbessiotis. A markovian dependability model with cascading failures. *IEEE Transactions on Computers*, 58(9):1238–1249, 2009.
- 21 David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.
- 22 Theodoros Lappas, Evimaria Terzi, Dimitrios Gunopulos, and Heikki Mannila. Finding effectors in social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1059–1068, 2010.
- 23 Hyunju Lee and Ji Hwan Cha. Point process approach to modeling and analysis of general cascading failure models. *Journal of Applied Probability*, 53(1):174–186, 2016.
- 24 Morten Lind. An overview of multilevel flow modeling. *International Electronic Journal of Nuclear Safety and Simulation*, 4, 2013.
- 25 Manxia Liu, Fabio Stella, Arjen Hommersom, Peter J. F. Lucas, Lonneke Boer, and Erik Bischoff. A comparison between discrete and continuous time bayesian networks in learning from clinical time series data with irregularity. *Artif. Intell. Medicine*, 95:104–117, 2019. doi:10.1016/j.artmed.2018.10.002.
- 26 Jyotiprasad Medhi. *Stochastic models in queueing theory*. Elsevier, 2002.
- 27 Allan H Murphy. The finley affair: A signal event in the history of forecast verification. *Weather and forecasting*, 11(1):3–20, 1996.
- 28 Upama Nakarmi and Mahshid Rahnamay-Naeini. A markov chain approach for cascade size analysis in power grids based on community structures in interaction graphs. In *2020 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, pages 1–6. IEEE, 2020.
- 29 Upama Nakarmi, Mahshid Rahnamay Naeini, Md Jakir Hossain, and Md Abul Hasnat. Interaction graphs for cascading failure analysis in power grids: A survey. *Energies*, 13(9):2219, 2020.
- 30 Praneeth Netrapalli and Sujay Sanghavi. Learning the graph of epidemic cascades. *ACM SIGMETRICS Performance Evaluation Review*, 40(1):211–222, 2012.
- 31 Uri Nodelman, Christian R Shelton, and Daphne Koller. Continuous time Bayesian networks. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI2002)*, 2002.
- 32 Uri D Nodelman. *Continuous time Bayesian networks*. PhD thesis, Stanford University, 2007.
- 33 Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.
- 34 Mahshid Rahnamay-Naeini, Zhuoyao Wang, Nasir Ghani, Andrea Mammoli, and Majeed M Hayat. Stochastic analysis of cascading-failure dynamics in power grids. *IEEE Transactions on Power Systems*, 29(4):1767–1779, 2014.
- 35 Marcello Rambaldi, Vladimir Filimonov, and Fabrizio Lillo. Detection of intensity bursts using hawkes processes: An application to high-frequency financial data. *Physical Review E*, 97(3):032318, 2018.
- 36 Jaxk Reeves, Jien Chen, Xiaolan L Wang, Robert Lund, and Qi Qi Lu. A review and comparison of changepoint detection techniques for climate data. *Journal of applied meteorology and climatology*, 46(6):900–915, 2007.
- 37 Yaacov Ritov, A Raz, and H Bergman. Detection of onset of neuronal activity by allowing for heterogeneity in the change points. *Journal of neuroscience methods*, 122(1):25–42, 2002.
- 38 Vicent Rodrigo, Moncef Chioua, Tore Hagglund, and Martin Hollender. Causal analysis for alarm flood reduction. *IFAC-PapersOnLine*, 49(7):723–728, 2016.

- 39 David Rybach, Christian Gollan, Ralf Schluter, and Hermann Ney. Audio segmentation for speech recognition using segment features. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4197–4200. IEEE, 2009.
- 40 Tore Schweder. Composable markov processes. *Journal of applied probability*, 7(2):400–410, 1970.
- 41 Christian R Shelton and Gianfranco Ciardo. Tutorial on structured continuous-time Markov processes. *Journal of Artificial Intelligence Research*, 51:725–778, 2014.
- 42 Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.
- 43 Simone Villa and Fabio Stella. Learning continuous time bayesian networks in non-stationary domains. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5656–5660. International Joint Conferences on Artificial Intelligence Organization, July 2018. doi:10.24963/ijcai.2018/804.
- 44 Yiming Wan, Fan Yang, Ning Lv, Haipeng Xu, Hao Ye, Weichang Li, Peng Xu, Liming Song, and Adam K Usadi. Statistical root cause analysis of novel faults based on digraph models. *Chemical Engineering Research and Design*, 91(1):87–99, 2013.
- 45 Haoyun Wang, Liyan Xie, Yao Xie, Alex Cuzzo, and Simon Mak. Sequential change-point detection for mutually exciting point processes. *Technometrics*, pages 1–13, 2022.
- 46 Jeremy C Weiss and David Page. Forest-based point process for event prediction from electronic health records. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*, pages 547–562. Springer, 2013.
- 47 Fan Yang, Sirish L Shah, Deyun Xiao, and Tongwen Chen. Improved correlation analysis and visualization of industrial alarm data. *ISA transactions*, 51(4):499–506, 2012.

## A Graphical information

The following proposition follows from Proposition 4 in [14] and Theorem 2 in [40].

► **Proposition 3.** *If  $A$  is ancestral, then the subprocess  $\mathbf{X}_A = (\mathbf{X}_i)_{i \in A}$  is a CTBN with transition matrices  $Q_{\mathbf{X}_i | Pa(\mathbf{X}_i)}$  and graph  $\mathcal{G}_A$ .*

**Proof of Proposition 2.** Note that  $A$ ,  $B$ , and  $C$  must be disjoint. We consider a connecting path between  $A$  and  $B$  in  $(\mathcal{G}_{\text{An}(A \cup B \cup C)})^m$  which does not intersect  $C$ ,

$$\mathbf{X}_{i_0} - \mathbf{X}_{i_1} - \dots - \mathbf{X}_{i_m}.$$

Let  $f : \mathbf{X} \rightarrow D$  be the unique map such that if  $f(\mathbf{X}_i) = D_l$ , then  $\mathbf{X}_i \in D_l$ . We consider the walk

$$f(\mathbf{X}_{i_0}) - f(\mathbf{X}_{i_1}) - \dots - f(\mathbf{X}_{i_m}).$$

and argue that this walk, or a subwalk, is present in  $(\mathcal{D}_{\text{An}(\underline{A} \cup \underline{B} \cup \underline{C})})^m$  and is not intersected by  $\underline{C}$ . Every node on the original walk is in  $\text{An}(A \cup B \cup C)$  in  $\mathcal{G}$ , so every node on the above walk is in  $\text{An}(\underline{A} \cup \underline{B} \cup \underline{C})$  in  $\mathcal{D}$ . We remove nodes such that no adjacent nodes are equal (note that the result is a nontrivial walk). If an edge on the original walk corresponds to a directed edge in  $\mathcal{G}$ , then it is also in  $\mathcal{D}$ . Assume it does not correspond to a directed edge on the original walk. It then corresponds to a “moral” edge,  $\mathbf{X}_{i_j} \rightarrow \mathbf{X}_k \leftarrow \mathbf{X}_{i_{j+1}}$  in  $\mathcal{G}$ , and these must be in different  $D_i$ . In this case,  $\mathbf{X}_{i_j} - \mathbf{X}_{i_{j+1}}$  is also in  $(\mathcal{D}_{\text{An}(\underline{A} \cup \underline{B} \cup \underline{C})})^m$ . No node can be in  $\underline{C}$  on this walk. We can reduce this to a path such that no node is repeated. Note that the end nodes are in  $\underline{A}$  and  $\underline{B}$ , respectively. ◀

► **Proposition 4.** *Let  $D_1, \dots, D_m$  be the strongly connect components of  $\mathcal{G}$ . If there are no edges between  $D_i$  and  $D_j$ ,  $i \neq j$ , in  $\mathcal{G}$ , then  $\overline{\mathbf{X}}(t)_{D_i} \perp\!\!\!\perp \overline{\mathbf{X}}(t)_{D_j} \mid \overline{\mathbf{X}}(t)_{p_i}$  or  $\overline{\mathbf{X}}(t)_{D_i} \perp\!\!\!\perp \overline{\mathbf{X}}(t)_{D_j} \mid \overline{\mathbf{X}}(t)_{p_j}$  where  $p_i = \bigcup_{D_k \in \text{Pa}(D_i)} D_k$  and  $p_j = \bigcup_{D_k \in \text{Pa}(D_j)} D_k$ .*

**Proof.** If there are no edges between any node in  $D_i$  and any node in  $D_j$ , then  $D_i$  and  $D_j$  are not adjacent in the condensation of  $\mathcal{G}$ . The condensation is acyclic, so we can without loss of generality assume that  $D_i$  is not a descendant of  $D_j$ . There are no descendants of  $D_j$  in  $\text{An}(\{D_i, D_j\} \cup \text{Pa}(D_j))$  and this means that  $D_i$  and  $D_j$  are separated by  $\text{Pa}(D_j)$  in  $(\mathcal{D}_{\text{An}(\{D_i, D_j\} \cup \text{Pa}(D_j))})^m$  where  $\mathcal{D}$  is the condensation of  $\mathcal{G}$ . The result follows from Propositions 2 and 1. ◀

## B Cascade identification

Informally, a cascade of events is a fast sequence of transitions; where fast is relative to the rest of the transitions that are observable during the evolution of the process. Starting from this informal definition, we can develop a naive approach to identifying such cascades in a trajectory. First of all we need to identify two quantities: -  $\lambda_{ft}$ : the *fast threshold* determines when two consecutive transitions are considered to occur *fast*. -  $\lambda_{mcl}$ : the *minimum cascade length* determines the minimum number of fast consecutive events to be considered a cascade.

Given the two parameters the identification procedure consists of iterating over the entire trajectory and identifying subsets of consecutive transitions with length at least  $\lambda_{mcl}$  and with a transition time between each pair of consecutive events of less than  $\lambda_{ft}$ . This approach can also be used to identify a *sentry state*. Indeed, once a cascade of events has been identified, the sentry state is the state from which the cascade begins.

The main limitation of this approach is the difficulty of identifying the correct parameters as it requires knowing in advance common durations and sizes of event cascades.

In addition, we define two simple quantities: *Naive Count* - the number of times a state starts a cascade, and *Naive Score* - the fraction of times that observing a specific state coincides with the start of a cascade.

## C Synthetic Experiments

■ **Table A1** Conditional Intensity Matrices used for the example in Figure 4a. Process A has no parents and therefore its transition rate only depends on its own state: If A is in state 0 (*off*), then its transition rate (to state 1 (*on*)) is 1.0. Process B has a single parent, process A. The states of processes A and B determine the transition rate of process B. If A is in state 0 (*off*) and B is in state 0 (*off*), then B transitions to state 1 (*on*) with rate 0.1. A CTBN is defined from its CIMs and its initial distribution. Its graph illustrates the dependence structure in the CIMs.

A	0	1				
0	-1.0	1.0				
1	5.0	-5.0				

A	B	0	1		
0	0	-0.1	0.1		
1	1	15.0	-15.0		
	0	15.0	-15.0		
	1	0.1	-0.1		

B	C	0	1
0	0	-0.1	0.1
1	1	15.0	-15.0
	0	-15.0	15.0
	1	0.1	-0.1



## 8:20 Analyzing Complex Systems with Cascades Using CTBNs

■ **Table A2** Values of EDNT, REDNT, Naive Score, and Naive Count for the CTBN depicted in Figure 4a. Higher values of REDNT indicate CTBN states that are more likely to be sentry states. One should note that high-scoring states with few alarms (bold rows) are more interesting in our application as they correspond to states that occur before strong cascading behavior.

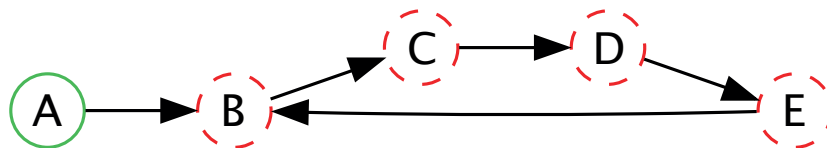
A	B	C	EDNT	REDNT	Naive Score	Naive Count
<b>1</b>	<b>0</b>	<b>0</b>	<b>6.316</b>	<b>1.589</b>	<b>0.35</b>	<b>304</b>
1	0	1	6.444	1.359	0.21	13
<b>0</b>	<b>1</b>	<b>0</b>	<b>5.394</b>	<b>1.357</b>	<b>0.16</b>	<b>41</b>
<b>0</b>	<b>0</b>	<b>1</b>	<b>4.740</b>	<b>1.192</b>	<b>0.03</b>	<b>26</b>
0	1	1	5.511	1.163	0.22	153
1	1	0	6.173	1.145	0.08	57
1	1	1	5.455	1.0	0.03	19
<b>0</b>	<b>0</b>	<b>0</b>	<b>3.976</b>	<b>1.0</b>	<b>0.02</b>	<b>24</b>

■ **Table A3** Values of EDNT, REDNT, Naive Score, and Naive Count of the states with at most one active alarm for the CTBN depicted in Figure 5a.

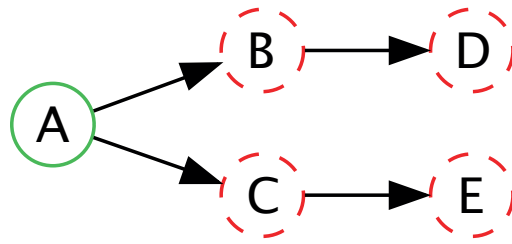
A	B	C	D	E	F	EDNT	REDNT	Naive Score	Naive Count
0	0	1	0	0	0	12.98	1.46	0.24	2172
0	0	0	1	0	0	11.33	1.28	0.25	2156
0	1	0	0	0	0	10.76	1.21	0.04	848
0	0	0	0	1	0	10.75	1.21	0.14	1533
1	0	0	0	0	0	10.24	1.15	0.02	341
0	0	0	0	0	1	9.90	1.12	0.03	651
0	0	0	0	0	0	8.87	1.0	0.01	426



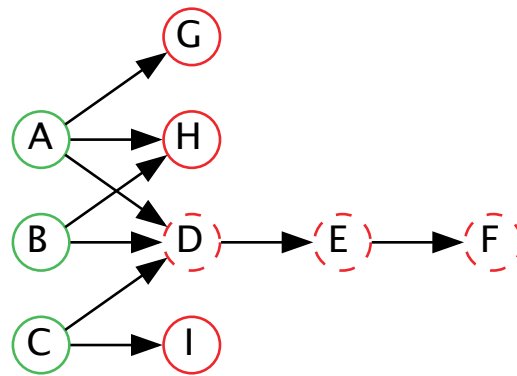
■ **Figure A1** Graph  $\mathcal{G}$  of a CTBN model consisting of a chain of five processes.



■ **Figure A2** Graph  $\mathcal{G}$  of a CTBN model consisting of five processes, including a cycle.



■ **Figure A3** Graph  $\mathcal{G}$  of a CTBN model consisting of five processes. The graph  $\mathcal{G}$  contains a bifurcation after the root node  $A$ .



■ **Figure A4** Graph  $\mathcal{G}$  of a CTBN model consisting of nine processes and with a more complex structure. The sentry state has three active alarms.




# A Sound and Complete Tableau System for Fuzzy Halpern and Shoham’s Interval Temporal Logic

Willem Conradie ✉ 🏠 

School of Mathematics, University of the Witwatersrand, Johannesburg, South Africa

Riccardo Monego ✉ 

Department of Mathematics and Computer Science, University of Ferrara, Italy

Emilio Muñoz-Velasco ✉ 🏠 

Department of Applied Mathematics, University of Málaga, Spain

Guido Sciavicco ✉ 🏠 

Department of Mathematics and Computer Science, University of Ferrara, Italy

Ionel Eduard Stan ✉ 🏠 

Faculty of Engineering, Free University of Bozen-Bolzano, Italy

---

## Abstract

Interval temporal logic plays a critical role in various applications, including planning, scheduling, and formal verification; recently, interval temporal logic has also been successfully applied to learning from temporal data. Halpern and Shoham’s interval temporal logic, in particular, stands out as a very intuitive, yet expressive, interval-based formalism. To address real-world scenarios involving uncertainty and imprecision, Halpern and Shoham’s logic has been recently generalized to the fuzzy (many-valued) case. The resulting language capitalizes on many-valued modal logics, allowing for a range of truth values that reflect multiple expert perspectives, but inherits the bad computational behaviour of its crisp counterpart. In this work, we investigate a sound and complete tableau system for fuzzy Halpern and Shoham’s logic, which, although possibly non-terminating, offers a semi-decision procedure for the finite case.

**2012 ACM Subject Classification** Theory of computation → Theory and algorithms for application domains

**Keywords and phrases** Interval temporal logic, many-valued logic, tableau system

**Digital Object Identifier** 10.4230/LIPICs.TIME.2023.9

**Funding** We acknowledge the support of the INDAM-GNCS project *Symbolic and Numerical Analysis of Cyberphysical Systems* (code CUP\_E53C22001930001), funded by INDAM, and of the FIRD project *Symbolic Geometric Learning*, funded by the University of Ferrara. Work is partially supported by the Spanish Project PID2022-140630NB-I00.

## 1 Introduction

Temporal logic is an essential framework for representing and reasoning about time. To accurately represent time, it is crucial to adopt suitable primitive ontological entities, usually categorized into point-based and interval-based ones. In this work, we take intervals as primary semantic objects. *Halpern and Shoham’s Modal Logic for Time Intervals (HS)* [14] is one of the most influential logical languages for time intervals, providing a robust and expressive formalism for reasoning about temporal relations between events with duration. Its applications range from planning, to scheduling, to formal verification; more recently, it has been shown how transparent and explainable interval temporal logic theories can be extracted from temporal data by exploiting the integration of HS in machine learning systems, including decision trees (e.g., see [6, 19]) and random forests (e.g., see [15, 16]).



© Willem Conradie, Riccardo Monego, Emilio Muñoz-Velasco, Guido Sciavicco, and Ionel Eduard Stan; licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 9; pp. 9:1–9:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

As it turns out, the satisfiability problem for HS is undecidable in all interesting cases of underlying linear order. Various strategies have been studied to obtain fragments of HS with better computational behaviour, such as restricting the set of modal operators [1, 3], constraining the underlying temporal structure [17], restricting the propositional power of the languages [5], and considering coarser logics based on relations that describe a less precise relationship between intervals [18]. On a more practical side, a few attempts to devise practical reasoning systems for HS and its fragments have been made; among them there are sound and complete procedures, respectively for the fragment of HS known as *PNL* [12, 4], and the coarser version of HS called *HS<sub>3</sub>* [18], among a few others; moreover, it has also been devised a similar procedure for the interval temporal logic *CDT*, introduced in [20], which generalizes HS to the case of ternary relations [11].

A unifying aspect of the work on interval temporal logic as we have presented it is the crisp (that is, based on the classic two-values Boolean algebra) semantics of all mentioned logics. In order to enhance the applicability and effectiveness in addressing real-world scenarios, it has been recently proposed to generalize the syntax and semantics of HS to accommodate the inherent uncertainty and imprecision when dealing with real-world data, including (multivariate) time series. A natural way to accomplish such a generalization is following the pioneering work of Fitting on fuzzy modal logics [8], in which both propositions and accessibility relations are no longer just true or false but can have different truth values. As a consequence, the definition of a fuzzy logic rests on a specific class of algebras. Typical choices are Heyting algebras and Łukasiewicz algebra; in the former case, the resulting logic embodies the perspectives of multiple experts whose opinions may not necessarily be independent, while in the latter case the idea is to represent intrinsic vagueness of data.

*Fuzzy Halpern and Shoham's Modal Logic for Time Intervals (FHS)* [7] is precisely the Fitting-style generalization of HS in the case of Heyting algebras. FHS inherits the bad computational behaviour of its crisp counterpart. In particular, in the case of chain Heyting algebras and the class of all (fuzzy) linearly ordered sets, the (fuzzy generalisation of the) satisfiability problem for FHS is undecidable, as well as in the case of all finite (fuzzy) linearly ordered sets, and it is believed so in the other two natural sub-classes of Heyting algebras, that is, the class of finite and the class of Boolean Heyting algebras. However, satisfiability of interval temporal logic formulas is much less studied in the fuzzy case than it is in its crisp counterpart. In this sense, there is a general lack of reasoning tools that are able to deal with fuzzy interval temporal logics, and with FHS in particular.

This work is a first step towards filling in this gap. In particular, we consider FHS in the case of finite Heyting algebras, and, following (again) Fitting [9], we study a tableau system for FHS in the case of all (fuzzy) linear orders. We shall prove that our tableau system, which generalizes tableau systems for crisp interval logics such as those proposed in [11, 12], is sound and complete for satisfiability (at a certain degree of truth or more), and that it is a semi-decision procedure for the case of all finite (fuzzy) linear orders.

This paper is organized as follows. In Section 2 we give some necessary background on HS, Heyting algebras and their properties, and FHS. Then, in Section 3 we present our tableau system, and prove its soundness and completeness, before concluding.

## 2 Background

While several different interval temporal logics have been proposed in the recent literature [13], *Halpern and Shoham's Modal Logic for Time Intervals (HS)* [14] is certainly the formalism that has received the most attention. Let  $\mathbb{D} = \langle D, < \rangle$  be a linear order with

■ **Table 1** Allen’s interval relations and HS modalities.

HS modality	Definition w.r.t. the interval structure	Example
$\langle A \rangle$ (after)	$[x, y]R_A[z, t] \Leftrightarrow y = z$	
$\langle L \rangle$ (later)	$[x, y]R_L[z, t] \Leftrightarrow y < z$	
$\langle B \rangle$ (begins)	$[x, y]R_B[z, t] \Leftrightarrow x = z \wedge t < y$	
$\langle E \rangle$ (ends)	$[x, y]R_E[z, t] \Leftrightarrow y = t \wedge x < z$	
$\langle D \rangle$ (during)	$[x, y]R_D[z, t] \Leftrightarrow x < z \wedge t < y$	
$\langle O \rangle$ (overlaps)	$[x, y]R_O[z, t] \Leftrightarrow x < z < y < t$	

domain  $D$ ; in the following, we shall use  $D$  and  $\mathbb{D}$  interchangeably. A *strict interval* over  $\mathbb{D}$  is an ordered pair  $[x, y]$ , where  $x, y \in \mathbb{D}$  and  $x < y$ . If we exclude the identity relation, there are 12 different binary ordering relations between two strict intervals on a linear order, often called *Allen’s interval 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 Tab. 1, and their *inverses*, that is,  $R_{\bar{X}} = (R_X)^{-1}$ , for each  $X \in \{A, L, B, E, D, O\}$ . We interpret interval structures as Kripke structures, with Allen’s relations playing the role of accessibility relations. Thus, we associate an *existential modality*  $\langle X \rangle$  with each Allen’s relation  $R_X$ . Moreover, for each  $X \in \{A, L, B, E, D, O\}$ , the *transpose* of modality  $\langle X \rangle$  is the modality  $\langle \bar{X} \rangle$  corresponding to the inverse relation  $R_{\bar{X}}$  of  $R_X$ . Now, let  $\mathcal{X} = \{A, \bar{A}, L, \bar{L}, B, \bar{B}, E, \bar{E}, D, \bar{D}, O, \bar{O}\}$ ; well-formed HS formulas are built from a set of *propositional letters*  $\mathcal{P}$ , the classical connectives  $\vee$  and  $\neg$ , and a modality for each Allen’s interval relation, as follows:

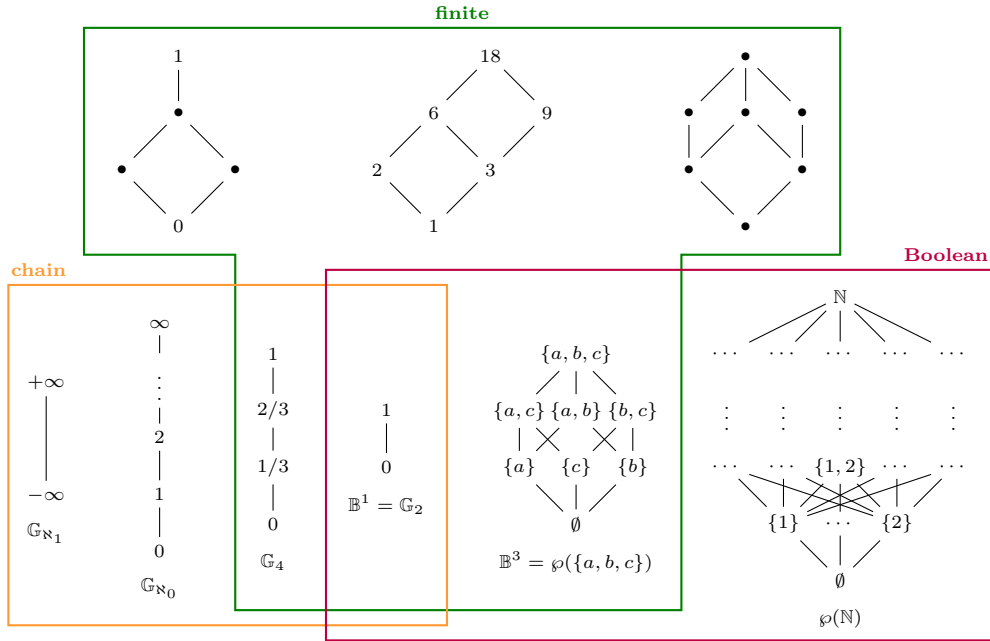
$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \psi \mid \langle X \rangle\varphi,$$

where  $p \in \mathcal{P}$  and  $X \in \mathcal{X}$ . The other propositional connectives and constants (i.e.,  $\psi_1 \wedge \psi_2 \equiv \neg\psi_1 \vee \neg\psi_2$ ,  $\psi_1 \rightarrow \psi_2 \equiv \neg\psi_1 \vee \psi_2$  and  $\top = p \vee \neg p$ ), as well as, for each  $X \in \mathcal{X}$ , the *universal modality*  $[X]$  (e.g.,  $[A]\varphi \equiv \neg\langle A \rangle\neg\varphi$ ), can be derived in the standard way. The set of all subformulas of a given HS formula  $\varphi$  is denoted by  $sub(\varphi)$ .

The strict semantics of HS is given in terms of *interval models* of the type  $M = \langle \mathbb{I}(\mathbb{D}), V \rangle$ , where  $\mathbb{D}$  is a linear order,  $\mathbb{I}(\mathbb{D})$  is the set of all strict intervals over  $\mathbb{D}$ , and  $V$  is a *valuation function*  $V : \mathcal{P} \rightarrow 2^{\mathbb{I}(\mathbb{D})}$  which assigns to every atomic proposition  $p \in \mathcal{P}$  the set of intervals  $V(p)$  on which  $p$  holds. The truth of a formula  $\varphi$  on a given interval  $[x, y]$  in an interval model  $M$ , denoted by  $M, [x, y] \Vdash \varphi$ , is defined by structural induction on the complexity of formulas, as follows:

$$\begin{array}{ll} M, [x, y] \Vdash p & \text{if and only if } [x, y] \in V(p), \text{ for each } p \in \mathcal{AP}, \\ M, [x, y] \Vdash \neg\psi & \text{if and only if } M, [x, y] \not\Vdash \psi, \\ M, [x, y] \Vdash \psi_1 \vee \psi_2 & \text{if and only if } M, [x, y] \Vdash \psi_1 \text{ or } M, [x, y] \Vdash \psi_2, \\ M, [x, y] \Vdash \langle X \rangle\psi & \text{if and only if there exists } [w, z] \text{ s.t. } [x, y]R_X[w, z] \text{ and } M, [w, z] \Vdash \psi, \end{array}$$

where  $X \in \mathcal{X}$ . Given a model  $M = \langle \mathbb{I}(\mathbb{D}), V \rangle$  and a formula  $\varphi$ , we say that  $M$  *satisfies*  $\varphi$  if there exists an interval  $[x, y] \in \mathbb{I}(\mathbb{D})$  such that  $M, [x, y] \Vdash \varphi$ . A formula  $\varphi$  is *satisfiable* if there exists an interval model that satisfies it. Moreover, a formula  $\varphi$  is *valid* if it is satisfiable at every interval of every (interval) model or, equivalently, if its negation  $\neg\varphi$  is *unsatisfiable*.



■ **Figure 1** Graphical representation of finite, Boolean, and chain algebras. Some examples have well-known names:  $\mathbb{G}$ s are Gödel algebras,  $\mathbb{B}$ s are Boolean algebras, and  $\wp(\mathbb{N})$  is the powerset of  $\mathbb{N}$ .

A *Heyting algebra* is a structure of the type

$$\mathcal{H} = (H, \cap, \cup, \leftrightarrow, 0, 1),$$

where  $(H, \cap, \cup, 0, 1)$  is a bounded lattice with *domain*  $H$ , with top (resp., bottom) element 1 (resp., 0); in the following, we shall use  $H$  and  $\mathcal{H}$  interchangeably. Recall that a bounded lattice is a set with internal operations  $\cap$  (*meet*<sup>1</sup>) and  $\cup$  (*join*), both commutative, associative, and connected by the absorption law, in which a partial order can be defined, as follows:

$$\alpha \preceq \beta \text{ iff } \alpha \cap \beta = \alpha \text{ iff } \alpha \cup \beta = \beta.$$

It is well-known that Heyting algebras are always distributive. In the following we use  $\bigcap$  (resp.,  $\bigcup$ ) to indicate the generalized  $\cap$  (resp.,  $\cup$ ), and we assume them to have the lowest priority in algebraic expressions; moreover, we omit the quantification domains when it is clear from the context. The symbols 0 and 1 denote, respectively, least and the greatest elements of  $\mathcal{H}$ . In other words, a Heyting algebra is a bounded distributive lattice in which the *relative pseudo-complement* of  $\alpha$  w.r.t.  $\beta$ , defined as

$$\bigcup \{ \gamma \mid \alpha \cap \gamma \preceq \beta \},$$

and denoted by  $\alpha \leftrightarrow \beta$  (it is also called *Heyting implication*), exists for every  $\alpha$  and  $\beta$  [10]. For instance, consider the Heyting algebra  $\mathbb{B}^3$  in Fig. 1. Then, as expected,  $0 \leftrightarrow 0 = 1$  and  $1 \leftrightarrow 0 = 0$ , where 0 is  $\emptyset$  and 1 is  $\{a, b, c\}$  (and, in general, this is true for every Heyting algebra, since it generalizes the Boolean case); moreover, we have that, for example,  $\{a, c\} \leftrightarrow 0 = \{b\}$ . A Heyting algebra is said to be *complete* if for every subset  $H' \subseteq H$ , both its least upper

<sup>1</sup> This is the classical nomenclature in lattice theory, and it should not be confused with Allen’s relation *meets*.



bound  $\bigcup H'$  and its greatest lower bound  $\bigcap H'$  exist. Moreover, a Heyting algebra is *finite* if its domain is finite, *Boolean* if it is isomorphic to a nonempty set of subsets of a given set closed under the set operations of union, intersection, and complement relative to that set, and a *chain* if  $\preceq$  is total. Graphical examples of such algebras can be found in Figure 1.

As in [7], assuming that  $\mathcal{H}$  is a complete Heyting algebra with domain  $H$  we define an *adequate fuzzy strictly linearly ordered set* as a structure of the type

$$\tilde{\mathbb{D}} = \langle D, \tilde{<}, \tilde{=} \rangle,$$

where  $D$  is a *domain* (and, again, we identify  $D$  with  $\mathbb{D}$ ) enriched with two functions  $\tilde{<}, \tilde{=} : D \times D \mapsto \mathcal{H}$ , for which the following conditions apply for every  $x, y$ , and  $z$ :

$$\begin{aligned} \tilde{=}(x, y) &= 1 \text{ iff } x = y, \\ \tilde{=}(x, y) &= \tilde{=}(y, x), \\ \tilde{<}(x, x) &= 0, \\ \tilde{<}(x, z) &\succeq \tilde{<}(x, y) \cap \tilde{<}(y, z), \\ \text{if } \tilde{<}(x, y) \succ 0 \text{ and } \tilde{<}(y, z) \succ 0 &\text{ then } \tilde{<}(x, z) \succ 0, \\ \text{if } \tilde{<}(x, y) = 0 \text{ and } \tilde{<}(y, x) = 0 &\text{ then } \tilde{=}(x, y) = 1, \\ \text{if } \tilde{=}(x, y) \succ 0 &\text{ then } \tilde{<}(x, y) \prec 1. \end{aligned}$$

An adequate fuzzy linear order is *finite* when  $D$  is finite. The above conditions are called *adequate fuzzy linear order axioms*.

Now, let us fix a complete Heyting algebra  $\mathcal{H}$ . Similarly to the crisp case, well-formed *Fuzzy Halpern and Shoham's Modal Logic for Time Intervals (FHS)* formulas are built from a set of propositional letters  $\mathcal{P}$ , the classical connectives  $\vee$  and  $\neg$ , and a modality for each Allen's interval relation, as follows:

$$\varphi ::= \alpha \mid p \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \varphi \rightarrow \psi \mid \langle X \rangle \varphi \mid [X] \varphi,$$

where  $\alpha \in \mathcal{H}$ ,  $p \in \mathcal{P}$ , and, as in the crisp case,  $X \in \mathcal{X}$ . As before, the set of all subformulas of a given FHS formula  $\varphi$  is denoted by  $sub(\varphi)$ .

As for the semantics of FHS formulas, given an adequate fuzzy strictly linearly ordered set we define the set of fuzzy strict intervals in  $\tilde{\mathbb{D}}$  as

$$\mathbb{I}(\tilde{\mathbb{D}}) = \{[x, y] \mid \tilde{<}(x, y) \succ 0\},$$

and, generalizing classical Boolean evaluation, propositional letters are directly evaluated in the underlying algebra by defining a *fuzzy valuation function*, as follows:

$$\tilde{V} : \mathcal{P} \times \mathbb{I}(\tilde{\mathbb{D}}) \mapsto H.$$

On top of the fuzzyfication of valuations we need to define how accessibility relations behave in the fuzzy context. The definition of fuzzy Allen's relations is obtained by generalizing the original, crisp definition, and substituting every  $=$  with  $\tilde{=}$  and every  $<$  with  $\tilde{<}$ :

$$\begin{aligned} \tilde{R}_A([x, y], [z, t]) &= \tilde{=}(y, z), \\ \tilde{R}_L([x, y], [z, t]) &= \tilde{<}(y, z), \\ \tilde{R}_B([x, y], [z, t]) &= \tilde{=}(x, z) \cap \tilde{<}(t, y), \\ \tilde{R}_E([x, y], [z, t]) &= \tilde{<}(x, z) \cap \tilde{=}(y, t), \\ \tilde{R}_D([x, y], [z, t]) &= \tilde{<}(x, z) \cap \tilde{<}(t, y), \\ \tilde{R}_O([x, y], [z, t]) &= \tilde{<}(x, z) \cap \tilde{<}(z, y) \cap \tilde{<}(y, t), \end{aligned}$$

and similarly for the inverse relations. Finally, we say that an  $\mathcal{H}$ -valued interval model (or *fuzzy interval model*) is a tuple of the type:

$$\tilde{M} = \langle \mathbb{I}(\tilde{\mathbb{D}}), \tilde{V} \rangle$$

## 9:6 A Sound and Complete Tableau System for FHS

where  $\widetilde{\mathbb{D}}$  is a fuzzy strictly linearly ordered set and  $\widetilde{V}$  is a fuzzy valuation function. We interpret an FHS formula in a fuzzy interval model  $\widetilde{M}$  and an interval  $[x, y]$  by extending the valuation  $\widetilde{V}$  of propositional letters as follows, where  $X \in \mathcal{X}$  and  $[z, t]$  varies in  $\mathbb{I}(\widetilde{\mathbb{D}})$ :

$$\begin{aligned}\widetilde{V}(\alpha, [x, y]) &= \alpha, \\ \widetilde{V}(\varphi \wedge \psi, [x, y]) &= \widetilde{V}(\varphi, [x, y]) \cap \widetilde{V}(\psi, [x, y]), \\ \widetilde{V}(\varphi \vee \psi, [x, y]) &= \widetilde{V}(\varphi, [x, y]) \cup \widetilde{V}(\psi, [x, y]), \\ \widetilde{V}(\varphi \rightarrow \psi, [x, y]) &= \widetilde{V}(\varphi, [x, y]) \hookrightarrow \widetilde{V}(\psi, [x, y]), \\ \widetilde{V}(\langle X \rangle \varphi, [x, y]) &= \bigcup \{ \widetilde{R}_X([x, y], [z, t]) \cap \widetilde{V}(\varphi, [z, t]) \}, \\ \widetilde{V}([X]\varphi, [x, y]) &= \bigcap \{ \widetilde{R}_X([x, y], [z, t]) \hookrightarrow \widetilde{V}(\varphi, [z, t]) \}.\end{aligned}$$

We say that a formula of FHS  $\varphi$  is  $\alpha$ -satisfied at an interval  $[x, y]$  in a fuzzy interval model  $\widetilde{M}$  if and only if

$$\widetilde{V}(\varphi, [x, y]) \succeq \alpha.$$

The formula  $\varphi$  is  $\alpha$ -satisfiable if and only if there exists a fuzzy interval model and an interval in that model where it is  $\alpha$ -satisfied. A formula is *satisfiable* if it is  $\alpha$ -satisfiable for some  $\alpha \in \mathcal{H}$ ,  $\alpha \neq 0$ . A formula is  $\alpha$ -valid if it is  $\alpha$ -satisfied at every interval in every model, and *valid* if it is 1-valid. Observe that since a Heyting algebra, in general, does not encompass classical negation, and since our definition of satisfiability is graded, instead of absolute, then the usual duality of satisfiability and validity does not hold anymore.

As shown in [7],  $\alpha$ -satisfiability of FHS formulas is undecidable in the case of chain algebras. Such a result cannot be immediately generalized to the case of *all* Heyting algebras, but, since as crisp HS is undecidable in every class of linearly ordered sets, one can expect that FHS is too, regardless the underlying algebra.

### 3 A Tableau System for FHS

In this section we consider the problem of reasoning with FHS formulas. Tableau systems have been introduced in [4, 11, 12, 18] for variants, fragments, and generalizations of crisp HS, and in [9] for fuzzy modal logics; as in the latter case, we limit ourselves to the case of finite Heyting algebras as truth value algebras.

A tableau for a FHS formula is a directed tree, in which every node is associated to a *truth judgment*, to a pair formula/interval, and to a finite constraint system. In Fitting's terminology, a truth judgement, as we use it, is a *signed formula with bounding implications* [9]. Such a system represents an adequate fuzzy linearly ordered set; its constraints come from both the formula whose satisfiability has to be checked and the axioms that every adequate fuzzy linear order must meet. It may be possible that such a constraint system cannot be satisfied at some node: this will cause the branch that contain that node to be closed.

► **Definition 1** (fuzzy constraint system). *Given a finite Heyting algebra  $\mathcal{H}$ , a fuzzy constraint system  $C$  is a finite set of elements  $\{x, y, \dots\}$  associated to a finite set of constraints of the following types:*

$$\begin{aligned}\cong(x, y) \bowtie \alpha, \\ \widetilde{\prec}(x, y) \bowtie \alpha, \\ F \quad \quad \quad F \text{ is an adequate fuzzy linear order axiom,}\end{aligned}$$

where  $\alpha \in \mathcal{H}$  and  $\bowtie \in \{\preceq, \succeq, \prec, \succ\}$ .

For the sake of convenience, when describing a constraint system we shall omit to include adequate fuzzy linear order axioms (as they do not vary from system to system). It is immediate to see that any fuzzy constraint system can be checked for satisfiability using a first-order reasoner, and that termination is guaranteed by the fact that the system is finite. In the following, we shall use  $x \in C$  to indicate that a certain element is mentioned in the fuzzy constraint system  $C$ . Intuitively,  $C$  represent a possibly incomplete adequate fuzzy linear order; if  $C$  can be satisfied, then it can be extended to a complete adequate fuzzy linear order. In the following, we say that  $C$  is *solved* if a value for  $\cong$  and a value for  $\prec$  has been chosen for every pair  $x, y \in C$  in such a way that all constraints are met; moreover, we say that  $C$  is *inconsistent* if a solution cannot be found. We assume that a constraint system  $C$  (once solved) can be *queried*, so that, for example, at any given time, we can know the value of the relation  $R_X([x, y], [z, t])$  for any points  $x, y, z, t \in C$ . In particular, for a given constraint system  $C$ , we assume that a function  $o(C)$  (resp.,  $n(C)$ ) is defined that returns a list of all possible *old* intervals  $[x, y]$  that can be formed with points in  $C$  (resp., all possible *new* intervals that can be formed in  $C$  using one or two points not currently in  $C$ ); clearly,  $o(C) \cap n(C) = \emptyset$ . Finally, observe that for a given non-inconsistent system  $C$  there may be more than one solution. The set of all possible constraint systems is denoted by  $\mathcal{C}$ .

Because classic negation is not available in the fuzzy case, following Fitting, our tableau is designed to answer the question of whether a given formula  $\varphi$  can be satisfied to a degree at least  $\alpha$  in  $\mathcal{H}$ -valued interval model, for a given finite Heyting algebra  $\mathcal{H}$ .

► **Definition 2** (decoration). *Given a Heyting algebra  $\mathcal{H}$ , an FHS formula  $\varphi$ , and a fuzzy constraint system  $C$ , a decoration is an object of the type*

$$Q(\alpha \rightarrow \varphi, [x, y], C), \text{ or } Q(\varphi \rightarrow \alpha, [x, y], C),$$

where  $\alpha \in \mathcal{H}$  and  $Q \in \{T, F\}$  is a judgment. The expression  $\alpha \rightarrow \varphi$  ( $\varphi \rightarrow \alpha$ ) is an assertion on  $[x, y] \in o(C)$ . The universe of all possible decorations is denoted by  $\mathcal{D}$ .

Intuitively, the assertion  $\alpha \rightarrow \varphi$  (resp.,  $\varphi \rightarrow \alpha$ ) on an interval  $[x, y]$  means that there exists a fuzzy model  $\tilde{M}$  with valuation function  $\tilde{V}$  such that  $\tilde{V}(\varphi, [x, y]) \succeq \alpha$  (resp., for every fuzzy model  $\tilde{M}$  and valuation functions  $\tilde{V}$  it is the case that  $\tilde{V}(\varphi, [x, y]) \preceq \alpha$ ); associating a judgment  $T$  (resp.,  $F$ ) to an assertion can be interpreted as (trying to) proving that the assertion holds (resp., does not hold).

► **Definition 3** (tableau for FHS). *Given an FHS formula  $\varphi$  and a finite Heyting algebra  $\mathcal{H}$ , the tableau  $\tau$  for  $\varphi$  and  $\alpha \in \mathcal{H}$  is an object of the type*

$$\tau = (\mathcal{V}, \mathcal{E}, d, f, c),$$

where  $(\mathcal{V}, \mathcal{E})$  is a tree with vertices (or nodes) in  $\mathcal{V}$  and edges in  $\mathcal{E}$ . The nodes in  $\tau$  are partially ordered by the relation  $\triangleleft$  (induced by the edges) and whose set of branches is denoted by  $\mathcal{B}$ ,

$$d : \mathcal{V} \rightarrow \mathcal{D},$$

is a node labeling function, which associates a decoration  $Q(\psi \rightarrow \alpha, [x, y], C)$  or  $Q(\alpha \rightarrow \psi, [x, y], C)$  to any node  $\nu$ , where  $\psi \in \text{sub}(\varphi)$  and  $x, y \in C$ , and

$$f : \mathcal{V} \rightarrow \{0, 1\}$$

is a node flag function, which determines which nodes have been already expanded,

$$c : \mathcal{B} \rightarrow \mathcal{C}$$

is a branch labeling function, which associates every branch to the constraint system in the decoration of its leaf, and it has been obtained starting from the initial tableau  $\tau_0$

$$(\{\nu_0\}, \emptyset, \{(\nu_0, T(\alpha \rightarrow \varphi, [x, y], \{x, y, \tilde{<(x, y) \succ 0\}))\}, \{(\nu_0, 0)\}, \{(\nu_0, \{x, y, \tilde{<(x, y) \succ 0\})\})\})$$

by iteratively applying the branch expansion rule in Fig. 2 to the closest-to-the-root node  $\nu$  such that  $f(\nu) = 0$  and every leaf  $\nu'$  such that  $\nu \triangleleft \nu'$ , until no further application is possible or all branches have been closed. The tableau is closed (resp., open) if all its branches (resp., at least one of its branches) are (resp., is) closed  $\blacktimes$  by some condition in Fig. 3 (resp., open  $\checkmark$ ).

Following the original terminology, the first four rules in Fig. 2 are referred to as *reverse* rules, the rules for nodes with a decoration that contain a propositional formula in the assertion are referred to as *propositional* rules, and the rules for nodes with a decoration that contain a temporal formulas in the assertion are referred to as *temporal* rules. Observe that the set actually covers all cases; those that are not covered can be treated by (the application of) a reverse rule.

The application of the branch expansion rule to a specific branch  $B$  in a tableau defined as above works as follows. First, the closest-to-the-root node  $\nu$  of  $B$ , such that  $f(\nu) = 0$  is chosen. Then, the consequent of the rule produces a new tree which is attached to the leaf of  $B$ ; observe that the constraint system used in the application is always the one currently at the leaf. Finally, the application is not possible if the nodes produced by it are already present on the branch.

Now, we move to proving that the tableau system is sound.

► **Lemma 4** (soundness). *Let  $\varphi$  be an FHS formula and  $\alpha \in \mathcal{H}$  a constant of a finite Heyting algebra. Then, if  $\varphi$  is  $\alpha$ -satisfiable, then the tableau  $\tau$  for  $\varphi$  and  $\alpha$  is open.*

**Proof.** Consider an FHS formula  $\varphi$ . Assume that  $\tau$  is the tableau for  $\varphi$  and  $\alpha \in \mathcal{H}$ , where  $\mathcal{H}$  is a fixed finite Heyting algebra. We proceed contrapositively to prove that if  $\tau$  is closed then  $\varphi$  is not  $\alpha$ -satisfiable. Given a node  $\nu$  in  $\tau$  such that  $C$  is the constraint system in  $d(\nu)$ , we define the set

$$S(\nu) = \{\nu' \mid \nu' \triangleleft \nu\}$$

and we say that  $S(\nu)$  is  $\alpha$ -satisfiable if and only if there is an  $\mathcal{H}$ -valued interval model

$$\tilde{M} = \langle \mathbb{I}(C^*), \tilde{V} \rangle,$$

where  $C^*$  is a fuzzy strictly linearly ordered set that extends  $C$ , such that

- for each node  $\nu' \in S(\nu)$  such that  $d(\nu') = T(\beta \rightarrow \psi, [x, y], C')$  (resp.,  $F(\psi \rightarrow \beta, [x, y], C')$ ), it is the case that  $\tilde{V}(\psi, [x, y]) \succeq \beta$  (resp.,  $\tilde{V}(\psi, [x, y]) \succeq \gamma$ , for some minimal  $\gamma$  not below  $\beta$ ), and
- for each node  $\nu' \in S(\nu)$  such that  $d(\nu') = T(\psi \rightarrow \beta, [x, y], C')$  (resp.,  $F(\beta \rightarrow \psi, [x, y], C')$ ), it is the case that  $\tilde{V}(\psi, [x, y]) \preceq \beta$  (resp.,  $\tilde{V}(\psi, [x, y]) \preceq \gamma$  for some maximal  $\gamma$  not above  $\beta$ ).

