26th International Symposium on Temporal Representation and Reasoning

TIME 2019, October 16–19, 2019, Málaga, Spain

Edited by

Johann Gamper
Sophie Pinchinat
Guido Sciavicco

Editors

Johann Gamper
Free University of Bozen-Bolzano, Italy
johann.gamper@unibz.it

Sophie Pinchinat
University of Rennes 1, France
sophie.pinchinat@irisa.fr

Guido Sciavicco
University of Ferrara, Italy
scvgdu@unife.it

ACM Classification 2012
Theory of computation → Logic; Information systems → Temporal data; Computing methodologies → Knowledge representation and reasoning

ISBN 978-3-95977-127-6

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/978-3-95977-127-6.

Publication date
October, 2019

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.

Digital Object Identifier: 10.4230/LIPIcs.TIME.2019.0

ISBN 978-3-95977-127-6 **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.

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

26th International Symposium on Temporal Representation and Reasoning (TIME 2019).
Editors: Johann Gamper, Sophie Pinchinat, and Guido Sciavicco
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

The 26th International Symposium on Temporal Representation and Reasoning (TIME 2019) took place in Málaga, Spain, from the 16th to the 19th of October, 2019. This year's edition was organized with the help of the University of Málaga, in particular the School of Industrial Engineering. TIME is a well-established symposium series which brings together researchers interested in reasoning about temporal aspects of information in all areas of computer science. The symposium aims to be interdisciplinary and to attract attendees from artificial intelligence, database management, logic and verification, and beyond.

TIME 2019 received 25 research paper submissions from France, Sweden, Greece, Italy, Spain, UK, Poland, Germany, Romania, Lebanon, and Algeria, written by 80 different authors. We would like to thank all authors for submitting outstanding contributions to the conference. The submissions were carefully evaluated by the 23 members of the program committee, who deserve a warm thanks for their high-quality and timely handling of the reviews. Each submission was reviewed by at least three PC members (two in a few cases). At the end, 17 contributions were accepted for inclusion in the conference proceedings and presentation at the conference.

The TIME 2019 scientific program includes also three keynote presentations given, respectively, by Michael Böhlen (University of Zurich), Estela Saquete Boró (University of Alicante), and Patricia Bouyer (CNRS). Thanks to the support of the University of Málaga, TIME 2019, for the first time, included two 3-hours tutorials on Computer Aided Synthesis and on Temporal Aspects of Inductive Logic Programming, given, respectively, by Véronique Bruyère (University of Mons) and Fabrizio Riguzzi (University of Ferrara). We are delighted that they all accepted our invitation, and are very grateful for their scientific contribution.

Finally, we would like to acknowledge the excellent work of all the people involved in the organization of the conference, in particular the local arrangement chair Emilio Muñoz-Velasco. His support was indispensable for a smooth organization and preparation of the conference. Deep thanks also go to Dr. Wagner and the LIPIcs team for the support during the preparation of the conference proceedings.

Johann Gamper
Sophie Pinchinat
Guido Sciavicco

26th International Symposium on Temporal Representation and Reasoning (TIME 2019).
Editors: Johann Gamper, Sophie Pinchinat, and Guido Sciavicco
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

List of Authors

Alex Abuin
Ikerlan Technology Research Center
Spain
aabuin@ikerlan.es

Ana de Almeida Borges
University of Barcelona
Spain
ana.agbv@gmail.com

Laura Bozelli
University of Napoli 'Federico II'
Italy
lr.bozzelli@gmail.com

Alexander Bolotov
University of Westminster
UK
a.bolotov@westminster.ac.uk

Sebastian Brandt
Siemens CT
Germany
sebastian-philipp.brandt@siemens.com

Julien Brunel
University of Touluse
France
julien.brunel@onera.fr

Andrea Brunello
University of Udine
Italy
andrea.brunello@uniud.it

Diego Calvanese
Free University of Bozen-Bolzano
Italy
calvanese@inf.unibz.it

Pasquale Cascarano
University of Bologna
Italy
pasquale.cascarano2@unibo.it

Serenella Cerrito
University of Paris-Saclay
France
serenella.cerrito@univ-evry.fr

David Chemouil
University of Touluse
France
david.chemouil@onera.fr

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

Juan José Conejero Rodriguez
University of Barcelona
Spain
juan.conejero@ub.edu

Pablo Cordero
University of Málaga
Spain
pcordero@uma.es

Amélie David
University of Paris-Saclay
France
amely.david@laposte.net

Unai Diáz de Cerio
Ikerlan Technology Research Center
Spain
udiazcerio@ikerlan.es

David Fernández Duque
University of Barcelona
Spain
davidstoteles@gmail.com

Inmaculada Fortes
University of Málaga
Spain
ifortes@ctima.uma.es

Stefano Gandolfi
University of Bologna
Italy
stefano.gandolfi@unibo.it

Mireia González Bedmar
University of Barcelona
Spain
m.gonzalezbedmar@ub.edu

26th International Symposium on Temporal Representation and Reasoning (TIME 2019).
Editors: Johann Gamper, Sophie Pinchinat, and Guido Sciavicco
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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

Elem Güzel Kalayci
Free University of Bozen-Bolzano
Italy
kalayci@inf.unibz.it

Inmaculada P. de Guzmán
University of Málaga
Spain
guzman@ctima.uma.es

Montserrat Hermo
University of the Basque Country
Spain
montserrat.hermo@ehu.es

Hsi-Ming Ho
University of Cambridge
UK
hsimho@gmail.com

Akash Hossain
University of Paris Diderot
France
akashoss42@gmail.com

Tomi Janhunen
University of Tampere
Finland
tomi.janhunen@aalto.fi

Timothy M. Jones
University of Cambridge
UK
tmj32@cl.cam.ac.uk

Joost J. Joosten
University of Barcelona
Spain
jjoosten@ub.edu

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

François Laroussinie
University of Paris Diderot
France
francoisl@irif.fr

Elena Loli Piccolomini
University of Bologna
Italy
elena.loli@unibo.it

Paqui Lucio
University of the Basque Country
Spain
paqui.lucio@ehu.es

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

Benjamin Mörzinger
Technische Universität Wien
Austria
moerzinger@ift.at

Barbara Oliboni
University of Verona
Italy
barbara.oliboni@univr.it

Anastasia Paparrizou
University of Artois
France paparrizou@cril.fr

Andrea Pascucci
University of Bologna
Italy andrea.pascucci@unibo.it

Adriano Peron
University of Napoli 'Federico II'
Italy
adrperon@unina.it

Quentin Peyras
University of Toulouse
France
quentin.peyras@onera.fr

Luca Poluzzi
University of Bologna
Italy
luca.poluzzi5@unibo.it

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

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

Vladislav Ryzhikov
University of London, Birkbeck
UK
vlad@dcs.bbk.ac.uk

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

Yakoub Salhi
University of Artois
France
salhi@cril.fr

Sixto Sánchez
University of Málaga
Spain
sixto@uma.es

Michael Sioutis
University of Aalto
Finland
michael.sioutis@aalto.fi

Heiner Stuckenshmidt
University of Mannheim
Germany
heiner@informatik.uni-mannheim.de

Luca Tavasci
University of Bologna
Italy
luca.tavasci2@unibo.it

Luca Viganó
King's College London
UK
luca.vigano@kcl.ac.uk

Przemysław Andrzej Wałega
University of Oxford
UK
p.a.walega@gmail.com

Melisachew Wudage Chekol
University of Mannheim
Germany
mel@informatik.uni-mannheim.de

Guohui Xiao
Free University of Bozen-Bolzano
Italy
xiao@inf.unibz.it

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

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

Michał Zawidzki
University of Łódź
Poland
michal.zawidzki@gmail.com

Kevin Zhou
University of Cambridge
UK
rkz20@cam.ac.uk

List of PC Members

Alessandro Artale

Beatrice Berard

Nathalie Bertrand

Dietmar Berwanger

Claudio Bettini

Davide Bresolin

Clare Dixon

Michael Fisher

Rajeev Gore

Giovanna Guerrini

Keijo Heljanko

Luke Hunsberger

Peter Jonsson

Bart Kuijpers

Orna Kupferman

Federica Mandreoli

Bastien Maubert

Emilio Muñoz-Velasco

Manuel Ojeda-Aciego

Mark Reynolds

François Schwarzentruber

Martin Theobald

Vassilis Tsotras

26th International Symposium on Temporal Representation and Reasoning (TIME 2019).
Editors: Johann Gamper, Sophie Pinchinat, and Guido Sciavicco
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Computing the Fourier Transformation over Temporal Data Streams

Michael H. Böhlen
University of Zürich, Switzerland
boehlen@ifi.uzh.ch

Muhammad Saad
University of Zürich, Switzerland
saad@ifi.uzh.ch

Abstract

In radio astronomy the sky is continuously scanned to collect frequency information about celestial objects. The inverse 2D Fourier transformation is used to generate images of the sky from the collected frequency information. We propose an algorithm that incrementally refines images by processing frequency information as it arrives in a temporal data stream. A direct implementation of the refinement with the discrete Fourier transformation requires $O(N^2)$ complex multiplications to process an element of the stream. We propose a new algorithm that avoids recomputations and only requires $O(N)$ complex multiplications.

2012 ACM Subject Classification Information systems → Stream management; Theory of computation → Data structures and algorithms for data management

Keywords and phrases Data streams, Fourier transform, time-varying data

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.1

Category Invited Talk

Funding *Muhammad Saad*: Partially supported by the Swiss National Science Foundation (SNSF) through project number 407550_167177.

1 Introduction

The discrete Fourier transform (DFT) converts a function of time into a function of frequency. The inverse of the DFT restores a time-varying function from its Fourier transform. In radio astronomy, high resolution images of celestial objects, such as stars and galaxies, are produced by directly measuring radio signals with an array of radio antennas. Each pair of antennas measures a visibility: a complex value that encodes amplitude and phase information of a radio signal. Each measured visibility value quantifies a single frequency component of the radio signal measured at a single time and a single (u, v) coordinate in the uv-plane. The main concept in radio astronomy is that the image of the sky has a Fourier transform that can be measured directly by a radio interferometer in the form of visibilities. A single visibility carries limited information about the spatial structure of the source. Hence, to produce a sky image with accurate spatial information, visibility values are measured at different coordinates in the uv-plane. The sky image is generated by taking the inverse 2D-DFT of the visibilities in the uv-plane.

Algorithms for computing the Fourier transform on streams can be divided into three classes: The first class computes DFT for sliding windows that overlap. If the sliding window advances for each point in the stream, then the DFT for that window can be computed efficiently using Sliding DFT algorithms [2], [3], [4], [6], [7]. The second class consists of algorithms when the window advances by multiple new points. The DFT of such hopping window can be computed as proposed in [5], [8]. The third class are variants of the FFT

algorithm [1], which is used when windows do not overlap. The methods require a quadratic number of complex multiplications. In this paper, we propose a new algorithm, the *Single Point Fourier Transform* (SPFT), for computing DFT of a stream where each new item in the stream updates a 2D-grid. Each time a 2D-grid is updated its DFT can be computed with a linear number of complex multiplications.

2 Background

A visibility value $V = c + di$ at coordinate (u, v) is measured for two antennas at a time point t_n. The antennas measure and generate a continuous and infinite stream S of visibilities: $S(ant_a, ant_b, t_i) = [V, (u, v)]$. As a running example we use the following stream of visibilities:

$$S(ant_1, ant_2, t_1) = [4 + 5i, (2, 2)], \quad S(ant_1, ant_3, t_2) = [-6 + 3i, (3, 1)],$$
$$S(ant_2, ant_3, t_3) = [4 + i, (2, 3)], \quad S(ant_1, ant_2, t_4) = [9 - 6i, (1, 1)],$$
$$S(ant_1, ant_2, t_5) = [4 + 3i, (2, 2)], \quad S(ant_1, ant_1, t_6) = [1 - 2i, (3, 0)], \ ...$$

$V(t_n)$ denotes the visibility grid that includes all visibilities from time t_0 to time t_n. $V_{u,v}$ denotes the cell of visibility grid V at coordinate (u, v). Multiple visibility values that fall into the same cell in grid V are added as illustrated in Table 1. For example, value $(8 + 8i)$ in cell $(2, 2)$ of $V(t_6)$ is the result of adding the visibilities of $S(ant_1, ant_2, t_1)$ and $S(ant_1, ant_2, t_5)$.

Table 1 Visibility grids generated by stream S at times t_2 and t_6.

$\mathbf{V(t_2)}$	0	1	2	3
0				
1				
2			$4 + 5i$	
3		$-6 + 3i$		

v (columns) / u (rows)

$\mathbf{V(t_6)}$	0	1	2	3
0				
1		$9 - 6i$		
2			$8 + 8i$	$4 + i$
3	$1 - 2i$	$-6 + 3i$		

v (columns) / u (rows)

$I(t_n)$ denotes the sky image that is computed as the inverse Fourier transform of $V(t_n)$. The cardinalities of the visibility grid and the image are identical. $I_{x,y}$ denotes the cell of the sky image I at coordinate (x, y), $x, y = 0, 1, 2, ..., N - 1$. The discrete Fourier transform (DFT) can be used to compute the value of $I_{x,y}$ at time t_n:

$$I_{x,y}(t_n) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} V_{u,v}(t_n) \cdot W^{(ux+vy)}, \qquad W = e^{\frac{i2\pi}{N}} \tag{1}$$

The exponents W^k, $k = 0, 1, 2, ..., N - 1$ are the primitive N^{th} roots of unity according to Euler's formula: $W^k = (e^{-\frac{i2\pi}{N}})^k = cos(\frac{2\pi}{N}k) + isin(\frac{2\pi}{N}k)$.

When a new visibility value $S(ant_i, ant_j, t_{n+1}) = [a + bi, (u, v)]$ arrives it is possible to compute $I(t_{n+1})$ incrementally from $I(t_n)$:

$$I_{x,y}(t_{n+1}) = I_{x,y}(t_n) + S(t_{n+1}) \cdot W^{ax+by} \tag{2}$$

$I(t_{n+1})$ can trivially be computed with N^2 complex multiplication operations as illustrated in Table 2.

Table 2 Incrementally computing $I(t_n)$ from $I(t_{n-1})$.

$\mathbf{I(t_n)}$	0	1	...	$N-1$
0	$I_{0,0}(t_{n-1}) + S(t_n) \cdot W^{0a+0b}$	$I_{0,1}(t_{n-1}) + S(t_n) \cdot W^{0a+1b}$	...	$I_{0,N-1}(t_{n-1}) + S(t_n) \cdot W^{0a+(N-1)b}$
1	$I_{1,0}(t_{n-1}) + S(t_n) \cdot W^{1a+0b}$	$I_{1,1}(t_{n-1}) + S(t_n) \cdot W^{1a+1b}$	...	$I_{1,N-1}(t_{n-1}) + S(t_n) \cdot W^{1a+(N-1)b}$
	...	...	...	...
$N-1$	$I_{N-1,0}(t_{n-1}) + S(t_n) \cdot W^{(N-1)a+0b}$	$I_{N-1,1}(t_{n-1}) + S(t_n) \cdot W^{(N-1)a+1b}$	...	$I_{N-1,N-1}(t_{n-1}) + S(t_n) \cdot W^{(N-1)a+(N-1)b}$

y (→) x (↓)

3 Single Point Fourier Transform

We propose the *Single Point Fourier Transform* (SPFT), a method that reduces the $O(N^2)$ complex multiplications to $O(N)$ complex multiplications. Let $S(ant_i, ant_j, t_{n+1}) = [c + di, (u, v)]$. The element-wise subtraction of visibility grid $V(t_n)$ from $V(t_{n+1})$ yields a $N \times N$ grid $Z(t_{n+1})$ where all elements are zero except value $c + di$ at coordinate (a, b):

$$Z(t_{n+1}) = \begin{cases} S(t_{n+1}) & if a = u, b = v \\ 0 & otherwise \end{cases}$$

The 2D-DFT $I(Z(t_{n+1}))$ can be computed as:

$$I(Z(t_{n+1})) = S(t_{n+1}) \cdot T(a, b) \tag{3}$$

where $T(a, b)$ is a 2D matrix consisting of twiddle factors as follows:

$$T_{x,y}(a, b) = W^{(ax+by)\%N} \tag{4}$$

We denote each row and column of twiddle factor matrix $T(a, b)$ by R_j and C_j, respectively. The k^{th} elements of these vectors are denoted by $R_j[k]$ and $C_j[k]$, respectively. The exponents of twiddle factors of the rows are the modulo of multiples of coordinate b with a constant z added, i.e., $(z + (i \times b))\%N$, where z is a multiple of a. For each row the multiple can be different. For columns $C_0, C_1, ..., C_{N-1}$ the roles of a and b are switched.

$$R_j[k] = W^{(j \times a + k \times b)\%N}, \qquad C_j[k] = W^{(j \times b + k \times a)\%N} \qquad \forall j, k = 0, 1, 2, ..., N-1$$

The key result is that either each row is a rotation of the first row or each column is a rotation of the first column.

$$R_j = CSHIFT(R_0, (j * nshift)\%N), \qquad C_j = CSHIFT(C_0, (j * nshift)\%N)$$

For coordinate (a, b) and a $2^m \times 2^m$ matrix, there can be a column or row shift or both. There is not a case when neither a row nor column shift is applicable.