Observe that  $\tilde{M}$  depends on  $\alpha$  since  $\nu_0 \in S(\nu)$ , for every  $\nu$ . Moreover, if  $S(\nu_0)$  is  $\alpha$ -satisfiable, then  $\varphi$  is  $\alpha$ -satisfiable. Now, we prove the following stronger statement: *if every branch containing a node  $\nu$  is closed, then the set  $S(\nu)$  is not  $\alpha$ -satisfiable.* Let us proceed by induction on the height  $h$  of the node  $\nu$ .

$$(T \succeq) \frac{T(\alpha \rightarrow \psi, [x, y], C)}{F(\psi \rightarrow \gamma, [x, y], c(B))}$$

where  $\alpha \neq 0$  and  $\gamma$  is any maximal element not above  $\alpha$ , i.e.,  $\gamma \not\leq \alpha$

$$(F \succeq) \frac{F(\alpha \rightarrow \psi, [x, y], C)}{T(\psi \rightarrow \beta_i, [x, y], c(B)) \mid \dots \mid T(\psi \rightarrow \beta_n, [x, y], c(B))}$$

where  $\alpha \neq 0$  and  $\beta_1, \dots, \beta_n$  are all maximal elements not above  $\alpha$ , i.e.,  $\beta_1, \dots, \beta_n \not\leq \alpha$

$$(T \preceq) \frac{T(\psi \rightarrow \alpha, [x, y], C)}{F(\gamma \rightarrow \psi, [x, y], c(B))}$$

where  $\alpha \neq 1$  and  $\gamma$  is any minimal element not below  $\alpha$ , i.e.,  $\gamma \not\geq \alpha$

$$(F \preceq) \frac{F(\psi \rightarrow \alpha, [x, y], C)}{T(\beta_i \rightarrow \psi, [x, y], c(B)) \mid \dots \mid T(\beta_i \rightarrow \psi, [x, y], c(B))}$$

where  $\alpha \neq 1$  and  $\beta_1, \dots, \beta_n$  are all minimal elements not below  $\alpha$ , i.e.,  $\beta_1, \dots, \beta_n \not\geq \alpha$

(a) Reverse rules.

$$(T \wedge) \frac{T(\alpha \rightarrow (\psi \wedge \xi), [x, y], C)}{T(\alpha \rightarrow \psi, [x, y], c(B)) \mid T(\alpha \rightarrow \xi, [x, y], c(B))}$$

where  $\alpha \neq 0$

$$(F \wedge) \frac{F(\alpha \rightarrow (\psi \wedge \xi), [x, y], C)}{F(\alpha \rightarrow \psi, [x, y], c(B)) \mid F(\alpha \rightarrow \xi, [x, y], c(B))}$$

where  $\alpha \neq 0$

$$(T \vee) \frac{T((\psi \vee \xi) \rightarrow \alpha, [x, y], C)}{T(\psi \rightarrow \alpha, [x, y], c(B)) \mid T(\xi \rightarrow \alpha, [x, y], c(B))}$$

where  $\alpha \neq 1$

$$(F \vee) \frac{F((\psi \vee \xi) \rightarrow \alpha, [x, y], C)}{F(\psi \rightarrow \alpha, [x, y], c(B)) \mid F(\xi \rightarrow \alpha, [x, y], c(B))}$$

where  $\alpha \neq 1$

$$(T \rightarrow) \frac{T(\alpha \rightarrow (\psi \rightarrow \xi), [x, y], C)}{F(\gamma \rightarrow \psi, [x, y], c(B)) \mid T(\gamma \rightarrow \xi, [x, y], c(B))}$$

where  $\alpha \neq 0$  and  $\gamma$  is any element below  $\alpha$  except 0, i.e.,  $0 \neq \gamma \preceq \alpha$

$$(F \rightarrow) \frac{F(\alpha \rightarrow (\psi \rightarrow \xi), [x, y], C)}{T(\beta_1 \rightarrow \psi, [x, y], c(B)) \mid \dots \mid T(\beta_n \rightarrow \psi, [x, y], c(B)) \mid F(\beta_1 \rightarrow \xi, [x, y], c(B)) \mid \dots \mid F(\beta_n \rightarrow \xi, [x, y], c(B))}$$

where  $\alpha \neq 0$  and  $\beta_1, \dots, \beta_n$  are all elements below  $\alpha$  except 0, i.e.,  $0 \neq \beta_1, \dots, \beta_n \preceq \alpha$

(b) Propositional rules.

$$(T \square) \frac{T(\alpha \rightarrow [X]\psi, [x, y], C)}{T((\alpha \cap \beta_1) \rightarrow \psi, [z_1, t_1], c(B)) \mid \dots \mid T((\alpha \cap \beta_n) \rightarrow \psi, [z_n, t_n], c(B))}$$

where  $\beta_i = R_X([x, y], [z_i, t_i])$ ,  $[z_i, t_i] \in o(c(B))$ ,  $\beta_i \succ 0$ , and  $\alpha \cap \beta_i \neq 0$

$$(T \diamond) \frac{T((X)\psi \rightarrow \alpha, [x, y], C)}{T((\psi \rightarrow (\beta_1 \leftrightarrow \alpha)), [z_1, t_1], c(B)) \mid \dots \mid T(\psi \rightarrow (\beta_n \leftrightarrow \alpha), [z_n, t_n], c(B))}$$

where  $\beta_i = R_X([x, y], [z_i, t_i])$ ,  $[z_i, t_i] \in o(c(B))$ ,  $\beta_i \succ 0$ , and  $\beta_i \leftrightarrow \alpha \neq 1$

$$(F \square) \frac{F(\alpha \rightarrow [X]\psi, [x, y], C)}{F((\alpha \cap \beta_1) \rightarrow \psi, [z_1, t_1], c(B)) \mid \dots \mid F((\alpha \cap \beta_n) \rightarrow \psi, [z_n, t_n], c(B))}$$

where  $\beta_i = R_X([x, y], [z_i, t_i])$ ,  $[z_i, t_i] \in o(c(B)) \cup n(c(B))$ ,  $\beta_i \succ 0$ , and  $\alpha \cap \beta_i \neq 0$

$$(F \diamond) \frac{F((X)\psi \rightarrow \alpha, [x, y], C)}{F(\psi \rightarrow (\beta_1 \leftrightarrow \alpha), [z_1, t_1], c(B)) \mid \dots \mid F(\psi \rightarrow (\beta_n \leftrightarrow \alpha), [z_n, t_n], c(B))}$$

where  $\beta_i = R_X([x, y], [z_i, t_i])$ ,  $[z_i, t_i] \in o(c(B)) \cup n(c(B))$ ,  $\beta_i \succ 0$ , and  $\beta_i \leftrightarrow \alpha \neq 1$

(c) Temporal rules.

■ **Figure 2** Branch expansion rules for a branch  $B$ , to be applied to  $\nu \in B$  under the conditions specified below each rule. The node flag is 0 when a rule is applied on a node with label at the top, modified into 1 after the application, and set to 0 on every produced node. When applying the rules  $(F \square)$  and  $(F \diamond)$ , the constraint system  $C$  is first solved, and, queried for  $o(C)$ , and finally, for  $n(C)$ , returning all possible (old and new) intervals relevant for the application.

$$\begin{array}{cc}
 (\mathbf{X1}) \frac{T(\alpha \rightarrow \beta, [x, y], C)}{\mathbf{X}} & (\mathbf{X2}) \frac{F(\alpha \rightarrow \beta, [x, y], C)}{\mathbf{X}} \\
 \text{where } \alpha \not\leq \beta & \text{where } \alpha \leq \beta \\
 \\
 (\mathbf{X3}) \frac{F(0 \rightarrow \psi, [x, y], C)}{\mathbf{X}} & (\mathbf{X4}) \frac{F(\psi \rightarrow 1, [x, y], C)}{\mathbf{X}} \\
 \\
 (\mathbf{X5}) \frac{T(\beta \rightarrow \psi, [x, y], C)}{\mathbf{X}} & (\mathbf{X6}) \frac{Q(\cdot, \cdot, C)}{\mathbf{X}} \\
 \text{where } \alpha \leq \beta & \text{where } C \text{ is inconsistent}
 \end{array}$$

■ **Figure 3** Branch closing conditions.

If  $h = 0$ , then there is exactly one branch that contains it. Since such branch is closed, one of the following must hold. First, for some  $\nu' \in S(\nu)$ ,  $d(\nu') = T(\beta \rightarrow \gamma, [x, y], C')$  but  $\beta \not\leq \gamma$  (condition **(X1)**), or  $d(\nu') = F(\beta \rightarrow \gamma, [x, y], C')$  but  $\beta \leq \gamma$  (condition **(X2)**). Second, for some  $\nu' \in S(\nu)$ ,  $d(\nu') = F(0 \rightarrow \psi, [x, y], C')$  (condition **(X3)**), or  $F(\psi \rightarrow 1, [x, y], C')$  (condition **(X4)**). Third, for some  $\nu' \in S(\nu)$ ,  $d(\nu') = Q(\cdot, \cdot, C)$  but  $C$  is inconsistent (condition **(X6)**). Or, fourth, for some  $\nu', \nu'' \in S(\nu)$ ,  $d(\nu') = F(\beta \rightarrow \psi, [x, y], C')$  and  $d(\nu'') = T(\gamma \rightarrow \psi, [x, y], C'')$ , but  $\gamma \leq \beta$  (condition **(X5)**). In all such cases,  $M$  cannot be realized, so  $S(\nu)$  is not  $\alpha$ -satisfiable, as we wanted.

Suppose, now, that  $h > 0$ . First, observe that if every branch that contains  $\nu$  is closed, then every branch that contains any of its successors must be closed too, so that the inductive hypothesis applies to them. Then, consider the node  $\nu' \in S(\nu)$  that has been expanded when  $\nu$  was a leaf, and let us analyze the possible rules that have been applied at  $\nu'$ . If  $d(\nu') = T(\psi \rightarrow \beta, [x, y], C')$ , then the immediate successor  $\nu''$  of  $\nu$  is such that  $d(\nu'') = F(\gamma \rightarrow \psi, [x, y], C'')$ , where  $\gamma$  is some minimal element of  $\mathcal{H}$  such that  $\gamma \not\leq \beta$  (rule  $(T \leq)$ ); by inductive hypothesis,  $S(\nu'')$  is not  $\alpha$ -satisfiable, but this implies that  $S(\nu)$  cannot be  $\alpha$ -satisfiable either. If  $d(\nu') = F(\beta \rightarrow \psi, [x, y], C')$ , then all immediate successors  $\nu_i$  of  $\nu$  are such that  $d(\nu_i) = T(\psi \rightarrow \gamma_i, [x, y], C_i)$ , where  $\gamma_i$  is a maximal element of  $\mathcal{H}$  such that  $\gamma_i \not\leq \beta$  (rule  $(F \geq)$ ); by inductive hypothesis,  $S(\nu_i)$  is not  $\alpha$ -satisfiable for any  $i$ , but this implies that  $S(\nu)$  cannot be  $\alpha$ -satisfiable either. The cases in which another reverse rule has been applied to  $\nu$  are similar. If  $d(\nu') = T(\beta \rightarrow (\psi \wedge \xi), [x, y], C')$ , then  $\nu$  has an immediate successor  $\nu_1$  with  $d(\nu_1) = T(\beta \rightarrow \psi, [x, y], C_1)$ , which in turn has an immediate successor  $\nu_2$  with  $d(\nu_2) = T(\beta \rightarrow \xi, [x, y], C_2)$  (rule  $(T \wedge)$ ). By inductive hypothesis,  $S(\nu_2)$ , in particular, is not  $\alpha$ -satisfiable, but this implies that  $S(\nu)$  is not  $\alpha$ -satisfiable either. If  $d(\nu') = F(\beta \rightarrow (\psi \wedge \xi), [x, y], C')$ , then  $\nu$  has two immediate successors  $\nu_1$  and  $\nu_2$  with  $d(\nu_1) = F(\beta \rightarrow \psi, [x, y], C_1)$  and  $d(\nu_2) = F(\beta \rightarrow \xi, [x, y], C_2)$  (rule  $(F \wedge)$ ). By inductive hypothesis, both  $S(\nu_1)$  and  $S(\nu_2)$  are not  $\alpha$ -satisfiable, but this implies that  $S(\nu)$  is not  $\alpha$ -satisfiable either. The cases in which another propositional rule has been applied to  $\nu$  are similar. If  $d(\nu') = T(\beta \rightarrow [X]\psi, [x, y], C')$  then  $\nu$  has a chain of successors  $\nu_1, \dots, \nu_n$ , such that  $d(\nu_i) = T((\beta \cap \gamma_i) \rightarrow \psi, [z_i, t_i], C_i)$  and  $\gamma_i = R_X([x, y], [z_i, t_i])$ , for all  $[z_i, t_i] \in o(c(B))$ , where  $1 \leq i \leq n$  (rule  $(T \square)$ ). Observe that asking that the evaluation of  $[X]\psi$  is above  $\beta$  is equivalent to asking that the evaluation of  $\psi$  is above  $\beta \cap \gamma_i$  on every interval  $[z_i, t_i]$ . Since, in particular,  $S(\nu_n)$  is not  $\alpha$ -satisfiable by inductive hypothesis,  $S(\nu)$  is not  $\alpha$ -satisfiable as well. The case in which  $(T \diamond)$  has been applied to  $\nu$  is similar. Finally, if  $d(\nu') = F(\langle X \rangle \psi \rightarrow \beta), [x, y], C'$  then every immediate successor  $\nu_i$  of  $\nu$  is such that  $d(\nu_i) = F(\psi \rightarrow (\gamma_i \leftrightarrow \beta), [z_i, t_i], C_i)$

and  $\gamma_i = R_X([x, y], [z_i, t_i])$ , for all  $[z_i, t_i] \in o(c(B)) \cup n(c(B))$ , where  $1 \leq i \leq n$  (rule  $(F\Diamond)$ ). Observe that asking that the evaluation of  $\langle X \rangle \psi$  is below  $\beta$  is equivalent to asking that the evaluation of  $\psi$  is below  $\gamma_i \multimap \beta$  on some interval  $[z_i, t_i]$ . Since all  $S(\nu_i)$  are not  $\alpha$ -satisfiable by inductive hypothesis,  $S(\nu)$  is not  $\alpha$ -satisfiable as well. The case in which  $(F\Box)$  has been applied to  $\nu$  is similar.  $\blacktriangleleft$

Finally, we turn our attention to proving completeness.

► **Lemma 5** (completeness). *Let  $\varphi$  be an FHS formula and  $\alpha \in \mathcal{H}$  a constant of a finite Heyting algebra. If  $\tau$  is an open tableau for  $\varphi$  and  $\alpha$ , then  $\varphi$  is  $\alpha$ -satisfiable.*

**Proof.** Consider an FHS formula  $\varphi$ , and assume that  $\tau$  is the tableau for  $\varphi$  and  $\alpha$ . Consider an open branch  $B$  in  $\tau$ , let  $C_\mu = \bigcup_{\nu \in B} C_\nu$ , where  $C_\nu$  is the constraint system in the label  $d(\nu)$ , and let  $C^*$  the complete extension of  $C_\mu$ . Then, consider the model

$$\widetilde{M} = \langle \mathbb{I}(C^*), \widetilde{V} \rangle,$$

where  $\widetilde{V}$  is the following fuzzy valuation function, defined for every propositional letter  $p$  and fuzzy strict interval  $[x, y]$  in  $\mathbb{I}(C^*)$ :

$$\widetilde{V}(p, [x, y]) = \begin{cases} \beta & \text{if } d(\nu) = T(\beta \rightarrow p, [x, y], C), \text{ for some } \nu \in B; \\ \gamma & \text{if } d(\nu) = F(\beta \rightarrow p, [x, y], C), \text{ for some } \nu \in B \text{ and } \gamma \not\leq \beta. \end{cases}$$

The model  $\widetilde{M}$  is the direct translation of the branch  $B$  into an (fuzzy) interval model; in particular, is an coherent assignment of truth values of all propositional letters on all intervals. As much as the case of the judgment  $F(\beta \rightarrow p, [x, y], C)$  is considered we need to associate any truth value  $\gamma$  such that  $\gamma \not\leq \beta$ . We want to prove that, for every  $\nu \in B$ ,

- if  $d(\nu) = T(\beta \rightarrow \psi, [z, t], C)$ , then  $\widetilde{V}(\psi, [z, t]) \succeq \beta$ ,
- if  $d(\nu) = T(\psi \rightarrow \beta, [z, t], C)$ , then  $\widetilde{V}(\psi, [z, t]) \preceq \beta$ ,
- if  $d(\nu) = F(\beta \rightarrow \psi, [z, t], C)$ , then  $\widetilde{V}(\psi, [z, t]) \not\leq \beta$ , and
- if  $d(\nu) = F(\psi \rightarrow \beta, [z, t], C)$ , then  $\widetilde{V}(\psi, [z, t]) \not\leq \beta$ .

Observe that the above implies that  $\varphi$  is  $\alpha$ -satisfiable on  $[x, y]$  in  $\widetilde{M}$ , that is, it is  $\alpha$ -satisfiable. Also, observe that  $\widetilde{M}$  is constructible and well-defined because  $B$  is open. Consider a node  $\nu \in B$  such that  $d(\nu)$  is a decoration with a judgment for a formula  $\psi$  on some interval  $[z, t]$ . We proceed by structural induction on  $\psi$ .

If  $\psi = p$  or  $\psi = \beta$ , then the claim is trivial.

If  $\psi = \xi \wedge \chi$ , then suppose, first, that  $d(\nu) = T(\beta \rightarrow (\xi \wedge \chi), [z, t], C)$ . Since  $\tau$  is fully expanded, rule  $(T\wedge)$  has been applied to  $\nu$ . It follows that  $B$  contains two nodes  $\nu_1$  and  $\nu_2$  such that  $d(\nu_1) = T(\beta \rightarrow \xi, [z, t], C)$  and  $d(\nu_2) = T(\beta \rightarrow \chi, [z, t], C)$ . By inductive hypothesis,  $\widetilde{V}(\xi, [z, t]) \succeq \beta$  and  $\widetilde{V}(\chi, [z, t]) \succeq \beta$ , which is equivalent to  $\widetilde{V}(\psi, [z, t]) \succeq \beta$ . Suppose, now, that  $d(\nu) = F(\beta \rightarrow (\xi \wedge \chi), [z, t], C)$ . Since  $\tau$  is fully expanded, rule  $(F\wedge)$  has been applied to  $\nu$ . It follows that  $B$  contains a node  $\nu'$  such that  $d(\nu') = F(\beta \rightarrow \xi, [z, t], C)$  or  $d(\nu') = F(\beta \rightarrow \chi, [z, t], C)$ . By inductive hypothesis,  $\widetilde{V}(\xi, [z, t]) \not\leq \beta$  or  $\widetilde{V}(\chi, [z, t]) \not\leq \beta$ , which is equivalent to  $\widetilde{V}(\psi, [z, t]) \not\leq \beta$ . The remaining propositional cases are similar.

Finally, if  $\psi = [X]\xi$ , then suppose, first, that  $d(\nu) = T(\beta \rightarrow [X]\xi, [z, t], C)$ . Since  $\tau$  is fully expanded, rule  $(T\Box)$  has been applied to  $\nu$ . This entails that, for every interval  $[z_i, t_i]$  in  $\mathbb{I}(C^*)$ ,  $B$  contains a node  $\nu_i$  such that  $d(\nu_i) = T((\beta \cap \gamma_i) \rightarrow \xi, [z_i, t_i], C_i)$ , that is,  $d(\nu_i) = T((\beta \cap R_X([z, t], [z_i, t_i])) \rightarrow \xi, [z_i, t_i], C_i)$ ; observe that this is guaranteed by the fact that the rule has been applied at a certain point of the construction on the  $n$  possible intervals that are constructible at that point, but then an additional  $(n + 1)$ -th node is also created at the end of the branch with the same decoration, ensuring that, should more



points be added at some later moment, the rule is applied, again, on them. By inductive hypothesis,  $\tilde{V}(\xi, [z_i, t_i]) \succeq (\beta \cap \gamma_i)$ . Therefore,  $\tilde{V}(\psi, [z, t]) = \bigcap_{[z_i, t_i]} \{R_X([z, t], [z_i, t_i]) \leftrightarrow \tilde{V}(\xi, [z_i, t_i])\} \succeq \beta$ . Suppose, now, that  $d(\nu) = F(\beta \rightarrow [X]\xi, [z, t], C)$ . Since  $\tau$  is fully expanded, rule  $(F\Box)$  has been applied to  $\nu$ . This entails that, for some interval  $[z_i, t_i]$  in  $\mathbb{I}(C^*)$ ,  $B$  contains a node  $\nu_i$  such that  $d(\nu_i) = F((\beta \cap \gamma_i) \rightarrow \xi, [z_i, t_i], C_i)$ , that is,  $d(\nu_i) = F((\beta \cap R_X([z, t], [z_i, t_i])) \rightarrow \xi, [z_i, t_i], C_i)$ . By inductive hypothesis,  $\tilde{V}(\xi, [z_i, t_i]) \not\succeq \beta \cap \gamma_i$ . Therefore,  $\tilde{V}(\psi, [z, t]) = \bigcap_{[z_i, t_i]} \{R_X([z, t], [z_i, t_i]) \leftrightarrow \tilde{V}(\xi, [z_i, t_i])\} \not\succeq \beta$ . The remaining temporal cases are similar.  $\blacktriangleleft$

► **Theorem 6** (semi-decision procedure). *The tableau system for FHS is sound and complete. Moreover, it is also a semi-decision procedure in the case of finite domains.*

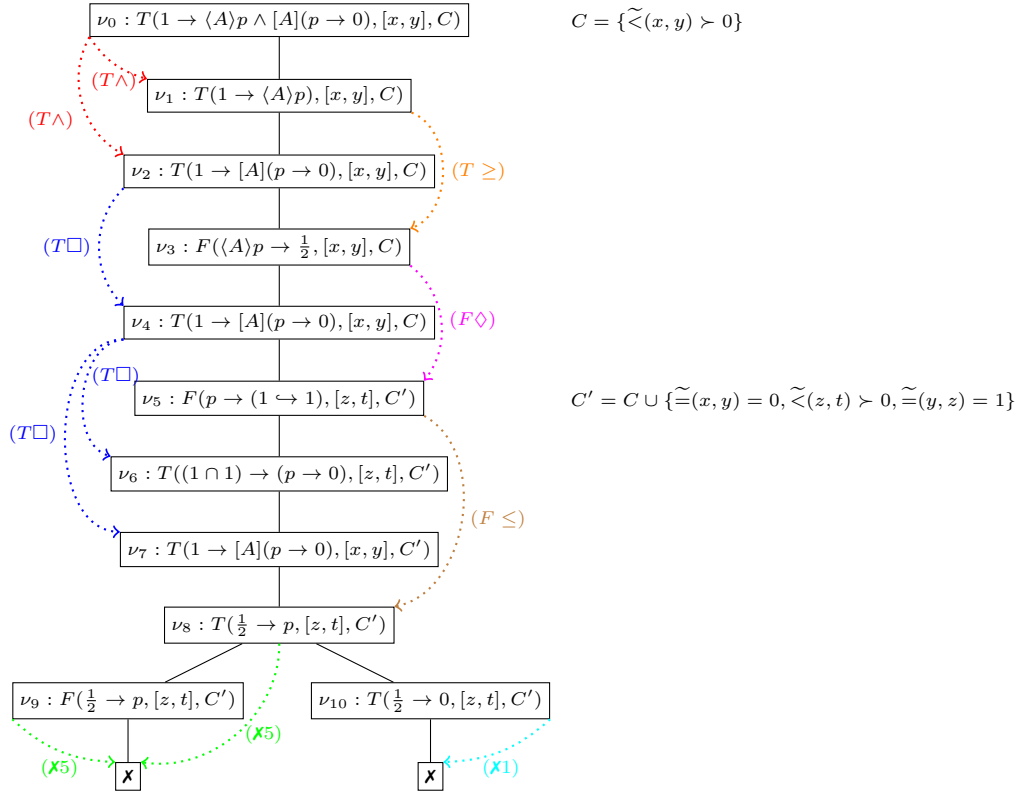
A semi-decision procedure for the problem of establishing if a given FHS formula  $\varphi$  is  $\alpha$ -satisfiable for some truth value  $\alpha$  of a given Heyting algebra, as stated by the above theorem, emerges naturally as the systematic application of the expansion rules (see Fig. 2) to the initial tableau. Termination is not guaranteed as there may exist formulas that are  $\alpha$ -satisfiable but only on an infinite domain: in such a case, no contradiction would be found within a finite amount of time.

We conclude this part with an example of application of the tableau system, illustrated in Fig. 4. In this example, we consider the formula  $\langle A \rangle p \wedge [A](p \rightarrow 0)$  and  $1 \in \mathbb{G}_3$ , where  $\mathbb{G}_3$  is the Gödel algebra from Fig. 1. To show how the system is applied, we tested if the truth value of the above formula is at least 1, that is,  $T(1 \rightarrow \langle A \rangle p \wedge [A](p \rightarrow 0), [x, y], C)$  on the interval  $[x, y]$  with  $C = \{\prec(x, y) \succ 0\}$ ; since all branches are closed, we verified, as expected, that it cannot. In the figure, only two branches are displayed.

## 4 Conclusion

Interval temporal logic is a crucial tool for planning, scheduling, and formal verification, and are also particular interesting for learning tasks, especially from continuous data. To deal with the uncertainty of real data, a fuzzy (many-valued) generalization of the most representative interval temporal logic (HS), called FHS, had been recently introduced and studied. The computational properties of FHS strongly depend on the underlying algebra on which it is based. Within the context of Heyting algebras, we considered, here, the finite case, and we devised a sound and complete tableau system for it. Our method builds on previous work by Fitting, and it is the first case of an implementable deduction procedure for fuzzy interval temporal logic, which could be applied as a reasoning system, for example, on formulas learned from real data in order to combine them with expert knowledge.

As future work, we plan to design an efficient implementation of the proposed tableau system. Observe that, in particular, such an implementation would be a generalization of its crisp counterpart. The experiments that have been carried on so far seem to indicate that the best implementation strategies are those based on the naive approaches as in [18], which is essentially different from the point-based case; therefore, in order to obtain a truly useful tool, an effort should be made to optimize the construction of such a tableau system in the crisp and fuzzy case alike.



■ **Figure 4** Some closed branches of the tableau for  $\langle A \rangle p \wedge [A](p \rightarrow 0)$  and  $1 \in \mathbb{G}_3$ .

## References


- 1 L. Aceto, D. Della Monica, V. Goranko, A. Ingólfssdóttir, A. Montanari, and Guido Sciavicco. A complete classification of the expressiveness of interval logics of Allen's relations: the general and the dense cases. *Acta Informatica*, 53(3):207–246, 2016.
- 2 J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
- 3 D. Bresolin, D. Della Monica, A. Montanari, P. Sala, and G. Sciavicco. Interval temporal logics over strongly discrete linear orders: Expressiveness and complexity. *Theoretical Computers Science*, 560:269–291, 2014.
- 4 D. Bresolin, D. Della Monica, A. Montanari, and G. Sciavicco. A tableau system for right propositional neighborhood logic over finite linear orders: An implementation. In *Proc. of the 22th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, volume 8123 of *LNCS*, pages 74–80. Springer, 2013.
- 5 D. Bresolin, A. Kurucz, E. Muñoz-Velasco, V. Ryzhikov, G. Sciavicco, and M. Zakharyashev. Horn fragments of the halpern-shoham interval temporal logic. *ACM Transactions on Computational Logic*, 18(3):22:1–22:39, 2017.
- 6 A. Brunello, G. Sciavicco, and I. E. Stan. Interval Temporal Logic Decision Tree Learning. In *Proceedings of the 16th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 11468 of *LNCS*, pages 778–793. Springer, 2019.
- 7 W. Conradie, D. Della Monica, E. Muñoz-Velasco, G. Sciavicco, and I. E. Stan. Fuzzy halpern and shoham's interval temporal logics. *Fuzzy Sets and Systems*, 456:107–124, 2023.
- 8 M. Fitting. Many-valued modal logics. *Fundamenta Informaticae*, 15(3-4):235–254, 1991.
- 9 M. Fitting. Tableaus for many-valued modal logic. *Studia Logica*, 55(1):63–87, 1995.

- 10 Nikolaos Galatos, Peter Jipsen, Tomasz Kowalski, and Hiroakira Ono. *Residuated lattices: an algebraic glimpse at substructural logics*. Elsevier, 2007.
- 11 V. Goranko, A. Montanari, P. Sala, and G. Sciavicco. A general tableau method for propositional interval temporal logics: Theory and implementation. *Journal of Applied Logics*, 4(3):305–330, 2006.
- 12 V. Goranko, A. Montanari, and G. Sciavicco. Propositional interval neighborhood temporal logics. *Journal of Universal Computer Science*, 9(9):1137–1167, 2003.
- 13 V. Goranko, A. Montanari, and G. Sciavicco. A road map of interval temporal logics and duration calculi. *J. of Applied Non-Classical Logics*, 14(1–2):9–54, 2004.
- 14 J. Y. Halpern and Y. Shoham. A Propositional Modal Logic of Time Intervals. *Journal of the ACM*, 38(4):935–962, 1991.
- 15 F. Manzella, G. Pagliarini, G. Sciavicco, and I. E. Stan. Interval Temporal Random Forests with an Application to COVID-19 Diagnosis. In *Proceedings of the 28th International Symposium on Temporal Representation and Reasoning (TIME)*, volume 206 of *LIPICs*, pages 7:1–7:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- 16 F. Manzella, G. Pagliarini, G. Sciavicco, and I. E. Stan. The voice of COVID-19: Breath and cough recording classification with temporal decision trees and random forests. *Artificial Intelligence in Medicine*, 137:102486, 2023.
- 17 A. Montanari, G. Sciavicco, and N. Vitacolonna. Decidability of Interval Temporal Logics over Split-Frames via Granularity. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 2424 of *LNCs*, pages 259–270. Springer, 2002.
- 18 E. Muñoz-Velasco, M. Pelegrín-Garcí, P. Sala, G. Sciavicco, and I. E. Stan. On coarser interval temporal logics. *Artificial Intelligence*, 266:1–26, 2019.
- 19 G. Sciavicco and I. E. Stan. Knowledge extraction with interval temporal logic decision trees. In *Proceedings of the 27th International Symposium on Temporal Representation and Reasoning (TIME)*, volume 178 of *LIPICs*, pages 9:1–9:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 20 Y. Venema. A modal logic for chopping intervals. *Journal of Logic and Computation*, 1(4):453–476, 1991.


# The Calculus of Temporal Influence

Florian Bruse ✉ 

Theoretical Computer Science / Formal Methods, University of Kassel, Germany

Marit Kastaun ✉ 

Didactics of Biology, University of Kassel, Germany

Martin Lange ✉ 

Theoretical Computer Science / Formal Methods, University of Kassel, Germany

Sören Möller ✉

Theoretical Computer Science / Formal Methods, University of Kassel, Germany

---

## Abstract

We present the Calculus of Temporal Influence, a simple logical calculus that allows reasoning about the behaviour of real-valued functions over time by making assertions that bound their values or the values of their derivatives. The motivation for the design of such a proof system comes from the need to provide the background computational machinery for tools that support learning in experimental subjects in secondary-education classrooms. The end goal is a tool that allows school pupils to formalise hypotheses about phenomena in natural sciences, such that their validity with respect to some formal experiment model can be checked automatically. The Calculus of Temporal Influence provides a language for formal statements and the mechanisms for reasoning about valid logical consequences. It extends (and deviates in parts from) previous work introducing the Calculus of (Non-Temporal) Influence by integrating the ability to model temporal effects in such experiments. We show that reasoning in the calculus is sound with respect to a natural formal semantics, that logical consequence is at least semi-decidable, and that one obtains polynomial-time decidability for a natural stratification of the problem.

**2012 ACM Subject Classification** Theory of computation → Theory and algorithms for application domains; Theory of computation → Automated reasoning; Applied computing → Interactive learning environments

**Keywords and phrases** temporal reasoning, formal models, continuous functions, polynomial decidability

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.10

## 1 Introduction

Digitalisation is an ongoing process that aims at providing better solutions in all kinds of areas in industry, science, society and everyday's life. The work presented here is motivated by efforts to provide digital technology in particular learning environments, mainly secondary-education classrooms in natural sciences like biology, physics, chemistry. Digitalisation in school classrooms is a well-studied topic in educational sciences, but it often just deals with the employment of digital equipment in order to enhance learning environments, like electronic whiteboards, tablets, the internet as an online source of information, or at most the acquisition of competences to use digital (software) tools.

An aspect that is fundamental to science education is the promotion of scientific literacy which targets “*the skills to use scientific knowledge, ask questions, and draw evidence-based conclusions to understand and make decisions about the natural world and the changes humans are making to it*” [11]. The *Circle of Inquiry* asks learners to form phenomena-based hypotheses and test them by experiment [6]. The curricula of natural science subjects therefore often contain experimental studies where pupils start from a given *research question* like “*does temperature influence the growth of yeast?*” which prompts them to formulate



© Florian Bruse, Marit Kastaun, Martin Lange, and Sören Möller;  
licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 10; pp. 10:1–10:19

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 10:2 The Calculus of Temporal Influence

a hypothesis like “*yeast grows when it is warmer*” or, better, “*yeast growth is maximal at temperatures between 32° and 38°, and higher temperatures influence the growth negatively.*” This is typically followed by a validation process in which they assemble and physically conduct an experiment in order to check the validity of their hypothesis, thus learning some form of scientific reasoning.

The design and use of learning tools, in the form of specialised software for instance, can take this a step further by making use of general digital *technology* rather than just equipment, see strategies devised by ministries and departments of education.<sup>1</sup> The next step is then of course to use computational resources like algorithms, formal models, etc. in order to further advance the digitalisation process in such areas.

Technology-based learning makes it possible to create differentiated learning methods and to enable individualised learning of subject-specific competencies using feedback [9]. It comes with several advantages.

- It is *resource efficient*, making it possible to run experiments more often and at smaller running costs after investing into generic and reusable hardware.
- It is *more scalable* since digital environments can easily be multiplied, depending only on the availability of digital hardware, thus making it possible to run experiments by smaller groups of pupils.
- It makes experimental lessons *more widely applicable* as it eliminates risk in handling dangerous substances, lifts the restriction to objects of manageable size, and allows time to be scaled up or down in experiments that would otherwise take a very long or too short time to be observed in reality.
- It *enhances learning efforts* by reducing the influence of *human factors*: broadly speaking, it forces pupils to concentrate on the learning material when interacting with a learning tool, rather than to look for clues to the right answers by interacting with a teacher directly, cf. [8].

In order to be effectively used by school pupils, a digitalised learning environment for experimental lessons needs to combine different kinds of digital technology, including intuitive web interfaces, secure and stable network communications, etc. Here we are concerned with the logic that is needed in order to automatically check the correctness of a formulated hypothesis w.r.t. some background knowledge about a modelled experiment. In previous work we have introduced the *Calculus of Influence* [4] which provides

- a simple language for making statements of the form

“*variable A influences variable B [ on data range  $[x, y]$  ] [ into values in  $[x', y']$  ]*  
[ *showing monotonic / antitonic / constant / arbitrary behaviour* ]”

where variables are partially ordered entities, determined by the experiment (for instance temperature and yeast), and  $[x, y]$  and  $[x', y']$  are intervals over the reals providing a common data type for all variables;

- a formal semantics for interpreting such statements in collections of real-valued continuous partial functions over partially ordered sets of variables, thus providing a formal model of such experiments in which influence of a variable  $B$  by a variable  $A$  is modelled by the existence of a function associated with the pair  $(A, B)$ ;

---

<sup>1</sup> See e.g. <https://www.kmk.org/themen/bildung-in-der-digitalen-welt/strategie-bildung-in-der-digitalen-welt.html> (in German) for a joint strategy paper on education in a digital world of the German federal states’ ministers for education.

- a simple proof system for derivability of a hypothesis from a set of statements representing the aforementioned background knowledge about the experiment. It turned out to be sound and polynomially decidable, and it was shown to be complete for a restricted class of formal experiment models.

The framework can be used to formalise known facts about the underlying experiment through sets of statements in the above sense, called an *influence scheme*. The pupils' task is then to formulate a hypothesis in the form of another statement, detecting some form of (simple) influence between the variables involved in the scheme. The underlying tool should then verify, based on the notion of logical consequence, whether the hypothesis follows from the facts given in the scheme. If this is the case, then the pupils could be given the possibility to further interact with the tool, for instance by providing an explanation for the discovered influence (which could be checked for correctness by matching it against the found proof). If it is not the case, then pupils can be guided towards a correct hypothesis by presenting them with a counterexample, i.e. a realisation of experimental behaviour that satisfies all the facts in the scheme but not the hypothesis, derived from the failed proof attempt in the underlying tool. Note that the proof rules and strategy are not open to interaction for the pupils but are hidden away in an automated decision procedure. Likewise, the formalisation of experiments as influence schemes is not something that the pupils are tasked with; this needs to be done by someone with further expertise, or it can be automatically extracted from sets of data points.

Note that influence in the above model is always *static*: it is possible to state that a growing temperature between  $10^\circ$  and  $20^\circ$  causes increasing yeast activity between  $0.14h^{-1}$  and  $0.26h^{-1}$ , but it is *not* possible to state how yeast activity causes an increasing volume of dough, as this keeps growing at a particular rate *over time*. Note that time cannot be modelled as a variable in this scenario like temperature or yeast activity, since neither of them cause particular values of time, and time alone does not cause particular volumes. Instead, the influence between yeast activity and dough volume is *dynamic*, more specifically it is time-dependent in the sense that a particular value of yeast activity causes particular volumes over a particular time interval.

In this paper we introduce the *Calculus of Temporal Influence* in order to provide a formal model for capturing further experiments and to allow hypotheses to be made about such time-dependent influences. In Sect. 2 we first introduce a simple formal language for formalising statements about background knowledge or hypotheses, incorporating time-dependency by allowing statements that assert an influence of a variable onto another one, resp. its *derivative* (w.r.t. time). This does not only extend the calculus by incorporating time as a special entity s.t. values of any variable are always implicitly time-dependent; it also enhances its ability to make more refined statements about the nature of influence, for instance asserting that the *gradient* of some growth rate – i.e. the values of the derivative function – falls into an interval  $[l, u]$ . For instance, monotonicity then corresponds to growth rates in the range  $[0, \infty)$ . Note, however, that the implicit time-dependency makes time play a special role in this setting. Moreover, in the Calculus of Temporal Influence, there are no direct formalisations available to express that one variable influences another, but rather indirect ways tightly connected to the fact that we model the behaviour of variables over time.

We then interpret such statements in collections containing a function of type  $\mathbb{R}^{\geq 0} \rightarrow \mathbb{R}$  for each variable, modelling its behaviour over time in the experiment. As it turns out, the introduction of time-dependency alleviates the need for ordering the variables as it is done in the non-temporal Calculus of Influence, thus extending it in this respect as well.

## 10:4 The Calculus of Temporal Influence

The formal semantics immediately gives rise to the question of logical consequence, explaining when a statement, resp. hypothesis  $H$  is seen to follow from a set of statements, resp. scheme  $\mathcal{C}$ . This provides the mathematical ground for an automatic correctness check for hypotheses  $H$  when  $\mathcal{C}$  models an experiment as a collection of known facts about time-dependent variable influences in it.

In Sect. 3 we present a simple proof system as the computational ground for such automatic correctness checks. The ultimate aim is polynomial decidability since the employment in a learning tool requires such correctness checks to be carried out efficiently – usually running on mobile devices – in order to provide instantaneous feedback to pupils formulating hypotheses. Another requirement posed by this application restricts the proof rules to formalisations of intuitive reasoning principles since an effective learning tool needs to be able to provide feedback to pupils about *why* a hypothesis might be false, i.e. why it cannot be inferred from the experiment model. This leads to a natural stratification of logical consequence w.r.t. the number of “difficult” rule applications, and we obtain polynomial-time decidability for each stratum in Sect. 4. We briefly report on a prototypical implementation of the deciding algorithm, implemented in Python, in Sect. 5.

Sect. 6 concludes with remarks on further work in this direction.

### 2 Modeling Influence over Time

**Variables and Statements.** Let  $\mathcal{V} = \{a, b, \dots\}$  be a finite set of so-called *variables* (like *temperature*, some bacteria’s *growth rate*, etc.). We introduce a small language for formalising statements about the behaviour of such variables over time. Below we introduce a formal semantics based on certain partial and continuous functions of the reals. This puts some underlying assumptions about such behaviours in place which is best motivated here in order to then proceed to the definition of the formal statements.

- There is a special variable  $t$  representing time which is not included in  $\mathcal{V}$ .
- Each variable – including  $t$  – is *real-valued*. This is in accordance with what is done in many areas in natural sciences, in particular in physics with the only exception of quantum physics perhaps. For simplicity we do not state units like seconds, kilograms, cubic meters, etc.; they are assumed to be given implicitly.
- The behaviour of a variable over time is a partial function whose domain is a single interval, i.e. values of variable  $a$  may only be defined after time point  $t_1$ , but if such values exist for time points  $t_1$  and  $t_2 > t_1$ , then they also exist for all time points  $t$  with  $t_1 \leq t \leq t_2$ . It is arguable that this puts restrictions on the structure of experiments to be modelled here, but there is also a benefit to it: it means that experiment models can be obtained from measurements at discrete time points, and while we may not know the exact values in between the measurements, we can assume them to exist and even make some reasonable assumption about their values. The domain of  $t$  is always  $\mathbb{R}^{\geq 0}$ .
- The behaviour of a variable over time is a continuous function on its domain. This is in line with typical behaviour in nature which is rarely discontinuous.
- The behaviour of a variable over time is a derivable function on its domain, and the derivative in time is also continuous. Again, this is in accordance with typical behaviour in nature; it is also a necessary consequence of introducing time into the study of influence between such variables. We want to allow statements that prescribe values of a variable at certain *later* moments, possibly depending on values at a current moment. A simple way to do this is to assert that the values of the derivative of the variable’s function are bounded, and for this to be well-founded such functions need to be derivable. We identify a variable  $a$  with an underlying function  $t \rightarrow a$  and write  $\dot{a}$  for its first derivative.



Such functions of type  $t \rightarrow a$  can of course easily be shown graphically. We remark, though, that they are only used here for a formal semantics for simple statements about temporal influences. Such statements – to be defined next – are the main objects of concern, as they will be used to symbolically reason about such functions.

Intervals of reals with rational bounds are written as  $[x, y]$  for  $x, y \in \mathbb{Q} \cup \{\infty, -\infty\}$  with  $x \leq y$ . We simply write  $[5.3, \infty]$  instead of  $[5.3, \infty)$  in order to avoid unnecessary case distinctions. Intervals are only (semi-)open on an infinite interval bound. We restrict ourselves to rational bounds to keep them easily representable. A *time interval* is one in which the bounds are non-negative.

► **Definition 1.** Let  $\mathcal{V}$  be given,  $a, b \in \mathcal{V}$ ,  $[l, u]$  and  $[l', u']$  be intervals in the sense above, and  $[t_1, t_2]$  be a time interval.

- A *time-value statement* (TVS) is of the form  $t \downarrow_{[t_1, t_2], [l, u], [l', u']} a$ . It states that the value of  $a$  is within  $[l, u]$  at time point  $t_1$ , is within  $[l', u']$  at time point  $t_2$ , and is within some interval  $[l'', u'']$  at any time point  $t' \in [t_1, t_2]$  that is obtained by linearly transforming  $[l, u]$  into  $[l', u']$  along the interval  $[t_1, t_2]$ . For details, see Def. 4 below.

Thus, such a TVS intuitively states that the portion of the graph of the function  $a$  in the interval  $[t_1, t_2]$  is contained in the trapezoid that has left vertical edge  $[l, u]$  at  $t_1$  and right vertical edge  $[l', u']$  at  $t_2$ .

Note that every rectangle is a trapezoid (with equal left and right vertical edges). We will write the special case of a TVS  $t \downarrow_{[t_1, t_2], [l, u], [l, u]} a$  also simply as  $t \downarrow_{[t_1, t_2], [l, u]} a$ .

Moreover, if  $t_2 = \infty$  then the “right edge” of the trapezoid is ill-defined, and we will assume that in such a case, the trapezoid degenerates to a rectangle in this sense, i.e. in a statement  $t \downarrow_{[t_1, t_2], [l, u], [l', u']} a$  with  $t_2 = \infty$  we will always have  $[l, u] = [l', u']$  and consequentially write it in its abbreviated form. A similar case arises with degenerate trapezoids of the form  $t \downarrow_{[t_1, t_2], [-\infty, \infty], [-\infty, \infty]} a$  which we also write as  $t \downarrow_{[t_1, t_2], [-\infty, \infty]} a$ .

- A *time-derivative statement* (TDS) is of the form  $t \downarrow_{[t_1, t_2], [l, u]} \dot{a}$ . It states that the gradient of the portion of the graph of the function  $a$  is bounded from below by  $l$  and from above by  $u$ , likewise that the portion of the *derivative* of the function  $a$  in this interval is contained in the rectangle that is formed by the horizontal interval  $[t_1, t_2]$  and the vertical interval  $[l, u]$ .
- A *value-derivative statement* (VDS) is of the form  $a \downarrow_{[l, u], [t_1, t_2], [l', u']} \dot{b}$ . It states: if at some point  $t$  in time, the function  $a$  has a value within  $[l, u]$ , then in the interval  $[t + t_1, t + t_2]$  the derivative of the function  $b$  has values between  $l'$  and  $u'$ .

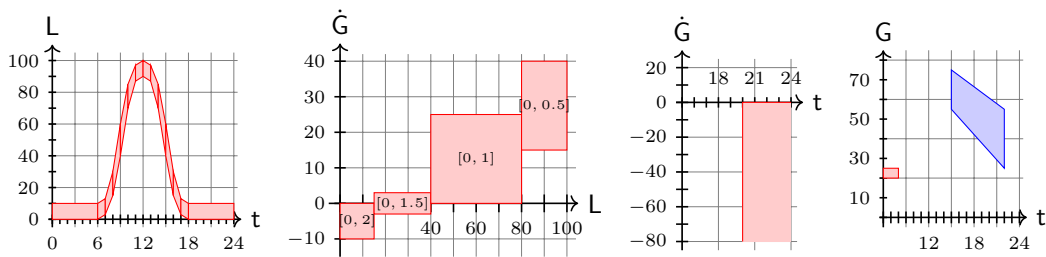
A  $\mathcal{V}$ -statement is either a TVS, a TDS or a VDS. Note that TVS and TDS assert that something holds in time, while a VDS is more reminiscent of a temporal logic formula like  $\Box(\varphi \rightarrow \Diamond\psi)$ .

Note the special role that time plays in the above definitions: the time variable  $t$  only ever appears in the context of descriptions of the behaviour of other variables over time, and the time axis only serves as the domain of functions, but never appears in the range of any function. The only exception are VDS, where time also appears in the context of a delay after which a certain statement is supposed to hold.

► **Definition 2.** A *temporal  $\mathcal{V}$ -influence scheme*  $\mathcal{C}$  is a finite set of  $\mathcal{V}$ -statements.

This introduces the main tool for formal modelling of temporal influence experiments. Intuitively, a temporal influence scheme collects abstract information about the way that the variables of an experiment behave over time and influence each other, in particular influence each other’s growth rate. We simply speak of statements and influence schemes if  $\mathcal{V}$  is clear from the context.

## 10:6 The Calculus of Temporal Influence



■ **Figure 1** Graphical presentation of some statements about temporal influence in the experiment on photosynthesis and cellular respiration, as explained in Ex. 3.

► **Example 3.** We consider a phenomenon that is routinely discussed in biology classes: photosynthesis produces glucose ( $G$ ) under the influence of light ( $L$ ), cell respiration consumes glucose. We model this as a temporal influence scheme over  $\mathcal{V} = \{L, G\}$ .<sup>2</sup>

A typical pattern of light intensity during a 24h day is shown in Fig. 1 left. It is modelled using 14 TVS like  $t \curvearrowright_{[9,10],[40,60],[75,85]} L$  that can be visualised as trapezoids in the plane for influences of type  $t \rightarrow L$ . The unit on the  $t$ -axis is hours, the one on the  $L$ -axis is percent.

Photosynthesis causes the production of glucose under light. On the other hand, cellular respiration consumes glucose at a fairly standard rate (which we assume here to be constant). We model this phenomenon by dividing the range of light intensity into four categories, leading to four VDS as follows.

- In lowest light or darkness (say 0–15%), glucose is solely consumed by cellular respiration and therefore decreases in overall availability:  $L \curvearrowright_{[0,15],[0,2],[-10,0]} \dot{G}$  actually states that the gradient of light over the next time interval of 2h lies within the range of staying constant to dropping by 10, say  $\mu g/h$ .
- At a lower light intensity (15–40%), the production and consumption of glucose in the processes of photosynthesis and cellular respiration balance each other out, and there is at most a small increase or decrease:  $L \curvearrowright_{[15,40],[0,1.5],[-3,3]} \dot{G}$ . Here we choose a shorter time interval – and then even shorter below – as bright light may be more prone to sudden changes as low light.
- At a higher intensity, production wins over consumption, and the growth rate is positive:  $L \curvearrowright_{[40,80],[0,1],[0,20]} \dot{G}$ .
- At highest intensity, the growth rate of glucose is even higher:  $L \curvearrowright_{[80,100],[0,0.5],[15,40]} \dot{G}$ .

Such VDS are less easy to depict graphically as they include *three* independent entities: a value range of  $L$ , a time interval, and a range for the gradient that values of  $G$  can follow over time. We introduce the graphical notation shown in Fig. 1 second to left, presenting those four statements from above.

A TDS seems to be of little use here in order to model the photosynthesis experiment; they are included in general because of common reasoning principles: if time influences light intensity, and light intensity influences the way that glucose levels change, then time indirectly influences the way that glucose levels change. Hence, a statement like “*glucose levels are falling after 20h*” – formalised as the TDS  $t \curvearrowright_{[20,\infty],[-\infty,0]} \dot{G}$  and shown in Fig. 1 second to right, actually saying that the levels are not rising instead of falling – may or may not be a valid conclusion from the TVS and VDS discussed beforehand.

<sup>2</sup> Clearly, such metabolism processes could be modelled at arbitrarily higher levels of detail, taking into account all sorts of involved biochemical agents. The view of the interaction focussing on these three agents is consistent with what can be done in biology school classes for instance. It already leaves out further factors like carbon dioxide, water and oxygen, which can either be seen as by-products or as constantly available.

At last, if time exerts an influence over the derivative of  $G$ , then it must also influence  $G$ . I.e. the picture of what kinds of influence between variables at certain value ranges or at certain time intervals exist in this model or can be derived from known facts in there, will not be complete without the possibility to include TVS for  $G$ . One such statement is shown in Fig. 1 right, in blue to distinguish it from the others as a hypothesis, i.e. a statement with which we associate the question “*Is this a correct logical consequence from the statements in the influence scheme?*” It predicts a glucose concentration whose expected value is falling slightly in the hours between 15 and 22.

It should be clear that in order to be able to answer this question we would need to fix initial values for  $G$  which can be done using another TVS like  $t \xrightarrow{[0,2],[20,25]} G$  shown in Fig. 1 right in red, and fixing a range of initial glucose concentration.

The problem of deciding whether a given hypothesis (in the form of a statement  $H$ ) is *correct* w.r.t. some experiment is modelled here as the question whether  $H$  follows logically from the temporal influence scheme  $\mathcal{C}$  modelling knowledge about the way that the variables of the experiment behave over time and possibly influence each other. For this to be well-defined we introduce a formal semantics for statements and schemes next.

**Semantics of Influence Schemes.** An *influence* is a function  $f: \mathbb{R}^{\geq 0} \rightarrow \mathbb{R}$  s.t. its domain  $\text{dom}(f)$  is a non-unit interval as per above, and  $f$  is derivable on its entire domain. An influence is used to specify the behaviour of a variable  $a \in \mathcal{V}$  over time. We write  $f(t) = \perp$  if  $t \notin \text{dom}(f)$ . Note that necessarily also  $t \notin \text{dom}(\dot{f})$ .

► **Definition 4.** Let  $\mathcal{V}$  be a set of variables. A  $\mathcal{V}$ -*influence experiment* is a collection  $\mathcal{F}$  of influences containing exactly one influence  $\mathcal{F}_a$  for each variable  $a \in \mathcal{V}$ .

■  $\mathcal{F}$  satisfies the TVS  $S = t \xrightarrow{[t_1, t_2], [l, u], [l', u']} a$ , written  $\mathcal{F} \models S$ , if

$$l + (l' - l) \cdot \frac{t - t_1}{t_2 - t_1} \leq \mathcal{F}_a(t) \leq u + (u' - u) \cdot \frac{t - t_1}{t_2 - t_1} \quad (1)$$

for all  $t \in [t_1, t_2]$ . In the special case where  $t_2 = \infty$  (and  $[l', u'] = [l, u]$  by convention) this is to be interpreted as  $t \in [l, u]$  for all  $t \geq t_1$ .

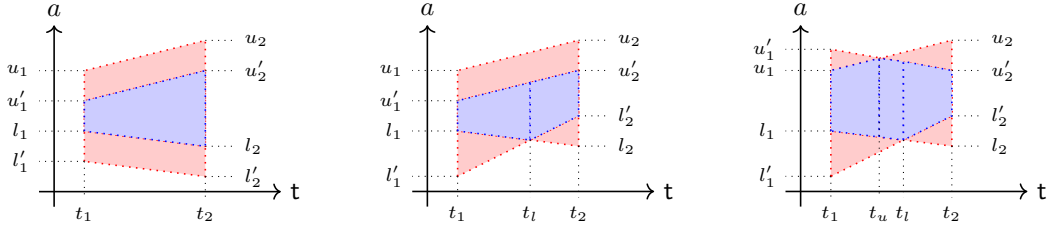
■  $\mathcal{F}$  satisfies the TDS  $S = t \xrightarrow{[t_1, t_2], [l, u]} \dot{a}$ , also written  $\mathcal{F} \models S$ , if  $l \leq \dot{\mathcal{F}}_a(t) \leq u$  for all  $t \in [t_1, t_2]$ .

■  $\mathcal{F}$  satisfies the VDS  $S = a \xrightarrow{[l, u], [t_1, t_2], [l', u']} \dot{b}$ , also written  $\mathcal{F} \models S$ , if  $l' \leq \dot{\mathcal{F}}_b(z) \leq u'$  for all  $z$  such that there is  $t$  with  $\mathcal{F}_a(t) \in [l, u]$  and  $z \in [t + t_1, t + t_2]$ .

$\mathcal{F}$  satisfies a  $\mathcal{V}$ -influence scheme  $\mathcal{C}$ , written  $\mathcal{F} \models \mathcal{C}$ , if  $\mathcal{F} \models S$  for all  $S \in \mathcal{C}$ .

An influence scheme  $\mathcal{C}$  is called *satisfiable* if there is some  $\mathcal{F}$  such that  $\mathcal{F} \models \mathcal{C}$ . We say that a statement  $S$  follows from an influence scheme  $\mathcal{C}$ , written  $\mathcal{C} \models S$ , if  $\mathcal{F} \models S$  for all  $\mathcal{F}$  such that  $\mathcal{F} \models \mathcal{C}$ . Hence, a temporal influence experiment models concrete behaviour in terms of particular, real-valued functions; a temporal influence scheme models this abstractly by collecting (a finite amount) of information about bounds on the expected temporal behaviour of variables and on the influence they exert on each other leading to such temporal behaviour. Likewise, a temporal influence scheme can be seen as a finite representation of a (typically infinite and even uncountable) number of influence experiments with behaviours within the bounds included in the statements of the scheme.

We also obtain a notion of logical equivalence for influence schemes: we say that  $\mathcal{C}$  and  $\mathcal{C}'$  are *equivalent*, written  $\mathcal{C} \equiv \mathcal{C}'$  iff for all influence experiments  $\mathcal{F}$  we have  $\mathcal{F} \models \mathcal{C}$  iff  $\mathcal{F} \models \mathcal{C}'$ . Note that in this case, for all statements  $S$  we have  $\mathcal{C} \models S$  iff  $\mathcal{C}' \models S$ . Hence, equivalent models can be seen as forming abstract representations of the same experimental setup that may differ syntactically.



■ **Figure 2** Intersecting two trapezoids (in red) over the same time interval into one, two or three adjacent trapezoids (in blue).

### 3 The Calculus of Temporal Influence

**Geometric Considerations.** We consider several subproblems – relaxation of a TVS, intersection and join of two TVS, and the effect of derivatives on TVS – arising with the problem of deciding (via a sound proof system) whether a given hypothesis follows from a given scheme, e.g. when two TVS make assertions about the same interval in time. Given the geometric interpretation of statements introduced in Ex. 3, the considerations carried out here can be seen as simple geometric principles.

Let  $S = \mathbf{t} \langle [t_1, t_2], [l_1, u_1], [l_2, u_2] \rangle_a$  be a TVS. By Def. 4,  $S$  implies that for all  $t \in [t_1, t_2]$ , the function  $\mathcal{F}_a : \mathbf{t} \rightarrow a$  satisfies the inequalities in Eq. 1. Clearly,  $S$  also implies conditions on subintervals of  $[t_1, t_2]$ , and less restrictive conditions w.r.t. the vertical extent of the two edges of the trapezoid implied by it. However, it is generally not correct to simply make the vertical intervals larger and to shorten the time interval. Instead, relaxation is formalised as follows.

► **Definition 5.** Let  $S$  be as above. The set of *relaxed TVS* w.r.t.  $S$ , written  $\text{relax}(S)$ , is the set of all TVS  $\mathbf{t} \langle [t'_1, t'_2], [l'_1, u'_1], [l'_2, u'_2] \rangle_a$  s.t.

- $t_1 \leq t'_1 < t'_2 \leq t_2$ , and
- $l'_1 \leq l_1 + (l_2 - l_1) \cdot \frac{t'_1 - t_1}{t_2 - t_1}$  and  $u'_1 \geq u_1 + (u_2 - u_1) \cdot \frac{t'_1 - t_1}{t_2 - t_1}$ , and
- $l'_2 \leq l_1 + (l_2 - l_1) \cdot \frac{t'_2 - t_1}{t_2 - t_1}$  and  $u'_2 \geq u_1 + (u_2 - u_1) \cdot \frac{t'_2 - t_1}{t_2 - t_1}$ .

Let  $S = \mathbf{t} \langle [t_1, t_2], [l_1, u_1], [l_2, u_2] \rangle_a$  and  $S' = \mathbf{t} \langle [t_1, t_2], [l'_1, u'_1], [l'_2, u'_2] \rangle_a$  be two TVS over the same time interval. Note that both induce a trapezoid in the graphical representation of the function  $a$ , and the horizontal range is  $[t_1, t_2]$  for both of these trapezoids. Clearly, if e.g.  $l_1 \leq l'_1$  and  $l_2 \leq l'_2$ , then the second lower bound implies the first, and similarly for the upper bounds. However, if  $l_1 < l'_1$  and  $l_2 > l'_2$ , then there is a unique inner point  $t_i \in [t_1, t_2]$  at which the two lower sides of the trapezoids intersect, namely such that  $l_1 + (l_2 - l_1) \cdot \frac{t_i - t_1}{t_2 - t_1} = l'_1 + (l'_2 - l'_1) \cdot \frac{t_i - t_1}{t_2 - t_1}$  and  $l_1 + (l_2 - l_1) \cdot \frac{t - t_1}{t_2 - t_1} < l'_1 + (l'_2 - l'_1) \cdot \frac{t - t_1}{t_2 - t_1}$  for all  $t \in [t_1, t_i)$ , and  $l_1 + (l_2 - l_1) \cdot \frac{t - t_1}{t_2 - t_1} > l'_1 + (l'_2 - l'_1) \cdot \frac{t - t_1}{t_2 - t_1}$  for all  $t \in (t_i, t_2]$ . Hence, splitting  $[t_1, t_2]$  into  $[t_1, t_i]$  and  $[t_i, t_2]$  would make it possible to have each lower bound be represented by a straight line in this case, the case where  $l_1 > l'_1$  and  $l_2 > l'_2$  works similarly. Moreover, this reasoning also applies to the upper bounds.

It should be clear that the bounds imposed by two trapezoids over the same time interval  $[t_1, t_2]$  can equally be represented by one (over  $[t_1, t_2]$ ), two (over  $[t_1, t_i]$  and  $[t_i, t_2]$ ) or three trapezoids (over  $[t_1, t_i]$ ,  $[t_i, t_u]$  and  $[t_u, t_2]$ , or  $[t_1, t_u]$ ,  $[t_u, t_i]$  and  $[t_i, t_2]$ , for some  $t_i, t_u$  determined by the two upper and lower sides of the trapezoids at hand and the order of their intersection points). A technical definition of the TVS  $\text{isect}_i([t_1, t_2], [l_1, u_1], [l_2, u_2], [l'_1, u'_1], [l'_2, u'_2])$

for  $i \in \{1, 2, 3\}$  is given in Appendix A. For the following it suffices to know that these terms refer to the first, second, or third (if they exist) trapezoid resulting from the intersection of  $S$  and  $S'$ , as shown in Fig. 2.

Another important reasoning principle is derived from the converse operation of joining two adjacent TVS  $S$  and  $S'$ . This is possible if the two trapezoids defined by them fit into the trapezoid defined by taking the left vertical interval of  $S$  and the right vertical interval of  $S'$  at their respective moments in time as new trapezoid-defining edges.

► **Definition 6.** Let  $S = \mathfrak{t}_{\langle [t_1, t_2], [l_1, u_1], [l_2, u_2] \rangle} a$  and  $S' = \mathfrak{t}_{\langle [t_2, t_3], [l'_1, u'_1], [l'_2, u'_2] \rangle} a$  be TVS. Let  $l = l_1 + (l'_2 - l_1) \cdot \frac{t_2 - t_1}{t'_2 - t_1}$  and let  $u = u_1 + (u'_2 - u_1) \cdot \frac{t_2 - t_1}{t'_2 - t_1}$ . If  $u \geq u_2, u \geq u'_1$  and  $l \leq l_2, l \leq l'_1$  then set  $\text{join}(t_1, t_2, t_3, [l_1, u_1], [l_2, u_2], [l'_1, u'_1], [l'_2, u'_2]) = \mathfrak{t}_{\langle [t_1, t_2], [l_1, u_1], [l'_2, u'_2] \rangle} a$ , otherwise set  $\text{join}(t_1, t_2, t_3, [l_1, u_1], [l_2, u_2], [l'_1, u'_1], [l'_2, u'_2]) = S$  to make it always defined.

Let  $S_1 = \mathfrak{t}_{\langle [t_1, t_2], [l_1, u_1], [l_2, u_2] \rangle} a$  be a TVS and  $\mathcal{F} \models S_1$ , i.e.  $\mathcal{F}_a: \mathfrak{t} \rightarrow a$  satisfies  $\mathcal{F}_a(t_1) \in [l_1, u_1]$  and  $\mathcal{F}_a(t_2) \in [l_2, u_2]$ . Let furthermore  $S_2 = \mathfrak{t}_{\langle [t'_1, t_1], [l_3, u_3] \rangle} \dot{a}$  and  $S_3 = \mathfrak{t}_{\langle [t_2, t'_2], [l_4, u_4] \rangle} \dot{a}$  be TDS. They imply  $\dot{\mathcal{F}}_a(t) \in [l_3, u_3]$  for all  $t \in [t'_1, t_1]$  and that  $\dot{\mathcal{F}}_a(t) \in [l_4, u_4]$  for all  $t \in [t_2, t'_2]$ . This entails  $l_1 + (t_1 - t) \cdot u_3 \leq \mathcal{F}_a(t) \leq u_1 - (t_1 - t) \cdot l_3$  for all  $t \in [t'_1, t_1]$  and  $l_2 + (t - t_2) \cdot l_4 \leq \mathcal{F}_a(t) \leq u_2 + (t - t_2) \cdot u_4$  for all  $t \in [t_2, t'_2]$ .

► **Definition 7.** Let  $S_1, S_2$  and  $S_3$  be as above. Define  $\text{l derivative}([t_1, t_2], [l_1, u_1], [l_3, l_4])$  as the TVS  $\mathfrak{t}_{\langle [t'_1, t_1], [l_1 + (t_1 - t'_1) \cdot u_3, u_1 + (t_1 - t'_1) \cdot l_3], [l_1, u_1] \rangle} a$  and  $\text{r derivative}([t_1, t_2], [l_2, u_2], [l_4, u_4])$  as the TVS  $\mathfrak{t}_{\langle [t_2, t'_2], [l_2, u_2], [l_2 + (t'_2 - t_2) \cdot l_4, u_2 + (t'_2 - t_2) \cdot u_4] \rangle} a$ .

Note the inverted role of  $u_3$  and  $l_3$  in  $\text{l derivative}$ , which is due to the reasoning from right to left that happens here.

**The Calculus.** We say that a  $\mathcal{V}$ -statement  $S$  is *provable* in the Calculus of Temporal Influence (CTI) w.r.t. an influence scheme  $\mathcal{C}$ , written  $\mathcal{C} \vdash S$ , if there is a finite proof for  $S$  in the proof system whose rules are shown in Fig. 3. We say that  $\mathcal{C}$  is *consistent* in CTI if there are no statements  $S, S'$  derivable from  $\mathcal{C}$  s.t. rules ( $\mathbf{S}_{\text{TVS}}$ ) or ( $\mathbf{S}_{\text{TDS}}$ ), when applied to  $S, S'$  as premises, would yield an ill-defined TVS or TDS, i.e. one where  $u < l$  in a vertical interval  $[l, u]$ . We briefly explain each rule and argue why it is sound.

- (**F**): This rule stipulates that any statement  $S$  that is already contained in  $\mathcal{C}$  is derivable. This is evidently sound.
- (**G<sub>TVS</sub>**) & (**G<sub>TDS</sub>**): These rules close gaps in the domain of a function or its derivative by asserting that the function, resp. its derivative be defined in the gap. This is sound due to the stipulation that functions and their derivatives are defined on intervals.
- (**W<sub>TVS</sub>**): This weakening rule stipulates the following kind of reasoning. Suppose we know that in the interval  $I_1$ , the function  $a$  is contained in the trapezoid generated by  $I_2$  and  $I_3$ . Then in any subinterval of  $I_1$  it is contained in any trapezoid that is larger in the vertical dimension, using the reasoning outlined before Def. 5. Note that this rule can be used to split a TVS in half on the time axis. The rule is easily seen to be sound already due to geometric reasoning.
- (**W<sub>TDS</sub>**): This weakening rule stipulates that if  $\dot{a}$  is contained in some rectangle, it must be contained in any rectangle that is smaller on the horizontal axis or larger on the vertical axis. Soundness is also by geometric reasoning.
- (**J<sub>TVS</sub>**) & (**J<sub>TDS</sub>**): These rules join two TVS or two TDS via the machinery introduced before Def. 6, resp. simple geometric reasoning. Soundness is by invoking weakening first to adjust vertical extents and then by straightforward joining of two trapezoids, resp. two rectangles.

## 10:10 The Calculus of Temporal Influence

$$\begin{array}{l}
\text{(F)} \quad \frac{}{S} \text{ if } S \in \mathcal{C} \qquad \text{(G}_{\text{TVS}}) \quad \frac{\mathfrak{t}_{\llbracket [t_1, t_2], I_2, I_3 \rrbracket} a \quad \mathfrak{t}_{\llbracket [t'_1, t'_2], I'_2, I'_3 \rrbracket} a}{\mathfrak{t}_{\llbracket [t_2, t'_1], [-\infty, \infty] \rrbracket} a} \text{ if } t_2 < t'_1 \\
\text{(G}_{\text{TDS}}) \quad \frac{\mathfrak{t}_{\llbracket [t_1, t_2], I_2 \rrbracket} \dot{a} \quad \mathfrak{t}_{\llbracket [t'_1, t'_2], I'_2 \rrbracket} \dot{a}}{\mathfrak{t}_{\llbracket [t_2, t'_1], [-\infty, \infty] \rrbracket} \dot{a}} \text{ if } t_2 < t'_1 \qquad \text{(W}_{\text{TVS}}) \quad \frac{S}{S'} \text{ if } S' \in \text{relax}(S) \\
\text{(W}_{\text{TDS}}) \quad \frac{\mathfrak{t}_{\llbracket I_1, I_2 \rrbracket} \dot{a}}{\mathfrak{t}_{\llbracket I'_1, I'_2 \rrbracket} \dot{a}} \text{ if } I'_1 \subseteq I_1, I_2 \subseteq I'_2 \qquad \text{(J}_{\text{TVS}}) \quad \frac{\mathfrak{t}_{\llbracket [t_1, t_2], I_2, I_3 \rrbracket} a \quad \mathfrak{t}_{\llbracket [t_2, t_3], I'_2, I'_3 \rrbracket} a}{\text{join}(t_1, t_2, t_3, I_2, I_3, I'_2, I'_3)} \\
\text{(J}_{\text{TDS}}) \quad \frac{\mathfrak{t}_{\llbracket [t_1, t_2], I_2 \rrbracket} \dot{a} \quad \mathfrak{t}_{\llbracket [t_2, t_3], I'_2 \rrbracket} \dot{a}}{\mathfrak{t}_{\llbracket [t_1, t_3], I_2 \cup I_3 \rrbracket} \dot{a}} \qquad \text{(VD)} \quad \frac{\mathfrak{t}_{\llbracket I_1, I_2, I_3 \rrbracket} a}{\mathfrak{t}_{\llbracket I_1, [-\infty, \infty] \rrbracket} a} \\
\text{(DV)} \quad \frac{\mathfrak{t}_{\llbracket I_1, I_2 \rrbracket} \dot{a}}{\mathfrak{t}_{\llbracket I_1, [-\infty, \infty] \rrbracket} a} \qquad \text{(S}_{\text{TVS}}) \quad \frac{\mathfrak{t}_{\llbracket I_1, I_2, I_3 \rrbracket} a \quad \mathfrak{t}_{\llbracket I_1, I'_2, I'_3 \rrbracket} a}{\text{isect}_i(I_1, I_2, I_3, I'_2, I'_3)} \text{ if } i \in \{1, 2, 3\} \\
\text{(S}_{\text{TDS}}) \quad \frac{\mathfrak{t}_{\llbracket I_1, I_2 \rrbracket} \dot{a} \quad \mathfrak{t}_{\llbracket I'_1, I'_2 \rrbracket} \dot{a}}{\mathfrak{t}_{\llbracket I_1 \cap I'_1, I_2 \cap I'_2 \rrbracket} \dot{a}} \qquad \text{(Der)} \quad \frac{\mathfrak{t}_{\llbracket [t_1, t_2], I_1 \rrbracket} a \quad a \llbracket [t'_1, t'_2], I_2 \rrbracket b}{\mathfrak{t}_{\llbracket [t_1 + t'_1, t_2 + t'_2], I_2 \rrbracket} b} \\
\text{(CDL)} \quad \frac{\mathfrak{t}_{\llbracket [t_2, t_3], I_1, I_2 \rrbracket} a \quad \mathfrak{t}_{\llbracket [t_1, t_2], I_3 \rrbracket} \dot{a}}{\text{l derivative}([t_1, t_2], I_1, I_3)} \qquad \text{(CDR)} \quad \frac{\mathfrak{t}_{\llbracket [t_1, t_2], I_1, I_2 \rrbracket} a \quad \mathfrak{t}_{\llbracket [t_2, t_3], I_3 \rrbracket} \dot{a}}{\text{r derivative}([t_2, t_3], I_2, I_3)}
\end{array}$$

■ **Figure 3** Proof rules for correctness of a statement w.r.t. an influence scheme  $\mathcal{C}$ .

- (VD) & (DV): These rules assert that a function is defined on some interval if its derivative is defined there, and vice versa. Soundness is by the definition of an influence.
- (S<sub>TVS</sub>) & (S<sub>TDS</sub>): The former rule can be used to derive up to three distinct TVS from two TVS defined on the same time interval, via the considerations outlined after Def. 5. Soundness follows from the discussion there; the technical definition is in Def. 13 in Appendix A. The latter rule is similar and for TDS instead, and, hence much simpler and readily seen to be sound.
- (Der): This core rule of the calculus can be thought of as a form of modus ponens. The right premise is a VDS, which always has the form of an implication that, if the graph of the function  $a$  is contained in the vertical interval  $[x, y]$  at some point, then the derivative of the function  $b$  is contained in the interval  $I_2$  during some interval dictated by  $[t'_1, t'_2]$ . The left premise of this rule is the assertion that  $a$  has the property demanded in the right premise of the rule, whence the conclusion of the right premise must hold, i.e. the derivative of  $b$  has the given properties in some interval. Soundness is due to the semantics of a VDS, applied to the whole interval  $[t_1, t_2]$ .
- (CDL) & (CDR): These rules combine the information that the graph of  $a$  is contained in some trapezoid (left premise) together with assertions on the derivative of  $a$  to the left (CDL) or right (CDR), in order to derive new trapezoids left, resp. right of the trapezoid given by the left premise. Soundness is due to the discussion before Def. 7.

Note that the left premise of rule (Der) is a rectangle-shaped TVS. It would be possible to formulate a similar rule more specialised to general trapezoids, but this would make it even more unwieldy, whence we refrain from this.

The following is a consequence of soundness of all the rules. It also entails that an inconsistent influence scheme is not satisfiable.

► **Theorem 8** (Soundness). *For any temporal influence scheme  $\mathcal{C}$  and any statement  $S$  we have: if  $\mathcal{C} \vdash S$  then  $\mathcal{C} \models S$ .*

#### 4 Decidability For Fixed Strata in the Consequence Relation

Let  $\mathcal{C}$  be an influence scheme and let  $S$  be a statement such that  $\mathcal{C} \vdash S$ . We say that a proof of  $S$  from  $\mathcal{C}$  has (Der)-depth  $k$  if, on any path from  $S$  to an axiom, there are at most  $k$  invocations of rule (Der). We say that  $S$  has (Der)-depth  $k$  w.r.t.  $\mathcal{C}$  if there is a proof of  $\mathcal{C} \models S$  with (Der)-depth at most  $k$ . Let  $\mathcal{C}$  be consistent. We write  $\text{CTI}[k](\mathcal{C})$  for the set of statements  $S$  such that  $\mathcal{C} \vdash S$  and  $S$  has (Der)-depth  $k$  w.r.t.  $\mathcal{C}$ . If  $\mathcal{C}$  is clear from context, we simply speak of (Der)-depth. We single out rule (Der) here since it is conceptually the most complicated, in particular for a pupil in secondary education. The stratification induced by the above definition also is very natural, as witnessed by Lemma 11.

Of course, if  $\mathcal{C}$  is not consistent,  $\mathcal{C}$  is not satisfiable, whence  $\mathcal{C} \models S$  trivially, but the proof of any statement witnessing inconsistency, for example by containing empty vertical intervals, might have (Der)-depth  $k'$  with  $k'$  being much greater than  $k$ . This is the reason for the restriction to consistent influence schemes. We now define two notions of normalisation.

► **Definition 9.** Let  $\mathcal{C}$  be a consistent  $\mathcal{V}$ -influence scheme, let  $a \in \mathcal{V}$ , let  $k \in \mathbb{N}$  and let  $\mathcal{S} = \{S_1, \dots, S_n\}$  be a set of TDS of (Der)-depth  $k$  or less, where  $S_j = \mathfrak{t}_{\langle [t_1^j, t_2^j], [l^j, u^j] \rangle} \dot{a}$ . We call  $\mathcal{S}$  *k-normalised* if the following hold:

- $\mathcal{S}$  is *separated*, i.e. for all  $j < k$ , we have  $t_2^j = t_1^{j+1}$ .
- $\mathcal{S}$  is *minimal*, i.e. for all TDS  $\mathfrak{t}_{\langle [t_1, t_2], [l, u] \rangle} \dot{a}$  of (Der)-depth (w.r.t.  $\mathcal{C}$ ) of  $k$  or less, if  $t_1 < t_2$  and there are  $j \leq j'$  such that  $t_1^j \leq t_1 < t_2 \leq t_2^{j'}$ , then for all  $j''$  s.t.  $j \leq j'' \leq j'$ , we have  $l^{j''} \geq l$  and  $u^{j''} \leq u$ .
- $\mathcal{S}$  is *representative*, i.e. for all TDS  $S = \mathfrak{t}_{\langle I_1, I_2 \rangle} \dot{a}$  of (Der)-depth (w.r.t.  $\mathcal{C}$ ) of  $k$  or less,  $S$  is derivable from  $\mathcal{S}$  via applications of rules ( $\text{W}_{\text{TDS}}$ ) and ( $\text{J}_{\text{TDS}}$ ) alone.

We call a set of TDS *k-normalised* if it is *k-normalised* for each individual variable.

The intuition here is that being separated removes temporal overlap between the individual TDS and that the union of their temporal domains forms an interval. Minimality means that  $\mathcal{S}$  cannot be strengthened any further without invocations of rule (Der), and being representative means that  $\mathcal{S}$  is a complete representation of  $\text{CTI}[k](\mathcal{C})$  w.r.t. the derivative of the function  $a$  in the sense that any TDS from the latter set can be derived from the former by very simple rules. In fact, derivability of such a TDS  $S$  can be decided by visual inspection: we have  $\mathcal{S} \vdash S$  iff the corridor defined by  $\mathcal{S}$  stretches at least over  $S$ 's time interval and is entirely surrounded by the rectangle defined by  $S$  there.

► **Definition 10.** Let  $\mathcal{C}$  be a consistent  $\mathcal{V}$ -influence scheme, let  $a \in \mathcal{V}$ , let  $k \in \mathbb{N}$  and let  $\mathcal{S} = \{S_1, \dots, S_n\}$  be a set of TVS of (Der)-depth  $k$  or less, where  $S_j = \mathfrak{t}_{\langle [t_1^j, t_2^j], [l^j, u^j], [l'^j, u'^j] \rangle} a$ . We call  $\mathcal{S}$  *k-pre-normalised* if the following hold:

- $\mathcal{S}$  is *separated*, i.e. for all  $j < k$ , we have  $t_2^j = t_1^{j+1}$ .
- $\mathcal{S}$  is *minimal*, i.e. for all TVS  $\mathfrak{t}_{\langle [t_1, t_2], [l', u'], [l'', u''] \rangle} a$  of (Der)-depth (w.r.t.  $\mathcal{C}$ ) of  $k$  or less, if  $t_1 < t_2$  and there are  $j \leq j'$  such that  $t_1^j \leq t_1 < t_2 \leq t_2^{j'}$ , then for all  $j''$  s.t.  $j \leq j'' \leq j'$ , and for all  $t \in [t_1^{j''}, t_2^{j''}]$  we have that

$$l + (l - l') \cdot t' \leq l^{j''} + (l'^{j''} - l^{j''}) \cdot t'' \leq u^{j''} + (u'^{j''} - u^{j''}) \cdot t'' \leq u + (u - u') \cdot t'$$



## 10:12 The Calculus of Temporal Influence

- where  $t' = \frac{t-t_1}{t_2-t_1}$  and  $t'' = \frac{t-t_1''}{t_2''-t_1''}$ .
- $\mathcal{S}$  is *representative*, i.e. for all TVS  $S = \mathfrak{t} \langle I_1, I_2, I_3 \rangle a$  of (Der)-depth (w.r.t.  $\mathcal{C}$ ) of  $k$  or less,  $S$  is derivable from  $\mathcal{S}$  via applications of rules ( $\mathsf{W}_{\text{TVS}}$ ) and ( $\mathsf{J}_{\text{TVS}}$ ) alone.
- Moreover, we call  $\mathcal{S}$  *k-normalised* if it is *k-pre-normalised* and the following holds:
- $\mathcal{S}$  is *derivative-reduced*, i.e. for all TDS  $\mathfrak{t} \langle I, [l, u] \rangle a$  of (Der)-depth (w.r.t.  $\mathcal{C}$ ) of  $k$  or less, and for all  $j \leq k$ , the following holds: if  $I \cap [t_1^j, t_2^j] \neq \emptyset$ , then (I) if  $l^j \neq -\infty$  and  $u^j \neq -\infty$ , then  $l \leq \frac{l^j - u^j}{t_2^j - t_1^j} \leq u$ , and (II) if  $u^j \neq \infty$  and  $l^j \neq \infty$ , then  $l \leq \frac{u^j - l^j}{t_2^j - t_1^j} \leq u$ .
  - $\mathcal{S}$  is (Der)-*ready*, i.e. if  $S$  can be obtained from some  $S_j$  via ( $\mathsf{W}_{\text{TVS}}$ ), and there is a VDS  $S'$  such that  $S$  and  $S'$  are possible premises for an application of rule (Der), then  $S$  and  $S_j$  agree on their time interval.

We call a set of TVS *k-normalised* if it is *k-normalised* w.r.t. each variable.

The intuition for the first three items is the same as in Def. 9, but the formulation of minimality is more complicated due to the more complex geometry involved. Minimality in geometric terms requires that any of the  $S_j$  that overlaps with any TVS  $S$  of (Der)-depth  $k$  or less must be contained in the trapezoid defined by  $S$  for the time interval of the overlap. Hence,  $S_j$  (or any TVS derived by splitting it) could not be strengthened via ( $\mathsf{S}_{\text{TVS}}$ ) using  $S$ .

The intuition for being derivative-reduced is the following: when we know that the graph of function  $a$  must pass through a given trapezoid that is bounded from e.g. above, and we know that the derivative of that function is bounded from above by  $u$  and from below by  $l$ , then unless the slope of the upper edge of the trapezoid is between  $l$  and  $u$ , there will be parts of the trapezoid that  $a$  cannot pass through without violating the bounds  $l$  and  $u$  on its derivative, and this fact can be derived via rules (CDL) or (CDR) after potentially splitting the TDS in question via ( $\mathsf{W}_{\text{TDS}}$ ), followed by ( $\mathsf{S}_{\text{TVS}}$ ). Similar reasoning applies for lower bounds or if the derivative is only properly bounded from one side. Note that for two neighbouring TVS such that the derivative in question is bounded from both above and below on the point in time of their intersection, being derivative-reduced entails that the upper and lower bounds of the TVS match due to continuity.

Finally, the point of being (Der)-ready is to ensure that a TVS can serve as the left premise of a given VDS either in its horizontal extent, or not at all.

► **Lemma 11.** *Let  $\mathcal{C}$  be a consistent  $\mathcal{V}$ -influence scheme, let  $a \in \mathcal{V}$ , let  $k \in \mathbb{N}$ . Let  $\mathcal{S}$  be a finite set of TVS and let  $\mathcal{S}'$  be a finite set of TDS s.t. all TVS and TDS of (Der)-depth (w.r.t.  $\mathcal{C}$ )  $k$  or less can be derived from  $\mathcal{S} \cup \mathcal{S}'$  without using rule (Der). Then there are  $\mathcal{S}_{\text{norm}}$  and  $\mathcal{S}'_{\text{norm}}$  that both are *k-normalised* (and therefore representative, ensuring equivalence to  $\mathcal{S}$  and  $\mathcal{S}'$  w.r.t.  $\vdash$ ). Moreover,  $\mathcal{S}_{\text{norm}}$  and  $\mathcal{S}'_{\text{norm}}$  can be computed from  $\mathcal{S}$  and  $\mathcal{S}'$  in polynomial time.*

The proof has been moved to Appendix B for space considerations. This lemma yields a polynomial-time decision procedure for  $\text{CTI}[k](\mathcal{C})$ :

► **Theorem 12.** *Let  $k$  be fixed. Let  $\mathcal{C}$  be a consistent  $\mathcal{V}$ -influence scheme and let  $S$  be  $\mathcal{V}$ -statement. It is decidable in polynomial time whether  $S \in \text{CTI}[k](\mathcal{C})$ .*

**Proof.** If  $S$  is a VDS then there is nothing to prove since no proof rule has a VDS as its conclusion. So assume that  $S$  is a TVS or a TDS.  $\mathcal{C}$  trivially contains subsets  $\mathcal{S}, \mathcal{S}'$  of TVS, resp. TDS that satisfy the premises of Lemma 11 for  $k = 0$ . Let  $\mathcal{S}_{\text{norm}}^0, \mathcal{S}'_{\text{norm}}^0$  be the 0-normalised, polynomially-sized sets from said lemma. If  $k = 0$  we are done.

Now assume that we have obtained *j-normalised* polynomially-sized sets  $\mathcal{S}_{\text{norm}}^j$  and  $\mathcal{S}'_{\text{norm}}^j$  for  $j \geq 0$ . We obtain sets  $\mathcal{S}^{j+1} \supseteq \mathcal{S}_{\text{norm}}^j$  and  $\mathcal{S}'^{j+1} \supseteq \mathcal{S}'_{\text{norm}}^j$  by extending  $\mathcal{S}_{\text{norm}}^j$  and  $\mathcal{S}'_{\text{norm}}^j$  the following way: (I) First we apply rule ( $\mathsf{W}_{\text{TVS}}$ ) to sets in  $\mathcal{S}_{\text{norm}}^j$  to obtain premises for all

instances of rule (Der) that can be obtained this way. Since  $\mathcal{S}^j$  is (Der)-ready, this is at most one instance per pair of TVS in  $\mathcal{S}^j$  and VDS in  $\mathcal{C}$ , since only vertical weakening is necessary. (II) Then we obtain all possible TDS that can be derived via rule (Der) from these pairs.

By the above, the sets  $\mathcal{S}^{j+1}$  and  $\mathcal{S}'^{j+1}$  are of polynomial size. Moreover, they satisfy the conditions of Lemma 11 for  $k = j + 1$ : let  $S'$  be a TVS or a TDS in  $\text{CTI}[j + 1](\mathcal{C})$ . If also  $S' \in \text{CTI}[j](\mathcal{C})$ , then due to  $j$ -normalisation of  $\mathcal{S}_{\text{norm}}^j$  and  $\mathcal{S}'_{\text{norm}}^j$ , we are done. Otherwise, in any proof of  $S'$  from  $\mathcal{C}$  that witnesses  $S' \in \text{CTI}[j + 1](\mathcal{C})$ , there are finitely many top-level applications of rule (Der), i.e. those such that on the path from the root  $S'$  to the application of (Der) in the proof tree, there is no second application of (Der). Let  $T$  be a conclusion of such a rule application, let  $T_l$  and  $T_r$  be its left- and right-hand premises. If we can show that  $T$  is derivable from  $\mathcal{S}^{j+1} \cup \mathcal{S}'^{j+1}$ , we are done since  $T$  is arbitrary.

By definition,  $T_l \in \text{CTI}[j](\mathcal{C})$  and, hence,  $T_l$  can be derived from  $\mathcal{S}_{\text{norm}}^j$  via applications of ( $J_{\text{TVS}}$ ) and ( $W_{\text{TVS}}$ ). W.l.o.g. the applications of ( $J_{\text{TVS}}$ ) happen last in the proof tree of  $T_l$ , i.e. there are  $T_1, \dots, T_m$  s.t.  $T_l$  can be obtained from  $\mathcal{S}_{\text{norm}}^j$  via ( $W_{\text{TVS}}$ ) for  $1 \leq i \leq m$ . By (Der)-readiness of  $\mathcal{S}_{\text{norm}}^j$ , each of the  $T_i$  is a valid premise for rule (Der) together with  $T_r$ , and the conclusions  $T'_1, \dots, T'_m$  are all in  $\mathcal{S}'^{j+1}$ . It is easily verified that  $T$  can be obtained from  $T'_1, \dots, T'_m$  via rules ( $W_{\text{TDS}}$ ) and ( $J_{\text{TDS}}$ ). Since  $T$  was arbitrary,  $S'$  is provable from  $\mathcal{S}^{j+1} \cup \mathcal{S}'^{j+1}$  without using rule (Der).

Using Lemma 11 on  $\mathcal{S}^{j+1}$  and  $\mathcal{S}'^{j+1}$  yields, in polynomial time,  $j+1$ -normalised  $\mathcal{S}_{\text{norm}}^{j+1}$  and  $\mathcal{S}'_{\text{norm}}^{j+1}$ . Continuing this sequence of mass-applications of rule (Der) and re-normalisations yields, in polynomial time,  $k$ -normalised  $\mathcal{S}_{\text{norm}}^k$  and  $\mathcal{S}'_{\text{norm}}^k$  from which it follows directly whether  $S \in \text{CTI}[k](\mathcal{C})$ . ◀

As a corollary, we obtain the following: given a consistent influence scheme  $\mathcal{C}$  and a statement  $S$ , it is semi-decidable whether  $\mathcal{C} \vdash S$ . If  $\mathcal{C} \vdash S$ , then there is a proof of  $S$  from  $\mathcal{C}$ , and it has (Der)-depth  $k$  for some  $k$ . Hence, by checking consecutively whether  $S \in \text{CTI}[k'](\mathcal{C})$  for  $k' = 0, \dots$  will yield a positive result when reaching  $k' = k$  at the latest. In fact, this procedure is polynomial for  $k$  given in unary.

## 5 A Prototypical Implementation

We have implemented the proof search algorithm given in Thm. 12 in Python.<sup>3</sup> The program accepts statements and hypotheses as lists of tuples, or from a CSV file.

The solver module can run in different modes: it can either generate all derivable statements up to a certain (Der)-depth, until it has covered a certain point in time with statements, or until it runs out of new statements to derive, which, in practice, happens quite frequently due to compounding imprecision growing with the distance from the initial values. This is an effect that is known from numerical methods for solving differential equations, cf. [12, Chp. 5]. It remains to be seen whether such effects are acceptable for the foreseen application in a digital learning tool, and whether mathematical methods can be employed to reduce such effects.

The implementation also comes with a plotter module to visualise the statements that were derived. Full implementation into a classroom application is still to be done. Finally, the implementation comes with some pre-implemented example problems, including the photosynthesis problem from Ex. 3.

<sup>3</sup> Available at [https://github.com/SoerenMoeller/timed\\_influence\\_solver](https://github.com/SoerenMoeller/timed_influence_solver).

## 10:14 The Calculus of Temporal Influence

Preliminary results show that the implementation is fast enough to be able to solve didactically meaningful problems in seconds even on mobile devices, which matches similar results for the Calculus of (Non-Temporal) Influence [4].

The implementation differs in some details from the variant presented in the proof. For example, the implementation keeps sets of TVS and TDS associated to a variable normalised, i.e. after each additional statement that is derived w.r.t. a given variable, its entire set of associated statements is re-normalised immediately, instead of doing this in bulk after exhausting certain derivation rules. The reason for this is that keeping a representation of derived statements normalised is advantageous from a computational point of view, but the structure of the proof above becomes simpler if normalisation is done in bulk. It is not hard to see that both approaches yield the same results.

### 6 Conclusion

This work introduced the Calculus of Temporal Influence with the aim to extend previously started work on the formalisation of experiments and phenomena in nature, specifically in natural science classes. The focus here is on allowing processes to be modelled that are largely driven by time.

There are various other proposals of formalisms that also provide means to model dynamic systems evolving in time, like timed automata [2], timed Petri nets [13], hybrid automata [7], etc. In fact, modelling biological, chemical or physics phenomena by means of discrete and/or continuous mathematical formalisms is an active field of research, cf. [15, 1, 10, 14, 5, 3]. The need to develop a new formalism is driven by its application in an environment that is largely determined by didactical considerations. For example, hybrid automata allow for very precise modelling which is needed in verification; the price to pay is undecidability, high complexity and potentially only approximative analysis. For the hypotheses that are typically created by secondary-education pupils, precision is much less relevant, and so the Calculi of (Temporal and Non-Temporal) Influence are designed to allow reasoning about imprecise models.

It is important to note that the Calculus of Temporal Influence is a suggestion; it will have to prove worthy under exactly those circumstances. This will require much further work, starting with a clear categorisation of natural science experiments to which a hierarchy of modelling formalisms w.r.t. expressiveness can be aligned. An important point here is that it is not necessarily desirable to be able to model every such phenomenon, since more complicated argument structures may not be suitable for secondary-education level. In particular for lower grades, explainability of the results and ease of use of the learning framework are considerably more important.

However, being able to model more phenomena is certainly interesting alongside a different axis, i.e. the scientific one. Something that immediately comes to mind is the ability to make statements about compound functions, e.g. on the sum of the values of two variables  $a$  and  $b$ . Apart from this and the obvious question of whether (polynomial) decidability could be obtained for  $\vdash$ , perhaps based on some pumping argument, there are also further aspects of future work, perhaps of more technical nature, that will have to be considered like completeness of the calculus: does  $\mathcal{C} \models H$  imply  $\mathcal{C} \vdash H$ ? In the non-temporal case, completeness does not hold, but it is possible to retain it for a large class of experiment models [4]. We suspect the obstacles incurring in the temporal version to be even bigger.

We are also planning integration of our implementation in the wider context of a digital tool that supports learning and teaching in experimental science classes. Beyond the obvious problems of providing students with an answer whether their hypothesis was correct, based on the Calculus of Temporal Influence, this includes also tasks such as visualisation of the data in question, automatic translations from textual statements into formal ones, and teacher support.