4 Conclusion and Outlook

We proposed an algorithm to efficiently compute the 2D-DFT of a temporal stream. The SPFT algorithm computes the 2D-DFT iteratively for each update by reusing computations to minimize the number of complex multiplications. Rather than recomputing the Fourier

Table 3 Number of complex multiplications required for a single point update.

Algorithm	# of complex multiplications	GridSize (N × N)				
		32 × 32	64 × 64	128 × 128	256 × 256	512 × 512
DFT	$4N^2$	4,096	16,384	65,536	262,144	1,048,576
FFT	$4N^2 \log N$	20,480	98,304	458,752	2,097,152	9,437,184
SPFT	$4N$	128	256	512	1,024	2,048

transform for all points of the temporal stream, our approach only computes DFT for the current point in a stream and adds it to the DFT of the previous values. Our approach requires 20-40 times less operations than FFT for different grid sizes.

In the future it would be interesting to investigate techniques to improve the cost for matrix additions, e.g., by distributing the computation of the additions. Further, the batching of observations, which benefits FFT, should be investigated for applications where this is feasible.

References

1 W. Cochran, J. Cooley, D. Favin, H. Helms, R. Kaenel, W. Lang, G. Maling, D. Nelson, C. Rader, and P. Welch. What is the fast Fourier transform? *IEEE Transactions on Audio and Electroacoustics*, 15(2):45–55, June 1967. doi:10.1109/TAU.1967.1161899.

2 K. Duda. Accurate, Guaranteed Stable, Sliding Discrete Fourier Transform [DSP Tips & Tricks]. *IEEE Signal Processing Magazine*, 27(6):124–127, November 2010. doi:10.1109/MSP.2010.938088.

3 E. Jacobsen and R. Lyons. The sliding DFT. *IEEE Signal Processing Magazine*, 20(2):74–80, March 2003. doi:10.1109/MSP.2003.1184347.

4 C. Park. Fast, Accurate, and Guaranteed Stable Sliding Discrete Fourier Transform [sp Tips & Tricks]. *IEEE Signal Processing Magazine*, 32(4):145–156, July 2015. doi:10.1109/MSP.2015.2412144.

5 C. Park and S. Ko. The Hopping Discrete Fourier Transform [sp Tips & Tricks]. *IEEE Signal Processing Magazine*, 31(2):135–139, March 2014. doi:10.1109/MSP.2013.2292891.

6 B. G. Sherlock and D. M. Monro. Moving discrete Fourier transform. *IEE Proceedings F - Radar and Signal Processing*, 139(4):279–282, August 1992. doi:10.1049/ip-f-2.1992.0038.

7 B.G Sherlock. Windowed discrete Fourier transform for shifting data. *Signal Processing*, 74(2):169–177, 1999. doi:10.1016/S0165-1684(98)00209-6.

8 A. Srivastava and V. Karwal. Windowed r-point update algorithm for Discrete Fourier Transform. In *2013 International Conference on Signal Processing and Communication (ICSC)*, pages 185–190, December 2013. doi:10.1109/ICSPCom.2013.6719780.

From Unstructured Data to Narrative Abstractive Summaries

Estela Saquete Boró
Department of Software and Computing Systems, University of Alicante,
Apdo. de Correos 99 E-03080, Alicante, Spain
https://www.dlsi.ua.es/eines/membre.cgi?id=cas&nom=stela
stela@dlsi.ua.es

Abstract

To provide with easy and optimal access to digital information, narrative summaries must have a coherent and natural structure. Depending on how a summary is produced, a distinction can be made between extractive and abstractive summaries. Using an abstractive summarization approach, the relevant information (e.g., who? what?, when?, where?,...) could be fused together, leading to the generation of one or more new sentences. However, in order to do this it is necessary to obtain and process the temporal information in a text. A very effective way is the generation of timelines starting from multiple documents so that the generation of summaries is supported by the generated timeline, without losing the relevant temporal information of the texts. In this proposal, a enriched timeline is generated automatically, and the process of generating abstractive summaries is presented using this timeline as a basis [1]. Finally, potential applications of the automatic timeline generation would be presented, as for example its application to Fake News detection.

2012 ACM Subject Classification Applied computing → Document management and text processing

Keywords and phrases Narrative summarization, Abstractive summarization, Timeline Generation, Temporal Information Processing, Natural Language Generation

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.2

Category Invited Talk

Funding This research work has been partially funded by the projects PROMETEU/2018/089 and RTI2018-094653-B-C22.

1 Introduction

As human beings, we tend to organize the flux of happening in structured units known as events. Each event is a fact that occurs in the (real or imaginary) world with a specific structure (the event structure), and denotes processes, activities, states, achievements or accomplishments [7]. An event involves participants [3] and other components that complete the event such as time, place, instruments, patients, etc. In ISO TimeML Working Group[2], the event was defined as "something that can be said to obtain or hold true, to happen or to occur". Moreover, relating and ordering the information extracted from *different documents* is an essential task to obtain this knowledge. This cross-document processing improves the traditional single-document extraction and uses information redundancy to its advantage.

Hence, event ordering is a crucial task within Natural Language Processing. Cross-Document Event Ordering implies the accomplishment of three sub-tasks [10]. First of all, the *extraction of events* and related entities from texts, because it is necessary to know which events appear in each document, and which entities are related to each one of them. Then *temporal information processing* is required in order to extract the temporal expressions and the temporal relationships established between these events, determining thus which events happen at the same time. Finally, *cross-document event coreference* is needed in order to cluster all the mentions that refer to the same event, regardless of the words used to express

them. The final aim of combining event extraction and temporal information processing with cross-document event coreference enables us to automatically build ordered timelines of events from written texts.

To provide users with relevant information, summaries must provide a coherent and natural structure [5]. Text summarization allows to condense the relevant information of different documents (e.g. news) [6].

The main objective of this work is to demonstrate how the use of automatic timelines can benefit multiple NLP applications, including the generation of narrative abstractive summaries based on a natural time ordering of events from a set of documents (news in this case) that deal with the same real events, as it was described in depth in [1]. This approach has two main components: (i) a cross-document timeline generation module that extracts events related to the same entity from several texts (cross-document) and the time slot in which each event occurs, arranging them in a timeline; and (ii) an abstractive summarization module that transforms these time-ordered events into a single text with a time-based chronological narrative structure.

2 Cross-document Enriched TimeLine Extraction

As previously explained, given a set of documents and a set of target entities, the original task of Cross-Document Timeline Extraction consists of building an event timeline for a target entity from a set of documents [8].

As presented in [11], theoretically, the main idea of our approach is that two events *e1* and *e2* will be coreferent if they are not only temporal compatible ($e1_t = e2_t$) but also if they refer to the same facts (semantic compatibility: $e1_s \simeq e2_s$):

$$coref(e1, e2) \rightarrow (e1_t = e2_t) \wedge (e1_s \simeq e2_s)$$

Our proposal extends the approach by enriching the event clusters with all the arguments extracted from these events in the different documents where they are presented (see [11] for further details). The steps of this module are:

- **Temporal clustering.** The input is a set of plain texts, and, therefore, the events in those texts must be automatically extracted. Furthermore, considering that the final aim is building a timeline, temporal expressions and temporal links between events and times are required. For this reason, the first step is performing Temporal Information Extraction and Processing, and TIPSem system [4] is used for this purpose. Considering the premise that two events mentions referring to the same event happen at the same time, and using the temporal annotation of the input texts (TimeML), the temporal clustering algorithm performs two steps:

  - *Within-document temporal clustering*: For each document, the temporal information of each event is extracted. Each event is anchored to a time anchor[1] when a temporal SIMULTANEOUS/ BEGIN/ INCLUDES link exists between this event and a temporal expression. After this, two events will be considered part of the same cluster if they are temporally compatible. This means that: a) two events are anchored to the same time anchor, or b) two events have a temporal SIMULTANEOUS link between them.

[1] A time anchor is always a DATE (as defined in TimeML) and its format follows the ISO-8601 standard.

- *Cross-document temporal clustering*: Considering that in the previous step all the events of each document were assigned to a time anchor, in this step, this information is merged in a single timeline, in which all the events of the different documents are clustered together if they are happening at the same time.

Finally, the temporal clusters are chronologically ordered.

- **Semantic clustering.** The events are clustered using event type information and distributional semantic knowledge. Two or more event mentions in the same time slot could refer to the same real event. To detect these corefential events, we have applied a clustering process based on two kinds of semantic information: the first one is the event type and the second one is the distributional semantic similarity between event mentions. During the event extraction process, each event mention has been classified according to its type of event following TimeML standard: occurrence, perception, reporting, aspectual, state, intentional state and intentional action. All the event mentions with the same time slot have been regrouped after also considering the type of event assigned. Next, our approach clusters coreferential events (identifies all the events that share the same time slot and the same type of event) according to the compositional-distributional semantic similarity between them. The semantics of the event structure is represented as a compositional-distributional vector. These vectors are called contextual vectors.
  In our approach each event structure is formed, on the one hand by the event head and, on the other hand by the nouns, verbs and adjectives of the main arguments. All this information is extracted by applying Freeling as Part of Speech tagger and Semantic Role Labeling system. Following the additive model [9], these word vectors are added in a single compositional vector that represents the distributional meaning of the whole event structure.
- **Event cluster enrichment.** In the original concept of timeline, only events are grouped together in the clusters and not the arguments involved in those events that constitute a fundamental part of the information. Therefore, in this step, all the arguments (semantic roles extracted in the previous step with Freeling) of the events in each cluster are added to the timeline, enriching the information provided for each event.

3 Abstractive Summarization

The aim of this module is to produce a narrative abstractive summary with chronological information given an enriched timeline as input. As presented in depth in [1], this summary is generated employing NLG techniques. In particular, we employ a hybrid surface realization approach, based on over-generation and ranking techniques. For each of the enriched cluster of events from the enriched timeline, the next steps are applied:

- Argument selection: in case there is more than one argument for the same semantic role, a statistical selection of the arguments from the timeline is performed.
- Obtaining verb frames: information about the frames corresponding to the verbs of each event is obtained to generate a sentence without the need to resort to grammar specifications.
- Sentence Generation: for each of the frames obtained a sentence is generated, based on the frame structure.
- Sentence Ranking: a ranking is performed for selecting only one sentence representing a specific event (cluster of event mentions) in the timeline.

4 Conclusions and Further Work

In this paper an integrated approach is presented on two basic aspects. First, it is based on the fact that humans tend to apply chronological ordering of events in the summarizing process, which implies the need for timelines [11]. Second, when using an abstractive summarization approach, rather than an extractive one, the relevant information could be fused together, leading to the generation of more complete sentences, and thus, more comprehensible and effective summaries [1]. The proposal comprises two main modules: i) Enriched Timeline Extraction module, and ii) Abstractive Summarization module.

Enriched timeline generation has multiple applications in Natural Language Processing approach. Apart from summarization, the timeline with arguments can be used to detect contradictions in different media outlets, which is one the foundamentals when detecting mis- or disinformation. Furthermore, it can be used in fact-checking purposes or misleading headlines.

References

1 Cristina Barros, Elena Lloret, Estela Saquete, and Borja Navarro-Colorado. NATSUM: Narrative abstractive summarization through cross-document timeline generation. *Information Processing & Management*, February 2019. doi:10.1016/j.ipm.2019.02.010.

2 ISO TimeML Working Group. ISO TimeML TC37 draft international standard DIS 24617-1, 2008. URL: http://semantic-annotation.uvt.nl/ISO-TimeML-08-13-2008-vankiyong.pdf.

3 Heng Ji, Ralph Grishman, Zheng Chen, and Prashant Gupta. Cross-document Event Extraction and Tracking: Task, Evaluation, Techniques and Challenges. In *RANLP*, pages 166–172. RANLP 2009 Organising Committee / ACL, 2009.

4 Hector Llorens, Estela Saquete, and Borja Navarro-Colorado. Applying Semantic Knowledge to the Automatic Processing of Temporal Expressions and Events in Natural Language. *Information Processing & Management*, 49(1):179–197, 2013.

5 Elena Lloret and Manuel Palomar. Text Summarisation in Progress: A Literature Review. *Artif. Intell. Rev.*, 37(1):1–41, January 2012. doi:10.1007/s10462-011-9216-z.

6 Inderjeet Mani and Mark T. Maybury, editors. *Advances in Automatic Text Summarization*. MIT Press, Cambridge, MA, USA, 1999.

7 Inderjeet Mani, James Pustejovsky, and Robert Gaizauskas. *The Language of Time*. Oxford University Press, Oxford, 2005.

8 Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begoña Altuna, Marieke van Erp, Anneleen Schoen, and Chantal van Son. MEANTIME, the NewsReader Multilingual Event and Time Corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, May 2016.

9 Jeff Mitchell and Mirella Lapata. Composition in Distributional Models of Semantics. *Cognitive Science*, 34:1388–1429, 2010.

10 Borja Navarro and Estela Saquete. GPLSIUA: Combining Temporal Information and Topic Modeling for Cross-Document Event Ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 820–824, Denver, Colorado, June 2015. Association for Computational Linguistics.

11 Borja Navarro-Colorado and Estela Saquete. Cross-document Event Ordering Through Temporal, Lexical and Distributional Knowledge. *Know.-Based Syst.*, 110(C):244–254, October 2016. doi:10.1016/j.knosys.2016.07.032.

On the Computation of Nash Equilibria in Games on Graphs

Patricia Bouyer
LSV, CNRS, ENS Paris-Saclay, Université Paris-Saclay, France
http://www.lsv.fr/~bouyer/
bouyer@lsv.fr

Abstract

In this talk, I will show how one can characterize and compute Nash equilibria in multiplayer games played on graphs. I will present in particular a construction, called the *suspect game construction*, which allows to reduce the computation of Nash equilibria to the computation of winning strategies in a two-player zero-sum game.

2012 ACM Subject Classification Theory of computation; Theory of computation → Solution concepts in game theory; Theory of computation → Verification by model checking

Keywords and phrases Multiplayer games, Nash equilibria

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.3

Category Invited Talk

Funding Work supported by ERC project EQualIS between 2013 and 2019.

Acknowledgements I would like to thank all my co-authors since I started working on multiplayer games played on graphs, that is, Nicolas Markey, Romain Brenguier [3], Daniel Stan [9], Michael Ummels and Nathan Thomasset.

1 Introduction

Multiplayer concurrent games over graphs allow to model rich interactions between players. Those games are played as follows. In a state, each player chooses privately and independently an action, defining globally a move (one action per player); the next state of the game is then defined as the successor (on the graph) of the current state using that move; players continue playing from that new state, and form a(n infinite) play. Each player then gets a reward given by a payoff function (one function per player). In particular, objectives of the players may not be contradictory: those games are non-zero-sum games, contrary to two-player games used for controller or reactive synthesis [10, 7].

Using solution concepts borrowed from game theory, one can describe the interactions between the players, and in particular describe their rational behaviours. One of the most basic solution concepts is that of Nash equilibria [8]. A Nash equilibrium is a strategy profile where no player can improve her payoff by unilaterally changing her strategy. The outcome of a Nash equilibrium can therefore be seen as a rational behaviour of the system. While very much studied by game theoretists (e.g. over matrix games), such a concept (and variants thereof) has been only rather recently studied over games on graphs. Probably the first works in that direction are [5, 4, 11, 12].

Computing Nash equilibria requires to (i) find a good behaviour of the system; (ii) detect deviations from that behaviour, and identify deviating players (called deviators); (iii) punish them. Variants of Nash equilibria (like subgame-perfect equilibria, robust equilibria, *etc*) require slightly different ingredients, but they are mostly of a similar vein.

In this talk, we will first recall some basics of game theory over matrix games. Those games are not sufficient in a verification context: indeed, explicit states are very useful when modelling systems or programs, but are missing in matrix games. However stability notions like Nash equilibria or other solution concepts borrowed from game theory, are very relevant. We will thus present the model of concurrent multiplayer games (played on graphs), which extends in a natural way standard models used in verification with multiplayer interactions. We will explain how Nash equilibria can be characterized and computed in such general games. We will also discuss some existence results for Nash equilibria.

A reference note for this talk is [1]. Related notes are [10], which discussed the use of two-player zero-sum games in verification, and [6], which discussed solution concepts in multiplayer turn-based games on graphs.

To give a taste of the approach, we informally discuss below a simple scenario. The general case with concurrent games and more general payoff functions will be handled by the suspect game abstraction, which somehow generalizes the simple scenario below. For readers not familiar at all with games, you can skip the discussion and wait until the talk.

2 Discussion on a Simple Scenario

We fix a turn-based and deterministic game $\mathcal{G}$ with set of players $\mathcal{P}$, and we assume that the payoff function for each player $A \in \mathcal{P}$ is given by a Boolean prefix-independent objective ϕ_A (that is, the player gets $+1$ is the play satisfies ϕ_A, and 0 otherwise). Each player A plays using a (deterministic) *strategy* σ_A, and once a strategy is fixed for every player, we have a *strategy profile* $\sigma = (\sigma_A)_{A\in\mathcal{P}}$. In such a game, the player objective is to make the generated play satisfy her formula. Hence, if the unique outcome of σ satisfies ϕ_A, then player A is satisfied. Otherwise player A will try to be more satisfied by changing her strategy; the new strategy is then called a (single-player) *deviation*. The strategy profile σ will then be a (pure) *Nash equilibrium* if it is resistant to single-player deviations.

In our simple setting, it will be rather easy to characterize deviations that are *profitable* to player A: once a deviation by player A has occurred, we assume that all other players (which we note as a coalition $\langle\!\langle -A \rangle\!\rangle$) play optimally in σ for the objective $\neg\phi_A$.[1] That way, if the outcome of the strategy does only visit winning states for $\langle\!\langle -A \rangle\!\rangle$ (which we denote $W_{\langle\!\langle A \rangle\!\rangle}$), then no deviation for player A can be profitable; conversely if the outcome visits a winning state for A (by determinacy, this is the negation of the previous case), then there will be a profitable deviation for player A. One can then characterize Nash equilibria as follows:

▶ **Proposition 1.** *Let ρ be an infinite path in $\mathcal{G}$ from the initial vertex v_{init}. Then, $\rho \models \Phi_{NE}$ if and only if there is a Nash equilibrium σ from v_{init} such that the outcome of σ is ρ, where*

$$\Phi_{NE} = \bigwedge_{A\in\mathcal{P}} \left(\neg\phi_A \Rightarrow \mathbf{G} W_{\langle\!\langle -A \rangle\!\rangle}\right)$$

The situation is illustrated on Figure 1. Note that by determinacy, "Player A_1 should lose" can be replaced by "Coalition $\{A_2, A_3\}$ prevents A_1 from winning".

A solution to compute Nash equilibria is then to compute for every $A \in \mathcal{P}$ the set $W_{\langle\!\langle -A \rangle\!\rangle}$ (or equivalently W_A), and to compute an infinite path in the game which satisfies formula Φ_{NE} (which can be done for instance by enumerating the possible set of losing players, and then finding an adequate ultimately periodic play). Obviously, for specific winning conditions, more efficient algorithms can be designed, see for instance [13, 2].

[1] Such a strategy is sometimes called a *threat* or a *trigger* strategy.

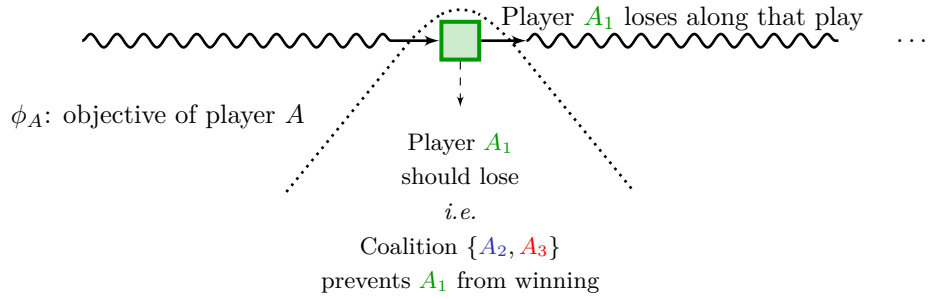


Figure 1 General shape of a Nash equilibrium in the simple setting (example with three players A_1, A_2 and A_3).

References

1. Patricia Bouyer. A Note on Game Theory and Verification. In *Proc. 17th International Symposium on Automated Technology for Verification and Analysis (ATVA'19)*, Lecture Notes in Computer Science. Springer, 2019. To appear.
2. Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Pure Nash Equilibria in Concurrent Games. *Logical Methods in Computer Science*, 11(2:9), 2015.
3. Romain Brenguier. *Nash Equilibria in Concurrent Games – Application to Timed Games*. PhD thesis, ENS Cachan, France, 2012.
4. Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdziński. Games with Secure Equilibria. *Theoretical Computer Science*, 365(1-2):67–82, 2006.
5. Krishnendu Chatterjee, Rupak Majumdar, and Marcin Jurdziński. On Nash Equilibria in Stochastic Games. In *Proc. 18th International Workshop on Computer Science Logic (CSL'04)*, volume 3210 of *Lecture Notes in Computer Science*, pages 26–40. Springer, 2004.
6. Erich Grädel and Michael Ummels. Solution Concepts and Algorithms for Infinite Multiplayer Games. In *New Perspectives on Games and Interaction*, volume 4 of *Texts in Logic and Games*, pages 151–178. Amsterdam University Press, 2008.
7. Thomas A. Henzinger. Games in system design and verification. In *Proc. 10th Conference on Theoretical Aspects of Rationality and Knowledge (TARK'05)*, pages 1–4, 2005.
8. John F. Nash. Equilibrium Points in n-Person Games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49, 1950.
9. Daniel Stan. *Randomized Strategies in Concurrent Games*. PhD thesis, Université Paris-Saclay, France, 2017.
10. Wolfgang Thomas. Infinite Games and Verification. In *Proc. 14th International Conference on Computer Aided Verification (CAV'02)*, volume 2404 of *Lecture Notes in Computer Science*, pages 58–64. Springer, 2002. Invited Tutorial.
11. Michael Ummels. Rational Behaviour and Strategy Construction in Infinite Multiplayer Games. In *Proc. 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 2006.
12. Michael Ummels. The Complexity of Nash Equilibria in Infinite Multiplayer Games. In *Proc. 11th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'08)*, volume 4962 of *Lecture Notes in Computer Science*, pages 20–34. Springer, 2008.
13. Michael Ummels. *Stochastic Multiplayer Games – Theory and Algorithms*. PhD thesis, RWTH Aachen, Germany, 2010.

A Modal Logic for Subject-Oriented Spatial Reasoning

Przemysław Andrzej Wałęga[1]
University of Oxford, Department of Computer Science, United Kingdom
University of Warsaw, Institute of Philosophy, Poland

Michał Zawidzki
University of Łódź, Department of Logic, Poland

Abstract

We present a modal logic for representing and reasoning about space seen from the subject's perspective. The language of our logic comprises modal operators for the relations "in front", "behind", "to the left", and "to the right" of the subject, which introduce the intrinsic frame of reference; and operators for "behind an object", "between the subject and an object", "to the left of an object", and "to the right of an object", employing the relative frame of reference. The language allows us to express nominals, hybrid operators, and a restricted form of distance operators which, as we demonstrate by example, makes the logic interesting for potential applications. We prove that the satisfiability problem in the logic is decidable and in particular PSpace-complete.

2012 ACM Subject Classification Theory of computation → Complexity theory and logic; Theory of computation → Modal and temporal logics; Theory of computation → Automata over infinite objects; Theory of computation → Verification by model checking

Keywords and phrases spatial logic, modal logic, subject-oriented, computational complexity

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.4

Funding *Przemysław Andrzej Wałęga*: the SIRIUS Centre for Scalable Data Access, the EPSRC projects DBOnto, AnaLOG, and ED[3]; and the Foundation for Polish Science (FNP).
Michał Zawidzki: the NCN grant DEC-2011/02/A/HS1/00395.

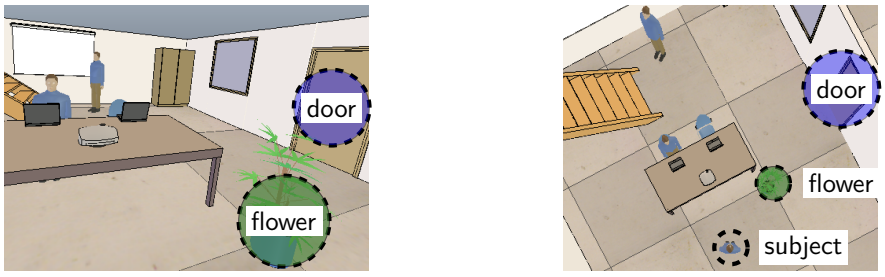
1 Introduction

Spatial reasoning is one of the most interesting abilities that humans possess and whose modeling is still a great challenge for AI. Research in this area is interesting from a theoretical point of view, for it may help us understand how people reason and use language, and from a practical perspective, as it may result in methods with a broad range of applications, e.g., in robotics [10] or geographical information systems [9].

There is a great variety of logical approaches to spatial reasoning which exploit the machinery of modal logics [32, 19, 21, 17, 5, 20, 22], relational algebras [24, 16, 28, 23], and first-order theories [8, 28, 27], among others. Various aspects of space are modeled by these systems, such as topology, directions, distance, orientation, size, shape, etc.; some use quantitative and other qualitative methods which, by the way, are often inspired by human-like reasoning. What is common for many of these approaches (especially for the ones based on modal logics) is that a *frame of reference* imposed on objects in order to describe and reason about their location is *absolute*. However, various studies show that users of Indo-European languages (e.g., English and Dutch) or Japanese mostly adopt the *intrinsic* frame of reference (also called *object-centered*, e.g., "something is to the left of me") or the *relative* one (called *egocentric*, e.g., "something is to the left of a given object with respect to my location"), when performing both linguistic and non-linguistic tasks [14, 18, 15].

[1] Corresponding author

To illustrate how space can be perceived from different points of view consider the spatial arrangement depicted in Figure 1. From the subject's point of view "the flower is in front of me" (intrinsic frame of reference) and "the flower is between the door and myself" (relative frame of reference). On the other hand, from the aerial perspective "the flower is to the south of the door" (absolute frame of reference).



(a) (b)

Figure 1 A spatial configuration viewed from the subject's (a) and aerial (b) perspectives.

A logical system modeling such a subject-oriented way of representing and reasoning about space could have a number of practical applications, e.g., in designing architectural objects which are required to be perceived by humans in a specific way, assisting blind persons, or giving directions to someone who is lost. In this paper we address this topic by constructing a modal logic for subject-oriented spatial reasoning (denoted by SOSL) which allows us to express spatial relations in both intrinsic and relative frames of reference, and which, to the best of our knowledge, is the first modal logic of this type.

To represent the two-dimensional space in a subject-oriented way, we use the polar coordinate system with a subject located at its center and with a spatial universe consisting of cells denoted by the polar coordinates of their central points, i.e., by a pair of a radius-coordinate, and a linear-coordinate (see Figure 2). To capture the intrinsic frame of reference we will introduce modal operators for the relations "in front of the subject" ($\blacklozenge$), "behind the subject" ($\blacklozenge$), "to the left of the subject" ($\blacklozenge$), and "to the right of the subject" ($\blacklozenge$), and to capture the relative frame of reference we will use operators expressing the relations "behind an object" ($\diamondsuit$), "between an object and the subject" ($\diamondsuit$), "to the left of an object" ($\diamondsuit$), and "to the right of an object" ($\diamondsuit$). Moreover, we will employ operators expressing the single-step relations between adjacent cells, namely "away from the subject" ($\uparrow$), "towards the subject" ($\downarrow$), "counter-clockwise" ($\circlearrowleft$), and "clockwise" ($\circlearrowright$). As we will show in the paper, such a language is expressive enough to define the universal modality (A), where $\mathsf{A}\varphi$ states that φ holds in all cells, nominals, i.e., atoms which hold in exactly one cell, satisfaction operators of the form $@_i$, where i is a nominal, stating that φ is satisfied in the cell in which i holds, and operators expressing distance, e.g., $\Diamond_{=d}\varphi$ for "φ holds in a cell with a not-smaller radius-coordinate which is in distance d" or $\Diamond_{<d}\varphi$ for "φ holds in a cell with a not-smaller radius-coordinate which is closer than d" etc. Then, in the intrinsic frame of reference the relation between the flower and the door from Figure 1 is described by the SOSL-formula $\blacklozenge\mathsf{flower}$, and in the relative frame of reference by $@_{\mathsf{door}}\diamondsuit\mathsf{flower}$.

We will prove that the satisfiability problem of SOSL-formulas is decidable (in contrast to many two-dimensional modal logics [32, 21, 12]) and in particular PSPACE-complete (the same as for one-dimensional Linear Temporal Logic [11, 3]). To show the lower bound we reduce the satisfiability problem of Linear Temporal Logic, and for the upper bound we present a relatively complex (due to the fact that SOSL is a two-dimensional logic with 12 modal

operators) construction of a generalized nondeterministic Büchi automaton which accepts precisely those words which describe models of a given SOSL-formula. As a result, we obtain a two-dimensional and subject-oriented logic which is decidable and enables us to express and reason about complex spatial configurations in the intrinsic and relative frames of reference. In what follows, we present examples demonstrating the expressive power of the language and its potential applications.

Modeling human cognition. Modeling human-centered cognition is required in various applications, e.g., in Computer-Aided Architecture Design (CAAD) systems to address problems such as indoor spatial awareness, visibility analysis, or wayfinding [6, 7]. The subject-oriented representation of space makes SOSL an adequate tool for the above-mentioned applications. For the sake of example consider the following SOSL-formulas specifying how household goods should be located with respect to a person sitting on a sofa, where sofa, tv, and window are propositional variables and $d \in \mathbb{R}$:

$$\mathsf{sofa} \wedge \blacklozenge(\mathsf{tv} \wedge \Diamond\!\!\!\!\rightarrow \mathsf{window}) \wedge \Diamond_{\leq d}\mathsf{lamp};$$

$$\mathsf{A}(\mathsf{tv} \vee \mathsf{window} \rightarrow \neg\Diamond\!\!\!\!\downarrow \mathsf{lamp}) \wedge \mathsf{A}(\mathsf{tv} \rightarrow \neg\Diamond\!\!\!\!\uparrow \mathsf{window}).$$

The first formula states that the configuration is described from the perspective of a person sitting on a sofa. There should be a tv standing in front of ($\blacklozenge$) the sofa and a window located to the right of ($\Diamond\!\!\!\!\rightarrow$) this tv with respect to the sofa. Furthermore, there should be a lamp located in a distance at most d ($\Diamond_{\leq d}$) from the sofa. The second formula describes the following universal constraints: no lamp can stand between ($\Diamond\!\!\!\!\downarrow$) any tv and the sofa or between any window and the sofa (not to block the view); and no window can be located behind ($\Diamond\!\!\!\!\uparrow$) any tv (for a better visibility). To determine whether the specification is spatially possible one can check whether the conjunction of presented SOSL-formulas is satisfiable.