---

## References

---

- 1 R. Alur, C. Belta, F. Ivančić, V. Kumar, M. Mintz, G. J. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biomolecular networks. In *Proc. 4th Int. Workshop on Hybrid Systems: Computation and Control, HSCC'01*, volume 2034 of *LNCS*, pages 19–32, 2001.
- 2 R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
- 3 J. Arias, M. Carro, Z. Chen, and G. Gupta. Modeling and reasoning in event calculus using goal-directed constraint answer set programming. *Theory Pract. Log. Program.*, 22(1):51–80, 2022. doi:10.1017/S1471068421000156.
- 4 F. Bruse, M. Lange, and S. Möller. Formal reasoning about influence in natural sciences experiments. In *Proc. 29th Int. Conf. on Computed-Aided Deduction, CADE'23*, LNAI. Springer, 2023. To appear.
- 5 C. Chaouiya. Petri net modelling of biological networks. *Briefings in Bioinformatics*, 8(4):210–219, 2007.
- 6 T. de Jong. Moving towards engaged learning in stem domains; there is no simple answer, but clearly a road ahead. *Journal of Computer Assisted Learning*, 35(2):153–167, 2019. doi:10.1111/jcal.12337.
- 7 T. A. Henzinger. The theory of hybrid automata. In *Proc. 11th Annual IEEE Symp. on Logic in Computer Science*, pages 278–292. IEEE, 1996.
- 8 T. Irion and K. Scheiter. Didaktische Potenziale digitaler Medien. Der Einsatz digitaler Technologien aus grundschul- und mediendidaktischer Sicht. *Grundschule aktuell*, 142:8–11, 2018.
- 9 M. Kastaun, M. Meier, N. Hundeshagen, and M. Lange. ProfiLL: Professionalisierung durch intelligente Lehr-Lernsysteme. In *Bildung, Schule, Digitalisierung*, pages 357–363. Waxmann-Verlag, 2020.
- 10 I. Koch. Petri nets – a mathematical formalism to analyze chemical reaction networks. *Molecular Informatics*, 29(12):838–843, 2010. doi:10.1002/minf.201000086.
- 11 OECD. *PISA 2006 – Science Competencies for Tomorrow's World: Volume 1: Analysis*. OECD Publishing, 2007. doi:10.1787/9789264040014-en.
- 12 R. S. Palais and R. A. Palais. *Differential Equations, Mechanics, and Computation*, volume 51 of *Student mathematical library; IAS/Park City mathematical subseries*. AMS, 2009.
- 13 L. Popova-Zeugmann. *Timed Petri Nets*, pages 139–172. Springer, 2013. doi:10.1007/978-3-642-41115-1\_4.
- 14 L. J. Steggle, R. Banks, and A. Wipat. Modelling and analysing genetic networks: From boolean networks to petri nets. In *International conference on computational methods in systems biology*, pages 127–141. Springer, 2006.
- 15 G. Theocharopoulou, C. Bobori, and P. Vlamos. Formal models of biological systems. In *Proc. 2nd World Congress on Genetics, Geriatrics and Neurodegenerative Disease, GeNeDis'16*, pages 325–338. Springer, 2017.

## A

 The Technical Definition of Intersecting Two Trapezoids

► **Definition 13.** Let  $S = \mathbf{t} \langle [t_1, t_2], [l_1, u_1], [l_2, u_2] \rangle_a$  and  $S' = \mathbf{t} \langle [t_1, t_2], [l'_1, u'_1], [l'_2, u'_2] \rangle_a$  be two TVS.

- If  $l_1 < l'_1$  and  $l_2 > l'_2$ , or if  $l_1 > l'_1$  and  $l_2 < l'_2$ , let  $t_l$  be the unique point in  $(t_1, t_2)$  such that  $\text{lo}_l = l_1 + (l_2 - l_1) \cdot \frac{t_l - t_1}{t_2 - t_1} = l'_1 + (l'_2 - l'_1) \cdot \frac{t_l - t_1}{t_2 - t_1}$ . Let  $\text{hi}_l = \min(u_1 + (u_2 - u_1) \cdot \frac{t_l - t_1}{t_2 - t_1}, u'_1 + (u'_2 - u'_1) \cdot \frac{t_l - t_1}{t_2 - t_1})$ .
- If  $u_1 < u'_1$  and  $u_2 > u'_2$ , or if  $u_1 > u'_1$  and  $u_2 < u'_2$ , let  $t_u$  be the unique point in  $(t_1, t_2)$  such that  $\text{hi}_u = u_1 + (u_2 - u_1) \cdot \frac{t_u - t_1}{t_2 - t_1} = u'_1 + (u'_2 - u'_1) \cdot \frac{t_u - t_1}{t_2 - t_1}$ . Let  $\text{lo}_u = \max(l_1 + (l_2 - l_1) \cdot \frac{t_u - t_1}{t_2 - t_1}, l'_1 + (l'_2 - l'_1) \cdot \frac{t_u - t_1}{t_2 - t_1})$ .

We now define three TVS  $S_i = \text{isect}_i([t_1, t_2], [l_1, u_1], [l_2, u_2], [l'_1, u'_1], [l'_2, u'_2])$  for  $i \in \{1, 2, 3\}$ , parameterized in the parameters of  $S, S'$  via:

- If both (I)  $l_1 \geq l'_1$  and  $l_2 \geq l'_2$  or  $l_1 \leq l'_1$  and  $l_2 \leq l'_2$ , and (II)  $u_1 \geq u'_1$  and  $u_2 \geq u'_2$ , or  $u_1 \leq u'_1$  and  $u_2 \leq u'_2$  hold, then  $S_1 = \mathbf{t} \langle [t_1, t_2], [\max(l_1, l'_1), \min(u_1, u'_1)], [\max(l_2, l'_2), \min(u_2, u'_2)] \rangle_a$  and  $S_2 = S_3 = S_1$ .
- If both (I)  $l_1 \geq l'_1$  and  $l_2 \geq l'_2$  or  $l_1 \leq l'_1$  and  $l_2 \leq l'_2$ , and (II)  $u_1 > u'_1$  and  $u_2 < u'_2$ , or  $u_1 < u'_1$  and  $u_2 > u'_2$  hold, then  $S_1 = \mathbf{t} \langle [t_1, t_u], [\max(l_1, l'_1), \min(u_1, u'_1)], [\text{lo}_u, \text{hi}_u] \rangle_a$  and  $S_2 = \mathbf{t} \langle [t_u, t_2], [\text{lo}_u, \text{hi}_u], [\max(l_2, l'_2), \min(u_2, u'_2)] \rangle_a$ , and  $S_3 = S_2$ .
- If both (I)  $l_1 < l'_1$  and  $l_2 > l'_2$  or  $l_1 > l'_1$  and  $l_2 < l'_2$ , and (II)  $u_1 \leq u'_1$  and  $u_2 \leq u'_2$ , or  $u_1 \geq u'_1$  and  $u_2 \geq u'_2$  hold, then  $S_1 = \mathbf{t} \langle [t_1, t_l], [\max(l_1, l'_1), \min(u_1, u'_1)], [\text{lo}_l, \text{hi}_l] \rangle_a$  and  $S_2 = \mathbf{t} \langle [t_l, t_2], [\text{lo}_l, \text{hi}_l], [\max(l_2, l'_2), \min(u_2, u'_2)] \rangle_a$ , and  $S_3 = S_2$ .
- If both (I)  $l_1 < l'_1$  and  $l_2 > l'_2$  or  $l_1 > l'_1$  and  $l_2 < l'_2$ , and (II)  $u_1 < u'_1$  and  $u_2 > u'_2$ , or  $u_1 < u'_1$  and  $u_2 > u'_2$  hold, then both  $t_l$  and  $t_u$  are defined. If  $t_l < t_u$ , then  $S_1 = \mathbf{t} \langle [t_1, t_l], [\max(l_1, l'_1), \min(u_1, u'_1)], [\text{lo}_l, \text{hi}_l] \rangle_a$  and  $S_2 = \mathbf{t} \langle [t_l, t_u], [\text{lo}_l, \text{hi}_l], [\text{lo}_u, \text{hi}_u] \rangle_a$ , and  $S_3 = \mathbf{t} \langle [t_u, t_2], [\text{lo}_u, \text{hi}_u], [\max(l_2, l'_2), \min(u_2, u'_2)] \rangle_a$ . The case for  $t_u < t_l$  is defined analogously. If  $t_l = t_u$ , then  $S_1 = \mathbf{t} \langle [t_1, t_l], [\max(l_1, l'_1), \min(u_1, u'_1)], [\text{lo}_l, \text{hi}_l] \rangle_a$  and  $S_2 = \mathbf{t} \langle [t_l, t_2], [\text{lo}_l, \text{hi}_l], [\max(l_2, l'_2), \min(u_2, u'_2)] \rangle_a$  and  $S_3 = S_2$ .

By setting  $S_3 = S_2$  or  $S_1 = S_2 = S_3$ , we make sure that, both for  $i = 2$  and  $i =$ , the TVS  $\text{isect}_i([t_1, t_2], [l_1, u_1], [l_2, u_2], [l'_1, u'_1], [l'_2, u'_2])$  are defined, even if the intersection produces less than three distinct TVS.

## B

 Proof of Lemma 11

Before we begin with the proof of Lemma 11, we study the interplay between TVS and TDS on adjacent time intervals. Let  $S_1 = \mathbf{t} \langle [t_1, t_2], [l_1, u_1], [l'_1, u'_1] \rangle_a$  and  $S_2 = \mathbf{t} \langle [t_2, t_3], [l_2, u_2], [l'_2, u'_2] \rangle_a$  be two adjacent TVS and let  $S_3 = \mathbf{t} \langle [t_1, t_2], [l_3, u_3] \rangle_a$  and  $S_4 = \mathbf{t} \langle [t_2, t_3], [l_4, u_4] \rangle_a$  be two adjacent TDS, all over the same variable  $a$  and with pairwise matching time intervals. This is a situation that appears in sets of separated TVS and TDS. The synchronisation of the time intervals can be achieved by using rules ( $\text{W}_{\text{TVS}}$ ) and ( $\text{W}_{\text{TDS}}$ ).

We call an individual TVS *derivative-reduced*, if it satisfies the conditions laid out in Def. 10 for derivative-reducedness of an entire set of TVS. Obviously, a set of TVS is derivative-reduced if all the TVS in it are so. By the above, we can assume that, for separated sets of TVS and separated and minimal TDS, the individual points where time intervals touch each other are the same. This can be achieved by splitting a TVS if two TDS for the same variable touch in an interior point of its time interval, and vice versa. Clearly, this produces polynomial blowup at most. Hence, for a given TVS, there is a unique strictest TDS in question that dictates whether the TVS is derivative-reduced. For our considerations, we assume that this is  $S_3$  for  $S_1$  and  $S_4$  for  $S_2$ .

We can now make  $S_1$  derivative-reduced by using rule ( $S_{TVS}$ ) to obtain TVS for the unit intervals  $[t_1, t_1]$  and  $[t_2, t_2]$  and using them as left premises for rules (CDR), resp. (CDL). The former of these rules produces an upper bound on the slope of the upper edge of the trapezoid generated by  $S_1$ , and a lower bound on the slope of the lower edge of said trapezoid. The latter rule produces a lower bound on the slope of the upper edge, and an upper bound on the slope of the lower edge. Notably, the actual slopes of the edges of the trapezoid defined by  $S_1$  can violate at most one of these bounds per slope. It follows that the intersection of the three TVS in question, i.e.  $S_1$ , the trapezoid obtained by using rule (CDR), and the one obtained by using rule (CDL), do not produce intersecting upper and lower edges unless there is inconsistency: The trapezoid obtained by using the former rule shares the right edge with the one defined by  $S_1$ , and the trapezoid obtained by using the latter rule shares a left edge with the one defined by  $S_1$ . Hence, neither of these produces a nontrivial intersection with  $S_1$ , and they cannot intersect with each other inside the trapezoid defined by  $S_1$  for simple geometric reasons unless  $S_1$  and  $S_3$  together are inconsistent. Hence, if we assume consistency, combining these three trapezoids using rule ( $S_{TVS}$ ) produces a single new trapezoid  $S'_1 = \mathfrak{t}_{\langle [t_1, t_2], [l_1^d, u_1^d], [l_1^d, u_1^d] \rangle} a$ . Moreover,  $S'_1$  shares at least two of  $l_1, u_1, l'_1, u'_1$ , since  $S_1$  violates at most one bound on the slope of its upper edge, and at most one on the slope of its lower edge. We write  $\text{reduce}(S_1, S_3)$  for the TVS  $S'_1$  obtained this way. We say that a TVS  $S$  is reduced w.r.t. a TDS  $S'$  if  $\text{reduce}(S, S') = S$ , i.e. if the trapezoid defined by it already satisfies the conditions on its slopes induced by  $S'$ . As outlined above, in a separated, minimal setting, w.l.o.g., for each TVS  $S$  there is a unique TDS  $S'$  such that  $S$  is derivative-reduced if it is derivative-reduced w.r.t.  $S'$ . The above also works if  $S_3$  contains infinite upper and lower bounds; in this case there are simply less restrictions.

Now imagine that we obtain  $S'_1 = \mathfrak{t}_{\langle [t_1, t_2], [l_1^d, u_1^d], [l_2^d, u_2^d] \rangle} a = \text{reduce}(S_1, S_3)$  and  $S'_2 = \mathfrak{t}_{\langle [t_2, t_3], [l_1^d, u_1^d], [l_2^d, u_2^d] \rangle} a = \text{reduce}(S_2, S_4)$ . If  $[l_2^d, u_2^d] = [l_1^d, u_1^d]$  then we are done. Otherwise we obtain the TVS  $\mathfrak{t}_{\langle [t_2, t_2], [\max(l_2^d, l_1^d), \min(u_2^d, u_1^d)] \rangle} a$ , and we can apply rules (CDL) and (CDR) together with  $S_2$  resp.  $S_4$  to further restrict the function  $\mathcal{F}_a$  on the intervals  $[t_1, t_2]$  and  $[t_2, t_3]$ . Hence, making a single TVS derivative-reduced is rather straightforward, but already with two, it is not clear that the process of making both of them derivative-reduced at the same time ever halts. The following is a crucial observation for the process of making a set of TVS derivative-reduced.

► **Observation 14.** *Let  $S' = \mathfrak{t}_{\langle [t_1, t_2], [l_1, u_1], [l_2, u_2] \rangle} a$  be the result of  $\text{reduce}(S, S'')$  for some TDS  $S''$ . Let  $[l'_1, u'_1] \subseteq [l_1, u_1]$  and  $[l'_2, u'_2] \subseteq [l_2, u_2]$  and let  $R_1 = \mathfrak{t}_{\langle [t_1, t_1], [l'_1, u'_1] \rangle} a$  and  $R_2 = \mathfrak{t}_{\langle [t_2, t_2], [l'_2, u'_2] \rangle} a$  be two TVS over unit intervals. Then there are unique, maximal intervals  $[l''_1, u''_1] \subseteq [l_1, u_1]$  and  $[l''_2, u''_2] \subseteq [l_2, u_2]$  s.t.  $S'_l = \mathfrak{t}_{\langle [t_1, t_2], [l'_1, u'_1], [l''_2, u''_2] \rangle} a$  and  $S'_r = \mathfrak{t}_{\langle [t_1, t_2], [l''_1, u''_1], [l'_2, u'_2] \rangle} a$  are derivative-reduced w.r.t.  $S''$ . Moreover, they can be obtained using rules (CDR), resp. rule (CDL) with the  $R_1$  resp.  $R_2$  as premises and then using ( $S_{TVS}$ ).*

The reason for this is that  $S$  is already derivative-reduced w.r.t.  $S''$ , i.e. the slopes of the upper and lower edges of the trapezoid defined by it are already within the bounds given by  $S''$ . Hence, making one of the vertical intervals smaller will only result in the other interval shrinking when making the TVS derivative-reduced again. We write  $\text{reduce}(S', S'', [l'_1, u'_1])$  or  $\text{reduce}(S', S'', [l'_2, u'_2])$  for the TVS  $S'_l$  and  $S'_r$  obtained this way, tacitly assuming that it is clear on which side the stricter vertical interval goes. Note that all of this also works if one of the TVS has infinite upper or lower bounds.

These re-reduced TVS are important since they give us an easy termination argument for the process of making a separated sequence of TVS derivative-reduced: Use the two-argument version of  $\text{reduce}$  to obtain a derivative-reduced version of the leftmost TVS. Let  $[l, u]$  be its right vertical interval, and let  $[l', u']$  be the left interval of the second TVS from the left.



Then use the three-argument version together with  $[l, u] \cap [l', u']$  to make the second TVS from the left derivative-reduced. This might yield a new left interval  $[l'', u'']$  for said second TVS, so we use the three-argument version of `reduce` on the first TVS to adjust it to the new interval bounds. Crucially, this will not change the right vertical interval of the first TVS, whence the process terminates.

The general procedure of making an entire separated sequence of TVS derivative-reduced works as follows: assume that the first  $k$  TVS are already derivative-reduced. Let  $[l, u]$  be the right vertical interval of the  $k$ th such TVS, and let  $[l', u']$  be the left vertical interval of its right neighbour, i.e. the  $k+1$ st TVS. Use the three-argument version of `reduce` on this from the left, together with the interval  $[l, u] \cap [l', u']$ . This makes the  $k+1$ st TVS derivative-reduced, and potentially introduces an even stricter interval  $[l'', u'']$  for the vertical interval between the  $k$ th and  $k+1$ st TVS. Use the three-argument version of `reduce` to adjust the  $k$ th TVS. This potentially yields a new interval bound between the  $k$ th and  $k-1$ st TVS, so use the three-argument version of `reduce` to adjust the  $k-1$ st TVS as well. Since the new, stricter intervals only propagate to the left, this process terminates when the first TVS is re-reduced, since it does not have a left neighbour. Now the first  $k+1$  TVS are derivative-reduced. Clearly this process works in polynomial time, since it propagates to the left at most once per TVS, which, in turn, only happens once for each of them.<sup>4</sup>

It follows that a sequence of separated TVS can be made derivative-reduced in polynomial time.

We are now ready to prove the following.

► **Lemma 11.** *Let  $\mathcal{C}$  be a consistent  $\mathcal{V}$ -influence scheme, let  $a \in \mathcal{V}$ , let  $k \in \mathbb{N}$ . Let  $\mathcal{S}$  be a finite set of TVS and let  $\mathcal{S}'$  be a finite set of TDS s.t. all TVS and TDS of (Der)-depth (w.r.t.  $\mathcal{C}$ )  $k$  or less can be derived from  $\mathcal{S} \cup \mathcal{S}'$  without using rule (Der). Then there are  $\mathcal{S}_{\text{norm}}$  and  $\mathcal{S}'_{\text{norm}}$  that both are  $k$ -normalised (and therefore representative, ensuring equivalence to  $\mathcal{S}$  and  $\mathcal{S}'$  w.r.t.  $\vdash$ ). Moreover,  $\mathcal{S}_{\text{norm}}$  and  $\mathcal{S}'_{\text{norm}}$  can be computed from  $\mathcal{S}$  and  $\mathcal{S}'$  in polynomial time.*

**Proof.** We begin by transforming  $\mathcal{S}'$  into a set  $\mathcal{S}'_c$  of TDS that is almost  $k$ -normalised for every variable. Note that the only rules that have a TDS as their conclusion are rules (Der), ( $\mathbf{G}_{\text{TDS}}$ ), ( $\mathbf{W}_{\text{TDS}}$ ), ( $\mathbf{S}_{\text{TDS}}$ ) and ( $\mathbf{VD}$ ). Obviously, rule (Der) plays no role here.

As a first step, we generate a candidate set  $\mathcal{S}'_c$  from  $\mathcal{S}'$  by making it separated for every variable. This is done by removing non-unit overlap between TDS with different time intervals using rule ( $\mathbf{W}_{\text{TDS}}$ ). We use this rule to split a TDS that overlaps with another w.r.t. their time intervals such that any two TDS in the set either have the same time interval, or time intervals that overlap in at most one point. Since each TDS overlaps at most with all the others in  $\mathcal{S}'$ , this produces at most polynomial blowup in the number of TDS.

We then use rule ( $\mathbf{S}_{\text{TDS}}$ ) if several TDS still overlap at non-unit intervals. By assumption, any such overlapping TDS have the same time interval, so it is easy to see that this produces at most one TDS per time interval with nontrivial overlap. We use rule ( $\mathbf{G}_{\text{TDS}}$ ) to close any uncovered gaps in the horizontal dimension. Hence,  $\mathcal{S}'_c$  is now separated. It is also minimal: Any witness to the contrary must be derivable from  $\mathcal{S}' \cup \mathcal{S}$  by assumption. Rules ( $\mathbf{G}_{\text{TDS}}$ ) and ( $\mathbf{VD}$ ) certainly cannot help to produce such a counterexample, as they produce TDS with infinite lower and upper bounds. Rule ( $\mathbf{W}_{\text{TDS}}$ ) can also be omitted since its premise would already be a counterexample. However, we just used rule ( $\mathbf{S}_{\text{TDS}}$ ) to its maximum extent, so  $\mathcal{S}'_c$  is minimal for each variable. It is not yet representative though, since there might

<sup>4</sup> In fact, the process can be optimised to only propagate to the left once. We simply state this here without an argument.



be a TDS derivable using rule (VD), but the premise of that rule is not in  $\mathcal{S}$ . However, by the reasoning above it is almost representative in the sense that these are the only TDS of (Der)-depth  $k$  or less not yet derivable from  $\mathcal{S}'_c$ .

Hence, we now transform  $\mathcal{S}$  into a candidate set  $\mathcal{S}_c$  that is almost  $k$ -pre-normalised for each variable. Again, we make  $\mathcal{S}_c$  separated by using rule ( $W_{\text{TVS}}$ ) to reduce overlaps to statements with either unit intervals, or the same time interval, and then ( $S_{\text{TVS}}$ ) to reduce overlap to unit intervals only. Note that rule ( $S_{\text{TVS}}$ ) may produce shorter time intervals due to the mechanics of intersecting trapezoids, however these are still only polynomially many, since each overlap produces at most three trapezoids. Using rules ( $G_{\text{TVS}}$ ) and (VD), we make  $\mathcal{S}_c$  separated for each variable. We then synchronise the time intervals for the TVS in  $\mathcal{S}_c$  and TDS  $\mathcal{S}'_c$  by splitting them, if necessary.

Using the procedure outlined after Obs. 14, we make  $\mathcal{S}_c$  derivative-reduced. Since  $\mathcal{S}'_c$  is minimal except for TDS with infinite upper and lower vertical bounds, the TDS in  $\mathcal{S}'_c$  do contain the strictest bounds on the respective derivative functions, so the result of the procedure is really derivative-reduced. Note that the procedure may have produced gaps in the sequence of TVS, if the time intervals in some of the TDS are large. We use rule ( $G_{\text{TVS}}$ ) to close these gaps, and rule (VD) to transport new information on the domains of the functions to the TDS. It is not hard to verify that both  $\mathcal{S}_c$  and  $\mathcal{S}'_c$  are now separated, minimal, and representative. Moreover,  $\mathcal{S}_c$  is derivative-reduced. We make it (Der)-ready by splitting intervals, if necessary, for, again, at most polynomial blowup.

Hence, the sets  $\mathcal{S}_c$  and  $\mathcal{S}'_c$  are now both  $k$ -normalised, and are our desired sets  $\mathcal{S}_{\text{norm}}$  and  $\mathcal{S}'_{\text{norm}}$ . ◀



# Detecting Causality in the Presence of Byzantine Processes: The Synchronous Systems Case

Anshuman Misra ✉

University of Illinois at Chicago, IL, USA

Ajay D. Kshemkalyani<sup>1</sup> ✉ 

University of Illinois at Chicago, IL, USA

---

## Abstract

Detecting causality or the happens before relation between events in a distributed system is a fundamental building block for distributed applications. It was recently proved that this problem cannot be solved in an asynchronous distributed system in the presence of Byzantine processes, irrespective of whether the communication mechanism is via unicasts, multicasts, or broadcasts. In light of this impossibility result, we turn attention to synchronous systems and examine the possibility of solving the causality detection problem in such systems. In this paper, we prove that causality detection between events can be solved in the presence of Byzantine processes in a synchronous distributed system. The positive result holds for unicast, multicast, as well as broadcast modes of communication. We prove the result by providing an algorithm. Our solution uses the Replicated State Machine (RSM) approach and vector clocks.

**2012 ACM Subject Classification** Computing methodologies → Distributed algorithms; Networks → Network algorithms

**Keywords and phrases** Byzantine fault-tolerance, causality, happens before, distributed system, message-passing, synchronous system

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.11

## 1 Introduction

### 1.1 Background

Causality is an important tool in understanding and reasoning about distributed executions [32]. Lamport formulated the “happens before” or the causality relation, denoted  $\rightarrow$ , between events in a distributed system [20]. Given two events  $e$  and  $e'$ , the *causality detection* problem asks to determine whether  $e \rightarrow e'$ . There are many applications of causality detection including determining consistent recovery points in distributed databases, deadlock detection, termination detection, distributed predicate detection, distributed debugging and monitoring, and the detection of race conditions and other synchronization errors [18].

The causality relation between events can be captured by tracking causality graphs [7], scalar clocks [20], vector clocks [5, 8, 22], and several other variants of logical clocks such as hierarchical clocks [35], plausible clocks [34], dotted version vectors [30], interval tree clocks [1], logical physical clocks [19], Bloom clocks [16, 23], incremental clocks [33], and resettable prime clocks [17, 29]. Some of these variants track causality accurately while others introduce approximations and inaccuracies as trade-offs in the interest of savings on the space and/or time and/or message complexity overheads. As stated by Schwarz and Mattern [32], the search for the holy grail of the ideal causality tracking mechanism is on. These above works in the literature assume that processes are correct (non-faulty). The causality detection problem for a system with Byzantine processes was recently introduced and studied in [25].

---

<sup>1</sup> Corresponding author



The related problem of causal ordering of messages asks that if the send event of message  $m$  happens before the send event of message  $m'$ , then  $m'$  should not be delivered before  $m$  at all the common destinations of  $m$  and  $m'$ . Under the Byzantine failure model, causal ordering has recently been studied in [2] for broadcast communication and in [24, 26, 27] for unicast, multicast, as well as broadcast communication.

## 1.2 Contributions

It was recently proved that the problem of detecting causality between a pair of events cannot be solved in an asynchronous system in the presence of Byzantine processes, irrespective of whether the communication is via unicasts, multicasts, or broadcasts [25]. In the multicast mode of communication, each send event sends a message to a group consisting of a subset of the set of processes in the system. Different send events can send to different subsets of processes. Communicating by unicasts and communicating by broadcasts are special cases of multicasting. It was shown in [25] that in asynchronous systems with even a single Byzantine process, the unicast and multicast modes of communication are susceptible to false positives and false negatives, whereas the broadcast mode of communication is susceptible to false negatives but no false positives. A false positive means that  $e \not\rightarrow e'$  whereas  $e \rightarrow e'$  is perceived/detected. A false negative means that  $e \rightarrow e'$  whereas  $e \not\rightarrow e'$  is perceived/detected.

1. In light of the impossibility result for asynchronous systems, this paper examines the solvability of causality detection in synchronous systems in the presence of Byzantine processes.
2. We prove that causality detection between events can be solved in the presence of Byzantine processes in a synchronous system. We provide an algorithm that solves the causality detection problem. The positive result holds for unicasts, multicasts, as well as broadcasts. Our solution uses the Replicated State Machine (RSM) approach [31], which works only in synchronous systems, in conjunction with vector clocks.
3. This is the first paper to establish this result. The paper uses a simple combination of RSMs and vector clocks and is yet significant, similar to results in [8, 22, 32], because it establishes a fundamental possibility result about causality detection in the presence of Byzantine processes in a synchronous system.
4. The results for multicasts, unicasts, and broadcasts are summarized in Table 1. In a system with  $n$  application processes, our RSM-based solution uses  $3t + 1$  process replicas per application process, where  $t$  is the maximum number of Byzantine processes that can be tolerated in a RSM. Thus, there can be at most  $nt$  Byzantine processes among a total of  $(3t + 1)n$  processes partitioned into  $n$  RSMs of  $3t + 1$  processes each, with each RSM having up to  $t$  Byzantine processes. By using  $(3t + 1)n$  processes and the RSM approach to represent  $n$  application processes, the malicious effects of Byzantine process behaviors are neutralized.

**Roadmap.** Section 2 gives the system model. Section 3 formulates the problem of detecting causality in the presence of Byzantine processes. Section 4 proves the results outlined under “Contributions” above. Section 5 gives a discussion and concludes.

■ **Table 1** Detecting causality between events under different communication modes in asynchronous and synchronous systems.  $FP$  is false positive,  $FN$  is false negative.  $\overline{FP}/\overline{FN}$  means no false positive/no false negative is possible.

Mode of communication	Detecting “happens before” in asynchronous systems	Detecting “happens before” in synchronous systems
Multicasts	Impossible [25] $FP, FN$	Possible, Theorem 11 $\overline{FP}, \overline{FN}$
Unicasts	Impossible [25] $FP, FN$	Possible, Corollary 12 $\overline{FP}, \overline{FN}$
Broadcasts	Impossible [25] $\overline{FP}, FN$	Possible, Corollary 13 $\overline{FP}, \overline{FN}$

## 2 System Model

This paper deals with a distributed system having Byzantine processes which are processes that can misbehave [21, 28]. A correct process behaves exactly as specified by the algorithm whereas a Byzantine process may exhibit arbitrary behaviour including crashing at any point during the execution. A Byzantine process cannot impersonate another process or spawn new processes.

The distributed system is modelled as an undirected graph  $G = (P, C)$ . Here  $P$  is the set of processes communicating in the distributed system. Let  $|P| = n$ .  $C$  is the set of (logical) communication links over which processes communicate by message passing. The channels are assumed to be FIFO.  $G$  is a complete graph.

The distributed system is assumed to be synchronous, i.e., there is a known fixed upper bound  $\delta$  on the message latency, and a known fixed upper bound  $\psi$  on the relative speeds of processors [6]. In contrast, an asynchronous system has been defined as one in which there is no upper bound on the message latency and on the relative speeds of processors [6]. A synchronous system guarantees that the relative speeds of non-faulty processors and messages is bounded, and this is equivalent to assuming that the system has synchronized real-time clocks [31].

Let  $e_i^x$ , where  $x \geq 1$ , denote the  $x$ -th event executed by process  $p_i$ . An event may be an internal event, a message send event, or a message receive event. Let the state of  $p_i$  after  $e_i^x$  be denoted  $s_i^x$ , where  $x \geq 1$ , and let  $s_i^0$  be the initial state. The execution at  $p_i$  is the sequence of alternating events and resulting states, as  $\langle s_i^0, e_i^1, s_i^1, e_i^2, s_i^2, \dots \rangle$ . The sequence of events  $\langle e_i^1, e_i^2, \dots \rangle$  is called the execution history at  $p_i$  and denoted  $E_i$ . Let  $E = \bigcup_i \{E_i\}$  and let  $T(E)$  denote the set of all events in (the set of sequences)  $E$ . The *happens before* [20] relation, denoted  $\rightarrow$ , is an irreflexive, asymmetric, and transitive partial order defined over events in a distributed execution that is used to define causality.

► **Definition 1.** The *happens before* relation  $\rightarrow$  on events  $T(E)$  consists of the following rules:

1. **Program Order:** For the sequence of events  $\langle e_i^1, e_i^2, \dots \rangle$  executed by process  $p_i$ ,  $\forall x, y$  such that  $x < y$  we have  $e_i^x \rightarrow e_i^y$ .
2. **Message Order:** If event  $e_i^x$  is a message send event executed at process  $p_i$  and  $e_j^y$  is the corresponding message receive event at process  $p_j$ , then  $e_i^x \rightarrow e_j^y$ .
3. **Transitive Order:** If  $e \rightarrow e' \wedge e' \rightarrow e''$  then  $e \rightarrow e''$ .

► **Definition 2.** The causal past of an event  $e$  is denoted as  $CP(e)$  and defined as the set of events  $\{e' \in T(E) \mid e' \rightarrow e\}$ .

### 3 Problem Formulation

The problem formulation is done similar to the way in [25]. An algorithm to solve the causality detection problem collects the execution history of each process in the system and derives causal relations from it.  $E_i$  is the *actual* execution history at  $p_i$ . For any causality detection algorithm, let  $F_i$  be the execution history at  $p_i$  as perceived and collected by the algorithm and let  $F = \bigcup_i \{F_i\}$ .  $F$  thus denotes the execution history of the system as perceived and collected by the algorithm. Analogous to  $T(E)$ , let  $T(F)$  denote the set of all events in  $F$ . Analogous to Definition 1, the *happens before* relation can be defined on  $T(F)$  instead of on  $T(E)$ . With a slight relaxation of notation, let  $T(E_i)$  and  $T(F_i)$  denote the set of all events in  $E_i$  and  $F_i$ , respectively.

Let  $e1 \rightarrow e2|_E$  and  $e1 \rightarrow e2|_F$  be the evaluation (1 or 0) of  $e1 \rightarrow e2$  using  $E$  and  $F$ , respectively. Byzantine processes may corrupt the collection of  $F$  to make it different from  $E$ . We assume that a correct process  $p_i$  needs to detect whether  $e_h^x \rightarrow e_i^*$  holds and  $e_i^*$  is an event in  $T(E)$ . If  $e_h^x \notin T(E)$  then  $e_h^x \rightarrow e_i^*|_E$  evaluates to *false*. If  $e_h^x \notin T(F)$  (or  $e_i^* \notin T(F)$ ) then  $e_h^x \rightarrow e_i^*|_F$  evaluates to *false*. We assume an oracle that is used for determining correctness of the causality detection algorithm; this oracle has access to  $E$  which can be any execution history such that  $T(E) \supseteq CP(e_i^*)$ .

Byzantine processes may collude as follows.

1. To delete  $e_h^x$  from  $F_h$  or in general, record  $F$  as any alteration of  $E$  such that  $e_h^x \rightarrow e_i^*|_F = 0$ , while  $e_h^x \rightarrow e_i^*|_E = 1$ , or
2. To add a fake event  $e_h^x$  in  $F_h$  or in general, record  $F$  as any alteration of  $E$  such that  $e_h^x \rightarrow e_i^*|_F = 1$ , while  $e_h^x \rightarrow e_i^*|_E = 0$ .

Without loss of generality, we have that  $e_h^x \in T(E) \cup T(F)$ . Note that  $e_h^x$  belongs to  $T(F) \setminus T(E)$  when it is a fake event in  $F$ .

► **Definition 3.** *The causality detection problem  $CD(E, F, e_i^*)$  for any event  $e_i^* \in T(E)$  at a correct process  $p_i$  is to devise an algorithm to collect the execution history  $E$  as  $F$  at  $p_i$  such that  $valid(F) = 1$ , where*

$$valid(F) = \begin{cases} 1 & \text{if } \forall e_h^x, e_h^x \rightarrow e_i^*|_E = e_h^x \rightarrow e_i^*|_F \\ 0 & \text{otherwise} \end{cases}$$

When 1 is returned, the algorithm output matches the actual (God's) truth and solves  $CD$  correctly. Thus, returning 1 indicates that the problem has been solved correctly by the algorithm using  $F$ . 0 is returned if either

- $\exists e_h^x$  such that  $e_h^x \rightarrow e_i^*|_E = 1 \wedge e_h^x \rightarrow e_i^*|_F = 0$  (denoting a false negative), or
- $\exists e_h^x$  such that  $e_h^x \rightarrow e_i^*|_E = 0 \wedge e_h^x \rightarrow e_i^*|_F = 1$  (denoting a false positive).

Using the state-machine replication approach, we show that  $F$  at a correct process can be made to exactly match  $E$ , hence there is no possibility of a false positive or of a false negative.

## 4 Solution based on Replicated State Machines (RSMs)

### 4.1 Background on RSMs

The discussion in this section is based on the survey by Schneider [31]. A process execution is modelled as the actions of a finite state machine. Two basic requirements are: (O1: FIFO order) Messages issued by a client to a state machine are processed in the order issued, and (O2: Causal order) If a message  $m1$  issued to a state machine  $sm$  by client  $c$  could have caused (i.e., causally preceded) a message  $m2$  issued by client  $c'$  to  $sm$ , then  $sm$  processes  $m1$  before  $m2$ .

A  $t$ -tolerant version of a state machine is implemented by replicating that state machine and running a *state machine replica*  $smr$  on different processors in an *ensemble*. If each replica run by a correct processor starts in the same initial state and executes the same requests in the same order, then each replica will execute the same step at each transition and produce the same output. Under Byzantine failures, an ensemble implementing a  $t$  tolerant RSM must have at least  $2t + 1$  replicas and the output of each (correct) replica in the ensemble is the output produced by  $t + 1$  replicas. To ensure that all replicas' actions and transitions are coordinated, all replicas in an ensemble must receive and process the same sequence of messages. This can be expressed as two requirements.

- Agreement: Every non-faulty replica receives every message.
  - Total order: Every non-faulty replica processes the messages it receives in the same order.
- Agreement requires that (IC1) for each message sent by a replica, all non-faulty replicas of the destination process agree on the contents of the message, and (IC2) if the transmitting replica is non-faulty, then all non-faulty replicas of the destination process use the transmitter's value as the one on which they agree. Any of the Byzantine agreement protocols in the literature can be used [21, 28]; they all require that the total number of replicas (of the destination process) is at least  $3t + 1$ . Furthermore, no deterministic algorithm can implement state machine replication, which requires agreement or consensus, in an asynchronous system [9]. So we assume a synchronous system.

Total order can be satisfied by assigning unique identifiers to messages sent and having the receiver's  $smrs$  process the messages as per a total order relation on these unique identifiers. For the RSM of application process  $p_j$ , its various  $3t + 1$   $smrs$  are denoted  $smr_{j,w}$ . A message is defined to be *stable* at  $smr_{j,w}$  once no message from a correct sender process replica (across all sender processes from various sender process ensembles) having a lower unique identifier can be subsequently delivered to  $smr_{j,w}$ . Total order is implemented by requiring a replica process to next process the stable request with the smallest stable identifier. Mechanisms for generating unique identifiers satisfying FIFO and causal order are given by Schneider [31]. These mechanisms are based on synchronized real-time clocks (which guarantees O1 and causal order O2 implicitly), or based on receiver replica-generated unique identifiers; the latter approach also requires for maintaining FIFO order and causal order (O1 and O2) that once a transmitter replica starts disseminating a message, it performs no other communication until the current message has been delivered to every receiver replica that is a destination of the current message. In a system with Byzantine processes, the replica-generated unique identifiers approach along with using the assumptions on synchronized real-time clocks can satisfy the total order. But note here that the requirement of synchronized real-time clocks forces us to assume a synchronous system.

## 4.2 Adapting RSMs to Our Solution

In our system model having  $n$  application processes, each process  $p_i$  modelled as a RSM is replicated  $3t + 1$ -way as  $p_{i,1}, \dots, p_{i,3t+1}$  and these processes form the ensemble  $p_i$ . Various RSM ensembles communicate in a peer-to-peer (P2P) manner with each other. When a RSM ensemble sends/receives a message, it is referred to as a sender/receiver RSM ensemble. Thus in a system having  $n$  application processes, there are  $(3t + 1)n$  processes (i.e., replicas) partitioned into  $n$  RSM ensembles and each ensemble can have at most  $t$  Byzantine processes. Each  $p_{i,a}$ , i.e.,  $smr_{i,a}$ , uses a sequence number denoted  $seq_{i,a}$  that is incremented for each message that it sends/multicasts as a sender RSM replica. The  $(3t + 1)n$  processes can be viewed as running in an application layer that is above the RSM layer which provides Agreement and Total Order.



Using the implementation of RSMs described by Schneider or any of the subsequent implementations proposed since then, Agreement and Total Order are guaranteed. Furthermore, Total Order is guaranteed in a receiver RSM ensemble for messages from multiple sender RSM ensembles. In addition, when each replica in the sender RSM ensemble does a multicast, the following version of the Agreement property needs to be implemented.

- Agreement– $M$ : Every non-faulty replica in every RSM ensemble that is included in the destination set of a multicast/broadcast receives the message multicast/broadcast.

Agreement- $M$  requires that (IC1- $M$ ) for each message sent by a replica, all non-faulty replicas of the destination processes of a multicast/broadcast agree on the contents of the message, and (IC2- $M$ ) if the transmitting replica is non-faulty, then all non-faulty replicas of the destination processes of a multicast/broadcast use the transmitter’s value as the one on which they agree.

When a RSM replica receives a message from the RSM layer satisfying Total Order and Agreement/Agreement- $M$ , we say that the message is *TOA-delivered* to that RSM replica. Under Byzantine failures, an ensemble implementing a  $t$  tolerant RSM in a system model disallowing cryptography must have at least  $3t + 1$  replicas and the output of each (correct) replica in an ensemble is the output produced by a *majority*  $= t + 1$  replicas. Henceforth, we treat *majority* as having the value  $t + 1$ . Since we are using RSMs for “clients” and “servers” in P2P mode, whenever a correct receiver replica is *TOA-delivered* (gets)  $t + 1$  identical messages  $M$  from the replicas of a sender ensemble, the (correct) receiver replica delivers the message to the layer above. We say that a message  $M$  is *SR-delivered* to a RSM replica if *majority*  $= t + 1$  identical copies of the message having the same  $seq_{j,*}$  from the replicas of a sender ensemble  $j$  have been *TOA-delivered* to it. On *SR-delivery* of a message to a RSM replica, that replica makes the next transition according to the local state machine. The Agreement and Total Order properties guarantee that if  $smr_{i,a}$  *SR-delivers* such a message, then every other correct receiver replica  $smr_{i,y}$  in that ensemble will also *SR-deliver* that same message  $M$  in exactly the order and sequence it was *SR-delivered* by  $smr_{i,a}$ . Note that there are at least  $t + 1$  votes for this message  $M$  from the sender replica ensemble and since there are at most  $t$  Byzantine processes in the sender replica ensemble, their state machines can send only up to  $t$  messages (for any particular sequence number  $seq_{j,*}$  from the sender ensemble  $j$ ) that are received by  $smr_{i,a}$  and that differ from the majority value of  $M$  received  $t + 1$  times by  $smr_{i,a}$ .

When  $smr_{i,a}$  sends a message to  $p_j$  at the application level, it sends it to all replicas  $smr_{j,b}$ . When  $smr_{i,a}$  *SR-delivers* a message, a receive event is said to have occurred at the application level. Henceforth, we also refer to  $smr_{i,a}$  as  $p_{i,a}$  and RSM  $i$  as  $p_i$ .

### 4.3 Data Structures and Algorithm

Algorithm 1 is an online algorithm in which each correct replica  $p_{i,a}$  records in  $F = \bigcup_k \{F_k\}$  its view of the execution history of RSM  $p_k$  via lines 1-21. This recording of  $F$  in the local replica is done by piggybacking control information on the application messages; no extra messages are used. There is also a module in Algorithm 1 lines 22-26 that takes as input two events  $e_h^x$  and  $e_i^*$  and produces output from  $\{true, false\}$  giving  $e_h^x \rightarrow e_i^*|_F$ . Theorem 9 shows that the set of events in  $E$  matches the set of events recorded in  $F$ , even though  $E$  is never recorded and is accessible only to an oracle. Next we show in Theorem 11 that using the output of the algorithm lines 22-26 function *test*, and Theorem 9, the causality detection problem is solved by Algorithm 1’s recording of  $F$  and function *test* using this  $F$ , i.e., there are no false positives nor false negatives.

■ **Algorithm 1** Processing of control information and testing for  $e_h^x \rightarrow e_i^*$ . Code at process  $p_{i,a}$ .

---

**Data:** Each process  $p_{i,a}$  maintains (i) an integer  $seq_{i,a}$ , (ii)  $F$  which is the union of sequences  $F_k$  (history of events at  $p_k$ ) for all  $k$ , (iii) integer matrix  $LASKALSJ[n, n]$ , (iv) integer matrix  $V[|T(F_i)|, n]$ .

**Input:**  $e_h^x, e_i^*$   
**Output:**  $e_h^x \rightarrow e_i^* |_F \in \{true, false\}$

- 1 **when**  $p_{i,a}$  needs to send application message  $M$  to  $p_{j,*}$ :  $\triangleright$  **Each other correct**  $p_{i,a'}$   
     **state machine will execute likewise**
- 2  $seq_{i,a} = seq_{i,a} + 1$
- 3 append current send event to  $F_i$ ;  $(\forall k)V[seq_{i,a}, k] = maxeventID(F_k)$
- 4  $(\forall k)$  include history from  $F_k$  after event  $LASKALSJ[j, k]$  in  $inc\_F$
- 5  $(\forall k) LASKALSJ[j, k] = maxeventID(F_k)$
- 6 send  $(M, inc\_F, seq_{i,a}, j)$  to each  $p_{j,*}$  via RSM layer (to satisfy RSM Total Order and Agreement for receiver ensemble  $p_j$ )
- 7 **when**  $p_{i,a}$  needs to send application message  $M$  to each  $p_{j,*}$  for each  $p_j \in G$ :  $\triangleright$  **Each other correct**  $p_{i,a'}$  **state machine will execute likewise**
- 8  $seq_{i,a} = seq_{i,a} + 1$
- 9 append current send event to  $F_i$ ;  $(\forall k)V[seq_{i,a}, k] = maxeventID(F_k)$
- 10  $(\forall k)$  include history from  $F_k$  after event  $\min_{p_j \in G}(LASKALSJ[j, k])$  in  $inc\_F$
- 11  $(\forall p_j \in G)(\forall k) LASKALSJ[j, k] = maxeventID(F_k)$
- 12 send  $(M, inc\_F, seq_{i,a}, G)$  to each  $p_{j,*}$  for each  $p_j \in G$  via RSM layer (to satisfy RSM Total Order and Agreement— $M$  for each receiver ensemble  $p_j$ )
- 13 **when**  $(M, inc\_F, seq_j, i/G)$  is *SR-delivered* to  $p_{i,a}$  from  $p_j$ :  $\triangleright$  **Happens when**  $t + 1$   
     **identical copies of**  $(M, inc\_F, seq_j, i/G)$  **for**  $seq_j$  **(which equals**  $seq_{j,*}$ **) are**  
     ***TDA-delivered* from**  $p_{j,*}$
- 14 **for all**  $k$  **do**
- 15     **if**  $maxeventID(F_k) < maxeventID(inc\_F_k)$  **then**
- 16         append history of events  $\langle maxeventID(F_k) + 1, \dots, maxeventID(inc\_F_k) \rangle$  from  
         |  $inc\_F_k$  to  $F_k$
- 17  $seq_{i,a} = seq_{i,a} + 1$
- 18 append current receive event to  $F_i$ ;  $(\forall k)V[seq_{i,a}, k] = maxeventID(F_k)$
- 19 **At internal event at**  $p_{i,a}$ :
- 20  $seq_{i,a} = seq_{i,a} + 1$
- 21 append current internal event to  $F_i$ ;  $(\forall k)V[seq_{i,a}, k] = maxeventID(F_k)$
- 22 **To determine**  $e_h^x \rightarrow e_i^*$  **at correct state machine**  $p_{i,a}$  **via call to**  $test(e_h^x \rightarrow e_i^*)$ :
- 23 **if**  $e_h^x$  **is in**  $F_h$  **and**  $*$   $\leq maxeventID(F_i)$  **then**
- 24     return  $(e_h^x \rightarrow e_i^* |_F) \triangleright$  **the test is whether**  $V[*, h] \geq x$
- 25 **else**
- 26     return(*false*)

---

Algorithm 1 gives the processing of control information done at a RSM replica  $p_{i,a}$ . Each RSM replica maintains the following data structures.

1. An integer  $seq_{i,a}$ , initialized to 0, that gives the sequence number of the latest local event at  $p_{i,a}$ .
2. A local  $F$  that is a set of sequences  $F_k$ .  $F$  contains  $p_{i,a}$ 's view of the recorded execution history  $F_k$  of each RSM  $p_k$ .

3. An integer matrix  $LASKALSJ[n, n]$ , where  $LASKALSJ[j, k]$  gives the sequence number of the latest send event by  $p_k$  (as per/from the local  $F_k$ ) at the point in time of the last send event to  $p_{j,*}$ .

This data structure is for efficiently identifying to send to  $p_j$  only the *incremental updates* that have occurred to the local  $F_k$  at  $p_{i,a}$  for each other process  $p_k$ , that need to be transmitted to the destinations  $p_j$  of a message send event since  $p_{i,a}$ 's last send to  $p_j$ .

4.  $p_{i,a}$  also maintains an auxiliary integer matrix  $V[|T(F_i)|, n]$ , where  $V[s, k]$  is  $maxeventID(F_k)$  in  $F(e_{i,a}^s)$ , i.e., the highest sequence number in  $F_k(\in F)$  when the  $s$ th local event  $e_{i,a}^s$  was executed at  $p_{i,a}$ .

Lines 1-6 give the processing for sending a unicast. If multicast can be implemented as a set of independent unicasts, similar code (but with a single increment in line 2) can be executed for sending to each destination of the multicast group. Otherwise a multicast send processing can be implemented via lines 7-12. When a message along with the incremental update  $inc\_F$  (containing the incremental updates for all  $p_k$  as per the sender) is *SR-delivered* to a RSM replica, it updates its  $F_k$  as shown in lines 13-18. A broadcast is a special case of multicast and is hence handled as a multicast. The test for the happens before relation using  $V$  is given in lines 22-26.

In the auxiliary matrix  $V$  at  $p_{i,a}$ , row  $V[w]$  is the vector timestamp [8, 22] of event  $e_{i,a}^w$  and could be stored along with the event in  $F_i$ .  $V[w, j]$  at  $p_{i,a}$  identifies (gives the sequence number of) the event at the surface of the causal past cone of event  $e_{i,a}^w$  at RSM  $p_j$ . At event  $seq_{i,a}$  for each type of event (unicast send (line 3), multicast send (line 9), delivery (line 18), internal (line 21)),  $V[seq_{i,a}, k]$  for all  $k$  is set to  $maxeventID(F_k)$ .  $V$  is used only to implement the test  $e_h^x \rightarrow e_i^*$ , viz.,  $V[*, h] \geq x$ .

#### 4.4 Correctness Proof

Events such as  $e_h^x$  with a single subscript which denotes the application-level process ID of  $p_h$ , are at the application level or RSM-ensemble level. Events such as  $e_{h,a}^x$  with two subscripts denote events at  $smr_{h,a}$ , i.e.,  $p_{h,a}$ , the individual state machine  $sm$  of replica  $a$  of RSM  $p_h$  in its RSM ensemble. Next, we adapt the definitions of  $E$ , of the happens before relation, and of causal past to abstract away the RSM details.

► **Definition 4.** Define  $E\_RSM$  to be the set of all events  $\{e_h^x\}$  such that the events  $e_{h,a}^x$  have occurred at at least majority ( $= t + 1$ ) number of processes  $p_{h,a}$ .

► **Definition 5.** The happens before relation  $\rightarrow_{RSM}$  on events in  $E\_RSM$  (which occur in ensembles of RSMs) consists of the following rules:

1. **Program Order:** For the sequence of events  $\langle e_i^1, e_i^2, \dots \rangle$  executed by RSM ensemble process  $p_i$ ,  $\forall x, y$  such that  $x < y$  we have  $e_i^x \rightarrow_{RSM} e_i^y$ .
2. **Message Order:** If event  $e_j^y$  is a message receive event executed at RSM ensemble process  $p_j$  (i.e., at at least a majority of processes  $p_{j,b}$ ) and there is a corresponding RSM send event  $e_i^x$  in RSM ensemble  $p_i$  (i.e., there are at least a majority events  $e_{i,a}^x$  that are the corresponding message send events at processes  $p_{i,a}$  to RSM ensemble  $p_j$ ), we have  $e_i^x \rightarrow_{RSM} e_j^y$ .
3. **Transitive Order:** If  $e \rightarrow_{RSM} e' \wedge e' \rightarrow_{RSM} e''$  then  $e \rightarrow_{RSM} e''$ .

► **Definition 6.** The RSM-causal past of an event  $e \in E\_RSM$  is denoted as  $CP\_RSM(e)$  and defined as the set of events  $\{e' \in E\_RSM \mid e' \rightarrow_{RSM} e\}$ .

In the causality graph  $(E\_RSM, \rightarrow_{RSM})$ , there is a RSM-causal path from any event in  $CP\_RSM(e)$  to  $e$  comprised of program order edges and message order edges.

► **Lemma 7.** *An event  $e_h^x \in E\_RSM$  occurs at each correct process  $p_{h,z}$  in the RSM ensemble  $p_h$ .*

**Proof.** By definition, an event  $e_h^x \in E\_RSM$  occurs at at least *majority* ( $= t + 1$ ) processes  $p_{h,a}$  in the RSM ensemble  $p_h$ . As at least one of these *majority* processes  $p_{h,a'}$  must be correct and executes  $e_{h,a'}^x$ , and from the Agreement/Agreement-M and Total Order properties of the RSM, each correct *smr*  $p_{h,z}$  will behave identically to  $p_{h,a'}$  and will execute  $e_{h,z}^x$ . ◀

► **Lemma 8.** *An event  $e_{h,z}^x$  that occurs at a correct process  $p_{h,z}$  also occurs as event  $e_h^x$  in the RSM ensemble  $p_h$ .*

**Proof.** As the RSM ensemble works in perfect unision, an event  $e_{h,z}^x$  that occurs at a correct process  $p_{h,z}$  also occurs at all the correct replicas in RSM ensemble  $p_h$  and thus at at least *majority* replicas in that ensemble. Then by Definition 4, the event is also said to occur as  $e_h^x$  in RSM ensemble  $p_h$ . ◀

► **Theorem 9.** *For an event  $e$  at a RSM  $p_i$  ( $e$  must occur at each correct process  $p_{i,z}$  by Lemma 7), the set of events  $T(F)$  when  $e$  is executed at each correct  $p_{i,z}$  is  $CP\_RSM(e)$ .*

**Proof.** There are two parts to this theorem.

1. If an event belongs to  $CP\_RSM(e)$ , the event must belong to  $T(F)$  when event  $e$  is executed at correct process  $p_{i,z}$ .

If event  $e_h^x \in CP\_RSM(e)$  then  $e_h^x$  must be genuine (not fake) and there is a “RSM-causal path” from  $e_h^x$  to  $e$  in  $(E\_RSM, \rightarrow_{RSM})$ . Such a RSM-causal path is comprised of program order edges and message order edges. For each message order edge under  $\rightarrow_{RSM}$  corresponding to a message hop along such a causal path, there are at least *majority* ( $= t + 1$ ) edges under  $\rightarrow$  from each of the at least *majority* ( $= t + 1$ ) correct processes in the sender RSM ensemble to the at least *majority* ( $= t + 1$ ) correct processes in the receiver RSM ensemble. Furthermore, for each program order edge at  $p_j$  under  $\rightarrow_{RSM}$  along the RSM-causal path, all the correct processes in the  $p_j$  ensemble preserve the content of  $F$  at the previous event along the corresponding program order edge under  $\rightarrow$ . Information about  $e_h^x$  gets propagated via  $inc\_F_h$  and  $F_h$  along all such message order edges and program order edges through the processes  $p_j$  along the “RSM-causal path” from  $e_h^x$  to  $e$  and gets inserted in  $F_h$  at  $p_{i,z}$ , as can be seen from lines (1-6) (for unicasts) or (7-12) (for multicasts), and lines (13-18). Once an entry is inserted in  $F_h$  at a correct process, it is never deleted. Note, event  $e \in E\_RSM$  and is specifically some event  $e_i^y$  that, by Lemma 7, must also occur as  $e_{i,z}$  (or  $e_{i,z}^y$ ) at each correct process  $p_{i,z}$ . Thus  $e_h^x \in CP\_RSM(e)$  implies  $e_h^x$  is in  $F_h$  when  $e$  is executed at  $p_{i,z}$ .

2. If an event  $e'$  belongs to  $T(F)$  when event  $e$  is executed at correct process  $p_{i,z}$  (the event  $e$  must also occur at RSM ensemble process  $p_i$  by Lemma 8), the event  $e'$  must belong to  $CP\_RSM(e)$ .

For event  $e_h^x$  in  $F_h$  when  $e_{i,z}^*$  occurs at  $p_{i,z}$ , there are two cases:  $h = i$  and  $h \neq i$ .

- a. First consider  $h = i$ .  $e_{i,z}^x$  (as  $e_i^x$ ) must have been inserted in  $F_i$  at  $e_{i,z}^x$  only in line 3 (for unicast send event) or line 9 (for multicast send event) or line 18 (for receive event) or line 21 (for internal event), and is never deleted by the correct process  $p_{i,z}$  as per the algorithm code. Clearly by Lemma 8,  $e_{i,z}^x$  and  $e_{i,z}^*$  and all events in between at  $p_{i,z}$  must have occurred at RSM ensemble process  $p_i$  as  $p_{i,z}$  is a correct process. As there exist program order edges under  $\rightarrow_{RSM}$  from  $e_i^x$  to  $e_i^*$  and  $p_{i,z}$  is a correct process, it must be that  $e_{h=i}^x \in CP\_RSM(e_i^*)$ .

- b. Consider now  $h \neq i$ . Set  $y$  to  $*$ ,  $i'$  to  $i$ ,  $z'$  to  $z$ .

As event  $e_h^x$  is in  $F_h$  when  $e_{i',z'}^y$  occurs at correct process  $p_{i',z'}$ , then  $e_h^x$  could have been inserted only in line 16 on the *SR-delivery* of a message  $m$  at an event  $e_{i',z'}$  in the causal past of  $e_{i',z'}^y$  (along program order edges under  $\rightarrow$  or under  $\rightarrow_{RSM}$ ) that resulted from the *TOA-delivery* from some (at least *majority*) of processes  $p_{j,a}$  that sent at event  $e_j^w$ . As at least *majority* ( $= t + 1$ ) processes in the preceding sender RSM ensemble reported the same *inc\_F* in the same message for the message to have been *SR-delivered* to  $p_{i,z}$ , the impact of Byzantine processes in the ensemble is filtered out. By Lemma 8 note that receive event  $e_{i'}$  occurs at RSM ensemble  $p_{i'}$  because  $e_{i',z'}$  occurs at a correct process. Also, send event  $e_j^w$  at RSM  $p_j$  must have occurred as at least a *majority* of processes  $p_{j,a}$  sent  $m$ , and so  $e_j^w$  must have occurred at all correct processes  $p_{j,c}$  in the ensemble  $p_j$ . The message  $m$  corresponds to a message order edge under  $\rightarrow_{RSM}$ . Moreover the send event  $e_j^w$  at RSM  $p_j$  belongs to  $CP\_RSM(e_{i'}^y)$  (and hence to  $CP\_RSM(e_{i,z}^* = e)$  by transitivity because  $e_{i'}^y \in CP\_RSM(e)$  as per the previous invocation (if any) of this case 2b).  $e_h^x$  must have existed in  $F_h$  at the time these correct  $p_{j,c}$  sent  $m$  at  $e_{j,c}^w$ . There are two subcases:  $j \neq h$  and  $j = h$ .

- i.  $j \neq h$ . Invoke case 2b but with  $y$  set to  $w$ ,  $i'$  set to  $j$ ,  $z'$  set to  $c$ . This case gets invoked at most  $n - 1$  times as there are  $n - 1$  processes (RSMs)  $p_j$  ( $j \neq h$ ) in the system and  $e_h^x$  gets added to  $F_h$  at correct replica processes  $c$  of any particular  $p_j$  at most once.
- ii.  $j = h$ :  $e_h^x$  must have been inserted in  $F_h$  at  $e_{h(=j),c}^x$  in line 3 (for unicast send event) or line 9 (for multicast send event) or in line 18/21 at a receive/internal event, at  $p_h$ , i.e., at each correct  $p_{j,c}$ . Clearly by Lemma 8,  $e_{j,c}^x$  and  $e_{j,c}^w$  and all events in between at  $p_{j,c}$  must have occurred at RSM ensemble process  $p_{j(=h)}$  as  $p_{j,c}$  is a correct process and hence there exist program order edges under  $\rightarrow_{RSM}$  from  $e_{j(=h)}^x$  to  $e_j^w$ . Moreover such an event  $e'$  (or  $e_h^x$ ) must belong to  $CP\_RSM(e_j^w)$  (and hence to  $CP\_RSM(e_{i,z}^* = e)$  by transitivity because  $e_j^w \in CP\_RSM(e)$  as shown above for case 2b of this invocation).

Combining transitively the above case invocations, it follows that event  $e_h^x$  is in  $F_h$  when  $e = e_{i,z}^*$  is executed at  $p_{i,z}$  implies  $e_h^x \in CP\_RSM(e)$  and  $e_h^x$  cannot be fake.

Both parts of the theorem thus stand proved.  $\blacktriangleleft$

Next we adapt the definition of the *CD* problem to deal with the RSM approach. We assume an oracle that is used for determining correctness of the causality detection algorithm at  $p_{i,z}^*$ ; this oracle has access to  $E\_RSM$  which can be any downward-closed superset of  $CP\_RSM(e_i^*)$ . Also let  $F(e_{i,z}^*)$  be the value of  $F$  at  $p_{i,z}$  when  $e_{i,z}^*$  is executed.

► **Definition 10.** *The causality detection problem  $CD(E\_RSM, F(e_{i,z}^*), e_{i,z}^*)$  for any event  $e_{i,z}^*$  at a correct process  $p_{i,z}$  (where  $e_i^* \in E\_RSM$ ) is to devise an algorithm to collect the execution history of events  $E\_RSM$  as  $F(e_{i,z}^*)$  at  $p_{i,z}$  such that  $valid(F) = 1$ , where*

$$valid(F) = \begin{cases} 1 & \text{if } \forall e_h^x, e_h^x \rightarrow e_{i,z}^* | E\_RSM = e_h^x \rightarrow e_{i,z}^* | F \\ 0 & \text{otherwise} \end{cases}$$

When 1 is returned, the algorithm output matches God's truth and solves *CD* correctly. Thus, returning 1 indicates that the problem has been solved correctly by the algorithm using  $F$ . 0 is returned if either

- $\exists e_h^x$  such that  $e_h^x \rightarrow e_{i,z}^* | E\_RSM = 1 \wedge e_h^x \rightarrow e_{i,z}^* | F = 0$  (denoting a false negative and  $(E\_RSM \cap CP\_RSM(e_{i,z}^*)) \setminus T(F(e_{i,z}^*)) \neq \emptyset$ ), or
- $\exists e_h^x$  such that  $e_h^x \rightarrow e_{i,z}^* | E\_RSM = 0 \wedge e_h^x \rightarrow e_{i,z}^* | F = 1$  (denoting a false positive and  $T(F(e_{i,z}^*)) \setminus E\_RSM \neq \emptyset$ ).

Algorithm 1 produces the output of  $e_h^x \rightarrow e_i^*|_F$  at  $p_{i,a}$  (lines 22-26) via recording  $F$  (lines 1-21). Theorem 9 showed that the set of events in  $E\_RSM$  matched the set of events recorded in  $F$ , even though  $E\_RSM$  is never recorded and is accessible only to an oracle. Next we show in Theorem 11 that using the output of the algorithm and Theorem 9, the causality detection problem  $CD(E\_RSM, F(e_{i,z}^*), e_{i,z}^*)$  is solved, i.e., there are no false positives nor false negatives.

► **Theorem 11.** *There are neither false negatives nor false positives in solving causality detection as per Algorithm 1 for the multicast mode of communication in synchronous systems.*

**Proof.** This theorem has two parts – no false negatives and no false positives – and the proof leverages the two cases in the proof of Theorem 9 which cover the multicast mode of communication. Recall our assumption in Definition 10 that  $p_{i,z}$  is a correct replica. By Lemma 8, event  $e_{i,z}^*$  occurs as  $e_i^*$  in  $E\_RSM$ . In what follows, we use  $CP\_RSM(e_{i,z}^*)$  instead of  $CP\_RSM(e_i^*)$  to emphasize that the reasoning is at  $e_{i,z}^*$  at  $p_{i,z}$ .

1.  $(E\_RSM \cap CP\_RSM(e_{i,z}^*)) \setminus T(F(e_{i,z}^*)) = \emptyset$ . This follows from the first case of Theorem 9 proof because each event in  $CP\_RSM(e_{i,z}^*)$  belongs to  $T(F)$  at  $e_{i,z}^*$ . Let  $e_h^x \in CP\_RSM(e_{i,z}^*)$ . The causality test in lines 22-26 of Algorithm 1 will return *true* because  $e_h^x \in T(F)$  at  $e_{i,z}^*$  and  $V[* , h] = \text{maxeventID}(F_h)$  (when  $e_h^x$  was added to  $T(F)$  at  $p_{i,z}$ , at or before  $e_{i,z}^*$  occurred)  $\geq x$ . Hence  $\nexists e_h^x$  such that  $e_h^x \rightarrow e_{i,z}^*|_{E\_RSM} = 1 \wedge e_h^x \rightarrow e_{i,z}^*|_F = 0$ . Hence there are no false negatives.
2.  $T(F(e_{i,z}^*)) \setminus E\_RSM = \emptyset$ . This follows from the second case of Theorem 9 proof because each event in  $T(F(e_{i,z}^*))$  must also belong to  $CP\_RSM(e_{i,z}^*)$  which is a subset of  $E\_RSM$  by definition. For the causality test of  $e_h^x \rightarrow e_i^*$  at  $p_{i,z}$  in lines 22-26 of Algorithm 1, consider the two cases:  $e_h^x$  is in  $F_h$  and not in  $F_h$ . If  $e_h^x$  is not in  $F_h$ , then by case 1 of Theorem 9 proof,  $e_h^x \notin CP\_RSM(e_{i,z}^*)$  and the test correctly returns *false*. If  $e_h^x$  is in  $F_h$ , then by case 2 of Theorem 9 proof,  $e_h^x \in CP\_RSM(e_{i,z}^*)$  and  $V[* , h] = \text{maxeventID}(F_h)$  (when  $e_h^x$  was added to  $T(F)$  at  $p_{i,z}$  at or before  $e_{i,z}^*$  occurred)  $\geq x$ . Hence the test correctly returns *true*. Hence  $\nexists e_h^x$  such that  $e_h^x \rightarrow e_{i,z}^*|_{E\_RSM} = 0 \wedge e_h^x \rightarrow e_{i,z}^*|_F = 1$ . Hence there are no false positives.

The theorem follows. ◀

As unicast and broadcast are special cases of multicast, the prevention of false positives and of false negatives for multicasts implies the prevention of false positives and of false negatives for unicasts and for broadcasts also. Thus we have the following corollaries to Theorem 11.

► **Corollary 12.** *There are neither false negatives nor false positives in solving causality detection as per Algorithm 1 for the unicast mode of communication in synchronous systems.*

► **Corollary 13.** *There are neither false negatives nor false positives in solving causality detection as per Algorithm 1 for the broadcast mode of communication in synchronous systems.*

## 5 Discussion and Conclusions

We proposed a RSM-based algorithm for solving the causality determination problem  $CD$  in synchronous systems that can have at most  $nt$  Byzantine processes among a total of  $(3t+1)n$  processes partitioned into  $n$  ensembles of  $3t+1$  processes each with each ensemble having up



to  $t$  Byzantine processes. By using  $(3t + 1)n$  processes and the RSM approach to represent  $n$  application processes, the malicious effects of Byzantine process behaviors are neutralized. This is true irrespective of whether the communication mode is by unicasting, multicasting, or broadcasting. The RSM approach works only in synchronous systems. This result is in contrast to the impossibility result for solving the *CD* problem in asynchronous systems in the presence of even a single Byzantine process [25]. It would be interesting to determine whether the *CD* problem can be solved in synchronous systems in the presence of Byzantine processes using a direct approach without using RSMs.

Detecting causality between a pair of events is a fundamental problem [32]. Other problems that use this problem as a building block include the following:

- detecting the interaction type between a pair of intervals at different processes [10],
- detecting the fine-grained modality of a distributed predicate [3, 14], and data-stream based global event monitoring using pairwise interactions between processes [4],
- detecting causality relation between two “meta-events” [11, 13, 15], each of which spans multiple events across multiple processes [12].

It can be shown that these problems in Byzantine failure-prone synchronous systems are solvable because they are reducible to causality detection in the presence of Byzantine processes in synchronous systems.

Byzantine-tolerant causal ordering of messages under unicast mode or multicast mode of communication has been proved to be unsolvable in asynchronous systems [26, 27]. Two forms of safety – *strong safety* (or unconditional safety) and a weaker form of safety called *weak safety* were defined [26], and it was also shown that Byzantine-tolerant causal ordering under broadcast mode of communication in asynchronous systems cannot satisfy strong safety [26] (in a system model in which cryptographic techniques are not allowed). Neither can the algorithm given in [2] for the broadcast mode of communication satisfy strong safety. Algorithms to provide weak safety and liveness of Byzantine-tolerant causal ordering were provided for synchronous systems in [24, 27] (implicitly for unicast mode, multicast mode, and broadcast mode). The use of the RSM approach can be seen to implicitly provide strong safety and liveness of Byzantine-tolerant causal ordering (of unicast mode, multicast mode, and broadcast mode of communication) in synchronous systems as order requirement O2 (Causal order) of the RSM specification is satisfied.

---

## References

- 1 Paulo Sérgio Almeida, Carlos Baquero, and Victor Fonte. Interval tree clocks. In *Proc. 12th International Conference on Principles of Distributed Systems, OPODIS*, pages 259–274, 2008. doi:10.1007/978-3-540-92221-6\_18.
- 2 Alex Auvolat, Davide Frey, Michel Raynal, and François Taïani. Byzantine-tolerant causal broadcast. *Theoretical Computer Science*, 885:55–68, 2021.
- 3 Punit Chandra and Ajay D. Kshemkalyani. Causality-based predicate detection across space and time. *IEEE Trans. Computers*, 54(11):1438–1453, 2005. doi:10.1109/TC.2005.176.
- 4 Punit Chandra and Ajay D. Kshemkalyani. Data-stream-based global event monitoring using pairwise interactions. *J. Parallel Distributed Comput.*, 68(6):729–751, 2008. doi:10.1016/j.jpdc.2008.01.006.
- 5 Bernadette Charron-Bost. Concerning the size of logical clocks in distributed systems. *Inf. Process. Lett.*, 39(1):11–16, 1991. doi:10.1016/0020-0190(91)90055-M.
- 6 Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988. doi:10.1145/42282.42283.



- 7 E.N. Elnozahy. Manetho: Fault tolerance in distributed systems using rollback-recovery and process replication, phd thesis. Technical report, Tech. Report 93-212, Computer Science Department, Rice University, 1993.
- 8 Colin J. Fidge. Logical time in distributed computing systems. *IEEE Computer*, 24(8):28–33, 1991. doi:10.1109/2.84874.
- 9 Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.
- 10 Ajay D. Kshemkalyani. Temporal interactions of intervals in distributed systems. *J. Comput. Syst. Sci.*, 52(2):287–298, 1996. doi:10.1006/jcss.1996.0022.
- 11 Ajay D. Kshemkalyani. Reasoning about causality between distributed nonatomic events. *Artif. Intell.*, 92(1-2):301–315, 1997. doi:10.1016/S0004-3702(97)00004-0.
- 12 Ajay D. Kshemkalyani. Causality and atomicity in distributed computations. *Distributed Comput.*, 11(4):169–189, 1998. doi:10.1007/s004460050048.
- 13 Ajay D. Kshemkalyani. A framework for viewing atomic events in distributed computations. *Theor. Comput. Sci.*, 196(1-2):45–70, 1998. doi:10.1016/S0304-3975(97)00195-3.
- 14 Ajay D. Kshemkalyani. A fine-grained modality classification for global predicates. *IEEE Trans. Parallel Distributed Syst.*, 14(8):807–816, 2003. doi:10.1109/TPDS.2003.1225059.
- 15 Ajay D. Kshemkalyani and Roshan Kamath. Orthogonal relations for reasoning about posets. *Int. J. Intell. Syst.*, 17(12):1101–1110, 2002. doi:10.1002/int.10062.
- 16 Ajay D. Kshemkalyani and Anshuman Misra. The bloom clock to characterize causality in distributed systems. In *The 23rd International Conference on Network-Based Information Systems, NBIS 2020*, volume 1264 of *Advances in Intelligent Systems and Computing*, pages 269–279. Springer, 2020. doi:10.1007/978-3-030-57811-4\_25.
- 17 Ajay D. Kshemkalyani, Min Shen, and Bhargav Voleti. Prime clock: Encoded vector clock to characterize causality in distributed systems. *J. Parallel Distributed Comput.*, 140:37–51, 2020. doi:10.1016/j.jpdc.2020.02.008.
- 18 Ajay D. Kshemkalyani and Mukesh Singhal. *Distributed Computing: Principles, Algorithms, and Systems*. Cambridge University Press, 2011. doi:10.1017/CB09780511805318.
- 19 Sandeep S. Kulkarni, Murat Demirbas, Deepak Madappa, Bharadwaj Avva, and Marcelo Leone. Logical physical clocks. In *Proc. 18th International Conference on Principles of Distributed Systems, OPODIS*, pages 17–32, 2014. doi:10.1007/978-3-319-14472-6\_2.
- 20 Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* 21, 7, pages 558–565, 1978.
- 21 Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982. doi:10.1145/357172.357176.
- 22 Friedemann Mattern. Virtual time and global states of distributed systems. In *Parallel and Distributed Algorithms*, pages 215–226. North-Holland, 1988.
- 23 Anshuman Misra and Ajay D. Kshemkalyani. The bloom clock for causality testing. In Diganta Goswami and Truong Anh Hoang, editors, *Proc. 17th International Conference on Distributed Computing and Internet Technology*, volume 12582 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2021. doi:10.1007/978-3-030-65621-8\_1.
- 24 Anshuman Misra and Ajay D. Kshemkalyani. Causal ordering in the presence of byzantine processes. In *28th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, 2022. doi:10.1109/ICPADS56603.2022.00025.
- 25 Anshuman Misra and Ajay D. Kshemkalyani. Detecting causality in the presence of byzantine processes: There is no holy grail. In *21st IEEE International Symposium on Network Computing and Applications (NCA)*, pages 73–80, 2022. doi:10.1109/NCA57778.2022.10013644.
- 26 Anshuman Misra and Ajay D. Kshemkalyani. Solvability of byzantine fault-tolerant causal ordering problems. In Mohammed-Amine Koulali and Mira Mezini, editors, *Networked Systems*, pages 87–103, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-17436-0\_7.

- 27 Anshuman Misra and Ajay D. Kshemkalyani. Byzantine fault-tolerant causal ordering. In *24th International Conference on Distributed Computing and Networking (ICDCN)*, pages 100–109, 2023. doi:10.1145/3571306.3571395.
- 28 Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980. doi:10.1145/322186.322188.
- 29 Tommaso Pozzetti and Ajay D. Kshemkalyani. Resettable encoded vector clock for causality analysis with an application to dynamic race detection. *IEEE Trans. Parallel Distributed Syst.*, 32(4):772–785, 2021. doi:10.1109/TPDS.2020.3032293.
- 30 Nuno M. Prego, Carlos Baquero, Paulo Sérgio Almeida, Victor Fonte, and Ricardo Gonçalves. Brief announcement: efficient causality tracking in distributed storage systems with dotted version vectors. In *ACM Symposium on Principles of Distributed Computing, PODC*, pages 335–336, 2012. doi:10.1145/2332432.2332497.
- 31 Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, 1990. doi:10.1145/98163.98167.
- 32 Reinhard Schwarz and Friedemann Mattern. Detecting causal relationships in distributed computations: In search of the holy grail. *Distributed Comput.*, 7(3):149–174, 1994. doi:10.1007/BF02277859.
- 33 Mukesh Singhal and Ajay D. Kshemkalyani. An efficient implementation of vector clocks. *Inf. Process. Lett.*, 43(1):47–52, 1992. doi:10.1016/0020-0190(92)90028-T.
- 34 Francisco J. Torres-Rojas and Mustaque Ahamad. Plausible clocks: Constant size logical clocks for distributed systems. *Distributed Computing*, 12(4):179–195, 1999. doi:10.1007/s004460050065.
- 35 Paul A. S. Ward and David J. Taylor. A hierarchical cluster algorithm for dynamic, centralized timestamps. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS 2001)*, pages 585–593, 2001. doi:10.1109/ICDSC.2001.918989.

# Embarrassingly Greedy Inconsistency Resolution of Qualitative Constraint Networks

Michael Sioutis   

LIRMM UMR 5506, Université de Montpellier & CNRS, France

---

## Abstract

In this paper, we deal with inconsistency resolution in qualitative constraint networks (QCN). This type of networks allows one to represent and reason about spatial or temporal information in a natural, human-like manner, e.g., by expressing relations of the form  $x \{is\ north\ of \vee\ is\ east\ of\} y$ . On the other hand, inconsistency resolution involves maximizing the amount of information that is consistent in a knowledge base; in the context of QCNs, this translates to maximizing the number of constraints that can be satisfied, via obtaining a qualitative solution (scenario) of the QCN that ignores/violates as few of the original constraints as possible. To this end, we present two novel approaches: a greedy constraint-based and an optimal Partial MaxSAT-based one, with a focus on the former due to its simplicity. Specifically, the greedy technique consists in adding the constraints of a QCN to a new, initially empty network, one by one, all the while filtering out the ones that fail the satisfiability check. What makes or breaks this technique is the ordering in which the constraints will be processed to saturate the empty QCN, and for that purpose we use many different strategies to form a portfolio-style implementation. The Partial MaxSAT-based approach is powered by Horn theory-based maximal tractable subsets of relations. Finally, we compare the greedy approach with the optimal one, commenting on the trade-off between obtaining repairs that are optimal and obtaining repairs in a manner that is fast, and make our source code available for anyone to use.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Constraint and logic programming; Computing methodologies  $\rightarrow$  Temporal reasoning

**Keywords and phrases** Spatial and Temporal Reasoning, Qualitative Constraints, Inconsistency Resolution, Maximizing Satisfiability, Greedy Algorithm, Partial MaxSAT Solver

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.12

**Supplementary Material** *Software:* <https://seafiler.lirmm.fr/d/6127423776d145bab11a/>

**Funding** *Michael Sioutis:* The work was partially funded by the Agence Nationale de la Recherche (ANR) for the “Hybrid AI” project that is tied to the chair of Dr. Sioutis, and the I-SITE program of excellence of Université de Montpellier that complements the ANR funding.

## 1 Introduction

Qualitative Spatio-Temporal Reasoning (QSTR) is a rich symbolic AI framework that deals with representing and reasoning about abstract, qualitative spatio-temporal information [8, 15]. Specifically, QSTR allows one to spatially or temporally relate one object with another object or oneself by using everyday, human-like natural language descriptions, and perform reasoning with those descriptions; as an example, consider a relation of the form  $x \{is\ north\ of \vee\ is\ east\ of\} y$ , which abstracts from numerical information and yet is very intuitive. Such QSTR descriptions or relations, and disjunctions thereof, can be modeled as a qualitative constraint network (QCN), a simplified example of which is provided in Figure 1a. Spatial or temporal information in the QSTR framework can, in general, pertain to any spatial or temporal aspects in the physical world. However, the literature has been deeply invested in point/interval-based calculi, with Allen’s Interval Algebra being the most representative example [1], as intervals can be used to represent and reason about



© Michael Sioutis;

licensed under Creative Commons License CC-BY 4.0

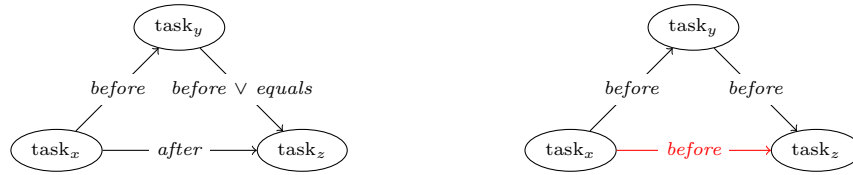
30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 12; pp. 12:1–12:12

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



(a) An inconsistent plan as a simplified QCN. (b) An *optimal* scenario of the simplified QCN.

■ **Figure 1** An illustration of the MAX-QCN problem of a qualitative constraint network (QCN) [6] and the terminology used here; the QCN in Figure 1a is inconsistent, and one solution of the MAX-QCN problem, viz., an *optimal* scenario, is depicted in Figure 1b, where  $\text{task}_x \{before\} \text{task}_z$  is the only relation that does not satisfy the respective constraint in Figure 1a..

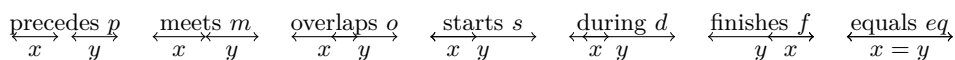
anything from durative actions in planning or tasks in robotics [18] to temporal abstractions in multivariate time series classification [17], among other applications; the interested reader is invited to explore the discussion in [26, 28, 9, 3].

### Context & Motivation

In this paper, we focus on the problem of maximizing satisfiability in a qualitative constraint network, formally called the MAX-QCN problem [6]. Specifically, given a QCN  $\mathcal{N}$ , the MAX-QCN problem is the problem of obtaining a spatial or temporal configuration that maximizes the number of satisfied constraints in  $\mathcal{N}$ ; see also Figure 1 for an example. The motivation behind studying this problem lies in the fact that representing spatial or temporal information may inevitably lead to inconsistencies, due to e.g. human error and/or inaccurate classifiers. As illustration, timetabling is an instance of scheduling where inconsistencies can naturally form due to the lack of resources for certain tasks, among other reasons [14]. Specifically, in timetabling the goal is to associate temporal intervals with a number of tasks requiring limited resources. In the context of a hospital, for example, an inconsistency can occur when two surgeons are allocated the same operating room in overlapping temporal intervals; the inconsistency must then be repaired by considering available temporal intervals and preferences alike, and minimizing changes so as to perturb the structure of the timetable as little as possible. In the broader context of neuro-symbolic AI architectures [13], classifiers may construct inconsistent spatio-temporal knowledge bases due to inaccurate predictions, and minimizing inconsistency (i.e., maximizing satisfiability) is an essential step of logical abduction (or other type of reasoning) in the neuro-symbolic cycle, see, e.g., Figure 1 in [31].

### State of the Art & Contribution

The state of the art in solving the MAX-QCN problem with respect to *constraints* and *SAT encodings* consists of the works in [6] and in [7], respectively. Specifically, both of these approaches try to obtain a refinement of the input QCN that maximizes the number of satisfied constraints in the QCN. In doing so, they are trying to solve two problems of different nature at the same time: extracting a scenario of the QCN, whilst ensuring that the extracted scenario is optimal. This is particularly crippling for the performance of the constraint-based approach in [6], as, should the constraint not be part of an optimal scenario in the end, taking a refinement of it in the beginning might create a huge branch in the search tree that is useless to explore. The clause learning of the SAT-based approach in [7] circumvents this issue, but, on the other hand, [7] does not exploit tractability properties for QCNs, viz., *Horn theories* and/or *maximal tractable subsets* of relations [22]; nevertheless,



■ **Figure 2** A representation of the 13 base relations  $b$  of IA, each one relating two potential intervals  $x$  and  $y$  as in  $x b y$ ; the converse of  $b$ , i.e.,  $b^{-1}$ , can be denoted by  $bi$  and is omitted in the figure.

it significantly outperforms [6]. Here, with respect to the previous discussion, we make the following contributions:

- (i) We offer a greedy constraint-based approach for tackling the MAX-QCN problem that treats the constraints of the input QCN in whole and, hence, may avoid – to a relatively greater extent – redundant exploration of search space;
- (ii) We introduce one of the most compact to date Partial MaxSAT encodings for the MAX-QCN problem by extending the SAT encoding of [20] (see also [30]), fully utilizing tractability properties (alongside chordal completions of the constraint graphs of QCNs);
- (iii) We pit the two approaches against each other in an experimental evaluation, and comment on the trade-off between obtaining repairs in an inconsistent QCN in a way that is optimal and coming close to a solution of the MAX-QCN problem in a manner that is fast, making our source code available for any interested researcher to use.

## Organization

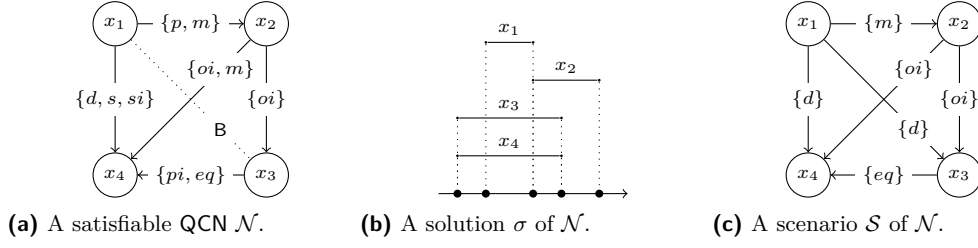
The rest of the paper is organized as follows. In Section 2 we provide definitions and notations regarding QSTR and the MAX-QCN problem that are necessary for following and understanding the paper. Then, Sections 3–5 expand on the contribution points (i)–(iii), respectively, that were listed earlier. Finally, in Section 6 we conclude 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, called *base relations* [15] and defined over an infinite domain  $D$  (e.g.,  $\mathbb{R}$ ). 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  $\text{Id}$ , 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  $\diamond$  [15]. For all  $r \in 2^B$ , we have that  $r^{-1} = \bigcup\{b^{-1} \mid b \in r\}$ . The weak composition ( $\diamond$ ) of two base relations  $b, b' \in B$  is defined as the smallest (i.e., most restrictive) relation  $r \in 2^B$  that includes  $b \circ b'$ , or, formally,  $b \diamond b' = \{b'' \in B \mid b'' \cap (b \circ b') \neq \emptyset\}$ , where  $b \circ b' = \{(x, y) \in D \times D \mid \exists z \in D \text{ such that } (x, z) \in b \wedge (z, y) \in b'\}$  is the (true) composition of  $b$  and  $b'$ . For all  $r, r' \in 2^B$ , we have that  $r \diamond r' = \bigcup\{b \diamond b' \mid b \in r, b' \in r'\}$ .

As illustration, consider the well-known qualitative temporal constraint language of Interval Algebra (IA) [1]. IA considers time intervals on the real line, and the set of base relations  $B = \{eq (= \text{Id}), p, pi, m, mi, o, oi, s, si, d, di, f, fi\}$  to encode knowledge about the temporal relations between such intervals, as described in Figure 2.

Representing and reasoning about qualitative spatio-temporal information pertaining to a set of base relations  $B$  can be facilitated by a *qualitative constraint network* (QCN):



■ **Figure 3** Figurative examples of QCN terminology using Interval Algebra (IA).

- **Definition 1.** A qualitative constraint network (QCN) is a tuple  $(V, C)$  where:
- $V = \{v_1, \dots, v_n\}$  is a non-empty finite set of variables (representing entities in  $\mathcal{D}$ );
  - and  $C$  is a mapping  $C : V \times V \rightarrow 2^{\mathcal{B}}$  such that,  $\forall v \in V$ ,  $C(v, v) = \{\text{ld}\}$ , and,  $\forall v, v' \in V$ ,  $C(v, v') = (C(v', v))^{-1}$ .

An example QCN of IA is shown in Figure 3a; for conciseness, converse relations or  $\text{ld}$  loops are not shown in the figure.

- **Definition 2.** Let  $\mathcal{N} = (V, C)$  be a QCN (Figure 3a), then:
- a solution of  $\mathcal{N}$  is a mapping  $\sigma : V \rightarrow \mathcal{D}$  such that,  $\forall (u, v) \in V \times V$ ,  $\exists b \in C(u, v)$  such that  $(\sigma(u), \sigma(v)) \in b$ ; and  $\mathcal{N}$  is satisfiable iff it admits a solution (see Figure 3b);
  - a sub-QCN (also known as refinement)  $\mathcal{N}'$  of  $\mathcal{N}$ , denoted by  $\mathcal{N}' \subseteq \mathcal{N}$ , is a QCN  $(V, C')$  such that,  $\forall u, v \in V$ ,  $C'(u, v) \subseteq C(u, v)$ ;
  - $\mathcal{N}$  is atomic iff,  $\forall v, v' \in V$ ,  $C(v, v')$  is a singleton relation, i.e., a relation  $\{b\}$  with  $b \in \mathcal{B}$ ;
  - a scenario  $\mathcal{S}$  of  $\mathcal{N}$  is an atomic satisfiable sub-QCN of  $\mathcal{N}$  (see Figure 3c);
  - the constraint graph of  $\mathcal{N}$ , denoted by  $\mathcal{G}(\mathcal{N})$ , is the graph  $(V, E)$  where  $\{u, v\} \in E$  iff  $C(u, v) \neq \mathcal{B}$  and  $u \neq v$ ;
  - for  $V' \subseteq V$ ,  $\mathcal{N}_{\downarrow V'}$  denotes  $\mathcal{N}$  restricted to  $V'$ ;
  - $\mathcal{N}$  is denoted by  $\mathcal{N}_{\top}$  when each of its constraints is universal, i.e., iff,  $\forall v, v' \in V$  with  $v \neq v'$ ,  $C(v, v') = \mathcal{B}$ .

### The MAX-QCN problem

The MAX-QCN problem has been introduced in the context of QSTR in [6]. Given a QCN  $\mathcal{N}$  over a set of variables  $V$ , the MAX-QCN problem is the problem of finding a scenario over  $V$  that maximizes the number of satisfied constraints in  $\mathcal{N}$ , or, equivalently, the problem of finding a scenario over  $V$  that minimizes the number of unsatisfied constraints in  $\mathcal{N}$ . Such scenarios are called *optimal* scenarios of  $\mathcal{N}$ . Clearly, if a QCN  $\mathcal{N}$  is satisfiable, any scenario of  $\mathcal{N}$  is also an optimal scenario of  $\mathcal{N}$ . The reader is kindly asked to revisit Figure 1 in the introduction for a simplified example of the MAX-QCN problem and a solution of it. Solving the MAX-QCN problem is clearly at least as difficult as solving the satisfiability checking problem of a QCN, which is NP-hard in general for most calculi [8].

## 3 Greedy Constraint-based Approach

In this section, we present a greedy approach to come close to, or even exactly identify, a maximum satisfiable subset of constraints of an original input QCN  $\mathcal{N} = (V, C)$  and, hence, tackle the MAX-QCN problem. This approach is presented in Algorithm 1, and it consists in consistently saturating a universal QCN (lines 4–13) with as many constraints as possible from  $\mathcal{N}$ , by using and iterating various different orderings of the constraints of  $\mathcal{N}$  (line 5).

■ **Algorithm 1** GREEDUS( $\mathcal{N}, \mathcal{A}$ ).

---

```

in      : A QCN  $\mathcal{N} = (V, C)$  and a set  $\mathcal{A}$  of bijections  $\alpha : E \rightarrow \{0, 1, \dots, |E| - 1\}$ , where
            $E = E(\mathbf{G}(\mathcal{N}))$  (i.e., roughly, a set of orderings of the constraints in  $\mathcal{N}$ )
out     : A subset  $p \subseteq E(\mathbf{G}(\mathcal{N}))$  corresponding to feasible constraints in  $\mathcal{N}$ 
1  $P \leftarrow \emptyset$ ;
2 foreach  $\alpha \in \mathcal{A}$  do
3    $p \leftarrow \emptyset$ ;
4    $\mathcal{N}' = (V, C') \leftarrow \mathcal{N}_\tau$ ;
5   for  $i$  from 0 to  $|E(\mathbf{G}(\mathcal{N}))| - 1$  do
6      $\{u, v\} \leftarrow \alpha^{-1}(i)$ ;
7      $C'(u, v) \leftarrow C(u, v)$ ;
8      $C'(v, u) \leftarrow C(v, u)$ ;
9     if SAT( $\mathcal{N}'$ ) then
10       $p \leftarrow p \cup \{\{u, v\}\}$ ;
11    else
12       $C'(u, v) \leftarrow \mathbf{B}$ ;
13       $C'(v, u) \leftarrow \mathbf{B}$ ;
14     $P \leftarrow P \cup \{p\}$ ;
15 return  $p \in \arg \max_{p' \in P} (|p'|)$ ;
```

---

Given a QCN  $\mathcal{N} = (V, C)$ , with  $E = E(\mathbf{G}(\mathcal{N}))$  denoting the set of edges in its constraint graph, GREEDUS runs in  $O(|E| \cdot \beta)$  time, where  $\beta$  is the runtime of a SAT oracle call. The SAT oracle here can be any solver that can solve the satisfiability checking problem of a QCN, be it SAT- or qualitative constraint-based; in our implementation of the algorithm, we opted for a qualitative constraint-based one, since it made the implementation of the algorithm more straightforward. Of course, we assume here that the size of the set  $\mathcal{A}$  of some orderings of the constraints in  $\mathcal{N}$  is upper bounded by a small constant  $k$  that is equal to the number of different strategies that will be used to obtain these orderings in the first place (a discussion on such strategies follows immediately after); this would be naturally the case, as exploring all possible orderings would defeat the purpose of being greedy. Finally, it is important to know that each iteration of the loop in line 5 can be run in parallel, as the calculation of a satisfiable subset of constraints  $p$  by the end of an iteration is completely independent to any other such  $p$ ; in the end, the largest such  $p$  is returned. However, in our implementation we maintained the sequential nature of the algorithm.

## Constraint Ordering Strategies

Given a QCN  $\mathcal{N} = (V, C)$ , the effectiveness of GREEDUS relies heavily on the set  $\mathcal{A}$  of some orderings of the constraints in  $\mathcal{N}$  that will be provided as part of its input, as this set has a direct effect on the quality of the satisfiable subset of constraints that will be obtained in the end. It is worth noting that the efficiency of GREEDUS does not rely all that much on  $\mathcal{A}$ , as the algorithm will go through all constraints anyway (of course, in the sequential version of the algorithm, some optimizations can be achieved by passing information from one iteration to the next one, to early stop the loop, for example).