Preprocessing spatial data. Spatial data gathered by artificial visual systems (e.g., in robotics) is often represented in the polar coordinate system [31], hence SOSL can be used to reason about such data. Indeed, the following example shows how SOSL-formulas allow us to filter noisy data and detect borders of obstacles, where obst is a propositional variable satisfied in cells where an obstacle was detected:

$$\mathsf{noise} \leftrightarrow \mathsf{obst} \wedge (\neg\Diamond\!\!\!\!\uparrow\mathsf{filt\text{-}obst} \wedge \neg\Diamond\!\!\!\!\downarrow\mathsf{filt\text{-}obst} \wedge \neg\Diamond\!\!\!\!\leftarrow\mathsf{filt\text{-}obst} \wedge \neg\Diamond\!\!\!\!\rightarrow\mathsf{filt\text{-}obst}) \tag{1}$$

$$\mathsf{filt\text{-}obst} \leftrightarrow \mathsf{obst} \wedge \neg\mathsf{noise} \tag{2}$$

$$\mathsf{bord\text{-}obst} \leftrightarrow \mathsf{filt\text{-}obst} \wedge (\Diamond\!\!\!\!\uparrow\neg\mathsf{filt\text{-}obst} \vee \Diamond\!\!\!\!\downarrow\neg\mathsf{filt\text{-}obst} \vee \Diamond\!\!\!\!\leftarrow\neg\mathsf{filt\text{-}obst} \vee \Diamond\!\!\!\!\rightarrow\neg\mathsf{filt\text{-}obst}) \tag{3}$$

The formula (1) marks as noise each cell recognized as an obstacle, for which there are no adjacent cells which are also recognized as obstacles. In (2), the input data is filtered by marking all non-noise cells which were recognized as obstacles with filt-obst. Then, (3) determines the borders of obstacles by marking with bord-obst the cells which are marked as filt-obst and have at least one adjacent cell which is not marked as filt-obst.

Combining qualitative and quantitative reasoning. The modal operators in SOSL allow us to express qualitative as well as quantitative relations, and use both to perform reasoning. Indeed, for any $d, d1, d2 \in \mathbb{R}$, any nominals objA, objB, objC, and subject being a nominal for the subject, the following formulas are tautologies in SOSL:

$$\mathsf{subject} \wedge \Diamond_{=d}\mathsf{objA} \wedge @_{\mathsf{objA}}\Diamond\!\!\!\!\rightarrow\mathsf{objB} \rightarrow \Diamond_{>d}\mathsf{objB} \tag{4}$$

$$\mathsf{subject} \wedge \Diamond_{=d}\mathsf{objA} \wedge \Diamond_{=d}\mathsf{objB} \rightarrow @_{\mathsf{objA}}(\mathsf{objB} \vee \Diamond\!\!\!\!\leftarrow\mathsf{objB} \vee \Diamond\!\!\!\!\rightarrow\mathsf{objB}) \tag{5}$$

$$@_{\mathsf{objA}}\Diamond_{=d1}\mathsf{objB} \wedge @_{\mathsf{objB}}\Diamond_{=d2}\mathsf{objC} \rightarrow @_{\mathsf{objA}}\Diamond_{\leq d1+d2}\mathsf{objC} \tag{6}$$

The formula (4) states that if objB is behind objA, then objB is further away from the subject than objA, (5) states that if objA and objB are situated in the same distance from the subject, then they occupy the same cell, or one of them is to the left of the other, and (6) expresses the (purely quantitative) triangle inequality theorem.

The remaining part of the paper is structured as follows. In Section 2 we define the syntax and semantics of SOSL, and in Section 3 we compare SOSL with other modal logics for spatial reasoning. Then, in Section 4, we determine the operators that are expressible in SOSL, and in Section 5 we prove PSPACE-completeness of the satisfiability problem in the logic. Finally, we conclude the paper in Section 6.

2 Syntax and Semantics

In this section we present formally the syntax and semantics of SOSL. The language of the logic includes the following elements:

- PROP – a countably infinite set of propositional variables;
- $\{\neg, \vee\}$ – a set of logical connectives;
- $\{\blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge, \uparrow, \downarrow, \leftarrow, \rightarrow\}$ – a set of SOSL modal operators;
- $\{(,)\}$ – a set of parentheses.

▶ **Definition 1.** *The set of* SOSL-formulas *is generated by the following grammar:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \blacklozenge\varphi,$$

where $p \in$ PROP *and* $\blacklozenge$ *is an* SOSL *modal operator.*

The logical constants and other connectives are defined in a standard way: $\top = p \vee \neg p$, $\bot = \neg\top$, $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$, $\varphi \to \psi = \neg\varphi \vee \psi$, $\varphi \leftrightarrow \psi = (\varphi \to \psi) \wedge (\psi \to \varphi)$, where $p \in$ PROP and φ, ψ are SOSL-formulas. Moreover, for $\blacklozenge \in \{\blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge\}$ we define a dual box operator $\blacksquare\varphi = \neg\blacklozenge\neg\varphi$. For any SOSL-formula φ, we will use $\uparrow^n\varphi$ as an abbreviation of $\overbrace{\uparrow \ldots \uparrow}^{n \text{ times}}\varphi$, and we adopt the analogous notation for iterations of $\downarrow$, $\leftarrow$, and $\rightarrow$. Furthermore, for $n \in \mathbb{N}$, we identify $\uparrow^{-n}$ with $\downarrow^n$; and $\leftarrow^{-n}$ with $\rightarrow^n$.

The universe of an SOSL-model is an infinite plane partitioned into two-dimensional cells, each of length 1 and angular width of 1 degree, and an additional cell representing the subject, located at the center of the plane – see Figure 2. Each cell is identified with the pair of polar coordinates of its centroid, where the subject is denoted by $\langle 0, 0\rangle$ and every non-subject cell by $\langle r, \theta\rangle$, for $r \in \mathbb{N}_+$ (we use $\mathbb{N}_+$ for the set of positive natural numbers and $\mathbb{N}$ for natural numbers with 0) and θ belonging to the set of angle-values, defined as follows:

$$\mathsf{ang} = \{(0.5 + i) \mid i \in \{-180, \ldots, 179\}\}.$$

Note that ang contains 360 elements. We define the set $\mathcal{C}$ of all cells as follows:

$$\mathcal{C} = \{\langle 0, 0\rangle\} \cup \{\langle r, \theta\rangle \mid r \in \mathbb{N}_+ \text{ and } \theta \in \mathsf{ang}\}. \tag{7}$$

We introduce 12 binary relations between cells in $\mathcal{C}$. First, we define 4 subject-orientation relations: *in front of the subject* ($R_{\blacklozenge}$), *behind the subject* ($R_{\blacklozenge}$), *to the left of the subject* ($R_{\blacklozenge}$), and *to the right of the subject* ($R_{\blacklozenge}$) – for a pictorial representation see Figure 2(a):

$$R_{\blacklozenge} = \Big\{\big(\langle 0, 0\rangle, \langle r, \theta\rangle\big) \mid -45 < \theta < 45\Big\}; \quad R_{\blacklozenge} = \Big\{\big(\langle 0, 0\rangle, \langle r, \theta\rangle\big) \mid \theta < -135 \text{ or } \theta > 135\Big\};$$

$$R_{\blacklozenge} = \Big\{\big(\langle 0, 0\rangle, \langle r, \theta\rangle\big) \mid 45 < \theta < 135\Big\}; \quad R_{\blacklozenge} = \Big\{\big(\langle 0, 0\rangle, \langle r, \theta\rangle\big) \mid -135 < \theta < -45\Big\},$$

where $\langle r, \theta\rangle \in \mathcal{C}$ and $r \neq 0$. The angle coordinates of cells cannot have integer values, so $-135, -45, 45, 135 \notin \mathsf{ang}$, thus $R_{\blacklozenge}$, $R_{\blacklozenge}$, $R_{\blacklozenge}$, and $R_{\blacklozenge}$ partition the set of non-subject cells.

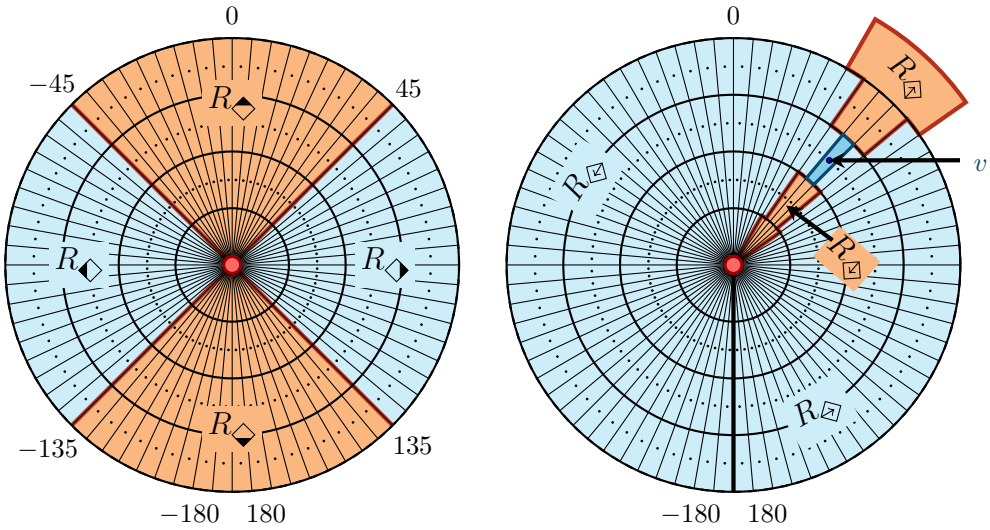


Figure 2 Relations in intrinsic (a) and relative (b) frames of reference. For a better readability we present 72 instead of all 360 cells in every ring around the subject.

The next 4 relations represent the location of one cell with respect to another (reference) cell (denoted by v in Figure 2(b)), from the subject's perspective: *behind a cell* ($R_{\diamondsuit\uparrow}$), *between a cell and the subject* ($R_{\diamondsuit\downarrow}$), *to the left of a cell* ($R_{\diamondsuit\leftarrow}$), and *to the right of a cell* ($R_{\diamondsuit\rightarrow}$), which are defined as follows:

$$R_{\diamondsuit\uparrow} = \Big\{\big(\langle r_1,\theta_1\rangle,\langle r_2,\theta_2\rangle\big) \mid r_1 < r_2 \text{ and } |\theta_1-\theta_2| \leq |r_1-r_2|\Big\};$$
$$R_{\diamondsuit\downarrow} = \Big\{\big(\langle r_1,\theta_1\rangle,\langle r_2,\theta_2\rangle\big) \mid r_2 < r_1 \text{ and } |\theta_1-\theta_2| \leq |r_1-r_2|\Big\};$$
$$R_{\diamondsuit\leftarrow} = \Big\{\big(\langle r_1,\theta_1\rangle,\langle r_2,\theta_2\rangle\big) \mid \theta_2 < \theta_1 \text{ and } |r_1-r_2| < |\theta_1-\theta_2|\Big\};$$
$$R_{\diamondsuit\rightarrow} = \Big\{\big(\langle r_1,\theta_1\rangle,\langle r_2,\theta\rangle\big) \mid \theta_1 < \theta_2 \text{ and } |r_1-r_2| < |\theta_1-\theta_2|\Big\},$$

where $\langle r_1,\theta_1\rangle, \langle r_2,\theta_2\rangle \in \mathcal{C}$, and $r_1, r_2 \neq 0$. The relations defined in this manner allow us to capture the subject's perspective. Observe that for each non-subject cell $\langle r_1,\theta_1\rangle$ the relations $R_{\diamondsuit\uparrow}$, $R_{\diamondsuit\downarrow}$, $R_{\diamondsuit\leftarrow}$, and $R_{\diamondsuit\rightarrow}$ partition the set of all non-subject cells distinct from $\langle r_1,\theta_1\rangle$.

The remaining 4 relations hold between adjacent cells: *away from the subject* ($R_{\uparrow}$), *towards the subject* ($R_{\downarrow}$), *counter-clockwise* ($R_{\ominus}$), and *clockwise* ($R_{\oplus}$). We define:

$$R_{\uparrow} = \Big\{\big(\langle r_1,\theta\rangle,\langle r_2,\theta\rangle\big) \mid r_2 = r_1+1\Big\} \cup \Big\{\big(\langle 0,0\rangle,\langle 1,\theta\rangle\big) \mid \theta \in \mathsf{ang}\Big\};$$
$$R_{\ominus} = \Big\{\big(\langle r,\theta_1\rangle,\langle r,\theta_2\rangle\big) \mid \theta_2 = \theta_1-1\Big\},$$

where $\langle r_1,\theta_1\rangle, \langle r_2,\theta_2\rangle \in \mathcal{C}$. We define $R_{\downarrow}$ and $R_{\oplus}$ as the converse relations of $R_{\uparrow}$ and $R_{\ominus}$.

▶ **Definition 2.** *An* SOSL-frame *is a pair* $\mathfrak{F} = (\mathcal{C}, \mathcal{R})$*, where* $\mathcal{C}$ *is defined as in (7) and* $\mathcal{R} = \{R_{\blacklozenge} \mid \blacklozenge$ *is an* SOSL*-operator*$\}$*. An* SOSL-model *is a pair* $\mathfrak{M} = (\mathfrak{F}, V)$*, where* $\mathfrak{F} = (\mathcal{C}, \mathcal{R})$ *is an* SOSL*-frame and* $V : \text{PROP} \longrightarrow \mathcal{P}(\mathcal{C})$*. The* forcing *relation* $\Vdash$ *is defined as follows:*

$$
\begin{array}{lll}
\mathfrak{M}, \langle r, \theta\rangle \Vdash p & \textit{iff} & \langle r, \theta\rangle \in V(p), \ \textit{for } p \in \text{PROP} \\
\mathfrak{M}, \langle r, \theta\rangle \Vdash \neg\varphi & \textit{iff} & \mathfrak{M}, \langle r, \theta\rangle \nVdash \varphi \\
\mathfrak{M}, \langle r, \theta\rangle \Vdash \varphi \vee \psi & \textit{iff} & \mathfrak{M}, \langle r, \theta\rangle \Vdash \varphi \ \textit{ or } \ \mathfrak{M}, \langle r, \theta\rangle \Vdash \psi \\
\mathfrak{M}, \langle r, \theta\rangle \Vdash \blacklozenge\varphi & \textit{iff} & \textit{there exists } \langle r', \theta'\rangle \textit{ such that } \langle r, \theta\rangle R_{\blacklozenge} \langle r', \theta'\rangle \textit{ and } \mathfrak{M}, \langle r', \theta'\rangle \Vdash \varphi,
\end{array}
$$

where $\langle r, \theta\rangle, \langle r', \theta'\rangle \in \mathcal{C}$*,* φ, ψ *are* $\mathcal{SOSL}$*-formulas, and* $\blacklozenge$ *is an* SOSL*-operator.*

We will denote an SOSL-model of the form $((\mathcal{C}, \mathcal{R}), V)$ by $(\mathcal{C}, \mathcal{R}, V)$. An SOSL-formula φ is satisfiable if there exists an SOSL-model $\mathfrak{M} = (\mathcal{C}, \mathcal{R}, V)$ and $\langle r, \theta\rangle \in \mathcal{C}$ such that $\mathfrak{M}, \langle r, \theta\rangle \Vdash \varphi$. Observe that $\langle 0, 0\rangle$ is a distinguished cell in every SOSL-model. In particular, a formula of the form $\blacklozenge\varphi$, for $\blacklozenge \in \{\diamondsuit, \diamondsuit, \diamondsuit, \diamondsuit\}$, can be satisfied only in $\langle 0, 0\rangle$, and no formula of the form $\blacklozenge\varphi$, for $\blacklozenge \in \{\diamondsuit, \diamondsuit, \diamondsuit, \diamondsuit\}$, can be satisfied in $\langle 0, 0\rangle$.

3 Related Work

In what follows we compare SOSL with some other modal logics for spatial reasoning. One of the main lines of distinction we consider is between types of reference systems used to describe the spatial location of an object in different logics. Usually, three main types of frames of reference are distinguished, namely intrinsic, relative, and absolute [14, 18, 15].