**Intuition.** We would like to delay the encounter of a constraint that causes inconsistency (line 9 in the algorithm) for as long as possible, as this should allow us to maximize the size of the set of satisfiable constraints. So, intuitively, we should order constraints from *more permissive* to *less permissive*, as this should increase our chances of a relatively more successful outcome. In the sequel, we list ways to assess the permissiveness of a constraint.



In qualitative constraint-based reasoning, the satisfiability checking of a QCN is done via the use of a backtracking algorithm [22], where the selection of the next constraint to process follows the *minimum remaining values* principle in traditional constraint programming [23] (commonly known as MRV); specifically, heuristics are used to select the more restrictive constraints first, as this should help the algorithm to explore a relatively sparser search tree. Here, we simply reverse the use of such constraint selection heuristics, making small adaptations where necessary, which we explain in what follows.

In sum, among other heuristics, we use the *local model* counting-based heuristics of [25], as well as the weighting-based ones of [27, 21], to order the constraints from more permissive to less permissive (or, equivalently, from less restrictive to more restrictive).

First, we need to recall and slightly adapt the definition of a local model from [25].

► **Definition 3** (local model, cf. [25]). *Given a QCN  $\mathcal{N} = (V, C)$  and an edge  $\{v, v'\} \in E(G(\mathcal{N}))$ , a local model of a base relation  $b \in C(v, v')$  is a scenario  $S = (V', C')$  of  $\mathcal{N} \downarrow_{V'}$ , where  $V' = \{v, v', u\}$  with  $u \in V$  ( $V'$  is a triple of variables in  $V$ ), and  $C'(v, v') = \{b\}$ .*

Now, we are ready to list all of the used constraint ordering strategies in this work. It is clear that, given a QCN  $\mathcal{N} = (V, C)$ , an exhaustive application of either of the following strategies for each of the (non-universal) constraints of  $\mathcal{N}$  provides an ordering of the constraints of  $\mathcal{N}$ ; we can then represent those orderings with bijections  $E \rightarrow \{0, 1, \dots, |E| - 1\}$ , where  $E = E(G(\mathcal{N}))$ , and form the required set of orderings for GREEDUS.

- **max**: choose the constraint that contains the base relation with the most local models.
- **min**: choose the constraint for which the base relation with the fewest local models has the most local models compared to such base relations of the rest of the constraints.
- **avg**: choose the constraint with the highest average count of local models (i.e., each of its base relations contributes a count and we take the average of these counts).
- **sum**: choose the constraint with the highest cumulative count of local models. (i.e., each of its base relations contributes a count and we take the sum of these counts).
- **weight**: choose the constraint with the largest weight; see, e.g., Figure 9 in [27] (the larger the weight, the more permissive the constraint).
- **card**: choose the constraint whose smallest decomposition into sub-relations of a (maximal) tractable subset  $\mathcal{S} \in 2^{\mathcal{B}}$  [21] (e.g., the ORD-Horn set for IA [20]) is the largest one.
- **card + weight**: the **card** heuristic, with the **weight** heuristic acting as tie-breaker (this is very typical in the literature e.g., [21]).
- **random**: choose a constraint randomly.

The reader can note that the aforementioned strategies are very different to one another, even contradictory at times (e.g., **max** and **min**). In fact, such a mix of different strategies ensures that our portfolio-style approach is diverse enough; diversity is an important aspect of any portfolio-based method.

## 4 Optimal Partial MaxSAT-based Approach

In this section, we introduce a Partial MaxSAT encoding for the MAX-QCN problem by extending the SAT encoding of [20]; we note that the aforementioned encoding pertains to the IA calculus, but the approach itself may be adapted to any calculus by using the hard clauses to encode a theory of the calculus and the soft ones to encode the constraints of an input QCN over that calculus – e.g., a similar encoding exists for RCC8 in [29]. It must be noted that, contrary to the approach of [7], which does not take into account a theory of a

calculus and aims to provide a generic approach that is based solely on the weak composition rules of that calculus, our extension may take full advantage of tractability properties for QCNs, viz., Horn theory-based *maximal tractable subsets* of relations [22], and is thus one of the most compact encodings for the MAX-QCN problem to date, see also Table 1 in [30].

First, we briefly introduce some notions about the Partial MaxSAT problem. A literal is a propositional variable or its negation, and a clause is a disjunction of literals. The maximum satisfiability problem (MaxSAT) is the problem of finding an assignment that satisfies as many clauses of a given set of clauses as possible [12]. Hence, the MAX-QCN problem can already be viewed as a version of the MaxSAT problem for QCNs. The Partial MaxSAT problem is an extension of the MaxSAT problem defined as follows: an instance  $\Omega$  of Partial MaxSAT [16, 5] is a set of clauses composed of hard and soft clauses, and a solution  $\omega$  of  $\Omega$  is an assignment that satisfies the hard clauses and maximizes the number of satisfied soft clauses. For the MAX-QCN problem, certain hard clauses are necessary to ensure the completeness of the approach, in particular, the clauses that pertain to a provided theory of a given calculus, as we will demonstrate in the sequel.

We first introduce our Partial MaxSAT encoding for a given QCN in an abstract way, and then give an example based on IA and the SAT encoding in [20]. Given a QCN  $\mathcal{N} = (V, C)$  over some calculus  $\mathcal{C}$ , the hard clauses in the Partial MaxSAT encoding are the ones encoding a theory of  $\mathcal{C}$ , the set of these clauses being denoted by  $\text{Th}_{\mathcal{C}}(\mathcal{N})$ , and the soft clauses in the Partial MaxSAT encoding are the ones encoding the constraints of  $\mathcal{N}$ , the set of these clauses being denoted by  $\text{In}_{\mathcal{C}}(\mathcal{N})$ . Specifically, regarding  $\text{In}_{\mathcal{C}}(\mathcal{N})$ , the soft clauses can be viewed as follows (an explanation of the symbols follows immediately after):

$$\bigwedge_{(i,j) \in E(\mathbf{G}(\mathcal{N})) \text{ s.t. } i < j} (r_{ij} \rightarrow \bigwedge_{l=1}^m c_l) \quad (1)$$

With respect to Equation (1) above,  $r_{ij}$  is an auxiliary variable associated with every  $(i, j) \in E(\mathbf{G}(\mathcal{N}))$  s.t.  $i < j$ , and complementing every clause  $c_l$  of a CNF formula  $c_1 \wedge c_2 \wedge \dots \wedge c_m$  corresponding to the constraint  $C(i, j)$  (here,  $m$  is some small constant that is particular to the CNF encoding of a constraint in a given calculus). The soft part in Equation (1) is simply the set of these  $r_{ij}$  unit clauses: maximizing the number of satisfied clauses of the form  $r_{ij}$  corresponds to maximizing the number of satisfied constraints of the form  $C(i, j)$ .

Let us ground the presentation so far in IA to facilitate the reader. A Horn theory of IA can be based on that of partial orders, as is done in [20]. We present this theory as follows:

$$\begin{aligned} x \leq z \wedge z \leq y \rightarrow x \leq y & \quad x = y \rightarrow x \leq y \\ x \leq y \wedge y \leq x \rightarrow x = y & \quad x = y \rightarrow y \leq x \\ x = y \wedge x \neq y \rightarrow \perp & \quad x \neq x \rightarrow \perp \end{aligned}$$

Then, we consider the usual domain  $\mathbf{D}$  of IA, which is defined as the set of intervals on the real line, i.e.,  $\mathbf{D} = \{x = (x^-, x^+) \in \mathbb{R} \times \mathbb{R} \mid x^- < x^+\}$ , where  $x^-$  and  $x^+$  denote the starting point and ending point of an interval  $x$ , respectively.

Given a QCN  $\mathcal{N} = (V, C)$  over IA, every interval variable  $x \in V$  can be translated with regard to the theory of partial orders as follows (remember that,  $\forall x \in V, x^- < x^+$ ):

$$x^- \leq x^+ \wedge x^- \neq x^+$$

In addition, for all distinct interval variables  $x, y, z \in V$ , we need to enforce the theory of partial orders mentioned earlier and obtain the respective translations for all of their starting and ending points (with respect to a chordal completion of  $E(\mathbf{G}(\mathcal{N}))$ ).

The hard clauses of  $\text{Th}_{\text{IA}}(\mathcal{N})$  can then be straightforwardly obtained by associating, for all  $s \in \{-, +\} \times \{\leq, =\} \times \{-, +\}$ , the propositional variables  $p_{xy}^s$  with every pair of interval variables  $x, y \in V$ , and retrieving the SAT encoding of the aforementioned translations. For example the formula corresponding to an interval variable (viewed within the theory of partial orders as above, viz.,  $x^- \leq x^+ \wedge x^- \neq x^+$ ) is as follows:

$$p_{xx}^{(-, \leq, +)} \wedge \neg p_{xx}^{(-, =, +)}$$

With respect to the soft clauses of  $\text{In}_{\text{IA}}(\mathcal{N})$ , and the SAT encoding of the constraints in particular, it can be easily obtained by considering the definition of each base relation of IA with respect to the starting and ending points of two intervals, and its subsequent translation with regard to the theory of partial orders. For example, the base relation *during* between two intervals  $x$  and  $y$  is defined as  $\{(x, y) \in \text{D} \times \text{D} \mid y^- < x^- \wedge x^+ < y^+\}$ ; we already saw earlier how  $<$  corresponds to  $\leq \wedge \neq$  with regard to the theory of partial orders, so the translation is obvious. By extension, the SAT encoding of composite relations (disjunctions of base relations) can be obtained via the disjunction of the SAT encodings of the base relations in the composite relation (which can then be transformed to CNF).

## 5 Experimentation

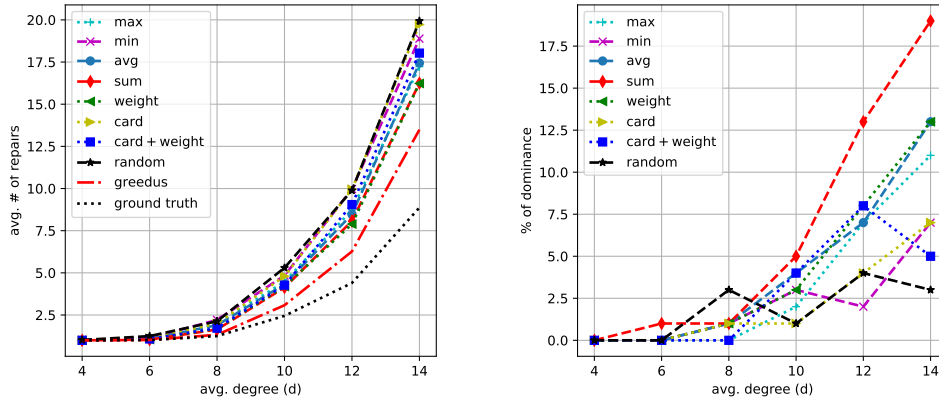
In this section, with respect to tackling the MAX-QCN problem, we perform an experimental evaluation between an in-house implementation of GREEDUS introduced in Section 3 (Algorithm 1), and an implementation of the Partial PaxSAT encoding introduced in Section 4 using the PySAT toolkit [10] and the RC2 MaxSAT solver offering there [11].

► Note 4. All the code is available at: <https://msioutis.gitlab.io/software/>

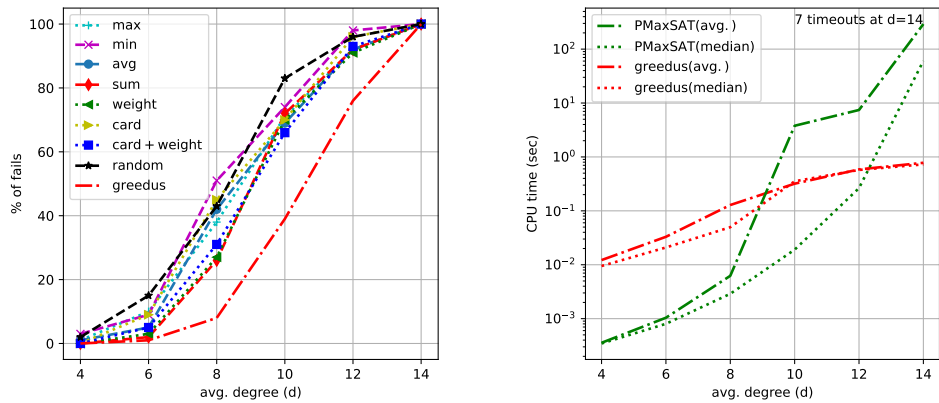
### Dataset & Setup

We kept the dataset consistent with what has been used in previous works on the MAX-QCN problem for comparability, cf. [6, 7]. Specifically, we considered IA network instances generated by the standard  $\text{A}(n, d, l)$  model [21], used extensively in the literature. In short,  $\text{A}(n, d, l)$  creates network instances of size  $n$ , average constraint graph degree  $d$ , and an average number  $l$  of base relations per constraint. We set  $n = 20$  and  $l = 6.5$ , and we considered 100 inconsistent network instances for *each* degree  $d$  between 4 and 14 with a 2-degree step; hence, 600 network instances in total. For this range of degrees  $d$ , the network instances of model  $\text{A}(n, d, l)$  lie within the *phase transition* region [19]. Again, the nature and size of the network instances is consistent with what has been used in the literature for the MAX-QCN problem in order to present results that are comparable and as complete as possible, cf. [6, 7] (see also the number of timeouts in Figure 4d for the dense instances). For the experiments we used an Intel® Core™ CPU i7-12700H @ 4.70GHz, 16 GB of RAM, and the Ubuntu Linux 22.04 LTS OS, and one CPU core per network. All coding/running was done in Python 3; however, we must note that the implementation of GREEDUS was sped up with PyPy,<sup>1</sup> which comes bundled with a just-in-time compiler, whereas the same is not possible for the implementation of the Partial MaxSAT encoding, because the RC2 MaxSAT solver in PySAT uses Glucose 3 [2] as the underlying SAT oracle, which is coded in C/C++.

<sup>1</sup> <https://www.pypy.org/>



(a) Avg. # of repairs required per approach. (b) % of dominance per heuristic.



(c) % of fails to find optimal value per approach. (d) Runtime performance of main approaches.

■ **Figure 4** Assessing the performance of an implementation of GREEDUS and of our Partial MaxSAT encoding, respectively, with Interval Algebra (IA) network instances of model  $A(n = 20, d, l = 6.5)$  [21]; a timeout occurs after 3600s, and in that case the runtime up to that point is not taken into account (only 7 such timeouts occurred, all for the Partial MaxSAT-based implementation at  $d = 14$ ).

### Results & Remarks

All of the experimental results are concisely presented in Figure 4. In Figure 4a we evaluate how the different strategies that are implemented under the hood of GREEDUS behave with respect to obtaining repairs in an inconsistent QCN if they are run standalone (see Section 3 for a description of these strategies), and how they define the respective behaviour of GREEDUS when taken all together; the ground truth here is the optimal value. The best performing strategies with respect to obtaining few repairs are *sum* and *weight*, and the worst performing one is *random*; however, as we will see in the sequel, no strategy goes to waste in this portfolio-style implementation. With respect to our last point, in Figure 4b we observe the percentage of times that a strategy *dominated* all others, where by “dominated” we mean that the strategy obtained a number of repairs that was strictly smaller than that of any other strategy. Somewhat surprisingly, the worst strategy when it comes to obtaining few repairs, viz., *random*, was still able to dominate all others at least a couple of times per avg.

degree  $d$ . This means that, by removing random, we would obtain a slightly worse result for GREEDUS in Figure 4a, or, in other words, that random, albeit not the most helpful of all strategies, can still be considered indispensable. In Figure 4c we observe the percentage of times that an approach fails to find the optimal value. The performance of the strategies here mirrors that of Figure 4a (the two measures, of course, correlate), but what we get from Figure 4c is that GREEDUS can find the optimal value for the majority of instances up to an avg. degree  $d$  of 10. Even though the situation might seem dramatic for an avg. degree  $d$  of 12 and 14, the distance to the optimal value, as reported in Figure 4a, is quite small, and a failure still registers as a failure even when a value of  $x + 1$  is reported instead of the optimal  $x$ . Finally, in Figure 4d we can see the implementation of GREEDUS scaling gracefully as the network instances become denser, whereas the performance of the implementation of the Partial MaxSAT encoding starts deteriorating drastically and even time-outs a few times when trying to solve the densest of instances.

► **Remark 5.** The time for generating the Partial MaxSAT encoding of a QCN was not taken into account in our evaluation and, in particular, in Figure 4d. This is because the encoding is currently not generated in an optimal way and it would skew the results in favor of the implementation of GREEDUS. However, some computational effort would be required in any case to produce the encoding, so what we see in Figure 4d for the implementation of the Partial MaxSAT encoding is a lower bound (with respect to our experimental evaluation here). In addition, despite the fact that the implementation of GREEDUS was sped up with PyPy, some overhead still remains, since it is fully coded in the high-level language of Python, which has an inherent performance disadvantage against low-level languages like C/C++. Thus, what we see in Figure 4d for the implementation of GREEDUS is an upper bound. In fact, based on algorithm design alone, it should be feasible to have an implementation of GREEDUS that would either match or exceed the performance of the implementation of the Partial MaxSAT encoding in all cases. The *main takeaway* regarding runtime performance here is that GREEDUS scales much better with respect to the average constraint graph degree of the network instances, and this scaling behaviour is accurately depicted in Figure 4d.

## 6 Conclusion and Future Work

In this paper, we focused on the problem of resolving inconsistency in qualitative constraint networks (QCNs), which can be viewed as knowledge bases of intuitive, human-like descriptions of spatio-temporal information like  $x$  *{is north of  $\vee$  is east of}*  $y$ . In particular, we presented two novel approaches for maximizing satisfiability in such networks: a greedy constraint-based and an optimal Partial MaxSAT-based one. The greedy technique adds the constraints of a given QCN to a new, initially empty network, one by one, filtering out the ones that fail the satisfiability check during the process; in doing so, it relies on many different strategies that create various orderings of the constraints to be processed, in a portfolio-style setting. The Partial MaxSAT encoding exploits to the fullest extent possible certain tractability properties associated with QCNs, viz., Horn theory-based *maximal tractable subsets* of relations [22], and is thus one of the most compact to date Partial MaxSAT encodings for the MAX-QCN problem, as evidenced also by the special case where all its clauses are assumed to be hard (the SAT case) [30]. We compared the two approaches against each other and provided some insight on the trade-off between obtaining repairs that are optimal and obtaining repairs in a manner that is fast. For future work, we would like to apply the techniques discussed here to other inconsistency-related reasoning tasks, such as the recently introduced one of decomposing QCNs into consistent components [24]. Further, we would like to explore more

on the use of SAT/MaxSAT solvers, especially solvers based on local search, e.g., [4], as we think that they would better suit our needs; in our experience, inconsistencies in QCNs tend to form locally. Finally, we are looking into ways of devising an optimal method out of our greedy approach.

---

## References

- 1 James F. Allen. Maintaining Knowledge about Temporal Intervals. *Commun. ACM*, 26:832–843, 1983.
- 2 Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon. Improving Glucose for Incremental SAT Solving with Assumptions: Application to MUS Extraction. In *SAT*, 2013.
- 3 Mehul Bhatt, Hans Guesgen, Stefan Wöflf, and Shyamanta Hazarika. Qualitative Spatial and Temporal Reasoning: Emerging Applications, Trends, and Directions. *Spatial Cognition & Computation*, 11:1–14, 2011.
- 4 Shaowei Cai and Zhendong Lei. Old techniques in new ways: Clause weighting, unit propagation and hybridization for maximum satisfiability. *Artif. Intell.*, 287:103354, 2020.
- 5 Shaowei Cai, Chuan Luo, John Thornton, and Kaile Su. Tailoring Local Search for Partial MaxSAT. In *AAAI*, 2014.
- 6 Jean-François Condotta, Ali Mensi, Issam Nouaouri, Michael Sioutis, and Lamjed Ben Said. A Practical Approach for Maximizing Satisfiability in Qualitative Spatial and Temporal Constraint Networks. In *ICTAI*, 2015.
- 7 Jean-François Condotta, Issam Nouaouri, and Michael Sioutis. A SAT Approach for Maximizing Satisfiability in Qualitative Spatial and Temporal Constraint Networks. In *KR*, 2016.
- 8 Frank Dylla, Jae Hee Lee, Till Mossakowski, Thomas Schneider, André van Delden, Jasper van de Ven, and Diedrich Wolter. A Survey of Qualitative Spatial and Temporal Calculi: Algebraic and Computational Properties. *ACM Comput. Surv.*, 50:7:1–7:39, 2017.
- 9 S.M. Hazarika. *Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions*. IGI Global, 2012.
- 10 Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, 2018.
- 11 Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an Efficient MaxSAT Solver. *J. Satisf. Boolean Model. Comput.*, 11:53–64, 2019.
- 12 David S. Johnson. Approximation Algorithms for Combinatorial Problems. *J. Comput. Syst. Sci.*, 9:256–278, 1974.
- 13 Jae Hee Lee, Michael Sioutis, Kyra Ahrens, Marjan Alirezaie, Matthias Kerzel, and Stefan Wermter. Neuro-Symbolic Spatio-Temporal Reasoning. *CoRR*, abs/2211.15566, 2022. doi: 10.48550/arXiv.2211.15566.
- 14 Joseph Y.-T. Leung, editor. *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004.
- 15 Gérard Ligozat. *Qualitative Spatial and Temporal Reasoning*. ISTE. Wiley, 2013.
- 16 Shuichi Miyazaki, Kazuo Iwama, and Yahiko Kambayashi. Database Queries as Combinatorial Optimization Problems. In *CODAS*, 1996.
- 17 Robert Moskovitch and Yuval Shahar. Classification of multivariate time series via temporal abstraction and time intervals mining. *Knowl. Inf. Syst.*, 45:35–74, 2015.
- 18 Lenka Mudrová and Nick Hawes. Task scheduling for mobile robots using interval algebra. In *ICRA*, 2015.
- 19 Bernhard Nebel. Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class. *Constraints*, 1:175–190, 1997.
- 20 Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen’s Interval Algebra. *J. ACM*, 42:43–66, 1995.
- 21 Jochen Renz and Bernhard Nebel. Efficient Methods for Qualitative Spatial Reasoning. *J. Artif. Intell. Res.*, 15:289–318, 2001.

## 12:12 Embarrassingly Greedy Inconsistency Resolution of Qualitative Constraint Networks

- 22 Jochen Renz and Bernhard Nebel. Qualitative Spatial Reasoning Using Constraint Calculi. In *Handbook of Spatial Logics*, pages 161–215. Springer, 2007.
- 23 Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.
- 24 Yakoub Salhi and Michael Sioutis. A Decomposition Framework for Inconsistency Handling in Qualitative Spatial and Temporal Reasoning. In *KR*, 2023. To appear.
- 25 Michael Sioutis and Diedrich Wolter. Dynamic Branching in Qualitative Constraint Networks via Counting Local Models. In *TIME*, 2020.
- 26 Michael Sioutis and Diedrich Wolter. Qualitative Spatial and Temporal Reasoning: Current Status and Future Challenges. In *IJCAI*, 2021.
- 27 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.
- 28 Matthias Westphal. *Qualitative Constraint-based Reasoning: Methods and Applications*. PhD thesis, Albert-Ludwigs-Universität Freiburg, 2014.
- 29 Matthias Westphal and Julien Hué. A Concise Horn Theory for RCC8. In *ECAI*, 2014.
- 30 Matthias Westphal, Julien Hué, and Stefan Wöflf. On the Propagation Strength of SAT Encodings for Qualitative Temporal Reasoning. In *ICTAI*, 2013.
- 31 Zhi-Hua Zhou. Abductive learning: towards bridging machine learning and logical reasoning. *Sci. China Inf. Sci.*, 62:76101:1–76101:3, 2019.



# Optimization of Nonsequenced Queries Using Log-Segmented Timestamps

Curtis E. Dyreson   

Department of Computer Science, Utah State University, Logan, UT, USA

---

## Abstract

In a period-timestamped, relational temporal database, each tuple is timestamped with a period. The timestamp records when the tuple is “alive” in some temporal dimension. *Nonsequenced semantics* is a query evaluation semantics that involves adding temporal predicates and constructors to a query. We show how to use log-segmented timestamps to improve the efficiency of temporal, nonsequenced queries evaluated using a non-temporal DBMS, i.e., a DBMS that has no special temporal indexes or query evaluation operators. A log-segmented timestamp divides the time-line into segments of known length. Any temporal period can be represented by a small number of such segments. The segments can be appended to a relation as additional columns. The advantage of log-segmented timestamps is that each segment can be indexed using standard database indexes, e.g., a B<sup>+</sup>-tree. A query optimizer can use the indexes to generate a lower cost query evaluation plan. This paper shows how to rewrite a query to use the additional columns and evaluates the time cost benefits and space cost disadvantages.

**2012 ACM Subject Classification** Information systems → Temporal data; Information systems → Relational database query languages

**Keywords and phrases** Temporal databases, nonsequenced semantics, query evaluation, query performance

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.13

**Supplementary Material** *Software (Source Code)*: <https://www.usu.edu/cs/people/CurtisDyreson/logsegmented/nonsequenced>

## 1 Introduction

In a tuple-timestamped, temporal relational database, the lifetime of a tuple in some temporal dimension is recorded using a period timestamp. The period timestamp represents the lifetime using a start time and an end time. Temporal relational database management systems process the timestamps in queries using two commonly recognized semantics for temporal query evaluation: *sequenced* [3] and *nonsequenced* [5]. Sequenced query evaluation, in effect, runs the query in every time instant, while nonsequenced semantics is about the evaluation of explicit temporal predicates, constructors and functions.

We previously showed how to use a different kind of timestamp, which we called a log-segmented timestamp, to implement sequenced semantics for queries in an unmodified relational DBMS [14] and that sequenced semantics can be leveraged to support other kinds of semantics [16]. This paper shows how to use log-segmented timestamps to implement nonsequenced semantics. The primary benefit of doing so is that the log segments can be indexed by non-temporal indexes, and the indexes can be used to (sometimes) lower the cost of query evaluation.

To illustrate nonsequenced query evaluation, consider the query given in Figure 1. The query computes the join on the `dept` attribute between the `Tesco` and `Walmart` relations; the relations are shown in Figure 2. The `OVERLAPS` temporal predicate in the `WHERE` clause determines if the timestamps for the `time` attributes in each relation overlap. The query can be translated by a layer into a Postgres SQL query as shown in Figure 3. Evaluating the query in Figure 3 gives the result in query Figure 2(c).



© Curtis E. Dyreson;

licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 13; pp. 13:1–13:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 13:2 Optimizing Nonsequenced with Log Segments

```
SELECT s.dept, OVERLAPS(r, s)
FROM tesco s, walmart r
WHERE r OVERLAPS s
```

■ **Figure 1** Query to compute the temporal join between two tables.

<i>Data</i>	<i>Time Metadata</i>	
<b>Dept</b>	<b>Start</b>	<b>Stop</b>
Shoe	1	5

(a) Relation Tesco

<i>Data</i>	<i>Time Metadata</i>	
<b>Dept</b>	<b>Start</b>	<b>Stop</b>
Shoe	2	3
Shoe	5	6

(b) Relation Walmart

<i>Data</i>	<i>Time Metadata</i>	
<b>Dept</b>	<b>Start</b>	<b>Stop</b>
Shoe	2	3
Shoe	5	5

(c) Result of the nonsequenced evaluation of the query in Figure 1.

■ **Figure 2** Example relations.

The focus of this paper is on the cost of query evaluation and whether that cost can be reduced. As an example, consider the evaluation of the query in Figure 3 on relations with 50K tuples. The query evaluation plan generated by the SQL compiler for the query is given in Figure 4. To improve query efficiency in the plan a two attribute index was created (`indexstartstop`) on the time attributes as well as individual indexes on each attribute. The index is used in the nested loops join, but the overall cost of the query (given in the top line of the plan) is 30,587,076.

The key research question addressed by this paper is whether this query evaluation plan can be improved using “off-the-shelf” relational DBMS technology, i.e., not using a specialized temporal index or other modifications of a DBMS. Using the techniques presented in this paper we show how to lower the cost to 1,376,011. The optimizer can choose between the plans to generate the lowest cost query.

This paper makes the following contributions.

- We describe how to extend a relation to store a temporal period using log segments.
- We show how to use log segments to evaluate a nonsequenced temporal predicate.
- We describe experiments with the Postgres DBMS that demonstrate the efficacy of our approach. Our experimental reproducibility package is available.<sup>1</sup>

This paper is organized as follows. The next section gives background material relevant to the paper. Section 3 presents the main technical content of the paper, how to store log segments and use the segments in nonsequenced query evaluation. Evaluation of the technique is given in Section 4 followed by a discussion of related work and a short conclusion with remarks on future work.

<sup>1</sup> <https://www.usu.edu/cs/people/CurtisDyreson/logsegmented/nonsequenced>

```

SELECT s.dept,
       GREATEST(r.time.start, s.time.start) AS start,
       LEAST(r.time.stop, s.time.stop) as stop
FROM tesco s, walmart r
WHERE ((r.start <= s.start AND s.start <= r.stop)
       OR (s.start <= r.start AND r.start <= s.stop))

```

■ **Figure 3** Query to compute the nonsequenced temporal join between two tables.

```

Nested Loop (cost=228.08..30587076.79 rows=524691358 width=20)
-> Seq Scan on empt r (cost=0.00..1662.00 rows=50000 width=20)
-> Bitmap Heap Scan on empt s (cost=228.08..454.30 rows=10494 width=8)
    Recheck Cond: ((r.start <= start) AND (start <= r.stop))
                OR ((start <= r.start) AND (r.start <= stop))
-> BitmapOr (cost=228.08..228.08 rows=11111 width=0)
    -> Bitmap Index Scan on foostart (cost=0.00..55.86 rows=5556 width=0)
        Index Cond: ((start >= r.start) AND (start <= r.stop))
    -> Bitmap Index Scan on foostartstop (cost=0.00..166.97 rows=5556 width=0)
        Index Cond: ((start <= r.start) AND (stop >= r.start))

```

■ **Figure 4** Query execution plan for a temporal join.

## 2 Preliminaries

In this section we describe background material pertinent to the paper.

### 2.1 Model of time

This research is orthogonal to assumptions about the time-line, number of temporal dimensions, representations of time, and data model. But for simplicity, we make the following assumptions.

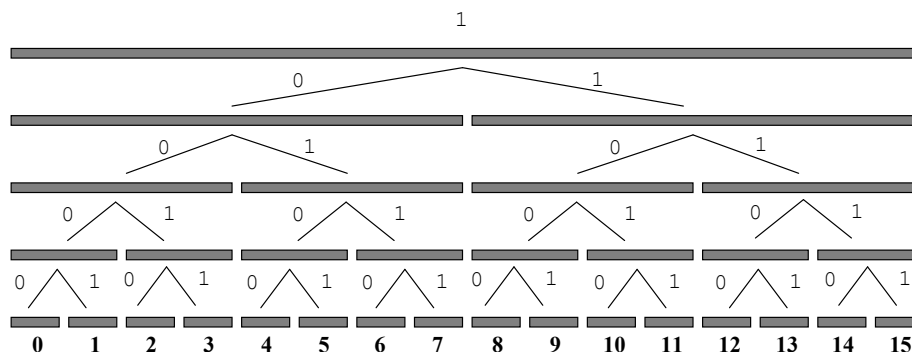
- We use a discrete time-line, with chronons ranging from time  $-\infty$  to time  $\infty$ .
- There is only one time dimension.
- We assume a relational data model (as either sets or bags of tuples) in which every tuple in every relation is annotated with *temporal metadata* that records the lifetime of the tuple in some time dimension. That is, it is a *tuple-timestamped* model [20].

### 2.2 Temporal Query Semantics

*Sequenced* and *nonsequenced* semantics were introduced as different semantics for the evaluation of a temporal operation such as a query or data modification, and both semantics are important [19]. Böhlen and Jensen trace the history and meaning of sequenced semantics [2], but, put simply, sequenced semantics evaluates an operation in each time instant using only the data alive at that time. Nonsequenced semantics, in contrast, means that an operation explicitly references and manipulates the timestamps in the data [5]. In some sense, nonsequenced semantics is the absence of a implicit temporal semantics, only explicit, direct manipulation of the timestamps is supported.

One important benefit of both semantics is that they *reduce* to non-temporal semantics. For sequenced semantics, the reducibility is called *snapshot reducibility* [23] or *S-reducibility* [4]. The temporal semantics is defined in terms of a (presumably easily understood) *slice* of temporal to non-temporal, and the non-temporal semantics of query evaluation (also, well understood).

## 13:4 Optimizing Nonsequenced with Log Segments



■ **Figure 5** Log segments on a time-line.

Nonsequenced semantics is also reductive. The time information is converted to data, and the non-temporal operation is evaluated on the data. Since time plays no special role in the evaluation, each tuple in the result has no (implicit) time. Instead, the times are manipulated through temporal functions, temporal predicates, and temporal constructors specified in the query. Some of the constructors can convert the data back into time.

Traditionally, the two semantics have been seen as different, though proposals for reconciling the differences exist [16].

### 2.3 Log-segmented Timestamps

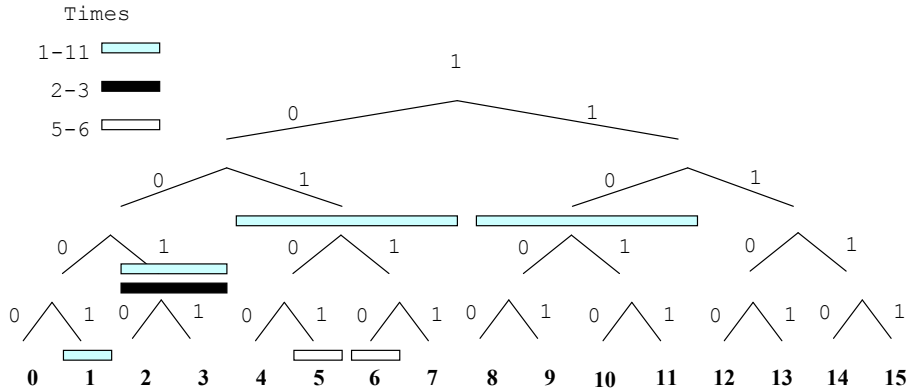
Most temporal database research and implementation uses period timestamps to annotate data with temporal metadata [22]. Period timestamping appends a timestamp to each data item to represent its lifetime. A variation of tuple-timestamped models is *attribute timestamping* where timestamps are appended to each attribute in a tuple rather than to the entire tuple [25].

Period timestamps are a poor fit for architectures that need to partition large data sets into smaller shards to process, e.g., mapreduce architectures [21] or hash joins in a DBMS. Consider, for instance a hash join operation. Data items that have the same join values hash to a common bucket, and the buckets are joined. The strategy is efficient since it ensures that only data items that actually will join are put into a bucket. A temporal join adds a further condition that two data items join only on the times at which they are both alive. For period timestamps this is computed as the temporal intersection of the timestamps. If the intersection is empty, the items do not join since they do not coexist at any point in time. The problem is that periods cannot be directly mapped to buckets in a way that ensures that the items within a bucket temporally intersect. Consider the periods  $[1, 2]$ ,  $[8, 9]$ , and  $[0, 10]$ .  $[1, 2]$  and  $[8, 9]$  should be placed in different buckets since they do not intersect, and hence, never represent data that coexists. But  $[0, 10]$  intersects both, it has to be placed into both. Since a period of size  $n$  has  $n(n + 1)/2$  sub-periods that could intersect, every period potentially needs to belong to many buckets.

To address this challenge we developed a *log-segmented timestamp* [13]. The timestamp uses a labelling scheme for pre-determined periods on a time-line. A label is a binary number that has the following meaning.

■ **Table 1** Some example labels for the time-line 0..15.

Label	Period	$t_x$	$t_y$
1	0 - 15	0	$15 = 0 + (2^4 - 1)$
10	0 - 7	$0 = 0 * 2^4$	$8 = 0 + (2^3 - 1)$
110	8 - 11	$8 = 1 * 2^3$	$11 = 8 + (2^2 - 1)$
1101	10 - 11	$10 = 1 * 2^3 + 1 * 2^1$	$11 = 10 + (2^1 - 1)$
10011	3 - 3	$3 = 1 * 2^1 + 1 * 2^0$	$3 = 3 + (2^0 - 1)$



■ **Figure 6** Log segments for the times in the relations in Figure 2 a) and b).

► **Definition 1 (Log-segment Label).** Let a (discrete) time-line consist of the times  $t_0, \dots, t_n$ , where  $n = 2^k - 1$ . Note that  $n$  can be represented using a binary number of length  $k$  with each digit set to 1. A label is a binary number,  $b_0..b_j$ , and  $b_0$  is always 1. The label  $1b_1..b_j$ ,  $j \leq k$ , represents the time period  $t_x$  to  $t_y$  where  $t_x = b_12^{k-1} + b_22^{k-2} + \dots + b_j2^{k-j}$  and  $t_y = t_x + (2^{k-j} - 1)$ .

The log segments for a time-line from 0 to 15 are depicted in Figure 5. The chronons in the time-line are numbered at the bottom of the figure. Each gray rectangle in the figure is a segment. A label for a segment is the concatenation of 1's and 0's along the path from the root to a segment. Some example labels are shown in Table 1. Note that only  $2n - 1$  of the  $(n + 1)(n + 2)/2$  possible periods in the timeline are labelled.

A *log-segmented timestamp* is the minimal set of segments that spans a given period. For example, the log-segmented timestamp representing the period [3,11] is {10011, 101, 110} (naming the periods {[3,3], [4,7], [8,11]}, respectively). The log-segmented timestamps for the times in the relations in Figure 2 a) and b) is graphically depicted in Figure 6.

Log-segmented timestamps have the following properties.

- Comprehensive - A time-line of size  $n$  has at most  $2n - 1$  labels. Each label will have a maximum length of  $1 + \lceil \log_2(n) \rceil$  bits. So a label of 64 bits (the size of a `long long` scalar in C++) can represent a time-line of  $2^{63} - 1$  time values, which encompasses a time-line longer than current estimates of the lifetime of the universe to the granularity of microseconds [17].
- Compact - The maximum number of segments in a log-segmented timestamp for a period,  $[t_x, t_y]$ , is  $2 * \lceil \log_2((1 + t_y) - t_x) \rceil$ . So assuming 64 bit labels, a log-segmented timestamp has at most  $2 * 64$  labels.

## 13:6 Optimizing Nonsequenced with Log Segments

<i>Data</i>		<i>Time Metadata</i>								
Dept	Start	Stop	s1	s2	s4	s8	s1x	s2x	s4x	s8x
...	1	11	10001	1001	101				110	
...	2	3		1001						
...	5	6	10101				10110			

■ **Figure 7** Example segment columns for the periods in Figure 6.

### 3 Timestamp Representation and Temporal Predicates

To process nonsequenced log-segmented queries, we choose to represent log segments by adding additional columns (attributes) to a table (relation). There are two kinds of additional columns, which we describe in this section: segment columns and prefix columns. It may seem counter intuitive to add columns in order to improve evaluation efficiency, but, in effect, we are trading space for time since the columns that we add will be indexed and the indexes used to lower the time cost.

#### 3.1 Segment Columns

A segment column is a column that stores a log segment as an integer. We observe that a log segmented timestamp has at most two segments with the same length. For instance in Figure 6 there are two segments with the same length, 101 and 110, in the set of segments for the period [1-11]. Hence  $2\log_2(n)$  columns are required to store all of the segments. Moreover, each column can be distinguished by the length of the segment that it stores, i.e., a segment with length  $n$  can be stored in the `segmentn` column, and the second segment (if present) in the `segmentnx` column. An example is shown in Figure 7. It depicts the segments for the log-segmented timestamps in Figure 6. The segment columns (e.g., `s1`) are appended to each row in the table. We assume a timeline for this example of only 16 chronons. An `s16` column is not needed since the entire timeline can be represented by the segments in `s8` and `s8x`.

Any missing segment column value is null, hence the additional columns will be relatively sparsely populated. Most modern DBMSs do not store null values, rather no space is allocated for a null, instead the column is marked as no size in the row header.

#### 3.2 Prefix Columns

The prefix columns record each segment that contains the starting (stopping) chronon of a period. For any given segment, the segment is contained in each segment that is a labelled with a prefix. For instance the segment 1101 is contained within the segments 110, 11, and 1.

Prefix columns are appended to each row in a table to store the prefixes for the start and stop chronons. Each prefix must have a different length, hence the length of the label can be used in the name of the column, e.g., `p4` for a start chronon prefix of length 4 and `p2e` for a stop chronon of length 2. Figure 8 shows the prefixes for the log-segmented timestamps in Figure 6.

Data		Time Metadata								
Dept	Start	Stop	p1	p2	p4	p8	p1e	p2e	p4e	p8e
...	1	11	10001	1000	100	10	11011	1101	110	11
...	2	3	10010	1001	100	10	10011	1001	100	10
...	5	6	10101	1010	101	10	10110	1011	101	10

■ **Figure 8** Example prefix columns for the periods in Figure 6.

### 3.3 Reasoning About Chronon Containment

The segment and prefix columns can be used to determine whether a start (or stop) time is contained within a period. Let chronon  $x$  have prefixes  $p_1, p_2, p_4, \dots, p_n$  and period  $[z, y]$  have segments  $s_1, s_2, s_4, \dots, s_n, s_1x, s_2x, \dots, s_nx$ , then  $x$  is contained in  $[z, y]$  if

$$\exists i(p_i = s_i \vee p_i = s_ix)$$

As examples consider the following using the segments of Figure 7 and the prefixes of Figure 8.

- Is 2 contained in [1-11]? The segments of [1-11] are

Dept	Start	Stop	s1	s2	s4	s8	s1x	s2x	s4x	s8x
...	1	11	10001	1001	101				110	

and the prefixes of 2 are as given below.

Dept	Start	Stop	p1	p2	p4	p8	p1e	p2e	p4e	p8e
...	2		10010	1001	100	10				

Since  $p_2 = s_2$ , it is contained within.

- Is 2 contained in [5-6]? The segments of [5-6] are

Dept	Start	Stop	s1	s2	s4	s8	s1x	s2x	s4x	s8x
...	5	6	10101				10110			

and the prefixes of 2 are as given below.

Dept	Start	Stop	p1	p2	p4	p8	p1e	p2e	p4e	p8e
...	2		10010	1001	100	10				

There is no  $i$  such that  $p_i = s_i$  or  $p_i = s_ix$ , hence it is not contained.

### 3.4 Temporal Predicates

Log-segmented timestamps give an alternative to using the start and stop times in a period to implement some temporal predicates. Examples include the following.

- $x$  OVERLAPS  $y$  - If the start chronon in  $x$  is contained in the period  $y$  or vice-versa then period  $x$  overlaps period  $y$ . Figure 9 shows the SQL to add to the WHERE clause to express an OVERLAPS predicate, assuming a timeline of  $2^{19}$  chronons. Note that the query could be rewritten using UNION or UNION-ALL to break up the large disjunctive condition in the WHERE clause.



## 13:8 Optimizing Nonsequenced with Log Segments

```
WHERE ...
  (x.s1 = y.p1 OR r.s2 = y.p2 OR ... OR x.s19 = y.p19
 OR x.s1 = y.p1e OR r.s2 = y.p2e OR ... OR x.s19 = y.p19e
 OR y.s1 = x.p1 OR y.s2 = x.p2 OR ... OR y.s19 = x.p19
 OR y.s1 = x.p1e OR y.s2 = x.p2e OR .. OR y.s19 = x.p19e)
```

■ **Figure 9** SQL for computing **OVERLAPS** in a time-line of  $2^{19}$ .

```
WHERE ...
  (x.start = y.start AND x.stop <> y.stop AND
   (y.s1 = x.p1e OR y.s2 = x.p2e OR ... OR y.s19 = x.p19e)
  )
```

■ **Figure 10** SQL for computing **STARTS** in a time-line of  $2^{19}$ .

- **x STARTS y** - If *x* and *y* have the same start chronons and different end chronons and the stop chronon of *x* is contained in period *y* then *x* starts *y*. Figure 10 shows the SQL to add to the **WHERE** clause to express a **STARTS** predicate, assuming a timeline of  $2^{19}$  chronons.
- **x CONTAINS y** - If the start and stop chronons of *y* are contained within *x* then *x* contains *y*.

Note that log segments do not provide an alternative for predicates that only involve period endpoints, such as **MEETS**. These predicates can still be evaluated using the start and stop chronons in the timestamp.

### 3.5 DBMS Implementation

It is highly unlikely that a user could manually manage the prefix and segment columns, for instance, populating these columns when inserting a tuple. To better support users we envision a stratum approach to implementation whereby a user interacts with the DBMS through a layer of middleware layer. The layer provides three key services.

1. Query rewriting - We observe that the schema supports query evaluation on both log-segmented timestamps and normal timestamps (it is not an either-or choice, both can be supported simultaneously). A query will be rewritten by replacing the nonsequenced and sequenced predicates and constructors in the query in two ways. First the query will be rewritten to evaluate with respect to the normal (non-log segmented timestamps). For example an **overlaps** predicate will be replaced with the SQL to compare **start** and **stop** timestamps. Note that this is how non-sequenced query evaluation is usually implemented. Second the query will be rewritten to use the log segmented timestamps. Both rewritten queries will be submitted to the query optimizer to determine which has an (estimated) lower cost, and that query evaluation plan will be chosen.
2. Schema modification - Schema modification statements, e.g., **CREATE TABLE** will be rewritten to manage the prefix and segment columns automatically. All indexes for the additional columns will be created or dropped as needed.
3. Data modification - Data modification statements will be rewritten to manage the prefix and segment columns automatically. Note that computing log segments is a simple calculation that can be done using an SQL function [14].

## 4 Evaluation

Log segments add columns to a relation and complicate the expression of temporal predicates, but they provide one key benefit: the segment and prefix columns can be indexed and the indexes can be utilized by the query optimizer to lower the time cost of query evaluation. This section describes an experimental evaluation of temporal predicates using log segmented timestamps.

### 4.1 Experiment Environment

The experiments were run on an Intel Core i7 CPU, 1.8 GHz clock speed, 16 GB of memory and 1 TB SSD drive running Windows 10 Pro 64-bit as the operating system. We used Postgres, version 14, and did not change any installation or configuration parameters from the standard (default) installation.

### 4.2 Schema for Experiments

We tested with two schemas: `periodStamped` and `segmentStamped`. The `periodStamped` schema has one table, an `Employee` table with the schema given below.

```
Employees(id, name, department, start, stop)
```

The `id` column is the primary key of the table (the snapshot versus temporal key is not relevant to the experiments) and an integer type, the `name` and `department` columns are string types, and the `start` and `stop` columns are integers. The `segmentStamped` schema has one table, an `Employee` table with the schema given below.

```
Employees(id, name, department, start, stop,
          s1, s2, ..., s19, s1x, s2x, ..., s19x,
          p1, p2, ..., p19, p1e, p2e, ..., p19e)
```

The added segment and prefix columns are integer types. We chose to represent a log segment using the time of the first chronon in the segment.