In the intrinsic frame of reference, the location of an object is determined by means of a binary relation between this object and a landmark object. The landmark object is "parsed" into its major parts, e.g., its front, back, left side, and right side, and then a named facet of the landmark object is used to describe the location of the first object, e.g., "the flower is in front of the door" [14]. The subject itself can be treated as the landmark object, and so "the flower is in front of me" is also a description in the intrinsic frame of reference [18].

In the relative frame of reference the position of an object is described by a ternary relation involving this object, a landmark object, and the subject (say myself), e.g., "the flower is between the door and myself" [18]. When describing the location of an object in the relative frame of reference, one also needs to perform "parsing" of objects and so it seems that the relative frame of reference cannot occur without the intrinsic one [26, 18].

In the absolute frame of reference, one uses fixed bearings or absolute coordinates such as north, south, west, and east, e.g., "the flower is to the south of the door". In the absolute frame of reference, in contrast to the intrinsic and relative ones, a spatial description does not change when the subject, not directly involved in the setting, changes their position [18].

In what follows we will compare SOSL with Compass Logic [32], Spatial Propositional Neighborhood Logic (SpPNL) [21], and Cone Logic [20]. Although spatial objects (constituting the universe of a model) and relations between them (interpreting modal operators) differ in these logics, the location of an object is always described in the absolute frame of reference. On the other hand, in SOSL we use both the intrinsic and relative frame of reference. Another difference is that in SOSL we adopt the polar coordinate system, whereas in the other logics the Cartesian system is used instead – see Table 1 for a cumulative comparison.

In Compass Logic [32, 19] spatial objects are points in the two-dimensional space, identified with their Cartesian coordinates. Modal operators express the following relations between points: "on the same horizontal line and above", "on the same horizontal line and below",

Table 1 Comparison of modal logics for spatial reasoning.

logic:	spatial object:	coordinates:	frame of reference:	complexity:
Compass	point	Cartesian	absolute	undecidable
SpPNL	rectangle			undecidable
Cone	set of points			PSpace
SOSL	cell	polar	intrinsic, relative	PSpace

"on the same vertical line and to the left", and "on the same vertical line and to the right". It is shown that the halting problem of a Turing machine reduces to the satisfiability problem of Compass Logic formulas, which makes the latter problem undecidable [19].

Similarly to Compass Logic, SpPNL uses the two-dimensional space with Cartesian coordinates. However, formulas are evaluated over rectangles rather than points. Intuitively, each spatial object is approximated with its minimum bounding rectangle, and modal operators express the following relations between these rectangles: "adjacent from above", "adjacent from below", "adjacent to the left", and "adjacent to the right". It is worth mentioning that similar relations are used in Rectangle Algebra [24]. Every formula from (undecidable) Compass Logic can be translated into an equisatisfiable SpPNL-formula, and so the satisfiability problem in SpPNL is also undecidable.

In Cone Logic [20] a spatial object is an arbitrary subset of the two-dimensional space. Each point is denoted by a pair of Cartesian coordinates and modal operators express the following cone-shaped relations between points: "to the north", "to the south", "to the west", and "to the east". The satisfiability problem in Cone Logic over the rational plane $\mathbb{Q} \times \mathbb{Q}$ is shown to be PSpace-complete using the tree (pseudo) model property [20].

To see how spatial representations differ in the above-discussed logics let us consider the configuration from Figure 1. As depicted in Figure 3, in Compass Logic "the flower is below and to the left of the door" (Figure 3a); in SpPNL "the minimal rectangle containing flower is (completely) to the left and (completely) below the minimal rectangle containing the door" (Figure 3b); in Cone Logic "the flower is to the south of the door" (Figure 3c); in SOSL "the flower is in front of the subject" (Figure 3d); and "the flower is between the door and the subject" (Figure 3e).

Observe that in Compass Logic, SpPNL, and Cone Logic the subject is not distinguished, and so their position has no influence on the spatial representation of a scene. On the other hand, the position of the subject is crucial when determining the relation between objects in SOSL. For instance, if in the scene from Figure 3 the subject was turned towards the west (instead of north), we would have "the flower is to the right of the subject" and if the subject was standing on the stairs, we would have "the flower is to the right of the door".

There is a number of other modal logics for spatial reasoning which, however, are less related to SOSL. For instance, $\mathsf{PDL}^{\mathsf{F}}_{\mathsf{M}}$ [22] is an extension of Propositional Dynamic Logic [13] in which relative movement of one object with respect to another is represented by means of a tuple of parameters. There are also logics based on topological spaces [5, 17, 25], where closure and interior operators are used to express topological relations such as "disconnected", "overlaps", and "inside" known from Region Connection Calculus [28]. Furthermore, there exist modal logics over affine spaces (allowing to express, e.g., the betweenness operator), logics of nearness, and logics of convexity, among others [1].

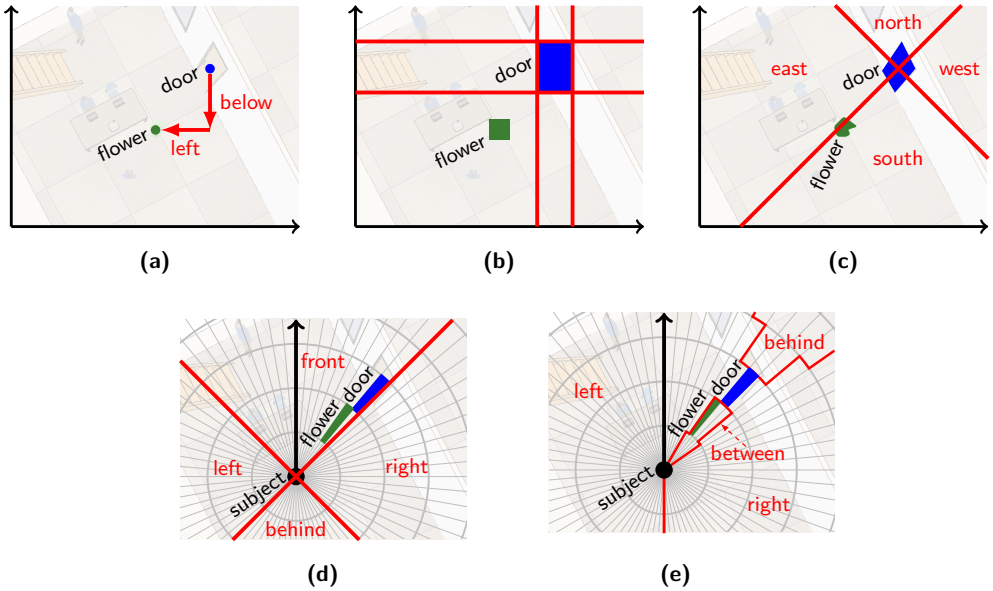


Figure 3 Spatial configuration from Figure 1 represented in Compass Logic (a), SpPNL (b), Cone Logic (c), SOSL (intrinsic frame of reference) (d), and SOSL (relative frame of reference) (e).

4 Expressive Power

As we have mentioned in Section 1, the language of SOSL is expressive enough to define a number of useful operators. First, we define the *difference* operator (D), where for an arbitrary SOSL-formula φ, $\mathsf{D}\varphi$ states that φ holds in some cell distinct from the current cell:

$$\mathsf{D}\varphi = \blacklozenge\varphi \vee \blacklozenge\varphi \vee \blacklozenge\varphi \vee \blacklozenge\varphi \vee$$
$$\blacklozenge\varphi \vee \blacklozenge\varphi \vee \blacklozenge\varphi \vee \blacklozenge\varphi \vee \blacklozenge\varphi \vee \blacklozenge\blacklozenge\varphi.$$

The first line of the definition expresses $\mathsf{D}\varphi$ if the current cell is the subject itself, namely the formula states that φ holds in some cell which is in front, behind, to the left, or to the right of the subject. The second line allows us to express $\mathsf{D}\varphi$ if the current cell is not the subject. In particular, it states that φ holds in some cell which is behind the current cell, between this cell and the subject, to the left or to the right of this cell, one cell towards the subject wrt to the current cell, or one cell towards the subject wrt to a cell which is between the current cell and the subject. The last two disjuncts are necessary to express the possibility that φ is satisfied in the subject. Then, the disjunction of the first and the second line expresses $\mathsf{D}\varphi$ in any cell. Observe that irreflexivity of relations interpreting SOSL modal operators is crucial for the definition presented above.

It is well-known that D allows us to express the *somewhere* operator (E) and its dual operator, the *everywhere* operator (A), as follows:

$$\mathsf{E}\varphi = \varphi \vee \mathsf{D}\varphi; \qquad \mathsf{A}\varphi = \neg\mathsf{E}\neg\varphi,$$

where $\mathsf{E}\varphi$ states that φ holds in some cell and $\mathsf{A}\varphi$ states that φ holds in all cells [2]. These operators can be used to simulate a *nominal* i with a propositional variable p_i, by expressing that p_i holds in exactly one cell [2]:

$$\mathsf{E}p_i \wedge \mathsf{A}(p_i \rightarrow \neg\mathsf{D}p_i).$$

Then, we can define a *satisfaction operator* $@_i$ (recall that $@_i\varphi$ states that φ holds in the cell in which i holds) as follows [2]:

$$@_i\varphi = \mathsf{E}(p_i \wedge \varphi).$$

Another interesting observation is that we can define a nominal $\mathsf{subject}$ which holds exactly in the cell $\langle 0, 0\rangle$:

$$\mathsf{subject} = \Diamond\!\!\!\!\blacklozenge\top.$$

Indeed, $\langle 0, 0\rangle$ is the only cell which has $R_{\Diamond\!\!\!\!\blacklozenge}$-successors, and so $\Diamond\!\!\!\!\blacklozenge\top$ holds only there. Furthermore, for any non-subject cell $\langle r, \theta\rangle$, we can define a formula $\mathsf{cell}_{\langle r,\theta\rangle}$ which holds exactly in this cell:

$$\mathsf{cell}_{\langle r,\theta\rangle} = \textcircled{\downarrow}^{r}\neg\textcircled{\downarrow}\top \ \wedge\ \textcircled{\leftarrow}^{\theta+179.5}\neg\textcircled{\leftarrow}\top.$$

The formula states that the current cell has exactly r $R_{\textcircled{\downarrow}}$-successors and exactly $\theta + 179,5$ $R_{\textcircled{\leftarrow}}$-successors, which is the case only if the current cell is $\langle r, \theta\rangle$.

Next, we show how to define operators expressing (a restricted form of) the distance between the central points of two cells. It is well-known that for two points with the polar coordinates, $\langle r_1, \theta_1\rangle$ and $\langle r_2, \theta_2\rangle$, we can compute the distance between them, denoted here by $dist(\langle r_1, \theta_1\rangle, \langle r_2, \theta_2\rangle)$, as follows:

$$dist(\langle r_1, \theta_1\rangle, \langle r_2, \theta_2\rangle) = \sqrt{r_1^2 + r_2^2 - 2 \cdot r_1 \cdot r_2 \cdot cos(\theta_1 - \theta_2)}.$$

Therefore, to compute the distance between $\langle r_1, \theta_1\rangle$ and $\langle r_2, \theta_2\rangle$, we need to determine the values of r_1, r_2, and $\theta_1 - \theta_2$, and then use them to perform arithmetic operations. As we show below, it can be done (in a restricted way) by means of an SOSL-formula. For any $d \in \mathbb{R}_+$ (where $\mathbb{R}_+$ is the set of positive real numbers) we can define an operator $\Diamond_{=d}$, such that $\Diamond_{=d}\varphi$ holds in $\langle r, \theta\rangle$ if φ holds in some $\langle r', \theta'\rangle$ such that $r \leq r'$ and $dist(\langle r, \theta\rangle, \langle r', \theta'\rangle) = d$:

$$\Diamond_{=d}\varphi = \textcircled{\uparrow}^{d} \vee \tag{8}$$

$$\bigvee_{\langle r,\theta\rangle \in \mathcal{C} \mid r < last(d)} \Big(\mathsf{cell}_{\langle r,\theta\rangle} \wedge \tag{9}$$

$$\bigvee_{\langle r',\theta'\rangle \in \mathcal{C} \mid r \leq r' \text{ and } dist(\langle r,\theta\rangle,\langle r',\theta'\rangle)=d} \textcircled{\uparrow}^{r'-r}\textcircled{\leftarrow}^{\theta'-\theta}\varphi\Big), \tag{10}$$

where $last(d) = \min\{n \in \mathbb{N}_+ \mid dist(\langle n, 0.5\rangle, \langle n, 1.5\rangle) > d\}$ and we assume that $\textcircled{\uparrow}^{d}\varphi$ is false if d is not an integer.

To capture the intuitions behind the definition above, assume that φ holds in $\langle r', \theta'\rangle$ which is in the distance d from $\langle r, \theta\rangle$ and such that $r \leq r'$. Clearly, we have either $\theta' = \theta$ or $\theta' \neq \theta$. In the first case, since the distance between $\langle r, \theta\rangle$ and $\langle r', \theta'\rangle$ equals d, we have $r' = r + d$, which is expressed by (8).

In the second case, we have $\theta' \neq \theta$. Suppose that $r \geq last(d)$. Then by the definition of $last(d)$ we can show that $dist(\langle r, \theta\rangle, \langle r', \theta'\rangle) > d$, which gives a contradiction with the assumption that $dist(\langle r, \theta\rangle, \langle r', \theta'\rangle) = d$. Therefore, $r < last(d)$, and so to determine the coordinates of $\langle r, \theta\rangle$ it suffices to check for what values of r and θ (where $r < last(d)$) the formula $\mathsf{cell}_{\langle r,\theta\rangle}$ is satisfied in the current cell, which is expressed by (9). Importantly, by the condition $r < last(d)$, the number of candidate values for r is bounded and so the disjunction in (9) is not infinite.

Assuming that we know the values of r and θ, it suffices to check whether there are values for r' and θ' such that $r \leq r'$, $dist(\langle r, \theta\rangle, \langle r', \theta'\rangle) = d$, and φ holds in $\langle r', \theta'\rangle$. Observe that for any values of r, θ, and d there is a bounded number of values of r' and θ' such that $r \leq r'$ and $dist(\langle r, \theta\rangle, \langle r', \theta'\rangle) = d$, because non-subject cells cannot have arbitrarily small dimensions (we can compute the size of the smallest non-subject cell). Hence, the size of the disjunction from (10) is bounded. Now, for all values of r' and θ' such that $r \leq r'$ and $dist(\langle r, \theta\rangle, \langle r', \theta'\rangle) = d$ it suffices to check whether φ holds in $\langle r', \theta'\rangle$, which is equivalent to checking if the formula $\textcircled{\uparrow}^{r'-r}\textcircled{\leftarrow}^{\theta'-\theta}\varphi$ is satisfied in $\langle r, \theta\rangle$, and which is expressed by (10).

In a similar way we can define operators $\Diamond_{<d}$ and $\Diamond_{\leq d}$:

$$\Diamond_{<d}\varphi = \bigvee_{k<d} \textcircled{\uparrow}^k\varphi \vee \bigvee_{\langle r,\theta\rangle | r<last(d)} \left(\mathsf{cell}_{\langle r,\theta\rangle} \wedge \bigvee_{\langle r',\theta'\rangle | r\leq r' \text{ and } dist(\langle r,\theta\rangle,\langle r,\theta\rangle)<d} \textcircled{\uparrow}^{r'-r}\textcircled{\leftarrow}^{\theta'-\theta}\varphi\right);$$

$$\Diamond_{\leq d}\varphi = \bigvee_{k\leq d} \textcircled{\uparrow}^k\varphi \vee \bigvee_{\langle r,\theta\rangle | r<last(d)} \left(\mathsf{cell}_{\langle r,\theta\rangle} \wedge \bigvee_{\langle r',\theta'\rangle | r\leq r' \text{ and } dist(\langle r,\theta\rangle,\langle r,\theta\rangle)\leq d} \textcircled{\uparrow}^{r'-r}\textcircled{\leftarrow}^{\theta'-\theta}\varphi\right),$$

where $k \in \mathbb{N}$ and $\langle r, \theta\rangle, \langle r', \theta'\rangle \in \mathcal{C}$.