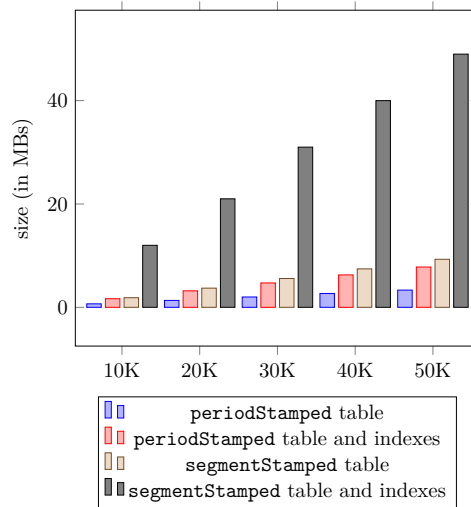
### 4.3 Database Generation

We synthetically generated a database for each of the schemas. We used 100 different departments and 90% (of the total number of tuples) different names when populating the table. We chose a timeline of  $2^{19}$  (enough to represent a span of 60 years to a granularity of seconds) and used timestamps randomly chosen within the timeline and of random length (from 1 to  $2^8$ ). We used test cases of 10000, 20000, 30000, 40000, and 50000 tuples and created one column indexes for every timestamp column (`start` through `p19e`) as well as a two column index on the `start` and `stop` columns. The resulting database sizes are shown in Figure 11. The log segmented tables are roughly twice the size of the period stamped tables, but the indexes for the log segmented tables approximately quadruple the storage cost.

### 4.4 Measuring Query Cost

To mitigate the impact of database buffering, we used the query cost as estimated by the Postgres query optimizer using `EXPLAIN`. The optimizer computes the cost in units that do not have an exact correspondence to running time, i.e., a cost of 100 does not mean 100 ms of time taken to evaluate a query, but rather are used to determine cheaper versus more expensive queries.

## 13:10 Optimizing Nonsequenced with Log Segments



■ **Figure 11** Size of data in database (in MBs).

We also measured actual query time using `EXPLAIN ANALYZE`. We took the minimum cost query over five runs (the variance was inconsequential). The measurements were taken with respect to a warm cache.

### 4.5 Predicate Evaluation

We measured the cost of three predicates on a fully temporal join, i.e., joining the two relations only on the temporal attributes. The first experiment measure the cost of overlaps. The results are shown in Figure 12. The log-segmented cost is estimated by the compiler to be much lower than that of the period stamped relations. The reason is a different query execution plan. The query execution plan for the period stamped relations was given previously (see Figure 4). Figure 16 shows the relevant part of the log segmented query execution plan. The compiler generates an efficient plan that uses bitmap indexes for matches between segment and prefix columns, yielding a lower cost query plan. The actual timings of the queries shown in Figure 14 show that the query optimizer produced an accurate estimate, and that the bitmap index use does speed up queries.

The second experiment measures the cost of contains. The results are shown in Figure 13. Contains is slightly more efficient for both period and log-segmented timestamped relations. Note that in the log-segmented plot the cost of the 50K join is less than that of the 40K cost. This reflects a change in the optimization strategy chosen by the query optimizer; at 50K tuples the optimizer chooses a parallel scan so gets some performance improvement. Figure 15 shows that the measured query times have the same profile as the estimates produced by the query optimizer.

The third experiment measures the cost of starts. The results are shown in Figure 17. Note that overall the costs are orders of magnitude lower than the other two predicates. This is because both the period and log segmented queries use an equality comparison on the start time, e.g., `r.start = s.start`, together with a test to determine whether the stop time is less. The start time condition can take advantage of the start index for both kinds of timestamp, and this in effect determines the cost of the query. The cost of testing end stop time is slightly worse for the log segmented timestamp, which increases its cost slightly. Note that the cost of the 50K case is better than the 40K case for the log segmented timestamps

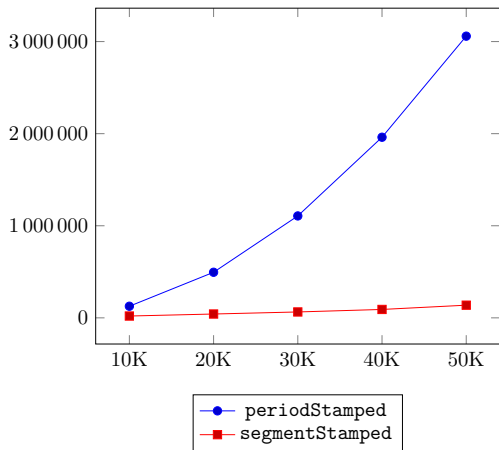


Figure 12 Optimizer estimate for overlaps.

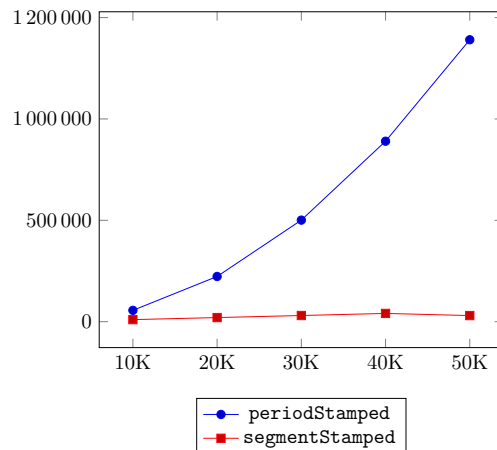


Figure 13 Optimizer estimate for contains.

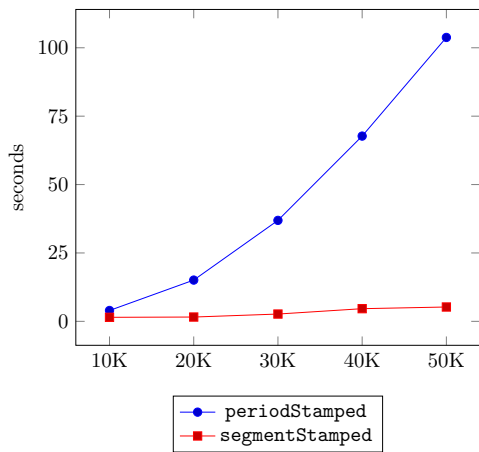


Figure 14 Measured Time for overlaps.

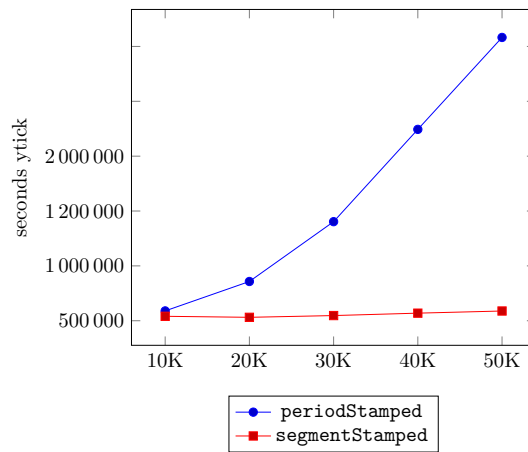


Figure 15 Measured Time for contains.

due to parallelization in the query execution plan. Figure 18 shows that period timestamped query have slightly lower times than the log-segmented timestamped, as predicted by the query optimizer.

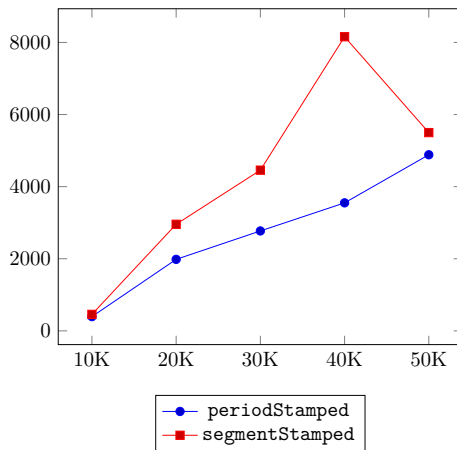
### 4.6 Discussion of Results

Appending columns to a relation to store log segments and prefixes of the start and stop chronons effectively doubles the size of a relation with few data columns. Adding indexes on the segment and prefix columns further increases the cost. But, in practice relations with 10 to 100 data columns are more common, so the storage cost difference would often be less in real-world situations. In some cases the query optimizer can use the added columns and indexes to generate a lower cost query evaluation plan at the cost of increasing the size and complexity of the WHERE clause predicate. But in many cases using the start and stop times and indexes offers a better plan as shown by the third experiment (the starts experiment). Utilizing log segments can be seen as a potential optimization technique that increases the space of potential plans, and the query optimizer can examine other constraints in the query to choose the best, lowest cost plan.

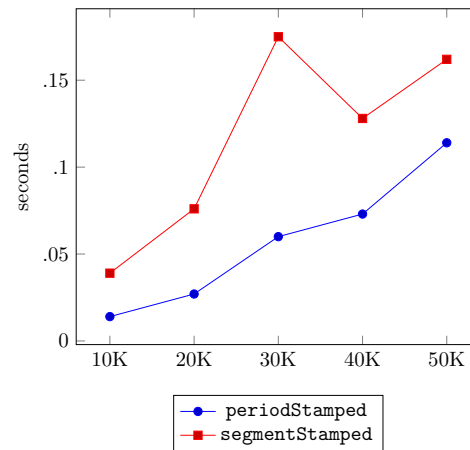
## 13:12 Optimizing Nonsequenced with Log Segments

```
Nested Loop (cost=12.13..102716.47 rows=120094 width=20)
-> Seq Scan on empt r (cost=0.00..417.00 rows=10000 width=176)
-> Bitmap Heap Scan on empt s (cost=12.13..18.11 rows=18 width=84)
Recheck Cond: ((s1 = r.p1) OR (s2 = r.p2) ... OR (s262144 = r.p262144x)
OR (s524288 = r.p524288x))
-> BitmapOr (cost=12.13..12.13 rows=18 width=0)
-> Bitmap Index Scan on foos1 (cost=0.00..0.30 rows=1 width=0)
Index Cond: (s1 = r.p1)
-> Bitmap Index Scan on foos2 (cost=0.00..0.30 rows=1 width=0)
Index Cond: (s2 = r.p2)
-> Bitmap Index Scan on foos4 (cost=0.00..0.30 rows=1 width=0)
...
-> Bitmap Index Scan on foos524288x (cost=0.00..0.30 rows=1 width=0)
Index Cond: (s524288 = s_1.p524288)
```

■ **Figure 16** Query execution plan using indexes on the segment and prefix columns for a temporal join.



■ **Figure 17** Optimizer for starts.



■ **Figure 18** Measured time for starts.

## 5 Related Work

There are many temporal extensions of query languages, c.f., [7,18,23,24]. This paper focuses on temporal SQL. The extensions have been broadly characterized in various ways but *sequenced* vs. *nonsequenced* distinguishes extensions, in part, by whether the time metadata is manipulated implicitly or explicitly. This paper is about nonsequenced semantics. Temporal languages have also been characterized as *abstract* vs. *concrete* based on whether their syntax and semantics depends on a specific representation of the time metadata [8]. This paper describes an abstract semantics, and proposes a concrete representation to optimize some nonsequenced queries.

Two implementation approaches are common for SQL-like temporal query languages. A *stratum*-approach adds a source-to-source translation layer to translate a query in a temporal extension into an equivalent query in the original, non-extended language [26,27]. Some constructs prove not possible to translate using period timestamps, e.g., sequenced outer join, so the only feasible approach is to extend the DBMS itself [11]. A related approach is to translate to a non-standard variant of SQL [15], in anticipation that SQL will one day evolve to incorporate the variant. We adopt a stratum approach in this paper whereby a nonsequenced

query is translated to a (non-temporal) SQL query and evaluated on a unmodified relational DBMS. We know of no other papers that explore optimization of nonsequenced queries using non-temporal indexes or without making other changes to the DBMS.

Hierarchical partitioning of intervals into smaller segments, similar to log segments, for the purposes of indexing has been explored recently [9]. Our research [13, 14] predates this effort and supports indexing by B-tree indexes.

There are several papers that also support the use of B-tree indexes in evaluating temporal constructors and predicates c.f., [1, 6, 10, 12]. In particular, it was proposed that a range query on a B-tree index combined with a UNION could be used to efficiently compute a non-sequenced join using an overlaps predicate [12]. While we found also found that UNION, or more specifically UNION-ALL, was useful in optimizing queries with an OR predicate in the WHERE clause, but care had to be taken to preserve duplicates or not over-produce duplicates in the query result. The UNION-ALL optimization could also be used for log segmented timestamps which have many OR predicates in overlap joins. One key difference is that we do not use range index queries, rather we use point index queries to evaluate the join. Techniques to augment the DBMS evaluation engine for improved join strategies [1] go beyond the scope of this paper, we focused on not altering the DBMS query evaluation engine.

## 6 Conclusion and Future Work

The primary contribution of this paper is to show a novel method for optimizing nonsequenced SQL queries. Temporal query languages often extend a non-temporal language by adding temporal predicates, constructors, and functions which directly manipulate the time metadata that annotates the data. A query is said to be nonsequenced if it explicitly includes one of these added temporal features. When a nonsequenced query is evaluated, the nonsequenced part of the query is evaluated against the time metadata, e.g., a temporal overlaps predicate checks if two timestamps overlap in time.

A tuple-timestamped relational database appends to each tuple a period timestamp for each temporal dimension. The start and stop times in the timestamp can be indexed, and often a query execution plan can use the indexes to lower the cost of query execution. This paper proposes adding a log-segmented timestamp to each tuple, in addition to a period timestamp. A log-segmented timestamp divides the time-line into segments of known length. Any temporal period can be represented by a small number of such segments. The segments can be used as an alternative to determining containment of a start or stop chronon within a period. We described how a relation can be extended with segment and prefix columns and how these columns can be used in the nonsequenced evaluation of temporal predicates such as OVERLAPS. We experimentally showed that an off-the-shelf relational DBMS can index the segments and the query optimizer can use the indexed segments to generate a lower cost query evaluation plan, though with higher space cost.

In future we plan to continue to investigate log segmented timestamps. We observe that such segments can be used to improve temporal hash joins with a specialized temporal hash join operator that can be added to a DBMS. The idea is that each tuple is first hashed to a data bucket, and if that data bucket becomes full, then further hashed to different time buckets within a data bucket by using log segments as the hash function. A time bucket joins with all time buckets within a data bucket that are prefixes of the segment. A second avenue to explore is prefix-based indexing. In this paper, we proposed precomputing the prefixes and storing them as additional columns, but the prefixes are actually visited in the

traversal of a  $B^+$  tree that stores the segments. By modifying the traversal, it should be possible to avoid precomputation and storage of the prefixes. A third area of future work is temporal constructors and functions. We believe that it is straightforward to articulate temporal constructors, such as the OVERLAPS constructor, but have yet to articulate the details. Finally, we are investigating the use of log segments in other temporal query languages such as temporal graph query languages.

## References


- 
- References
- 1 Andreas Behrend, Anton Dignös, Johann Gamper, Philip Schmiegelt, Hannes Voigt, Matthias Rottmann, and Karsten Kahl. Period Index: A Learned 2D Hash Index for Range and Duration Queries. In *Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD 2019, Vienna, Austria, August 19-21, 2019*, pages 100–109. ACM, 2019. doi:10.1145/3340964.3340965.
  - 2 Michael H. Böhlen and Christian S. Jensen. Sequenced semantics. In *Encyclopedia of Database Systems*, pages 2619–2621. 2009. doi:10.1007/978-0-387-39940-9\_1053.
  - 3 Michael H. Böhlen and Christian S. Jensen. Sequenced semantics. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems, Second Edition*. Springer, 2018. doi:10.1007/978-1-4614-8265-9\_1053.
  - 4 Michael H. Böhlen, Christian S. Jensen, and Richard T. Snodgrass. Temporal Statement Modifiers. *ACM Trans. Database Syst.*, 25(4):407–456, 2000. URL: <http://portal.acm.org/citation.cfm?id=377674.377665>.
  - 5 Michael H. Böhlen, Christian S. Jensen, and Richard T. Snodgrass. Nonsequenced semantics. In *Encyclopedia of Database Systems*, pages 1913–1915. 2009. doi:10.1007/978-0-387-39940-9\_1052.
  - 6 Matteo Ceccarello, Anton Dignös, Johann Gamper, and Christina Khnaisser. Indexing Temporal Relations for Range-Duration Queries. *CoRR*, abs/2206.07428, 2022. doi:10.48550/arXiv.2206.07428.
  - 7 Cindy Xinmin Chen and Carlo Zaniolo.  $Sql^{st}$ : A spatio-temporal data model and query language. In *ER*, pages 96–111, 2000. doi:10.1007/3-540-45393-8\_8.
  - 8 Jan Chomicki and David Toman. Abstract versus concrete temporal query languages. In *Encyclopedia of Database Systems*, pages 1–6. 2009. doi:10.1007/978-0-387-39940-9\_1559.
  - 9 George Christodoulou, Panagiotis Bouros, and Nikos Mamoulis. Hint: A hierarchical index for intervals in main memory, 2021. arXiv:2104.10939.
  - 10 Carlo Combi and Pietro Sala. Interval-based temporal functional dependencies: specification and verification. *Ann. Math. Artif. Intell.*, 71(1-3):85–130, 2014. doi:10.1007/s10472-013-9387-1.
  - 11 Anton Dignös, Michael H. Böhlen, and Johann Gamper. Temporal Alignment. In *SIGMOD*, pages 433–444, 2012. doi:10.1145/2213836.2213886.
  - 12 Anton Dignös, Michael H. Böhlen, Johann Gamper, Christian S. Jensen, and Peter Moser. Leveraging Range Joins for the Computation of Overlap joins. *VLDB J.*, 31(1):75–99, 2022. doi:10.1007/s00778-021-00692-3.
  - 13 Curtis E. Dyreson. Using CouchDB to Compute Temporal Aggregates. In *18th IEEE International Conference on High Performance Computing and Communications (HPCC)*, pages 1131–1138. IEEE Computer Society, 2016. doi:10.1109/HPCC-SmartCity-DSS.2016.0159.
  - 14 Curtis E. Dyreson and M. A. Manazir Ahsan. Achieving a Sequenced, Relational Query Language with Log-Segmented Timestamps. In *TIME 2021*, volume 206 of *LIPICs*, pages 14:1–14:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.TIME.2021.14.



- 15 Curtis E. Dyreson and Venkata A. Rani. Translating Temporal SQL to Nested SQL. In *23rd International Symposium on Temporal Representation and Reasoning, TIME*, pages 157–166. IEEE Computer Society, 2016. doi:10.1109/TIME.2016.24.
- 16 Curtis E. Dyreson, Venkata A. Rani, and Amani Shatnawi. Unifying Sequenced and Non-sequenced Semantics. In *22nd International Symposium on Temporal Representation and Reasoning, TIME*, pages 38–46. IEEE Computer Society, 2015. doi:10.1109/TIME.2015.22.
- 17 Curtis E. Dyreson and Richard T. Snodgrass. Timestamp semantics and representation. *Information Systems*, 18(3):143–166, 1993. doi:10.1016/0306-4379(93)90034-X.
- 18 Fabio Grandi. T-SPARQL: A TSQL2-like Temporal Query Language for RDF. In *ADBIS*, pages 21–30, 2010. URL: <http://ceur-ws.org/Vol-639/021-grandi.pdf>.
- 19 Fabio Grandi, Federica Mandreoli, Riccardo Martoglia, and Wilma Penzo. Unleashing the power of querying streaming data in a temporal database world: A relational algebra approach. *Inf. Syst.*, 103:101872, 2022. doi:10.1016/j.is.2021.101872.
- 20 C. S. Jensen and C. E. Dyreson (editors). A Consensus Glossary of Temporal Database Concepts - February 1998 Version. In *Temporal Databases: Research and Practice, Lecture Notes in Computer Science 1399*, pages 367–405. Springer-Verlag, 1998.
- 21 Seyed Nima Khezr and Nima Jafari Navimipour. Mapreduce and its applications, challenges, and architecture: a comprehensive review and directions for future research. *J. Grid Comput.*, 15(3):295–321, 2017. doi:10.1007/s10723-017-9408-0.
- 22 R. T. Snodgrass. Introduction to TSQL2. In R. T. Snodgrass, editor, *The TSQL2 Temporal Query Language*, chapter 2, pages 19–31. Kluwer Academic Publishers, 1995.
- 23 Richard T. Snodgrass. The Temporal Query Language TQuel. *ACM Transactions on Database Systems*, 12(2):247–298, 1987. doi:10.1145/22952.22956.
- 24 Richard T. Snodgrass, editor. *The TSQL2 Temporal Query Language*. Kluwer, 1995.
- 25 A. U. Tansel. Modelling temporal data. *Information and Software Technology*, 32(8):514–520, October 1990.
- 26 Kristian Torp, Christian S. Jensen, and Michael H. Böhlen. Layered temporal DBMS: concepts and techniques. In *DASFAA*, pages 371–380, 1997.
- 27 Kristian Torp, Christian S. Jensen, and Richard T. Snodgrass. Stratum Approaches to Temporal DBMS Implementation. In *IDEAS*, pages 4–13, 1998. doi:10.1109/IDEAS.1998.694346.



# An Event Calculus for Run-Time Reasoning

Periklis Mantenoglou  

National and Kapodistrian University of Athens, Greece  
NCSR “Demokritos”, Athens, Greece

---

## Abstract

In stream reasoning, the task is to derive high level abstractions of large data streams with minimal latency, as required by contemporary applications. This work presents an Event Calculus-based approach to stream reasoning, highlighting its core features and recent extensions.

**2012 ACM Subject Classification** Computing methodologies → Temporal reasoning

**Keywords and phrases** Event Calculus, temporal pattern matching, complex event recognition

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.14

**Category** Extended Abstract

**Funding** This work was supported by the ENEXA project (No 101070305), and by the Hellenic Foundation for Research and Innovation (HFRI) under the 3rd Call for HFRI PhD Fellowships (Fellowship Number: 6011).

## Motivation

Modern applications require the processing of large, high-velocity data streams that are being generated continuously. A stream reasoning system derives instances of spatio-temporal pattern satisfaction, based on the actions/events reported in such data streams, with minimal latency. These spatio-temporal patterns may define a set of situations of interest in the target application domain. In maritime situational awareness, e.g., a stream reasoning system can be used to detect vessel activities that may be suspicious, illegal, dangerous or have a negative environmental impact, based the position and velocity signals that are continuously being emitted by sailing vessels [7]. For instance, spatio-temporal patterns may specify an illegal fishing activity in a prohibited area or an unexpected halt in signal transmissions.

There are several requirements for effective stream reasoning. First, a stream reasoning system should be based on a formal pattern specification language, in order to allow the user to express situations of interest without ambiguity. Second, this language has to be expressive enough to support all situations of interest that need to be detected in the target application. Third, the system should be equipped with highly-efficient algorithms for detecting these patterns, taking into consideration that input actions/events cannot be stored in memory en masse when operating in a streaming setting.

## The Event Calculus for Run-Time Reasoning

Towards addressing the requirements of stream reasoning, we proposed “The Event Calculus for Run-Time Reasoning” (RTEC), a logic-based, formal computational framework that is optimised for stream reasoning [2, 6, 5]. RTEC is based on a logic programming implementation of the Event Calculus [3], a temporal formalism for representing and reasoning about events and their effects. The Event Calculus dialect used by RTEC is many-sorted and includes events, fluents, i.e., properties that may have different values at different points in time, and a linear time model with integer time-points. The built-in Event Calculus predicates of RTEC are used to express event occurrences and changes in the values of fluents, and specify the time periods during which fluent-value pairs (FVPs) hold continuously.



© Periklis Mantenoglou;

licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 14; pp. 14:1–14:3

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$\text{happensAt}(E, T)$  denotes that event  $E$  takes place at time-point  $T$ , while  $\text{initiatedAt}(F = V, T)$  (resp.  $\text{terminatedAt}(F = V, T)$ ) expresses that a time period during which fluent  $F$  has the value  $V$  is initiated (terminated) at time-point  $T$ .  $\text{holdsFor}(F = V, I)$  states that fluent  $F$  has the value  $V$  continuously in the maximal intervals included in list  $I$ . Finally,  $\text{holdsAt}(F = V, T)$  states that fluent  $F$  has the value  $V$  at time-point  $T$ . Moreover, RTEC adopts the specification of the common-sense law of inertia used in the Event Calculus, expressing that FVPs persist through time, unless an event that affects the value of the fluent takes place.

RTEC features two types of fluents, called “simple” and “statically determined”. The conditions under which event occurrences may affect the values of simple fluents are expressed through domain-specific  $\text{initiatedAt}$  and  $\text{terminatedAt}$  rules. Given the “initiation points” and the “termination points” of some simple fluent  $F$  with value  $V$ , RTEC computes  $\text{holdsFor}(F = V, I)$ , i.e., the maximal intervals  $I$  in which  $F = V$  holds continuously. In the case of a statically determined fluent  $F$ , RTEC the maximal intervals of  $F = V$  directly, i.e., without computing the initiation and termination points of  $F = V$ , using an domain-specific  $\text{holdsFor}(F = V, I)$  rule, defining the maximal intervals  $I$  of  $F = V$  in terms of the maximal intervals of other FVPs via interval operations, such as union, intersection and relative complement.

RTEC supports stream reasoning applications by integrating the aforementioned representation and reasoning formalism with caching, indexing, windowing and a “forget” mechanism that removes redundant events and FVP intervals from its knowledge base. Moreover, RTEC is restricted to hierarchical knowledge bases that allow bottom-up processing, thus avoiding re-computations. The complexity analysis of RTEC is available in [2].

## Recent Extensions

The specifications of modern applications may include cyclic dependencies. In maritime situational awareness, e.g., a fishing trip consists of several stages, such as approaching a fishing area, fishing and returning to a port. These stages form a cycle, as each stage depends on the previous one. Moreover, situations of interest are often defined as temporal combinations of other situations, which are typically durative and take place within temporal intervals. The corresponding patterns can be expressed using Allen’s interval relations [1]. For instance, we may detect the suspicious situation where a vessel stops signal transmissions while being close to another vessel using the “during” Allen relation.

We extended RTEC for expressing patterns featuring cyclic dependencies and Allen relations [6, 5]. Our theoretical analysis of the resulting framework highlighted its semantics, correctness and complexity, while our empirical evaluation demonstrated its effectiveness when reasoning over large streams of benchmark and real data from modern applications. This extended version of RTEC is publicly available<sup>1</sup>.

## Further Work

Uncertainty is inherent in modern applications. In maritime situational awareness, e.g., malfunctions in signal transmitters may lead to data streams that include empty fields or erroneous field values. A recent work tackles uncertainty by associating input stream items with probability values, serving as a confidence estimate, and then employing probabilistic reasoning to derive the probabilities of pattern satisfaction instances based on such an input [4]. In the future, we would like to extend RTEC with probabilistic reasoning techniques.

---

<sup>1</sup> <https://github.com/aartikis/RTEC>

---




**References**

---

- 1 J. Allen. Maintaining knowledge about temporal intervals. *Comm. of the ACM*, 26(11):832–843, 1983.
- 2 Alexander Artikis, Marek Sergot, and Georgios Paliouras. An event calculus for event recognition. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):895–908, 2015.
- 3 Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–96, 1986.
- 4 Periklis Mantenoglou, Alexander Artikis, and Georgios Paliouras. Online probabilistic interval-based event calculus. In *ECAI*, volume 325, pages 2624–2631, 2020.
- 5 Periklis Mantenoglou, Dimitrios Kelesis, and Alexander Artikis. Complex event recognition with Allen relations. In *KR*, 2023.
- 6 Periklis Mantenoglou, Manolis Pitsikalis, and Alexander Artikis. Stream reasoning with cycles. In *KR*, pages 544–553, 2022.
- 7 Manolis Pitsikalis, Alexander Artikis, Richard Dreo, Cyril Ray, Elena Camossi, and Anne-Laure Joussemme. Composite event recognition for maritime monitoring. In *DEBS*, pages 163–174. ACM, 2019.



# SSTRESED: Scalable Semantic Trajectory Extraction for Simple Event Detection over Streaming Movement Data

Nikos Giatrakos   

Technical University of Crete, Chania, Greece  
Athena RC, Marousi, Greece

---

## Abstract

We describe SSTRESED, a prototype focused on the real-time, online detection of simple, durative events over streaming movement data. It is the first prototype that establishes a direct connection between semantic trajectory extraction and simple event detection. SSTRESED is highly scalable by incorporating parallel processing in two separate, but connected, training and event detection pipelines implemented on state-of-the-art platforms, directly deployable in cloud environments.

**2012 ACM Subject Classification** Computer systems organization → Architectures

**Keywords and phrases** Semantic Trajectory, Event Processing, Data Streams

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.15

**Category** Extended Abstract

**Funding** This work is funded by the European Union under Horizon Europe agreement No 101070430.

## 1 Introduction & Motivation

Detecting Simple, Derived Events (SDEs) is the first step towards Complex Event Recognition (CER) [3, 4, 5]. In time critical-applications [1, 6], such as safe robot navigation in dynamic smart factory environments, SDE detection should be performed continuously over voluminous streams of movement data arriving at high speeds. In such scenarios, extracting SDEs out of raw streams is a challenging task engaging (a) online neural network training for continuously maintaining an up-to-date model for SDE labelling purposes and (b) semantic-aware trajectory processing for identifying homogeneous movement portions, defining the SDE duration, before using the neural model for labelling it. By definition, output SDEs are simple pieces of information (Listing 2), but the volume and velocity of the original raw streams (Listing 1) in large scale smart factory applications call for scaling out (parallelizing) the computation to a number of machines to ensure real-time processing. Therefore, both (a) and (b) should be set up in state-of-the-art, relevant platforms [7, 9] to allow for direct deployment over computer clusters and/or the cloud. To tackle these challenges we develop SSTRESED, a prototype for scalable SDE detection over streaming movement data. For the first time, SSTRESED establishes a direct connection between semantic trajectory computation and SDE detection in the streaming context. This is in contrast to prior art [9, 10] which uses predetermined, application-defined time windows to a priori restrict eligible SDE durations.

## 2 The SSTRESED Prototype

SSTRESED (Figure 1) composes two connected pipelines distributed across worker machines running in the cloud. In the robotic scenario of Section 1, truthful, timestamped and labeled movement streams are continuously produced by robotic simulators, such as <https://github.com/rock-simulation>, as SDEs and their raw features, per robot (Listing 1).



© Nikos Giatrakos;

licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

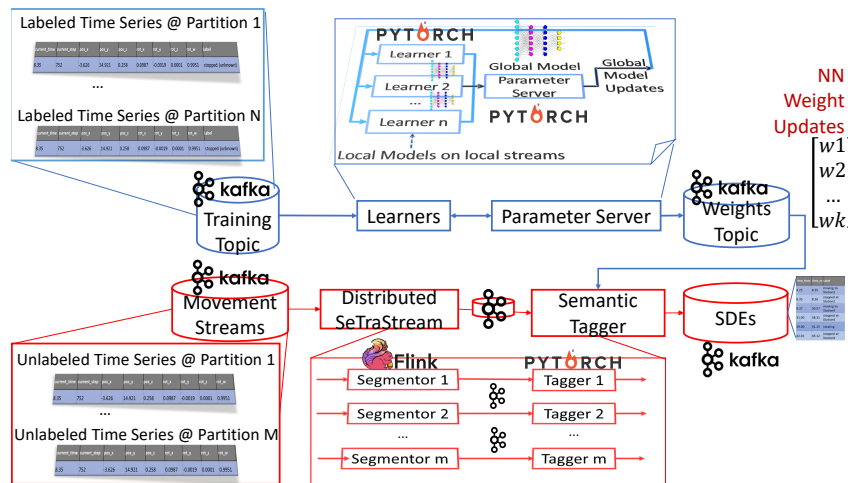
Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 15; pp. 15:1–15:4

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





■ **Figure 1** SSTRESED Architecture. Training (blue) and SDE Detection (red) Pipelines.

■ **Listing 1** Example training stream for a single simulated robot. Unlabelled movement streams lack a SDE label (last column in the figure). Thousands of such streams can be ingested by SSTRESED in large scale applications.

time	...	pos_x	pos_y	pos_z	...	rot_w	SDE
8.35	...	-3.626	14.921	0.258	...	0.9951	stopped at Station1
30.57	...	...	...	...	...	...	...
41.15	...	-7.446	23.866	0.257	...	0.0977	moves to Station3
41.12	...	-7.444	23.867	0.258	...	0.0972	rotating

The training pipeline (blue-colored path in Figure 1) continuously receives these robot movement time series ingested in Apache Kafka partitions of the `Training Topic`. The `Training Topic` is read by parallel PyTorch `Learners`. Each such learner, utilizes an identical neural model (specified by the application), but performs the training process on a separate set of robots. The local models learned at each `Learner i` (top of Figure 1) are synchronized into a global neural model maintained by a `Parameter Server` [2]. At a global model update, new weights of the neural network are written to a `Weights Topic` of Kafka.

The SDE detection pipeline (red-colored path in Figure 1) receives raw, unlabeled streaming movement data, partitioned in the `Movement Streams` Kafka Topic. These incoming tuples, ingested directly from the application field, have the same schema as those of the `Training Topic`, but lack a label/SDE field. Ingested `Movement Streams` of robots (or, optionally, samples of them [8, 11]) are processed by a distributed version of `SeTraStream` [12] developed in Apache Flink. `Distributed SeTraStream` uses each parallel `Segmentor i` to continuously identify homogeneous movement portions based on the ingested features per robot, thus semantically and temporally segmenting each trajectory. In that, the duration of a SDE is determined, which also bounds the feature tensors that should then be used for labeling the SDE. Each parallel `Segmentor i` writes the result of its processing to an intermediate Kafka topic connecting `Distributed SeTraStream` with a PyTorch `Semantic Tagger` in the red-colored path. Each parallel `Tagger i` (bottom of Figure 1) of the `Semantic Tagger`, at any given time instance, reads the up-to-date weights from the `Weights Topic` and uses the updated neural model to label SDEs. The final SSTRESED output goes to the `SDEs` Kafka topic in the form of tuples as illustrated in Listing 2 (per robot).

■ **Listing 2** SSTRESED output SDE Stream for the movement of a single robot.

Time_from	Time_to	SDE
4.25	8.35	moving to Station2
8.35	8.36	stopped at Station2
...	...	...
39.00	41.15	rotating

## References

- 1 Alexander Artikis, Nikos Katzouris, Ivo Correia, Chris Baber, Natan Morar, Inna Skarbovsky, Fabiana Fournier, and Georgios Paliouras. A prototype for credit card fraud management: Industry paper. In *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems, DEBS 2017, Barcelona, Spain, June 19-23, 2017*, pages 249–260. ACM, 2017.
- 2 Antonios Deligiannakis, Nikos Giatrakos, Yannis Kotidis, Vasilis Samoladas, and Alkis Simitsis. Extreme-scale interactive cross-platform streaming analytics - the INFORE approach. In *Proceedings of the 2nd Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores (SEA-Data 2021) co-located with 47th International Conference on Very Large Data Bases (VLDB 2021), Copenhagen, Denmark, August 20, 2021*, volume 2929 of *CEUR Workshop Proceedings*, pages 7–13. CEUR-WS.org, 2021.
- 3 Ioannis Flouris, Nikos Giatrakos, Antonios Deligiannakis, and Minos N. Garofalakis. Network-wide complex event processing over geographically distributed data sources. *Inf. Syst.*, 88, 2020.
- 4 Ioannis Flouris, Nikos Giatrakos, Minos N. Garofalakis, and Antonios Deligiannakis. Issues in complex event processing systems. In *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20-22, 2015, Volume 2*, pages 241–246. IEEE, 2015.
- 5 Nikos Giatrakos, Elias Alevizos, Alexander Artikis, Antonios Deligiannakis, and Minos N. Garofalakis. Complex event recognition in the big data era: a survey. *VLDB J.*, 29(1):313–352, 2020.
- 6 Nikos Giatrakos, Antonios Deligiannakis, Konstantina Bereta, Marios Vodas, Dimitris Zissis, Elias Alevizos, Charilaos Akasiadis, and Alexander Artikis. Processing big data in motion: Core components and system architectures with applications to the maritime domain. In Edward Curry, Sören Auer, Arne J. Berre, Andreas Metzger, María S. Pérez, and Sonja Zillner, editors, *Technologies and Applications for Big Data Value*, pages 497–518. Springer, 2022. doi:10.1007/978-3-030-78307-5\_22.
- 7 Nikos Giatrakos, Eleni Kougioumtzi, Antonios Kontaxakis, Antonios Deligiannakis, and Yannis Kotidis. Easyflinkcep: Big event data analytics for everyone. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 3029–3033. ACM, 2021.
- 8 Antonis Kontaxakis, Nikos Giatrakos, and Antonios Deligiannakis. A synopsis data engine for interactive extreme-scale analytics. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 2085–2088. ACM, 2020.
- 9 Emmanouil Ntoulías, Elias Alevizos, Alexander Artikis, Charilaos Akasiadis, and Athanasios Koumparos. Online fleet monitoring with scalable event recognition and forecasting. *GeoInformatica*, 26(4):613–644, 2022.
- 10 Kostas Patroumpas, Elias Alevizos, Alexander Artikis, Marios Vodas, Nikos Pelekis, and Yannis Theodoridis. Online event recognition from moving vessel trajectories. *GeoInformatica*, 21(2):389–427, 2017.
- 11 Marios Vodas, Konstantina Bereta, Dimitris Kladis, Dimitris Zissis, Elias Alevizos, Emmanouil Ntoulías, Alexander Artikis, Antonios Deligiannakis, Antonios Kontaxakis, Nikos Giatrakos, David Arnu, Edwin Yaqub, Fabian Temme, Mate Torok, and Ralf Klinkenberg. Online distributed maritime event detection & forecasting over big vessel tracking data. In *2021*

## 15:4 SSTRESED: Scalable Semantic Trajectory Extraction for Simple Event Detection

*IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, December 15-18, 2021*, pages 2052–2057. IEEE, 2021.

- 12 Zhixian Yan, Nikos Giatrakos, Vangelis Katsikaros, Nikos Pelekis, and Yannis Theodoridis. Setrastream: Semantic-aware trajectory construction over streaming movement data. In *Advances in Spatial and Temporal Databases - 12th International Symposium, SSTD 2011, Minneapolis, MN, USA, August 24-26, 2011, Proceedings*, volume 6849 of *Lecture Notes in Computer Science*, pages 367–385. Springer, 2011.

# A Decomposition Framework for Inconsistency Handling in Qualitative Spatial and Temporal Reasoning

Yakoub Salhi   

CRIL UMR 8188, Université d'Artois & CNRS, France

Michael Sioutis   

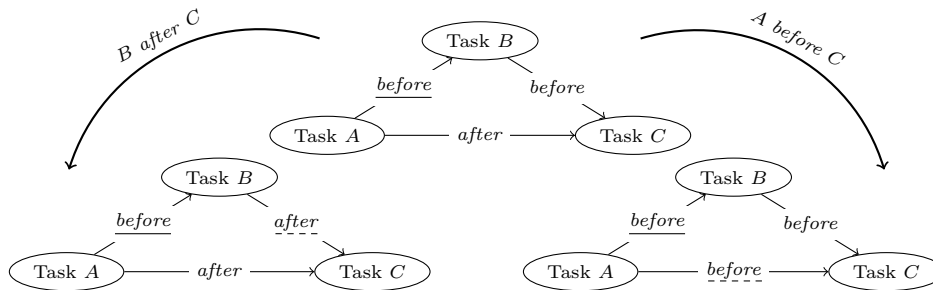
LIRMM UMR 5506, Université de Montpellier & CNRS, France

---

## Abstract

---

Dealing with inconsistency is a central problem in AI, due to the fact that inconsistency can arise for many reasons in real-world applications, such as context dependency, multi-source information, vagueness, noisy data, etc. Among the approaches that are involved in inconsistency handling, we can mention argumentation, non-monotonic reasoning, and paraconsistency, e.g., see [2, 3, 10]. In the work of [7], we are interested in dealing with inconsistency in the context of Qualitative Spatio-Temporal Reasoning (QSTR) [6]. QSTR is an AI framework that aims to mimic, natural, human-like representation and reasoning regarding space and time. This framework is applied to a variety of domains, such as qualitative case-based reasoning and learning [5] and visual sensemaking [9]; the interested reader is referred to [8] for a recent survey.



■ **Figure 1** A decomposition of an inconsistent qualitative constraint network (QCN) into consistent subnetworks (components).

**Motivation.** In [7], we study the decomposition of an inconsistent constraint network into consistent subnetworks under, possible, mandatory constraints. To illustrate the interest of such a decomposition, we provide a simple example described in Figure 1. The QCN depicted in the top part of the figure corresponds to a description of an inconsistent plan. Further, we assume that the constraint Task A {before} Task B is mandatory. To handle inconsistency, this plan can be transformed into a decomposition of two consistent plans, depicted in the bottom part of the figure; this decomposition can be used, e.g., to capture the fact that Task C must be performed twice. More generally, network decomposition can be involved in inconsistency handling in several ways: it can be used to identify potential contexts that explain the presence of inconsistent information; it can also be used to restore consistency through a compromise between the components of a decomposition, e.g., by using belief merging [4]; in addition, QCN decomposition can be used as the basis for defining inconsistency measures.

**Contributions.** We summarize the contributions of [7] as follows. First, we propose a theoretical study of a problem that consists in decomposing an inconsistent QCN into a bounded number of consistent QCNs that may satisfy a specified part in the original QCN; intuitively, the required common part corresponds to the constraints that are considered necessary, if any. To this end, we provide upper bounds for the minimum number of components in a decomposition as well as computational complexity results. Secondly, we provide two methods for solving our decomposition



© Yakoub Salhi and Michael Sioutis;  
licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 16; pp. 16:1–16:3

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

problem. The first method corresponds to a greedy constraint-based algorithm, a variant of which involves the use of spanning trees; the basic idea of this variant is that any acyclic constraint graph in QSTR is consistent, and such a graph can be used as a starting point for building consistent components. The second method corresponds to a SAT-based encoding; every model of this encoding is used to construct a valid decomposition. Thirdly, we consider two optimization versions of the initial decomposition problem that focus on minimizing the number of components and maximizing the similarity between components, respectively. The similarity between two QCNs is quantified by the number of common non-universal constraints; the interest in maximizing the similarity lies mainly in the fact that it reduces the number of constraints that allow each component to be distinguished from the rest. Of course, our previous methods are adapted to tackle these optimization versions, too. Additionally, we introduce two inconsistency measures based on QCN decomposition, which can be seen as counterparts of measures for propositional KBs introduced in [11, 1], and show that they satisfy several desired properties in the literature. Finally, we provide implementations of our methods for computing decompositions and experimentally evaluate them using different metrics.

**2012 ACM Subject Classification** Theory of computation → Constraint and logic programming; Computing methodologies → Temporal reasoning; Computing methodologies → Spatial and physical reasoning

**Keywords and phrases** Spatial and Temporal Reasoning, Qualitative Constraints, Inconsistency Handling, Decomposition, Inconsistency Measures

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.16

**Category** Extended Abstract

**Related Version** This is an extended abstract of our KR 2023 paper.

*Full Version:* <https://proceedings.kr.org/2023/59/> [7]

**Supplementary Material** *Software:* <https://seafiler.lirmm.fr/d/cff42be4169d433d88a7/>

**Funding** *Michael Sioutis:* The work was partially funded by the Agence Nationale de la Recherche (ANR) for the “Hybrid AI” project that is tied to the chair of Dr. Sioutis, and the I-SITE program of excellence of Université de Montpellier that complements the ANR funding.

---

## References


- 1 Meriem Ammoura, Yakoub Salhi, Brahim Oukacha, and Badran Raddaoui. On an MCS-based inconsistency measure. *Int. J. Approx. Reasoning*, 80:443–459, 2017.
- 2 Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. MIT Press, 2008.
- 3 Gerhard Brewka, Jürgen Dix, and Kurt Konolige. *Nonmonotonic Reasoning: An Overview*. CSLI Lecture Notes. CSLI Publications, Stanford, CA, 1997.
- 4 Jean-François Condotta, Souhila Kaci, Pierre Marquis, and Nicolas Schwind. A Syntactical Approach to Qualitative Constraint Networks Merging. In *LPAR*, 2010.
- 5 Thiago Pedro Donadon Homem, Paulo Eduardo Santos, Anna Helena Reali Costa, Reinaldo Augusto da Costa Bianchi, and Ramón López de Mántaras. Qualitative case-based reasoning and learning. *Artif. Intell.*, 283:103258, 2020.
- 6 Gérard Ligozat. *Qualitative Spatial and Temporal Reasoning*. ISTE. Wiley, 2013.
- 7 Yakoub Salhi and Michael Sioutis. A Decomposition Framework for Inconsistency Handling in Qualitative Spatial and Temporal Reasoning. In *KR*, 2023.
- 8 Michael Sioutis and Diedrich Wolter. Qualitative Spatial and Temporal Reasoning: Current Status and Future Challenges. In *IJCAI*, 2021.

- 9 Jakob Suchan, Mehul Bhatt, and Srikrishna Varadarajan. Commonsense visual sensemaking for autonomous driving - On generalised neurosymbolic online abduction integrating vision and semantics. *Artif. Intell.*, 299:103522, 2021.
- 10 Koji Tanaka, Francesco Berto, Edwin D. Mares, and Francesco Paoli, editors. *Paraconsistency: Logic and Applications*. Logic, Epistemology, and the Unity of Science. Springer, 2013.
- 11 Matthias Thimm. On the expressivity of inconsistency measures. *Artif. Intell.*, 234:120–151, 2016.





# Answer Set Automata: A Learnable Pattern Specification Framework for Complex Event Recognition

Nikos Katzouris ✉ 🏠 

Institute of Informatics, National Center for Scientific Research “Demokritos”, Athens, Greece

Georgios Paliouras ✉ 🏠 

Institute of Informatics, National Center for Scientific Research “Demokritos”, Athens, Greece

---

## Abstract

Complex Event Recognition (CER) systems detect event occurrences in streaming input using predefined event patterns. Techniques that learn event patterns from data are highly desirable in CER. Since such patterns are typically represented by symbolic automata, we propose a family of such automata where the transition-enabling conditions are defined by Answer Set Programming (ASP) rules, and which, thanks to the strong connections of ASP to symbolic learning, are learnable from data. We present such a learning approach in ASP, capable of jointly learning the structure of an automaton and its transition guards’ definitions from building-block predicates, and a scalable, incremental version thereof that progressively revises models learnt from mini-batches using Monte Carlo Tree Search. We evaluate our approach on three CER datasets and empirically demonstrate its efficacy.

**2012 ACM Subject Classification** Computing methodologies → Logic programming and answer set programming

**Keywords and phrases** Event Pattern Learning, Answer Set Programming

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.17

**Category** Extended Abstract

**Funding** *Nikos Katzouris*: This work is supported by the project EVENFLOW – “Robust Learning and Reasoning for Complex Event Forecasting”, which has received funding from the European Union’s Horizon research and innovation programme under grant agreement No 101070430.