We can also define operators $\Diamond_{>d}$ and $\Diamond_{\geq d}$, however, due to the limitation of space and relatively complex combinatorial properties which are encoded by these modalities, we omit their definitions.

The reader might have noticed that our distance operators are not symmetrical, i.e., they only define the distance between a cell $\langle r, \theta\rangle$ and another cell $\langle r', \theta'\rangle$ with $r \leq r'$. We can define, to a certain extent, symmetrical versions of these operators using hybrid machinery. By way of example, we can express that the cells in which nominals i and j are satisfied, are in distance d, as follows:

$$@_i\Diamond_{=d}j \vee @_j\Diamond_{=d}i.$$

We finish this section by presenting several tautologies of SOSL (formulas that are satisfied in every cell of every model) which demonstrate what kind of reasoning can be captured in this logic:

$$\textcircled{\uparrow}\textcircled{\uparrow}p \rightarrow \textcircled{\uparrow}p; \qquad p \rightarrow \boxed{\uparrow}\textcircled{\downarrow}p; \qquad \textcircled{\leftarrow}\boxed{\uparrow}p \rightarrow \boxed{\uparrow}(\textcircled{\leftarrow}\top \rightarrow \textcircled{\leftarrow}p); \qquad \textcircled{\blacktriangle}(i \wedge \textcircled{\rightarrow}j) \rightarrow \neg\textcircled{\blacklozenge}j,$$

where p is a propositional variable, and i, j, and k are nominals. The first formula informs that $R_{\textcircled{\uparrow}}$ is a transitive relation (the analogous formulas for $\textcircled{\downarrow}$, $\textcircled{\leftarrow}$, and $\textcircled{\rightarrow}$ are also tautologies); the second states that $R_{\textcircled{\downarrow}}$ is the converse relation for $R_{\textcircled{\uparrow}}$ (similarly to the fact that $R_{\textcircled{\rightarrow}}$ is the converse for $R_{\textcircled{\leftarrow}}$); the third encodes a restricted form of confluence of $R_{\textcircled{\leftarrow}}$ and $R_{\textcircled{\uparrow}}$; and the fourth claims that whenever i is situated in front of the subject, and j is to the right of i, it cannot be the case that j is located to left of the subject. For more tautologies of SOSL see (4)–(6) in Section 1.

5 Computational Complexity

In this section, we will show that SOSL-satisfiability is PSPACE-complete. We will prove the lower bound by reducing the satisfiability problem of Linear Temporal Logic and the upper bound by constructing a generalized nondeterministic Büchi automaton.

For the lower bound we reduce the satisfiability problem of Linear Temporal Logic whose language contains the following operators: *in the next time point* (X), *in the previous time point* (Y), *everywhere in the future* (G), and *everywhere in the past* (H), to the

SOSL-satisfiability. Linear Temporal Logic with the operators X, Y, G, and H is often referred to as $\mathsf{LTL}(X, Y, G, H)$, but since we do not consider any other linear temporal logic in this paper, we denote it simply by LTL [11].

The set of LTL-formulas is generated by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid Y\varphi \mid G\varphi \mid H\varphi,$$

where $p \in \textsc{prop}$.

An LTL-model is a pair $\mathcal{M} = (\mathbb{N}, V)$, where $\mathbb{N}$ is the set of natural numbers strictly ordered by $<$ and $V : \textsc{prop} \longrightarrow \mathcal{P}(\mathbb{N})$. The *forcing* relation $\Vdash$ is defined in a standard way (see [11, Section 2.2]). An LTL-formula φ is satisfiable if there exists an LTL-model $\mathcal{M}$ such that $\mathcal{M}, 0 \Vdash \varphi$. The satisfiability problem of LTL is known to be PSPACE-complete ([11, 3]).

▶ **Theorem 3.** SOSL-*satisfiability is* PSPACE-*hard.*

Proof sketch. Our reduction of LTL-satisfiability to SOSL-satisfiability is based on the idea that we can simulate an LTL-model with the left-most ray of an SOSL-model. To capture this correspondence formally, we will introduce a translation f of LTL-models into SOSL-models and a translation g of SOSL-models into LTL-models.

Let $\mathcal{M} = (\mathbb{N}, V)$ be an LTL-model. We define $f(\mathcal{M}) = (\mathcal{C}, \mathcal{R}, V')$ as an SOSL-model such that for all $\langle r, \theta\rangle \in \mathcal{C}$ and $p \in \textsc{prop}$ (without loss of generality we assume that the sets of propositional variables in LTL and SOSL coincide):

$$\langle r, \theta\rangle \in V'(p) \quad \text{iff} \quad \theta = -179.5 \text{ and there is } n \in \mathbb{N} \text{ such that } n \in V(p) \text{ and } r = n + 1.$$

Hence, the valuation of propositional variables on the left-most ray of $f(\mathcal{M})$ is the same as the valuation on the time line in $\mathcal{M}$, whereas in cells of $f(\mathcal{M})$ which do not belong to the left-most ray, no propositional variable is satisfied.

On the other hand, for an SOSL-model $\mathfrak{M} = (\mathcal{C}, \mathcal{R}, V)$, we define $g(\mathfrak{M}) = (\mathbb{N}, V')$ as an LTL-model such that for all $n \in \mathbb{N}$ and $p \in \textsc{prop}$:

$$n \in V'(p) \quad \text{iff} \quad \langle n + 1, -179.5\rangle \in V(p).$$

Thus, the valuation of propositional variables in $g(\mathfrak{M})$ coincides with the valuation in the left-most ray of $\mathfrak{M}$.

Now, let φ be an abitrary LTL-formula. Based on the above-presented correspondence between LTL-models and SOSL-models, we will construct an SOSL-formula φ' which is equisatisfiable with φ. First, we define a translation tr of LTL-formulas into SOSL-formulas:

$$\begin{array}{rclcrcl}
\mathsf{tr}(p) & = & p & \qquad & \mathsf{tr}(X\psi) & = & \textcircled{\uparrow}\mathsf{tr}(\psi) \\
\mathsf{tr}(\neg\psi) & = & \neg\mathsf{tr}(\psi) & & \mathsf{tr}(Y\psi) & = & \textcircled{\downarrow}\textcircled{\downarrow}\top \wedge \textcircled{\downarrow}\mathsf{tr}(\psi) \\
\mathsf{tr}(\psi \vee \chi) & = & \mathsf{tr}(\psi) \vee \mathsf{tr}(\chi) & & \mathsf{tr}(G\psi) & = & \boxed{\uparrow}(\boxed{\leftarrow}\bot \rightarrow \mathsf{tr}(\psi)) \\
 & & & & \mathsf{tr}(H\psi) & = & \boxed{\downarrow}(\boxed{\leftarrow}\bot \wedge \textcircled{\downarrow}\top \rightarrow \mathsf{tr}(\psi)),
\end{array}$$

where $p \in \textsc{prop}$ and φ, ψ are LTL-formulas. This translation allows us to simulate LTL-formulas with SOSL-formulas in the sense that the following implications hold for every LTL-formula ψ, LTL-model $\mathcal{M}$, SOSL-model $\mathfrak{M}$, and $n \in \mathbb{N}$:

- If $\mathcal{M}, n \Vdash \psi$, then $f(\mathcal{M}), \langle n + 1, -179, 5\rangle \Vdash \mathsf{tr}(\psi)$;
- If $\mathfrak{M}, \langle n + 1, -179.5\rangle \Vdash \mathsf{tr}(\psi)$, then $g(\mathfrak{M}), n \Vdash \psi$.

Indeed, we can prove these implications by induction on the complexity of ψ (we omit the details due to space limitations). Then, we define φ' as follows:

$$\varphi' = \mathsf{tr}(\varphi) \wedge \mathsf{cell}_{\langle 1,-179,5\rangle},$$

where $\mathsf{cell}_{\langle 1,-179,5\rangle}$ is the formula which was defined formally in Section 4 and which is satisfied exactly in $\langle 1, -179, 5\rangle$.

Assume that φ is LTL-satisfiable, so there is an LTL-model $\mathcal{M}$ such that $\mathcal{M}, 0 \Vdash \varphi$. Thus $f(\mathcal{M}), \langle 1, -179, 5\rangle \Vdash \varphi'$, and so φ' is SOSL-satisfiable. For the opposite direction, assume that φ' is SOSL-satisfiable. Then, there is an SOSL-model $\mathfrak{M}$ such that $\mathfrak{M}, \langle 1, -179.5\rangle \Vdash \varphi'$ (note that φ' can only be satisfied in $\langle 1, -179.5\rangle$ by the definition of $\mathsf{cell}_{\langle 1,-179,5\rangle}$). Hence, $g(\mathfrak{M}), 0 \Vdash \varphi$, and so φ is LTL-satisfiable. It follows that φ is LTL-satisfiable if and only if φ' is SOSL-satisfiable. Since $\varphi_{\langle 1,-179,5\rangle}$ is of a constant size and tr is a logarithmic-space translation, the whole reduction is in logarithmic space, which finishes the proof. ◀

In the remainder of this section we show that SOSL-satisfiability is in PSpace. The main part of the proof consists of checking in PSpace whether there exists an SOSL-model such that a given SOSL-formula φ is satisfied in this model in $\langle 0, 0\rangle$. To do it, we will first guess in NPSpace a set K (called a *kernel*) of formulas which are satisfied in $\langle 0, 0\rangle$ (thus $\varphi \in K$). Then, we will construct a generalized Büchi automaton $G_{\varphi,K}$ whose states determine formulas which are satisfied in cells sharing the same radius coordinate (a set of such cells forms a *ring* around $\langle 0, 0\rangle$); each symbol of the automaton's alphabet is a sequence of 360 sets of propositional variables (which encodes a valuation of propositional variables in cells located within one ring); the transition relation forces the consecutive rings to match each other; and the accepting sets assure that every "promise" (e.g., that p holds somewhere in front of the subject) will eventually be kept. As we will show, an infinite word w is accepted by $G_{\varphi,K}$ (i.e., there is a run of $G_{\varphi,K}$ on w, in which at least one state from each accepting set occurs infinitely many times during the run) if and only if w describes an SOSL-model in which φ is satisfied in $\langle 0, 0\rangle$.

First, we define the closure of an SOSL-formula φ, denoted by $\mathsf{cl}(\varphi)$, as the set of all subformulas of φ and their negations (where ψ and $\neg\neg\psi$ are identified). The neighbourhood closure of φ, referred to as $\mathsf{ncl}(\varphi)$, is the set of all formulas of the form $\textcircled{\uparrow}^n\psi$, $\textcircled{\downarrow}^n\psi$, and their negations, for $0 \leq n \leq 360$ and $\psi \in \mathsf{cl}(\varphi)$. Moreover, $\mathsf{cl}_\diamond(\varphi)$ is the set of subformulas of φ of the form $\diamondsuit\psi$, $\diamondsuit\psi$, $\diamondsuit\psi$, and $\diamondsuit\psi$, and their negations. A set $X \subseteq \mathsf{ncl}(\varphi)$ is:

(con) *consistent wrt propositional logic* if for all $\psi, \varphi_1 \vee \varphi_2 \in \mathsf{ncl}(\varphi)$ the following hold:
- if $\psi \in X$, then $\neg\psi \notin X$;
- $\varphi_1 \vee \varphi_2 \in X$ iff $\varphi_1 \in X$ or $\varphi_2 \in X$;

(max) *maximal* if for all $\psi \in \mathsf{ncl}(\varphi)$:
- if $\psi \notin X$, then $\neg\psi \in X$.

We will denote the set of propositional variables occurring in φ by $\textsc{prop}(\varphi)$.

From now on, unless stated otherwise, we assume that φ is a fixed SOSL-formula. We define a *kernel* for φ as a set of formulas which are to be satisfied in $\langle 0, 0\rangle$ (consequently, a kernel contains φ).

▶ **Definition 4.** *A φ-kernel is a set $K \subseteq \mathsf{ncl}(\varphi)$ satisfying* (con) *and* (max)*, and such that for all $\psi \in \mathsf{ncl}(\varphi)$ the following hold:*

1. $\diamondsuit\psi, \diamondsuit\psi, \diamondsuit\psi, \diamondsuit\psi, \textcircled{\downarrow}\psi, \ominus\psi, \ominus\psi \notin K$;
2. $\varphi \in K$.

A kernel contains formulas satisfied in $\langle 0, 0\rangle$, whereas *rings* (which are to be states of the automaton) contain formulas satisfied in cells located in consecutive rings around $\langle 0, 0\rangle$.

▶ **Definition 5.** *A* φ-ring *is a function* R *whose domain is* $\mathsf{ang} \cup \{\diamond\}$*, such that:*

- $R(\theta) \subseteq \mathsf{ncl}(\varphi)$ *and sastisfies* (con) *and* (max)*, for all* $\theta \in \mathsf{ang}$*;*
- $R(\diamond) \subseteq \mathsf{cl}_\diamond(\varphi)$*.*

Moreover, for all $\psi \in \mathsf{ncl}(\varphi)$*,* $m, n \in \mathbb{N}$*, and* $\theta \in \mathsf{ang}$*:*

1. $\ominus^n \psi \in R(\theta)$ *iff* $\psi \in R(\theta - n)$*;*
2. $\oplus^n \psi \in R(\theta)$ *iff* $\psi \in R(\theta + n)$*;*
3. $\boxplus \psi \in R(\theta)$ *iff there are* $n > 0$ *and* $m < n$ *such that* $\uparrow^m \psi \in R(\theta - n)$ *or* $\downarrow^m \psi \in R(\theta - n)$*;*
4. $\boxplus \psi \in R(\theta)$ *iff there are* $n > 0$ *and* $m < n$ *such that* $\uparrow^m \psi \in R(\theta + n)$ *or* $\downarrow^m \psi \in R(\theta + n)$*;*
5. $\blacklozenge \psi, \blacklozenge \psi, \blacklozenge \psi, \blacklozenge \psi \notin R(\theta)$*.*

Intuitively, conditions 1-5 guarantee that a ring is locally consistent. By conditions 1-4, a formula of the form $\ominus^n \psi$, $\oplus^n \psi$, $\boxplus \psi$, or $\boxplus \psi$, belongs to $R(\theta)$ if and only if it has a witness on the same ring; and by condition 5 formulas of the form $\blacklozenge \psi, \blacklozenge \psi, \blacklozenge \psi$, and $\blacklozenge \psi$ cannot occur in any $R(\theta)$ since such formulas can be satisfied only in $\langle 0, 0 \rangle$.

Next, we define conditions which need to be fulfilled by a ring succeeding a kernel, and by a ring succeeding another ring. First, we impose conditions on the formulas preceded with $\blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge$, and their negations.

▶ **Definition 6.** *Let* $X \subseteq \mathsf{ncl}(\varphi)$ *be a* φ*-kernel or* $R(\diamond)$ *from a* φ*-ring* R*. The set* X $\diamond$-matches *a* φ*-ring* R' *if the following conditions are met:*

1. *if* $\blacklozenge \psi \in X$ *and* $\psi \notin R'(\theta)$ *for all* $\theta \in \mathsf{ang}$ *such that* $-45 < \theta < 45$*, then* $\blacklozenge \psi \in R'(\diamond)$*;*
2. *if* $\neg \blacklozenge \psi \in X$*, then* $\neg \blacklozenge \psi \in R'(\diamond)$ *and* $\neg \psi \in R'(\theta)$*, for all* $\theta \in \mathsf{ang}$ *such that* $-45 < \theta < 45$*,*

and the analogous conditions (obtained by modifying the range of θ*) for* $\blacklozenge$*,* $\blacklozenge$*, and* $\blacklozenge$*.*

Intuitively, the first condition states that $\blacklozenge \psi$ needs to be present in consecutive rings until ψ is present in $R(\theta)$ for some ring R and θ such that $-45 < \theta < 45$. The second condition states that if $\neg \blacklozenge \psi$ is present in some ring, then it needs to be present in the consecutive ring.

▶ **Definition 7.** *A* φ*-ring* R' *is a* successor *of a* φ*-kernel* K *if* R' $\diamond$*-matches* K *and:*

1. *for all* $\uparrow \psi \in \mathsf{ncl}(\varphi)$*:* $\uparrow \psi \in K$ *iff there is* $\theta \in \mathsf{ang}$ *such that* $\psi \in R'(\theta)$*;*
2. *for all* $\downarrow \psi \in \mathsf{ncl}(\varphi)$ *the following are equivalent:* $\psi \in K$*; there is* $\theta \in \mathsf{ang}$ *such that* $\downarrow \psi \in R'(\theta)$*; for all* $\theta \in \mathsf{ang}$ $\downarrow \psi \in R'(\theta)$*;*
3. *for all* $\boxplus \psi \in \mathsf{ncl}(\varphi)$ *and* $\theta \in \mathsf{ang}$*:* $\boxplus \psi \notin R'(\theta)$*.*

Similarly to the case of the 1st ring which is the successor of a kernel, the $(n+1)$th ring needs to fulfil particular conditions to be the successor of the nth ring. In the following definition we present the conditions which have to be satisfied by consecutive rings:

▶ **Definition 8.** *A* φ*-ring* R' *is a* successor *of a* φ*-ring* R *if* R' $\diamond$*-matches* $R(\diamond)$ *and for all* $\theta \in \mathsf{ang}$ *the following hold:*

1. *for all* $\uparrow \psi \in \mathsf{ncl}(\varphi)$*:* $\uparrow \psi \in R(\theta)$ *iff* $\psi \in R'(\theta)$*;*
2. *for all* $\downarrow \psi \in \mathsf{ncl}(\varphi)$*:* $\downarrow \psi \in R'(\theta)$ *iff* $\psi \in R(\theta)$*;*
3. *for all* $\boxplus \psi \in \mathsf{ncl}(\varphi)$*:* $\boxplus \psi \in R(\theta)$ *iff at least one of the following holds:* $\psi \in R'(\theta - 1)$*,* $\psi \in R'(\theta)$*,* $\psi \in R'(\theta + 1)$*,* $\boxplus \psi \in R'(\theta - 1)$*,* $\boxplus \psi \in R'(\theta)$*, or* $\boxplus \psi \in R'(\theta + 1)$*;*
4. *for all* $\boxplus \psi \in \mathsf{ncl}(\varphi)$*:* $\boxplus \psi \in R'_\theta$ *iff at least one of the following holds:* $\psi \in R(\theta - 1)$*,* $\psi \in R(\theta)$*,* $\psi \in R(\theta + 1)$*,* $\boxplus \psi \in R(\theta - 1)$*,* $\boxplus \psi \in R(\theta)$*, or* $\boxplus \psi \in R(\theta + 1)$*.*

Now, we are ready to define a generalized Büchi automaton for an $\mathcal{SOSL}$-formula φ and a φ-kernel K:

▶ **Definition 9.** *For a φ-kernel K we define a generalized nondeterministic Büchi automaton $G_{\varphi,K} = (\Sigma, Q, Q_0, \delta, \mathcal{F})$, where:*

- *Σ is the set of all functions $\sigma : \mathsf{ang} \longrightarrow \mathcal{P}(\text{PROP}(\varphi))$;*
- *Q is the set of all φ-rings;*
- *$Q_0 = \{R \in Q \mid R \text{ is a successor of } K\}$;*
- *$\delta : Q \times \Sigma \longrightarrow \mathcal{P}(Q)$ is such that for all $R \in Q$ and $\sigma \in \Sigma$:*

  $$\delta(R, \sigma) = \{R' \in Q \mid R' \text{ is a successor of } R \text{ and } R'(\theta) \cap \text{PROP}(\varphi) = \sigma(\theta), \text{ for all } \theta \in \mathsf{ang}\};$$

- *$\mathcal{F} = \{\mathcal{F}^{\theta}_{\textcircled{\scriptsize\diamond}\psi} \mid \theta \in \mathsf{ang} \text{ and } \textcircled{\scriptsize\diamond}\psi \in \mathsf{ncl}(\varphi)\} \cup \{\mathcal{F}_{\blacklozenge\psi} \mid \blacklozenge \in \{\blacktriangle, \blacktriangledown, \blacktriangleleft, \blacktriangleright\} \text{ and } \blacklozenge\psi \in \mathsf{cl}_{\diamond}(\varphi)\}$, where:*
  - *$\mathcal{F}^{\theta}_{\textcircled{\scriptsize\diamond}\psi} = \{R \in Q \mid \textcircled{\scriptsize\diamond}\psi \notin R(\theta) \text{ or } \psi \in R(\gamma), \text{ for some } \gamma \in \mathsf{ang}\}$;*
  - *$\mathcal{F}_{\blacklozenge\psi} = \{R \in Q \mid \blacklozenge\psi \notin R(\diamond)\}$.*

Intuitively, the definition of a state of the automaton (i.e., the Definition 5 of a ring) is responsible for capturing the meaning of formulas preceded with $\circleddash, \circleddash, \textcircled{\scriptsize\leftrightarrow}$, and $\textcircled{\scriptsize\diamond}$; the transition relation (based on the Definition 7 of a successor ring) captures the meaning of formulas preceded with $\textcircled{\scriptsize\uparrow}, \textcircled{\scriptsize\downarrow}$, and $\textcircled{\scriptsize\diamond}$; whereas the accepting sets allow us to capture the meaning of formulas preceded with $\blacktriangle, \blacktriangledown, \blacktriangleleft, \blacktriangleright$, and $\textcircled{\scriptsize\diamond}$. As we show below, the automaton accepts only those words that represent SOSL-models in which φ is satisfied in $\langle 0, 0\rangle$.

▶ **Lemma 10.** *For every SOSL-formula φ the following statements are equivalent:*

1. *There exists a φ-kernel K such that the language of $G_{\varphi,K}$ is non-empty;*
2. *There exists an SOSL-model $\mathfrak{M}$ such that $\mathfrak{M}, \langle 0, 0\rangle \Vdash \varphi$.*

Proof sketch.

(1 ⇒ 2) Let $w = \sigma^2\sigma^3\ldots$ be an infinite word accepted by $G_{\varphi,K}$ and let $R^1, R^2, \ldots$ be an infinite sequence of states in an accepting run of $G_{\varphi,K}$ on w. We define an SOSL-model $\mathfrak{M} = (\mathcal{C}, \mathcal{R}, V)$ such that for all $p \in \text{PROP}(\varphi)$:

- $\langle 0, 0\rangle \in V(p)$ iff $p \in K$;
- for all $\langle r, \theta\rangle \in \mathcal{C}$ distinct from $\langle 0, 0\rangle$ we have $\langle r, \theta\rangle \in V(p)$ iff $p \in R^r(\theta)$.

We can show that for each $\psi \in \mathsf{ncl}(\varphi)$ the following implications hold:

$$\text{If } \psi \in K, \text{ then } \mathfrak{M}, \langle 0, 0\rangle \Vdash \psi; \qquad \text{If } \psi \in R^r(\theta), \text{ then } \mathfrak{M}, \langle r, \theta\rangle \Vdash \psi.$$

We prove these implications by simultaneous induction on the complexity of ψ. The proof has a great number of cases (as the language of SOSL contains 12 modal operators) and uses numerous conditions from Definitions 4–9. Due to space limitations we omit the details.

By Definition 4, we have $\varphi \in K$, and so, by the first of the proved implications, we obtain $\mathfrak{M}, \langle 0, 0\rangle \Vdash \varphi$.

(1 ⇐ 2) Let $\mathfrak{M} = (\mathcal{C}, \mathcal{R}, V)$ be an SOSL-model such that $\mathfrak{M}, \langle 0, 0\rangle \Vdash \varphi$. We define:

$$K = \{\psi \in \mathsf{ncl}(\varphi) \mid \mathfrak{M}, \langle 0, 0\rangle \Vdash \psi\}.$$

Since $\mathfrak{M}$ is an SOSL-model, the set K satisfies the conditions from Definition 4, so K is a φ-kernel. Next, we will show that the language of $G_{\varphi,K} = (\Sigma, Q, Q_0, \delta, \mathcal{F})$ is non-empty. Let $w = \sigma^2\sigma^3\ldots$ be an infinite word such that for all $r \in \mathbb{N}_+$ and $\theta \in \mathsf{ang}$ we have:

$$\sigma^r(\theta) = \{p \in \text{PROP}(\varphi) \mid \mathfrak{M}, \langle r, \theta\rangle \Vdash p\}.$$

Next, let $\mathfrak{R} = R^1, R^2, \dots$ be an infinite sequence of functions R^r over the domain $\mathsf{ang} \cup \{\diamond\}$, such that for all $r \in \mathbb{N}_+$, $\theta \in \mathsf{ang}$, and $\blacklozenge \in \{\blacklozenge, \blacklozenge, \blacklozenge, \blacklozenge\}$ we have:

$$R^r(\diamond) = \{\blacklozenge\psi \in K \mid \psi \notin R^l(\gamma), \text{ for all } \langle l, \gamma\rangle \in \mathcal{C} \text{ such that } \langle 0,0\rangle R_{\blacklozenge} \langle l, \gamma\rangle \text{ and } l \leq r\} \cup \{\neg\blacklozenge\psi \mid \neg\blacklozenge\psi \in K\};$$
$$R^r(\theta) = \{\psi \in \mathsf{ncl}(\varphi) \mid \mathfrak{M}, \langle r, \theta\rangle \Vdash \psi\}.$$

Each R^r satisfies the conditions from Definition 5, so $\mathfrak{R}$ is a sequence of φ-rings, i.e., a sequence of states of $G_{\varphi,K}$. It remains to show that $\mathfrak{R}$ is an accepting run of $G_{\varphi,K}$ on w, i.e., that the following hold (since the proof of these conditions is long and standard, we omit it):

- $R^1 \in Q_0$;
- for all $r \in \mathbb{N}_+$ we have $R^{r+1} \in \delta(R^r, \sigma^{r+1})$;
- for every set in $\mathcal{F}$ there are infinitely many elements of $\mathfrak{R}$ belonging to this set.

As a result, the language of $G_{\varphi,K}$ is non-empty (as it contains w), which finishes the proof. ◀

Using Lemma 10 we can show the upper bound for the complexity of SOSL-satisfiability.

▶ **Theorem 11.** SOSL-*satisfiability is in* PSPACE.

Proof. To check whether an SOSL-formula φ is SOSL-satisfiable, we define:

$$\varphi' = \varphi \vee \blacklozenge\varphi \vee \blacklozenge\varphi \vee \blacklozenge\varphi \vee \blacklozenge\varphi.$$

Clearly, φ' is of polynomial size wrt the size of φ (denoted by $|\varphi|$) and φ is satisfiable if and only if there exists an SOSL-model in which φ' is satisfied in $\langle 0,0\rangle$, which by Lemma 10 is equivalent to the existence of a φ'-kernel K such that the language of $G_{\varphi',K}$ is non-empty.

To check this condition we nondeterministically guess a φ'-kernel K. By Definition 4 we have $K \subseteq \mathsf{ncl}(\varphi')$ and since $\mathsf{ncl}(\varphi')$ is of a polynomial size wrt $|\varphi'|$, the kernel K is also of a polynomial size wrt $|\varphi'|$, and thus we can guess it in NPSPACE (note that NPSPACE = PSPACE [30]). Although we cannot construct $G_{\varphi',K}$ in PSPACE (as the number of states of the automaton is exponential wrt $|\varphi'|$), we can examine emptiness of its language by guessing "on-the-fly" (one-by-one) the successor states in an accepting run [4]. Indeed, each state of $G_{\varphi',K}$ is of polynomial size wth $|\varphi'|$, so the guessing can be done in NPSPACE. To confirm that this run is accepting it suffices to check if the first guessed state is an initial state, every next state matches the previous state wrt to the transition relation, and the accepting states are visited infinitely often, which can be done in PSPACE. Hence, the whole procedure is in PSPACE. ◀

As a result of Theorem 3 and Theorem 11 we obtain the following tight complexity bound for the satisfiability problem:

▶ **Corollary 12.** SOSL-*satisfiability is* PSPACE-*complete.*

It is worth noticing that, in general, the satisfiability problem in two-dimensional logics in which each dimension is a linear order with an infinite ascending chain is undecidable [29, 12]. Decidability (and in particular membership in PSPACE) of the satisfiability problem of SOSL is obtained mainly as a result of using the polar coordinate system in which one dimension (corresponding to angular coordinates) is finite.

6 Conclusions

We have constructed a two-dimensional modal logic for subject-oriented spatial reasoning, denoted by SOSL. The approach employs the polar coordinate system to divide an infinite plane into cells of constant length and constant angle-width. Spatial representation reflects the subject's perspective and exploits the relations "in front", "behind", "to the left", and "to the right" of the subject; and "behind an object", "between the subject and an object", "to the left of an object", and "to the right of an object". Such relations are commonly used in most Indo-European languages, where the intrinsic frame of reference (the first 4 relations) and the relative frame of reference (the remaining 4 relations) are most often used to represent and reason about space. We have shown that the language of SOSL allows us to express, e.g., nominals, satisfaction operators, and modal operators for distance. We have proved decidability and PSPACE-completeness of the satisfiability problem in SOSL. In the future we plan to search for low-complexity fragments and modifications of this logic.

References

1 Marco Aiello and Johan Van Benthem. A modalwalk through space. *Journal of Applied Non-Classical Logics*, 12(3-4):319–363, 2002.

2 Carlos Areces, Patrick Blackburn, and Maarten Marx. The computational complexity of hybrid temporal logics. *Logic Journal of IGPL*, 8(5):653–679, 2000.

3 Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. The Complexity of Clausal Fragments of LTL. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning: 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, pages 35–52, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

4 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

5 Brandon Bennett. Modal logics for qualitative spatial reasoning. *Logic Journal of the IGPL*, 4(1):23–45, 1996.

6 Mehul Bhatt, Carl Schultz, and Minqian Huang. The shape of empty space: Human-centred cognitive foundations in computing for spatial design. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 33–40. IEEE, 2012.

7 Francis DK Ching. *Architecture: Form, space, and order*. John Wiley & Sons, 2014.

8 Bowman L Clark et al. Individuals and points. *Notre Dame Journal of Formal Logic*, 26(1):61–75, 1985.

9 Matt Duckham, Jenny Lingham, Keith Mason, and Michael Worboys. Qualitative reasoning about consistency in geographic information. *Information Sciences*, 176(6):601–627, 2006.

10 Manfred Eppe and Mehul Bhatt. Narrative based postdictive reasoning for cognitive robotics. *arXiv preprint*, 2013. `arXiv:1306.0665`.

11 Michael Fisher. A Resolution Method for Temporal Logic. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence – Volume 1*, IJCAI'91, pages 99–104, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

12 Dov M Gabbay, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. *Many-dimensional modal logics: Theory and applications*. Elsevier North Holland, Amsterdam, Boston, 2003.

13 Dexter Kozen and Rohit Parikh. An elementary proof of the completeness of PDL. *Theoretical Computer Science*, 14(1):113–118, 1981.

14 Stephen C Levinson. Frames of reference and Molyneux's question: Crosslinguistic evidence. *Language and Space*, 109:169, 1996.

15 Stephen C Levinson. Language and space. *Annual Review of Anthropology*, 25(1):353–382, 1996.

16 Gérard Ligozat. On Generalized Interval Calculi. In Thomas L. Dean and Kathleen McKeown, editors, *AAAI*, pages 234–240. AAAI Press / The MIT Press, 1991.

17 Carsten Lutz and Frank Wolter. Modal logics of topological relations. *arXiv preprint*, 2006. `arXiv:cs/0605064`.

18 Asifa Majid, Melissa Bowerman, Sotaro Kita, Daniel BM Haun, and Stephen C Levinson. Can language restructure cognition? The case for space. *Trends in Cognitive Sciences*, 8(3):108–114, 2004.

19 Maarten Marx and Mark Reynolds. Undecidability of Compass Logic. *Journal of Logic and Computation*, 9(6):897–914, 1999.

20 Angelo Montanari, Gabriele Puppis, and Pietro Sala. A decidable spatial logic with cone-shaped cardinal directions. In *Computer Science Logic*, pages 394–408. Springer, 2009.

21 Antonio Morales, Isabel Navarrete, and Guido Sciavicco. A new modal logic for reasoning about space: Spatial propositional neighborhood logic. *Annals of Mathematics and Artificial Intelligence*, 51(1):1–25, 2007.