*Georgios Paliouras*: This work is supported by the project EVENFLOW – “Robust Learning and Reasoning for Complex Event Forecasting”, which has received funding from the European Union’s Horizon research and innovation programme under grant agreement No 101070430.

## 1 Introduction

Complex Event Recognition (CER) systems [3] detect occurrences of *complex events* (CEs) in streaming input, using temporal patterns consisting of *simple events*, e.g. sensor data, or other complex events. CE patterns are typically defined by domain experts in some *event specification language*. Despite the diversity of such languages and the variety of the proposed event processing operators [4], a minimal set of such operators that every ECL should support [3, 4] includes *sequence* and *iteration* (*Kleene Closure*), implying respectively that some particular events should succeed one another temporally, or that an event should occur iteratively in a sequence, and the *filtering* operator, which matches input events that satisfy a set of predefined predicates.

Taken together, these three operators point to a computational model for CER based on symbolic finite automata [1] (SFA), i.e., automata where the transition-enabling conditions are predicates than need to be evaluated against the input, rather than mere symbols. As a



© Nikos Katzouris and Georgios Paliouras;

licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 17; pp. 17:1–17:3

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

result, in most existing CER systems CE patterns are either defined directly as SFA, or are compiled into such at runtime. Prominent areas of CER research, then, concern the study of trade-offs between ECLs’ expressive power and pattern matching complexity, in addition to more practical considerations, such as scaling-up concrete pattern matching algorithms, or dealing with fault tolerance and distributed processing.

CE pattern learning is a less studied CER topic, which, however, is of utmost importance, since CE patterns are not always known in advance, or they frequently need to be revised. A few learning approaches have been proposed, which have several limitations. Some are restricted to sequence-based ESLs that do support operators such as iteration [7, 5, 2], others do not allow for filtering predicates [6], most offer very limited support for reasoning with background knowledge during pattern induction and none supports CE pattern revision.

To address such issues we propose *answer set automata learning*, a framework that allows to specify SFA-based CE patterns in the form of answer set programs (ASP), which, thanks to the strong connections of ASP to symbolic learning, are directly learnable and revisable from data. We begin by encoding CE patterns specified in any ESL that supports the core CER operators of filtering, sequence, iteration, disjunction and conjunction into *answer set automata*, i.e. executable ASP specifications of the SFA that correspond to the initial CE patterns. Our CE patterns-to-ASP programs translation comes with a correctness property, stating that a pattern will be matched against a particular finite piece of input when run with a CER engine, iff its corresponding answer set automaton program satisfies a particular query, when run on the same input with an ASP solver.

The established connection between event pattern matching and logical reasoning allows to learn the ASP program equivalent of a CE pattern from labeled event traces, via abductive learning w.r.t. to an SFA interpreter. Our learning approach, implemented directly on top of an ASP solver, allows to synthesize patterns utilizing the core CER operators by jointly learning the structure of the corresponding SFA and the definitions of its transition guards, consisting of boolean combinations of building-block, background knowledge predicates.

To scale-up the abductive learning core of our SFA synthesis method to large training sets, we propose an incremental learning technique utilizing SFA revision in a Monte Carlo Tree Search (MCTS) that continuously revises programs learnt from mini-batches of the data, in an effort to approximate a global optimum. The revision operators can modify the structure of an automaton, by adding/removing states and transitions, or the structure of the transition guard rules, by adding/removing conditions from such rules’ bodies. These revision operators are realized via same abductive learning technique that handles batch learning, using constraints generated from the labeled traces in each mini-batch to guide the search for optimal “local” revisions. MCTS is used to stochastically search in the massive space of SFA structures, while balancing exploitation, i.e. revising already identified, globally-good SFA, in an effort to further improve their quality, with exploration, i.e. revising less promising SFA, in an effort to escape local optima.

We evaluate our SFA learning approach on three CER datasets and empirically demonstrate its efficacy. We also compare our technique to classical automata learning methods on univariate input and show its superiority, both in terms of predictive accuracy and scalability.

---

## References

- 1 Loris D’Antoni and Margus Veanes. The power of symbolic automata and transducers. In *International Conference on Computer Aided Verification*, pages 47–67. Springer, 2017.
- 2 Lars George, Bruno Cadonna, and Matthias Weidlich. Il-miner: instance-level discovery of complex event patterns. *Proceedings of the VLDB Endowment*, 10(1):25–36, 2016.

- 3 Nikos Giatrakos, Elias Alevizos, Alexander Artikis, Antonios Deligiannakis, and Minos N. Garofalakis. Complex event recognition in the big data era: a survey. *VLDB J.*, 29(1):313–352, 2020.
- 4 Alejandro Grez, Cristian Riveros, Martín Ugarte, and Stijn Vansummeren. A formal framework for complex event recognition. *ACM Transactions on Database Systems (TODS)*, 46(4):1–49, 2021.
- 5 Sarah Kleest-Meißner, Rebecca Sattler, Markus L Schmid, Nicole Schweikardt, and Matthias Weidlich. Discovering multi-dimensional subsequence queries from traces—from theory to practice. *BTW 2023*, 2023.
- 6 Yan Li and Tingjian Ge. Imminence monitoring of critical events: A representation learning approach. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1103–1115, 2021.
- 7 Alessandro Margara, Gianpaolo Cugola, and Giordano Tamburrelli. Learning from the past: automated rule generation for complex event processing. In *Proceedings of the 8th ACM international conference on distributed event-based systems*, pages 47–58, 2014.



# A Benchmark for Early Time-Series Classification

Petro-Foti Kamberi ✉

Institute of Informatics & Telecommunications, NCSR “Demokritos”, Athens, Greece

Evgenios Kladis ✉

Institute of Informatics & Telecommunications, NCSR “Demokritos”, Athens, Greece

Charilaos Akasiadis ✉ 

Institute of Informatics & Telecommunications, NCSR “Demokritos”, Athens, Greece

---

## Abstract

The objective of Early Time-Series Classification (ETSC) is to predict the class of incoming time-series by observing the fewest time-points possible. Although many approaches have been proposed in the past, not all techniques are suitable for every problem type. In particular, the characteristics of the input data may impact performance. To aid researchers and developers with deciding which kind of method suits their needs best, we developed a framework that allows the comparison of five existing ETSC algorithms, and also introduce a new method that is based on the selective truncation of time-series principle. To promote results reproducibility and the alignment of algorithm comparisons, we also include a bundle of datasets originating from real-world time-critical applications, and for which the application of ETSC algorithms can be considered quite valuable.

**2012 ACM Subject Classification** Computing methodologies → Machine learning algorithms

**Keywords and phrases** Time-series analysis, Classification, Benchmark

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.18

**Category** Extended Abstract

**Related Version** *Previous Version*: <https://arxiv.org/abs/2203.01628>

**Supplementary Material** *Software*: <https://github.com/xarakas/ETSC>  
archived at `swh:1:dir:9b337165e0682df9285e5f66c99e4722a71c2988`

**Funding** This work has received funding from the EU project CREXDATA (under grant agreement No. 101092749).

## 1 Extended Abstract

Latest technological advancements drive the generation of large volumes of time-series data. Sea vessels, for example, utilize integrated sensory and telecommunication devices to continuously report trajectory information in the form of time-series data. This abundance of data is leveraged by machine learning techniques to address various problems. In the life sciences domain, simulators are incorporated to test the effectiveness of new experimental drugs “in-silico”. Such simulations often require long time and large amounts of computational resources, which, in the case of unsuccessful drug treatment cases being simulated, are consumed in vain [1]. It would be desirable to be able to predict such outcomes early on by observing the simulations course as time-series, and to terminate not interesting trials so to speed up the whole drug discovery process. To this end, the domain of Early Time-Series Classification (ETSC) has an objective to classify time-series at the earliest point possible, before the entire series is observed [5].

Meanwhile, despite the numerous proposed methods for ETSC, there is a notable absence of a dedicated experimental evaluation and comparison framework in this field. Furthermore, ETSC methods are predominantly evaluated and compared against only a limited set of



© Petro-Foti Kamberi, Evgenios Kladis, and Charilaos Akasiadis;  
licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 18; pp. 18:1–18:3

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

alternative algorithms. In order to fill this gap, we have developed a framework that allows to empirically compare five existing approaches as well as a newly introduced one, using a curated bundle of datasets from real-world applications. This framework is utilized to highlight the ETSC algorithms merits and shortcomings when applied to cases with different features, e.g. dataset size, observation variability, class imbalance, etc. The framework can be easily extended to include more datasets and algorithms, and is openly available online [2].

**Evaluation metrics.** In the ETSC domain, apart from the predictive performance (accuracy and  $F_1$ -score), there is also the objective to optimize the earliness of the generated predictions. These two metrics can also be combined into a single metric as a harmonic mean [9]. Training and testing times are also of interest to consider in real-world applications.

**Algorithms.** Our framework includes five existing ETSC algorithms, i.e. *ECEC* [7], *ECONOMY-K* [3], *ECTS* [10], *EDSC* [11], and *TEASER* [9]. *ECEC* calculates confidence thresholds above which a class label prediction is considered to be reliable. *ECONOMY-K* performs clustering on the training data, and estimates the cost of having to observe more time points to generate a prediction. *ECTS* utilizes nearest neighbors and reverse nearest neighbors sets for its decisions. *EDSC* extracts shapelets of the training data that are then matched with the input test data. *TEASER* trains classifiers on overlapping prefixes of the training data, and then applies an one-class SVM that validates the class label prediction.

In addition, we propose a new method that can be configured to utilize different state-of-the-art full time-series classification algorithms for ETSC. It relies on iteratively truncating time-series into prefixes of gradually increasing length and then applying *Minirocket* [4], *MLSTM* [6], and *WEASEL* [8] to the truncated examples for predicting the corresponding class labels. Thus, for each dataset, a fixed earliness is determined throughout the training phase. Although this might be suboptimal in the sense that for particular instances the prediction could have been generated earlier than for others, it constitutes a comprehensive baseline for diagnosing if applying ETSC to particular datasets and domains would be successful, i.e. if accurate predictions can be generated earlier, before the whole time-series is observed. Our approach can incorporate any full time-series classification algorithm.

**Datasets.** We have collected 10 publicly available datasets from the UEA & UCR Time-series Classification Repository, and also introduced two new cases, one originating from the drug discovery domain, and the other from the field of maritime intelligence. In turn, these datasets are categorized according to particular characteristics: size, coefficient of variability, levels of class imbalance, number of distinct class labels, and number of variables.

**Comparison results.** Judging by our experimental results, we can see that *ECEC* achieves the most accurate and early predictions for datasets with lengthy time-series, but it requires higher training times. When the number of examples in the dataset increases, the *MLSTM* variant of our method competes with *ECEC* and *TEASER* in terms of the harmonic mean between accuracy and earliness, but it has higher training times compared to both. In applications with high variance in measurements and high class imbalance, *ECEC* and *MLSTM* achieve the highest harmonic mean scores. In multi-class classification cases, *MLSTM* is the best choice with the lowest earliness scores, followed by *Minirocket*, which has high accuracy and reduced training times. For the rest of the datasets we tested, *Minirocket* is the most suitable algorithm for ETSC in terms of harmonic mean. It has very low earliness scores and training times, although its predictive accuracy is worse than *ECEC*, *ECONOMY-K*, and *ECTS*, which however achieve higher earliness scores.

As concerns future work, we are already in the process of further enriching our framework with additional ETSC algorithms and more datasets. We believe that establishing consolidated benchmarking procedures will be of great benefit for the ETSC community.

---

## References




---

- 1 C. Akasiadis, M. Ponce-de Leon, A. Montagud, E. Michelioudakis, A. Atsidakou, E. Alevizos, A. Artikis, A. Valencia, and G. Paliouras. Parallel model exploration for tumor treatment simulations. *Computational Intelligence*, 38(4):1379–1401, 2022. doi:10.1111/coin.12515.
- 2 Charilaos Akasiadis, Evgenios Kladis, Evangelos Michelioudakis, Elias Alevizos, and Alexander Artikis. Early time-series classification algorithms: An empirical comparison. *arXiv preprint arXiv:2203.01628*, 2022.
- 3 A. Dachraoui, A. Bondu, and A. Cornuéjols. Early classification of time series as a non myopic sequential decision making problem. In *Joint European Conf. on Machine Learning and Knowledge Discovery in Databases*, pages 433–447. Springer, 2015.
- 4 A. Dempster, D. F. Schmidt, and G. I. Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proc. of the 27th ACM SIGKDD Conf. on Knowledge Discovery & Data Mining*, pages 248–257, 2021.
- 5 A. Gupta, H. P. Gupta, B. Biswas, and T. Dutta. Approaches and applications of early classification of time series: A review. *IEEE Trans. Artif. Intell.*, 1(1):47–61, 2020.
- 6 F. Karim, S. Majumdar, H. Darabi, and S. Harford. Multivariate LSTM-FCNs for time series classification. *Neural Networks*, 116:237–245, 2019.
- 7 J. Lv, X. Hu, L. Li, and P.-P. Li. An effective confidence-based early classification of time series. *IEEE Access*, 7:96113–96124, 2019.
- 8 P. Schäfer and U. Leser. Fast and accurate time series classification with WEASEL. In *Proc. of the 2017 ACM Conf. on Information and Knowledge Management*, pages 637–646, 2017.
- 9 P. Schäfer and U. Leser. Teaser: early and accurate time series classification. *Data mining and knowledge discovery*, 34(5):1336–1362, 2020.
- 10 Z. Xing, J. Pei, and P. S. Yu. Early classification on time series. *Knowledge and information systems*, 31(1):105–127, 2012.
- 11 Z. Xing, J. Pei, P. S. Yu, and K. Wang. Extracting interpretable features for early classification on time series. In *Proceedings of the 2011 SIAM international conference on data mining*, pages 247–258. SIAM, 2011.







# Time-Aware Robustness of Temporal Graph Neural Networks for Link Prediction

Marco Sälzer   

School of Electrical Engineering and Computer Science, University of Kassel, Germany

Silvia Beddar-Wiesing  

School of Electrical Engineering and Computer Science, University of Kassel, Germany

---

## Abstract

We present a first notion of a time-aware robustness property for Temporal Graph Neural Networks (TGNN), a recently popular framework for computing functions over continuous- or discrete-time graphs, motivated by recent work on time-aware attacks on TGNN used for link prediction tasks. Furthermore, we discuss promising verification approaches for the presented or similar safety properties and possible next steps in this direction of research.

**2012 ACM Subject Classification** Computing methodologies → Neural networks; Security and privacy → Logic and verification

**Keywords and phrases** graph neural networks, temporal, verification

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.19

**Category** Extended Abstract

## Introduction

Graph Neural Networks (GNN) provide a framework for computing functions over graphs based on learnable parameters and have gained much attention in recent years [7]. The most popular GNN models, so-called convolutional GNN or message-passing GNN apply a neighborhood aggregation procedure to each node in a graph to compute its output. Usually, such GNNs are used for classification or prediction tasks over static graphs. However, this limits their applicability in contexts like social networks or knowledge graphs, where underlying graphs change stepwise or time-continuously. Temporal Graph Neural Networks<sup>1</sup> (TGNN) [5, 6] try to close this gap. The general idea of TGNN is to generalize the neighborhood aggregation procedure mentioned above to temporal graphs, usually represented as a tuple of a base graph with a series of time-stamped observed changes. In most applications involving Neural Network based models, giving reliable safety certificates is highly desirable but also a significant challenge, especially because of the blackbox nature of neural models. In this extended abstract, we address the topic of verifying TGNN, which is an unexplored area of research. We present a time-aware robustness property for TGNN used for link prediction tasks, which is motivated by recent work on similar time-aware attacks [3]. Additionally, we discuss our ongoing work regarding promising verification approaches for the introduced (or similar) safety property.

## Preliminaries

*Temporal Graphs:* A Continuous-Time Temporal Graph (CTG) is a tuple  $(G, \mathcal{O})$  where  $G$  is a graph, often called start graph, and  $\mathcal{O}$  is a finite set of time-stamped observations, including events like node or edge additions or deletions. We denote by  $G_{\leq t}^{\mathcal{O}}$  for some  $t \in \mathbb{Q}^{\geq 0}$  the

---

<sup>1</sup> Equivalently, these models are also called Dynamic Graph Neural Networks (DGNN).



graph constructed by applying the observations from  $\mathcal{O}$  with a timestamp  $t' \leq t$  to  $G$ . Note that each CTG can be seen as a finite sequence of graphs by unfolding  $\mathcal{O}$  in a stepwise fashion. For more details on the notions of temporal graphs used in the context of temporal graph learning, see [4]. *Link Prediction for CTG*: Given a CTG  $C = (G, \mathcal{O})$ , the link prediction task is to predict for a time  $t \in \mathbb{Q}^{\geq 0}$  and pair of nodes  $u, v$  present in  $G_{\leq t}^{\mathcal{O}}$  whether an edge  $(u, v)$  will be present in the graph at time  $t$ . We denote the output of a TGNN for the above-described link prediction task by  $N(C, (u, v), t)$ .

### Time-aware Robustness in the Context of Link Prediction

Chen et al. [3] present a notion of pointwise (adversarial) attacks on link predicting TGNN, exploiting the time component of temporal graphs. Similarly, we present the following definition of a pointwise time-aware (adversarial) robustness certificate for TGNN.

► **Definition 1.** *Let  $N$  be a TGNN,  $C = (G, \mathcal{O})$  a CTG and  $B = (\mathcal{O}_1, \mathcal{O}_2)$  an adversarial budget of two sets  $\mathcal{O}_1, \mathcal{O}_2$  of observations where  $\mathcal{O}_1 \subseteq \mathcal{O}$ . We say that  $N$  is robust for nodes  $u, v$  and time  $t$  under influence of  $B$  if  $N(C, (u, v), t) = N(C', (u, v), t)$  where  $C' = (G, (\mathcal{O} \setminus \mathcal{O}_1) \cup \mathcal{O}_2)$ .*

While this exact robustness certificate is desirable, a computationally feasible and complete verification algorithm is unlikely, as recent results about the decidability and complexity of similar safety properties for GNN [9] indicate. Therefore, we propose two approaches: (A) one focuses on the development of non-complete verification algorithms, similar to [10] for GNN, or (B) one gives up on exact verification and relaxes Def. 1 to a probabilistic certificate, similar to [2] for GNN. The two approaches have advantages and disadvantages: (A) allows for exact verification but most likely depends on the underlying TGNN model, making model-specific verification algorithms necessary. Approach (B) can be model-agnostic but can only give probabilistic certificates.

### Outlook

We introduced a first notion of robustness for TGNN in the context of link prediction, inspired by common (adversarial) attack and robustness certificates for Neural Network based models, and discussed possible verification approaches. However, this can only be seen as a first step in developing a well-founded framework for the verification of TGNN. Next to developing efficient verification algorithms, a desirable goal is to combine TGNN verification with well-founded specification languages or temporal logic. Since TGNNs work over finite sequences or traces of graphs, a similar logic to Linear Temporal Logic (LTL) on finite traces [8] could be promising. However, the traces considered here work over infinite domains, making more expressive LTL variants necessary, like in [1].

---



### References

- 1 Artale et al. First-order temporal logic on finite traces: Semantic properties, decidable fragments, and applications. *CoRR*, abs/2202.00610, 2022. URL: <https://arxiv.org/abs/2202.00610>.
- 2 Bojchevski et al. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1003–1013. PMLR, July 2020. URL: <https://proceedings.mlr.press/v119/bojchevski20a.html>.

- 3 Chen et al. Time-aware gradient attack on dynamic network link prediction. *IEEE Trans. Knowl. Data Eng.*, 35(2):2091–2102, 2023. doi:10.1109/TKDE.2021.3110580.
- 4 Kazemi et al. Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.*, 21:70:1–70:73, 2020. URL: <http://jmlr.org/papers/v21/19-447.html>.
- 5 Longa et al. Graph Neural Networks for Temporal Graphs: State of the Art, Open Challenges, and Opportunities. *arXiv preprint arXiv:2302.01018*, 2023.
- 6 Skarding et al. Foundations and Modeling of Dynamic Networks using Dynamic Graph Neural Networks: A Survey. *IEEE Access*, 9:79143–79168, 2021.
- 7 Wu et al. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24, 2021. doi:10.1109/TNNLS.2020.2978386.
- 8 Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 854–860. IJCAI/AAAI, 2013. URL: <http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997>.
- 9 Marco Sälzer and Martin Lange. Fundamental limits in formal verification of message-passing neural networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL: <https://openreview.net/forum?id=W1bG82OmRH->.
- 10 Daniel Zügner and Stephan Günnemann. Certifiable robustness and robust training for graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 246–256. ACM, 2019. doi:10.1145/3292500.3330905.



# Converting Simple Temporal Networks with Uncertainty into Dispatchable Form – Faster

Luke Hunsberger  

Vassar College, Poughkeepsie, NY, USA

Roberto Posenato   

University of Verona, Italy

---

## Abstract

---

In many sectors of real-world industry, it is necessary to plan and schedule tasks allocated to agents participating in complex processes. Temporal planning aims to schedule tasks while respecting temporal constraints such as release times, maximum durations, and deadlines, which requires quantitative temporal reasoning. Over the years, several major application developers have highlighted the need for the explicit representation of actions with uncertain durations; efficient algorithms for determining whether plans involving such actions are controllable; and efficient algorithms for converting such plans into forms that enable them to be executed in real time with minimal computation, while preserving maximum flexibility.

A Simple Temporal Network with Uncertainty (STNU) is a data structure for reasoning about time constraints on actions that may have uncertain durations. An STNU is a triple  $(\mathcal{T}, \mathcal{C}, \mathcal{L})$  where  $\mathcal{T}$  is a set of real-valued variables called *timepoints*,  $\mathcal{C}$  is a set of constraints of the form  $Y - X \leq \delta$ , where  $X, Y \in \mathcal{T}$  and  $\delta \in \mathbf{R}$ , and  $\mathcal{L}$  is a set of *contingent links* of the form  $(A, x, y, C)$ , where  $A, C \in \mathcal{T}$  and  $0 < x < y < \infty$ . A contingent link  $(A, x, y, C)$  represents an uncertain duration where  $A$  is the *activation timepoint*,  $C$  is the *contingent timepoint*, and  $y - x$  is the uncertainty in the duration  $C - A$ . Typically, an executor controls the execution of  $A$ , but only observes the execution of  $C$  in real time. Although uncontrollable, the duration is guaranteed to satisfy  $C - A \in [x, y]$ . We let  $n = |\mathcal{T}|$ ,  $m = |\mathcal{C}|$  and  $k = |\mathcal{L}|$ .

An STNU graph is a pair  $(\mathcal{T}, \mathcal{E})$ , where the timepoints in  $\mathcal{T}$  serve as nodes in the graph, and the edges in  $\mathcal{E}$  correspond to the constraints in  $\mathcal{C}$  and contingent links in  $\mathcal{L}$ . For each  $Y - X \leq \delta$  in  $\mathcal{C}$ ,  $\mathcal{E}$  contains an *ordinary* edge  $X \xrightarrow{\delta} Y$ . For each  $(A, x, y, C) \in \mathcal{L}$ ,  $\mathcal{E}$  contains a *lower-case* (LC) edge,  $A \xrightarrow{-x} C$ , and an *upper-case* (UC) edge,  $C \xrightarrow{-y} A$ , representing the respective possibilities that  $C - A$  might take its minimum or maximum value. The *LO-edges* are the LC or ordinary edges; the *OU-edges* are the ordinary or UC edges.

For any STNU, it is important to determine whether it is *dynamically controllable* (DC) (i.e., whether it is possible, in real time, to schedule its non-contingent timepoints such that all constraints will necessarily be satisfied no matter what durations turn out for the contingent links). Polynomial-time algorithms are available to solve this *DC-checking* problem. Each uses rules to generate new edges aiming to bypass certain kinds of edges in the STNU graph. Morris'  $O(n^4)$ -time DC-checking algorithm [3] starts from LC edges, propagating forward along OU-edges, looking for opportunities to generate new OU-edges that bypass the LC edges. Morris'  $O(n^3)$ -time algorithm [4] starts from negative OU-edges, propagating backward along LO-edges, aiming to bypass negative edges with non-negative edges. The  $O(mn + k^2n + kn \log n)$ -time  $RUL^-$  algorithm [1] starts from UC edges, propagating backward along LO-edges, aiming to bypass UC edges with ordinary edges. After propagating, each algorithm checks for certain kinds of negative cycles to decide DC-vs.-non-DC.

However, being DC only asserts the *existence* of a dynamic scheduler. It is also crucial to be able to *execute* a DC STNU efficiently in real time. For maximum flexibility and minimal space and time requirements, a dynamic scheduler for an STNU is typically computed *incrementally*, in real time, so that it can react to observations of contingent executions as they occur. An efficient dynamic scheduler can be realized by first transforming an STNU into an equivalent *dispatchable* form [6, 8]. Then, to execute the dispatchable STNU, it suffices to maintain *time-windows* for each timepoint and, as each timepoint  $X$  is executed, only updating time-windows for neighbors of  $X$  in the graph. Dispatchable STNUs are very important in applications that demand quick responses to observations of contingent events.



© Luke Hunsberger and Roberto Posenato;

licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 20; pp. 20:1–20:3

Leibniz International Proceedings in Informatics



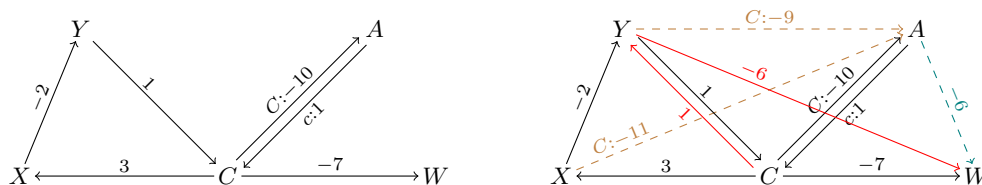
LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 20:2 Converting STNUs into Dispatchable Form – Faster

Of the existing DC-checking algorithms, only Morris'  $O(n^3)$ -time algorithm necessarily generates a dispatchable STNU for DC inputs. This abstract describes a faster,  $O(mn + kn^2 + n^2 \log n)$ -time algorithm for converting DC STNUs into dispatchable form. (The full journal article is available elsewhere [2].) This improvement is significant for applications (e.g., modeling business processes) where networks are typically sparse. For example, if  $m = O(n \log n)$  and  $k = O(\log n)$ , then our algorithm runs in  $O(n^2 \log n) \ll O(n^3)$  time.

Our new *Fast Dispatch* algorithm,  $\text{FD}_{\text{STNU}}$ , has three phases. The first phase is similar to the  $\text{RUL}^-$  DC-checking algorithm, but generates an order-of-magnitude fewer edges overall, while also generating new UC edges that correspond to *wait constraints*. The second phase is a version of Morris' 2006 algorithm that propagates forward from LC edges, but only along LO-edges, aiming to generate *ordinary* bypass edges. The third phase focuses on the subgraph of *ordinary* edges, which comprise a Simple Temporal Network (STN). It uses an existing dispatchability algorithm for STNs [8] to convert that ordinary subgraph into a dispatchable STN. After completing the three phases, the STNU is guaranteed to be dispatchable.

The left-hand graph below is the graph for a sample STNU that happens to be DC.



The right-hand graph shows the edges inserted by our  $\text{FD}_{\text{STNU}}$  algorithm. The first phase applies the modified  $\text{RUL}^-$  algorithm, propagating backward from the original UC edge  $(C, c:1, A)$  to generate the (brown, dashed) wait edges  $(Y, C: -9, A)$  and  $(X, C: -11, A)$ . (The original  $\text{RUL}^-$  algorithm only generates ordinary edges.) The second phase propagates *forward* from the LC edge  $(A, c:1, C)$  to generate the (teal, dashed) edge  $(A, -6, W)$ . The third phase runs the pre-existing STN-dispatchability algorithm on the *ordinary* STN subgraph, inserting the (red) edges  $(C, 1, Y)$  and  $(Y, -6, W)$ . Using the mathematical analysis of dispatchability due to Morris [5], it is not hard to confirm that this STNU is dispatchable.

We provide the source code of a Java implementation of the considered algorithms (Morris,  $\text{RUL}^-$ , and  $\text{FD}_{\text{STNU}}$ ) [7] and the benchmarks used to compare their performances.

**2012 ACM Subject Classification** Computing methodologies  $\rightarrow$  Temporal reasoning; Theory of computation  $\rightarrow$  Dynamic graph algorithms

**Keywords and phrases** Temporal constraint networks, contingent durations, dispatchable network

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.20

**Category** Extended Abstract

**Related Version** *Full Version*: <https://doi.org/10.1016/j.ic.2023.105063> [2]

**Supplementary Material** *Software (Source code)*: <https://profs.scienze.univr.it/~posenato/software/cstnu/> [7]

**Funding** *Luke Hunsberger*: National Science Foundation [Grant No. 1909739].

---

### References

- 1 Massimo Cairo, Luke Hunsberger, and Romeo Rizzi. Faster Dynamic Controllability Checking for Simple Temporal Networks with Uncertainty. In *25th International Symposium on Temporal Representation and Reasoning (TIME-2018)*, volume 120 of *LIPIcs*, pages 8:1–8:16, 2018. doi:10.4230/LIPIcs.TIME.2018.8.



- 2 Luke Hunsberger and Roberto Posenato. A Faster Algorithm for Converting Simple Temporal Networks with Uncertainty into Dispatchable Form. *Information and Computation*, 293(105063):1–21, 2023. doi:10.1016/j.ic.2023.105063.
- 3 Paul Morris. A Structural Characterization of Temporal Dynamic Controllability. In *Principles and Practice of Constraint Programming (CP-2006)*, volume 4204, pages 375–389, 2006. doi:10.1007/11889205\_28.
- 4 Paul Morris. Dynamic controllability and dispatchability relationships. In *CPAIOR 2014*, volume 8451 of *LNCS*, pages 464–479. Springer, 2014. doi:10.1007/978-3-319-07046-9\_33.
- 5 Paul Morris. The Mathematics of Dispatchability Revisited. In *26th International Conference on Automated Planning and Scheduling (ICAPS-2016)*, pages 244–252, 2016. doi:10.1609/icaps.v26i1.13739.
- 6 Nicola Muscettola, Paul H. Morris, and Ioannis Tsamardinos. Reformulating Temporal Plans for Efficient Execution. In *6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR-1998)*, pages 444–452, 1998.
- 7 Roberto Posenato. CSTNU Tool: A Java library for checking temporal networks. *SoftwareX*, 17:100905, 2022. doi:10.1016/j.softx.2021.100905.
- 8 Ioannis Tsamardinos, Nicola Muscettola, and Paul Morris. Fast Transformation of Temporal Plans for Efficient Execution. In *15th National Conf. on Artificial Intelligence (AAAI-1998)*, pages 254–261, 1998.



# Towards Infinite-State Verification and Planning with Linear Temporal Logic Modulo Theories

Luca Geatti ✉

University of Udine, Italy

Alessandro Gianola ✉

Free University of Bozen-Bolzano, Italy

Nicola Gigante ✉

Free University of Bozen-Bolzano, Italy

---

## Abstract

In this extended abstract, we discuss about *Linear Temporal Logic Modulo Theories over finite traces* ( $LTL_f^{MT}$ ), a temporal logic that we recently introduced with the goal of providing an equilibrium between generality of the formalism and decidability of the logic. After recalling its distinguishing features, we discuss some future applications.

**2012 ACM Subject Classification** Theory of computation → Logic and verification

**Keywords and phrases** Linear Temporal Logic, Satisfiability Modulo Theories

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.21

**Category** Extended Abstract

## 1 Overview

Linear Temporal Logic (LTL) [11] is arguably the most common language for the specification of system properties in the field of *formal verification*. In recent years, its *finite-traces* counterpart,  $LTL_f$  [4], also took traction in the field of *artificial intelligence*, where reasoning about a finite execution (*e.g.* in planning) is more natural. These formalisms are *propositional logics*, and are therefore suitable to specify and reason about *finite-state* systems. However, many complex scenarios, for example involving arithmetic constraints, complex data types, or relational databases, require to go beyond finite-state systems.

Here, we discuss our take on the problem of specifying and reasoning about infinite-state systems. To this aim, we recently introduced  $LTL_f$  *Modulo Theories* ( $LTL_f^{MT}$ ) [6], an extension of  $LTL_f$  where propositions are replaced by first-order formulas interpreted over arbitrary theories, similarly to how *satisfiability modulo theories* (SMT) [1] extends the classic Boolean satisfiability problem, and where first-order variables referring to different time points can be compared.

$LTL_f^{MT}$  is, in general, undecidable, but for decidable underlying theories it is *semi-decidable*<sup>1</sup>, with an effective semi-decision procedure based on the encoding of a tree-shaped tableau system into first-order logic, handled directly by off-the-shelf SMT solvers. The technique is implemented in the BLACK<sup>2</sup> temporal reasoning framework [7, 8], providing interesting performance. This puts this approach in contrast with other previous studies of first-order extensions of LTL: on one hand, such extensions have been thoroughly studied only from a theoretical perspective, without providing practical reasoning tools [9]; on the other hand, other practice-oriented approaches resulted in efficient tools for ad-hoc extensions (*e.g.* [3]), but difficult to extend or generalize. In contrast, our framework provides a most

---

<sup>1</sup> This would not be true if we interpreted the logic over *infinite traces* instead.



general and theoretically well-founded ground which can also lead to effective reasoning tools. We discuss here future applications of these framework in the context of formal verification and artificial intelligence.

## 2 Applications

The first-order setting of  $LTL_f^{MT}$  naturally allows one to specify and reason about structured and complex scenarios. An important use case is that of *data-aware* systems, *i.e.* systems that manipulate unbounded data. Such data can come from numeric variables, or other kinds of unbounded data types. In the most general setting, data comes from *relational databases*, which are naturally grounded in first-order logic and are therefore perfectly suitable to be modeled in  $LTL_f^{MT}$ , including relations with primary and foreign key constraints, and many other features.

These considerations lead to the definition of *knowledge-base-driven systems* (KDS), a kind of transition system whose behavior depends on the content of a mutable relational data store, that is updated by the transitions of the system. Work defining and studying KDSs is under review. This concept is of uttermost generality, potentially subsuming many different approaches found in the literature [2, 5], while still being directly handled by the BLACK solver.

On the other hand, inspired by the tight relationship between LTL satisfiability and *classical planning* [10], the same framework can be adapted to approach *data-aware planning problems*, a scenario still unexplored in the planning literature. In such problems, the agent is required to reason about actions whose preconditions depend on the content of an unbounded relational data store, and whose effects update such data store. These kind of planning problems could find applications in a wide range of scenarios such as planning for data warehouses, business process management, and ontology-driven systems.

The  $LTL_f^{MT}$  framework is still in its infancy, and interesting developments are on its path.

---

### References

- 1 Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. IOS Press, 2009. doi:10.3233/978-1-58603-929-5-825.
- 2 Diego Calvanese, Silvio Ghilardi, Alessandro Gianola, Marco Montali, and Andrey Rivkin. SMT-based verification of data-aware processes: a model-theoretic approach. *Math. Struct. Comput. Sci.*, 30(3):271–313, 2020. doi:10.1017/S0960129520000067.
- 3 Alessandro Cimatti, Alberto Griggio, Enrico Magnago, Marco Roveri, and Stefano Tonetta. Smt-based satisfiability of first-order LTL with event freezing functions and metric operators. *Inf. Comput.*, 272:104502, 2020. doi:10.1016/j.ic.2019.104502.
- 4 Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 854–860. IJCAI/AAAI, 2013.
- 5 Alin Deutsch, Yuliang Li, and Victor Vianu. Verification of hierarchical artifact systems. *ACM Trans. Database Syst.*, 44(3):12:1–12:68, 2019.
- 6 Luca Geatti, Alessandro Gianola, and Nicola Gigante. Linear temporal logic modulo theories over finite traces. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 2641–2647. ijcai.org, 2022. doi:10.24963/ijcai.2022/366.

---

<sup>2</sup> <https://www.black-sat.org>

- 7 Luca Geatti, Nicola Gigante, and Angelo Montanari. A sat-based encoding of the one-pass and tree-shaped tableau system for LTL. In *Proceedings of the 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, volume 11714 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2019. doi:10.1007/978-3-030-29026-9\_1.
- 8 Luca Geatti, Nicola Gigante, and Angelo Montanari. BLACK: A fast, flexible and reliable LTL satisfiability checker. In *Proceedings of the 3rd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis*, volume 2987 of *CEUR*, pages 7–12, 2021.
- 9 Roman Kontchakov, Carsten Lutz, Frank Wolter, and Michael Zakharyashev. Temporalising tableaux. *Stud Logica*, 76(1):91–134, 2004. doi:10.1023/B:STUD.0000027468.28935.6d.
- 10 Marta Cialdea Mayer, Carla Limongelli, Andrea Orlandini, and Valentina Poggioni. Linear temporal logic as an executable semantics for planning languages. *Journal of Logic, Language and Information*, 16(1):63–89, 2007. doi:10.1007/s10849-006-9022-1.
- 11 A. Pnueli. The Temporal Logic of Programs. In *Proc. of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.32.



# Qualitative past Timeline-Based Games

Renato Acampora  

University of Udine, Italy

Luca Geatti   

University of Udine, Italy

Nicola Gigante   

Free University of Bozen-Bolzano, Italy

Angelo Montanari   

University of Udine, Italy

---

## Abstract

This extended abstract discusses timeline-based planning, a modeling approach that offers a unique way to model complex systems. Recently, the timeline-based planning framework has been extended to handle general nondeterminism in a game-theoretic setting, resulting in timeline-based games. In this context, the problem of establishing whether a timeline-based game admits a winning strategy and synthesizing such a strategy have been addressed. We propose exploring simpler yet expressive fragments of timeline-based games by leveraging results about the role of past operators in synthesis from temporal logic specifications. The qualitative fragment of timeline-based planning is a good starting point for this exploration. We suggest introducing syntactic restrictions on synchronization rules so that they only constrain the behavior of the system before the current time point, which is expected to lower the complexity of synthesizing timeline-based games to EXPTIME.

**2012 ACM Subject Classification** Computing methodologies → Planning for deterministic actions

**Keywords and phrases** Automata, Planning, Temporal Reasoning

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2023.22

**Category** Extended Abstract

## 1 State of the Art

Timeline-based planning, initially proposed for planning and scheduling of space operations [12], offers a unique approach to modelling complex systems. Unlike action-based planning paradigms, such as STRIPS [5], the timeline-based one does not explicitly separate states, actions, and goals; rather, it models the domain as a set of independent yet interacting components, whose behaviour over time is governed by “synchronization rules”. A solution plan is a set of timelines describing a behaviour of the system components that satisfies all rules. This approach has been successfully deployed in systems like the Hubble Space Telescope scheduling and control system (HSTS) [11]. The timeline-based planning framework has recently been extended to handle general nondeterminism in a game-theoretic setting (*timeline-based games*) [7]. The resulting formalism allows one to meet time constraints without being affected by the environment’s choices, reducing re-planning issues present in systems that only deal with temporal uncertainty [9]. Establishing the existence of a winning strategy for a timeline-based planning game has been proved to be 2EXPTIME-complete.

In [1], Acampora et al. addressed the problem of establishing whether a timeline-based game admits a winning strategy and, if this is the case, synthesizing such a strategy. In particular, they outlined an algorithm, based on a non-trivial construction of a Deterministic Finite Automaton (DFA), that recognizes solution plans for timeline-based planning problems. As it commonly happens, the synthesis of controllers reduces to the “compilation” of the specification into a DFA, which then serves as a game arena. This deterministic model is



© Renato Acampora, Luca Geatti, Nicola Gigante, and Angelo Montanari;  
licensed under Creative Commons License CC-BY 4.0

30th International Symposium on Temporal Representation and Reasoning (TIME 2023).

Editors: Alexander Artikis, Florian Bruse, and Luke Hunsberger; Article No. 22; pp. 22:1–22:3

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

required for playing games, and it is necessary to meet optimal computational complexity. The winning strategy for one of the players can then be determined through simple reachability games [8]. In [1], a compilation procedure that serves as the core for the DFA's construction is provided, and a controller implementing the specification is built. As a matter of fact, it was the first effective procedure for the synthesis of a controller starting from a timeline-based game specification.

## 2 Qualitative Past Timeline-Based Games

In view of the high complexity of timeline-based games, we aim at exploring simpler yet expressive fragments. One promising approach is to leverage results about the role of *past operators* in synthesis from temporal logic specification. Recently, it has been shown that writing the specification using past operators can make the synthesis problem exponentially more efficient [2, 6]. Drawing parallels with co-safety properties in Linear Temporal Logic (LTL), where properties express the fact that something good will eventually happen, we plan to apply these findings to timeline-based games. Specifically, we are thinking of introducing suitable syntactic restrictions on synchronization rules so that they only constrain the behavior of the system before the current time point. In such a way, we expect to lower the complexity of synthesizing timeline-based games to EXPTIME, offering an interesting direction for future research.

To this end, a good starting point is the *qualitative* fragment of timeline-based planning [10], a simpler planning formalism that only considers qualitative (ordering) features of timelines. To solve the synthesis problem for such a formalism, one can represent synchronization rules as partial orders and construct an automaton whose states are downward-closed subsets of such partial orders. Each single synchronization rule leads to a deterministic automaton taking care of matching the elements of the partial order when reading the word representing the solution plan. By generating the union of all the automata (one for each rule), we derive an automaton for the whole system. The size of such a deterministic automaton is exponential in the size of the planning problem. Since the automaton can be built on-the-fly and solving the reachability problem requires (nondeterministic) logarithmic space in the size of the automaton, we get that the problem of identifying solution plans for the suggested fragment belongs to PSPACE. Most importantly, the automaton is deterministic and can serve as the game arena for achieving synthesis in exponential time.

## 3 Temporal Logic Characterization and Symbolic Algorithms

We conclude the abstract with a discussion of two other future directions. In [4], is given the bounded variant of *Timed Propositional Temporal Logic with Past* (TPTL<sub>b</sub>+P) used to capture timeline-based problems. It seems natural to look for a *cosafety fragment* of TPTL<sub>b</sub>+P that captures past timeline-based problems, restricting the syntax of TPTL<sub>b</sub>+P to formulas of the form  $F(\alpha)$ , where  $F$  is the *eventually* operator and  $\alpha$  is a pure past TPTL<sub>b</sub>+P formula. Modern algorithms for the synthesis of temporal logic specifications deal with a *symbolic* representation of automata, representing the DFA equivalent to the initial formula by means of Boolean formulas only, in contrast to the explicit-state representation where states and transitions of the automaton are represented as memory locations and pointers. The benefits of using a symbolic representation are known from almost 30 years [3]. Producing a symbolic DFA starting from a past temporal logic specification has proven to be very effective [2, 6]. We expect to lift this result to the case of past timeline-based games.



## References

- 1 Renato Acampora, Luca Geatti, Nicola Gigante, Angelo Montanari, and Valentino Picotti. Controller synthesis for timeline-based games. In Pierre Ganty and Dario Della Monica, editors, *Proceedings of the 13th International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2022, Madrid, Spain, September 21-23, 2022*, volume 370 of *EPTCS*, pages 131–146, 2022. [Tdoi:10.4204/EPTCS.370.9](https://doi.org/10.4204/EPTCS.370.9).
- 2 Alessandro Artale, Luca Geatti, Nicola Gigante, Andrea Mazzullo, and Angelo Montanari. Complexity of safety and cosafety fragments of linear temporal logic. In *Proceedings of the AAAI Conference on Artificial Intelligence 37(5)*, pages 6236–6244, 2023. [Tdoi:10.1609/aaai.v37i5.25768](https://doi.org/10.1609/aaai.v37i5.25768).
- 3 Jerry R Burch, Edmund M Clarke, Kenneth L McMillan, David L Dill, and Lain-Jinn Hwang. Symbolic model checking: 1020 states and beyond. *Information and computation*, 98(2):142–170, 1992.
- 4 Dario Della Monica, Nicola Gigante, Angelo Montanari, Pietro Sala, and Guido Sciavicco. Bounded timed propositional temporal logic with past captures timeline-based planning with bounded constraints. In Carles Sierra, editor, *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1008–1014, 2017. [Tdoi:10.24963/ijcai.2017/140](https://doi.org/10.24963/ijcai.2017/140).
- 5 Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1971. [Tdoi:10.1016/0004-3702\(71\)90010-5](https://doi.org/10.1016/0004-3702(71)90010-5).
- 6 Giuseppe De Giacomo, Antonio Di Stasio, Francesco Fuggitti, and Sasha Rubin. Pure-past linear temporal and dynamic logic on finite traces. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4959–4965. [ijcai.org](https://ijcai.org), 2020. [Tdoi:10.24963/ijcai.2020/690](https://doi.org/10.24963/ijcai.2020/690).
- 7 Nicola Gigante, Angelo Montanari, Andrea Orlandini, Marta Cialdea Mayer, and Mark Reynolds. On timeline-based games and their complexity. *Theoretical Computer Science*, 815:247–269, May 2020. [Tdoi:10.1016/j.tcs.2020.02.011](https://doi.org/10.1016/j.tcs.2020.02.011).
- 8 Swen Jacobs, Roderick Bloem, Romain Brenguier, Rüdiger Ehlers, Timotheus Hell, Robert Könighofer, Guillermo A Pérez, Jean-François Raskin, Leonid Ryzhyk, Ocan Sankur, et al. The first reactive synthesis competition (syntcomp 2014). *International journal on software tools for technology transfer*, 19:367–390, 2017.
- 9 Marta Cialdea Mayer, Andrea Orlandini, and Alessandro Umbrico. Planning and execution with flexible timelines: a formal account. *Acta Informatica*, 53(6-8):649–680, 2016. [Tdoi:10.1007/s00236-015-0252-z](https://doi.org/10.1007/s00236-015-0252-z).
- 10 Dario Della Monica, Nicola Gigante, Salvatore La Torre, and Angelo Montanari. Complexity of qualitative timeline-based planning. In *27th International Symposium on Temporal Representation and Reasoning, TIME 2020, September 23-25, 2020, Bozen-Bolzano, Italy*, volume 178 of *LIPICs*, pages 16:1–16:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. [Tdoi:10.4230/LIPICs.TIME.2020.16](https://doi.org/10.4230/LIPICs.TIME.2020.16).
- 11 Nicola Muscettola. HSTS: Integrating Planning and Scheduling. In Monte Zweben and Mark S. Fox, editors, *Intelligent Scheduling*, chapter 6, pages 169–212. Morgan Kaufmann, 1994. URL: <https://apps.dtic.mil/sti/pdfs/ADA266991.pdf>.
- 12 Nicola Muscettola, Stephen F. Smith, Amedeo Cesta, and Daniela D’Aloisi. Coordinating space telescope operations in an integrated planning and scheduling architecture. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA, USA, 9-11 April 1991*, pages 1369–1376. IEEE Computer Society, 1991. [Tdoi:10.1109/ROBOT.1991.131803](https://doi.org/10.1109/ROBOT.1991.131803).