22 Emilio Muñoz-Velasco, Alfredo Burrieza, and Manuel Ojeda-Aciego. A logic framework for reasoning with movement based on fuzzy qualitative representation. *Fuzzy Sets and Systems*, 242:114–131, 2014.

23 Isabel Navarrete, Antonio Morales, and Guido Sciavicco. Consistency Checking of Basic Cardinal Constraints over Connected Regions. In *IJCAI*, pages 495–500, 2007.

24 Isabel Navarrete, Antonio Morales, Guido Sciavicco, and M. Antonia Cardenas-Viedma. Spatial reasoning with rectangular cardinal relations. *Annals of Mathematics and Artificial Intelligence*, 67(1):31–70, 2013.

25 Werner Nutt. On the translation of qualitative spatial reasoning problems into modal logics. In *Annual Conference on Artificial Intelligence*, pages 113–124. Springer, 1999.

26 Eric Pederson. How many reference frames? In *International Conference on Spatial Cognition*, pages 287–304. Springer, 2002.

27 Ian Pratt and Dominik Schoop. A complete axiom system for polygonal mereotopology of the real plane. *Journal of Philosophical Logic*, 27(6):621–658, 1998.

28 David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions and Connection. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, pages 165–176. Morgan Kaufmann, 1992.

29 Mark Reynolds and Michael Zakharyaschev. On the products of linear modal logics. *Journal of Logic and Computation*, 11(6):909–931, 2001.

30 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, 4(2):177–192, 1970.

31 V. Javier Traver and Alexandre Bernardino. A review of log-polar imaging for visual perception in robotics. *Robotics and Autonomous Systems*, 58(4):378–398, 2010.

32 Yde Venema. Expressiveness and completeness of an interval tense logic. *Notre Dame Journal of Formal Logic*, 31(4):529–547, 1990.

A Appendix (proofs)

▶ **Theorem 3.** SOSL*-satisfiability is* PSPACE*-hard.*

Proof. To finish the proof we need to show that the following implications hold for any LTL-formula ψ and any $n \in \mathbb{N}$:

$$\text{If} \quad \mathcal{M}, n \Vdash \psi, \quad \text{then} \quad f(\mathcal{M}), \langle n+1, -179, 5\rangle \Vdash \mathsf{tr}(\psi); \tag{11}$$

$$\text{If} \quad \mathfrak{M}, \langle n+1, -179.5\rangle \Vdash \mathsf{tr}(\psi), \quad \text{then} \quad g(\mathfrak{M}), n \Vdash \psi. \tag{12}$$

We conduct the proofs of both implications by induction on the complexity of ψ. Throughout the proof we use the definition for the $\Vdash$ relation for LTL taken from [11, Section 2.2].

Proof of (11). We omit the boolean cases since they are straightforward.

case 1: $\psi = X\chi$ Before we prove the implication in question, note that by the definition of $R_{\textcircled{\uparrow}}$ we have that for each $n \in \mathbb{N}$ $R_{\textcircled{\uparrow}}(\langle n+1, -179.5\rangle) = \langle n+2, -179.5\rangle$. Now, assume that $\mathcal{M}, n \Vdash X\chi$. By the definition of $\Vdash$ for the formulas preceded by the X-operator, we know that $\mathcal{M}, n+1 \Vdash \chi$. By the inductive hypothesis it means that $f(\mathcal{M}), \langle n+2, -179.5\rangle \Vdash \mathsf{tr}(\chi)$, which, in turn, yields $f(\mathcal{M}), \langle n+1, -179.5\rangle \Vdash \textcircled{\uparrow}\mathsf{tr}(\chi)$. Finally, by the definition of tr we obtain $f(\mathcal{M}), \langle n+1, -179.5\rangle \Vdash \mathsf{tr}(X\chi)$.

case 2: $\psi = Y\chi$ This case is analogous to the previous one with the proviso that n has to be greater than 0.

case 3: $\psi = G\chi$ Assume that $\mathcal{M}, n \Vdash G\chi$. By the definition of $\Vdash$, for all m such that $m > n$ we have that $\mathcal{M}, m \Vdash \chi$. By the inductive hypothesis, for every such m we obtain $f(\mathcal{M}), \langle m+1, -179.5\rangle \Vdash \mathsf{tr}(\chi)$. Since for all $k \in \mathbb{N}$ we have $f(\mathcal{M}), \langle k+1, -179.5\rangle \Vdash \boxed{\leftarrow}\bot$ and, at the same time, $f(\mathcal{M}), \langle k+1, \theta\rangle \nVdash \boxed{\leftarrow}\bot$ for any $\theta \neq -179.5$, then, *a fortiori*, for each $m > n$ we have $f(\mathcal{M}), \langle m+1, -179.5\rangle \Vdash \boxed{\leftarrow}\bot \wedge \mathsf{tr}(\chi)$. Consequently, $f(\mathcal{M}), \langle n+1, -179.5\rangle \Vdash \boxed{\uparrow}(\boxed{\leftarrow}\bot \to \mathsf{tr}(\chi))$, hence $f(\mathcal{M}), \langle n+1, -179.5\rangle \Vdash \mathsf{tr}(G\chi)$.

case 4: $\psi = H\chi$ We handle this case in an analogical manner to the previous one with the proviso that for $n = 0$ all H-formulas are vacuously satisfied in 0 and all $\boxed{\downarrow}$-formulas are vacuously satisfied in $\langle 1, -179.5\rangle$, which automatically yields the proof for $n = 0$. ◁

Proof of (12). We omit the boolean cases since they are straightforward.

case 1: $\mathsf{tr}(\psi) = \mathsf{tr}(X\chi)$ Assume that $\mathfrak{M}, \langle n+1, -179.5\rangle \Vdash \mathsf{tr}(X\chi)$. By the definition of tr, $\mathfrak{M}, \langle n+1, -179.5\rangle \Vdash \textcircled{\uparrow}\mathsf{tr}(\chi)$ and further, by the definition of $\Vdash$, $\mathfrak{M}, \langle n+2, -179.5\rangle \Vdash \textcircled{\uparrow}\mathsf{tr}(\chi)$. By the inductive hypothesis we get $g(\mathfrak{M}), n+1 \Vdash \chi$, which, by the semantics of the X operator, yields $g(\mathfrak{M}), n \Vdash X\chi$.

case 2: $\mathsf{tr}(\psi) = \mathsf{tr}(Y\chi)$ This case is analogous to the previous one with the proviso that n has to be greater than 0.

case 3: $\mathsf{tr}(\psi) = \mathsf{tr}(G\chi)$ Assume that $\mathfrak{M}, \langle n+1, -179.5\rangle \Vdash \mathsf{tr}(G\chi)$. By the definition of tr, $\mathfrak{M}, \langle n+1, -179.5\rangle \Vdash \boxed{\uparrow}(\boxed{\leftarrow}\bot \to \mathsf{tr}(\chi))$. Since for all $k \in \mathbb{N}$ it holds that $\mathfrak{M}, \langle k+1, -179.5\rangle \Vdash \boxed{\leftarrow}\bot$, then by the definition of $R_{\textcircled{\uparrow}}$ we obtain $\mathfrak{M}, \langle m+1, -179.5\rangle \Vdash \mathsf{tr}(\chi)$, for each $m > n$. By the inductive hypothesis it follows that for each $m > n$ $g(\mathfrak{M}), m \Vdash \chi$, whence we get $g(\mathfrak{M}), n \Vdash G\chi$.

case 4: $\mathsf{tr}(\psi) = \mathsf{tr}(H\chi)$ We handle this case in an analogical manner to the previous one with the proviso that for $n = 0$ all H-formulas are vacuously satisfied in 0 and all $\boxed{\downarrow}$-formulas are vacuously satisfied in $\langle 1, -179.5\rangle$, which automatically yields the proof for $n = 0$. ◁

◀

▶ **Lemma 10.** *For every* SOSL*-formula* φ *the following statements are equivalent:*

1. *There exists a* φ*-kernel* K *such that the language of* $G_{\varphi,K}$ *is non-empty;*
2. *There exists an* SOSL*-model* $\mathfrak{M}$ *such that* $\mathfrak{M}, \langle 0, 0\rangle \Vdash \varphi$.

Proof.

(1 ⇒ 2) To finish the proof of the left-to-right implication, we now show that for each $\psi \in \mathsf{ncl}(\varphi)$ the following implications hold:

(K) If $\psi \in K$, then $\mathfrak{M}, \langle 0, 0\rangle \Vdash \psi$;

(R) If $\psi \in R^r(\theta)$, then $\mathfrak{M}, \langle r, \theta\rangle \Vdash \psi$.

We prove them by simultaneous induction on the complexity of ψ. In the proof we leave aside the most straightforward cases, such as, e.g., the base case which follows directly from the definition of V.

case 1: $\psi = \neg\chi$

subcase 1(a): $\psi = \neg\Diamond\!\!\!\!\blacklozenge\chi$ or $\psi = \neg\Diamond\!\!\!\!\blacklozenge\chi$ or $\psi = \neg\Diamond\!\!\!\!\blacklozenge\chi$ or $\psi = \neg\Diamond\!\!\!\!\blacklozenge\chi$ **(K)** Suppose, for the sake of contradiction, that $\neg\Diamond\!\!\!\!\blacklozenge\chi \in K$ and $\mathfrak{M}, \langle 0,0\rangle \nVdash \neg\Diamond\!\!\!\!\blacklozenge\chi$. By Definition 2 it means that $\mathfrak{M}, \langle 0,0\rangle \Vdash \Diamond\!\!\!\!\blacklozenge\chi$, thus for some $\langle r,\theta\rangle \in \mathcal{C}$ such that $\langle 0,0\rangle R_{\Diamond\!\!\!\!\blacklozenge}\langle r,\theta\rangle$ it holds that $\mathfrak{M}, \langle r,\theta\rangle \Vdash \chi$, and further, by Definition 2, that $\mathfrak{M}, \langle r,\theta\rangle \nVdash \neg\chi$. By the inductive hypothesis it follows that $\neg\chi \notin R^r(\theta)$ and by the fact that $R^r(\theta)$ satisfies (max) we have $\chi \in R^r(\theta)$. Since $\mathfrak{R}$ is an accepting run, by Definition 6 (2) we know that for every $n \in \mathbb{N}_+$, $\neg\Diamond\!\!\!\!\blacklozenge\chi \in R^n(\diamond)$. In particular, $\neg\Diamond\!\!\!\!\blacklozenge\chi \in R^r(\diamond)$. By Definition 6 (2) it follows that $\neg\chi \in R^r(\theta)$, which gives a contradiction with the fact that $R^r(\theta)$ satisfies (con). The proof of the remaining cases proceeds analogically.

(R) Assume that $\neg\Diamond\!\!\!\!\blacklozenge\chi \in R^r(\theta)$. By the definition of $R_{\Diamond\!\!\!\!\blacklozenge}$, $\langle r,\theta\rangle$ has no $R_{\Diamond\!\!\!\!\blacklozenge}$-successors, hence, by Definition 2, $\mathfrak{M}, \langle r,\theta\rangle \Vdash \neg\Diamond\!\!\!\!\blacklozenge\chi$. The proof of the remaining cases proceeds analogically.

subcase 1(b): $\psi = \neg\Diamond\!\!\!\!\blacklozenge\chi$ **(K)** Assume that $\neg\Diamond\!\!\!\!\blacklozenge\chi \in K$. By the definition of $R_{\Diamond\!\!\!\!\blacklozenge}$, $\langle 0,0\rangle$ has no $R_{\Diamond\!\!\!\!\blacklozenge}$-successors, thus by Definition 2, $\mathfrak{M}, \langle 0,0\rangle \Vdash \neg\Diamond\!\!\!\!\blacklozenge\chi$.

(R) Assume that $\neg\Diamond\!\!\!\!\blacklozenge\chi \in R^r(\theta)$. By Definition 5, $R^r(\theta)$ satisfies (con), i.e., $\Diamond\!\!\!\!\blacklozenge\chi \notin R^r(\theta)$. Since $\mathfrak{R}$ is an accepting run, R^r and R^{r+1} satisfy the conditions from Definition 8. By the third condition of this definition we obtain $\chi \notin R^{r+1}(\theta-1)$ and $\chi \notin R^{r+1}(\theta)$ and $\chi \notin R^{r+1}(\theta+1)$ and $\Diamond\!\!\!\!\blacklozenge\chi \notin R^{r+1}(\theta-1)$ and $\Diamond\!\!\!\!\blacklozenge\chi \notin R^{r+1}(\theta)$, and $\Diamond\!\!\!\!\blacklozenge\chi \notin R^{r+1}(\theta+1)$. By the fact that R^{r+1} satisfies (max) we further get $\neg\chi \in R^{r+1}(\theta-1)$ and $\neg\chi \in R^{r+1}(\theta)$ and $\neg\chi \in R^{r+1}(\theta+1)$ and $\neg\Diamond\!\!\!\!\blacklozenge\chi \in R^{r+1}(\theta-1)$ and $\neg\Diamond\!\!\!\!\blacklozenge\chi \in R^{r+1}(\theta)$, and $\neg\Diamond\!\!\!\!\blacklozenge\chi \in R^{r+1}(\theta+1)$. By the inductive hypothesis it follows that $\mathfrak{M}, \langle r+1,\theta-1\rangle \Vdash \neg\chi$ and $\mathfrak{M}, \langle r+1,\theta\rangle \Vdash \neg\chi$, and $\mathfrak{M}, \langle r+1,\theta+1\rangle \Vdash \neg\chi$. With $\neg\Diamond\!\!\!\!\blacklozenge\chi \in R^{r+1}(\theta-1)$ and $\neg\Diamond\!\!\!\!\blacklozenge\chi \in R^{r+1}(\theta)$, and $\neg\Diamond\!\!\!\!\blacklozenge\chi \in R^{r+1}(\theta+1)$ we repeat the reasoning for this case. It follows that for all $n > 0$ and all $\gamma \in$ ang such that $\gamma = \theta \pm m$, where $0 \leq m \leq n$, we have $\neg\chi \in R^{r+n}(\gamma)$. From the definition of $R_{\Diamond\!\!\!\!\blacklozenge}$ we know that $\langle r,\theta\rangle R_{\Diamond\!\!\!\!\blacklozenge}\langle r',\theta'\rangle$ for all $r' \in \mathbb{N}_+$ and $\theta' \in$ ang such that $r < r'$ and $|\theta - \theta'| \leq |r - r'|$. We have just shown that for all such $\langle r',\theta'\rangle$ it holds that $\mathfrak{M}, \langle r',\theta'\rangle \Vdash \neg\chi$, and so by Definition 2 we get $\mathfrak{M}, \langle r,\theta\rangle \Vdash \neg\Diamond\!\!\!\!\blacklozenge\chi$.

case 2: $\psi = \Diamond\!\!\!\!\blacklozenge\chi$ or $\psi = \Diamond\!\!\!\!\blacklozenge\chi$ or $\psi = \Diamond\!\!\!\!\blacklozenge\chi$ or $\psi = \Diamond\!\!\!\!\blacklozenge\chi$ **(K)** Suppose towards a contradiction that $\Diamond\!\!\!\!\blacklozenge\chi \in K$ and $\mathfrak{M}, \langle 0,0\rangle \nVdash \Diamond\!\!\!\!\blacklozenge\chi$, which, by Definition 2, means that $\mathfrak{M}, \langle 0,0\rangle \Vdash \neg\Diamond\!\!\!\!\blacklozenge\chi$. It follows that there is no $\langle r,\theta\rangle \in \mathcal{C}$ such that $\langle 0,0\rangle R_{\Diamond\!\!\!\!\blacklozenge}\langle r,\theta\rangle$ and $\mathfrak{M}, \langle r,\theta\rangle \Vdash \chi$, so for all $\langle r,\theta\rangle$ such that $\langle 0,0\rangle R_{\Diamond\!\!\!\!\blacklozenge}\langle r,\theta\rangle$ it holds that $\mathfrak{M}, \langle r,\theta\rangle \nVdash \chi$. By the inductive hypothesis we obtain that for all $n \in \mathbb{N}_+$ and all $\theta \in$ ang, $\chi \notin R^n(\theta)$. By Definition 6 (1) it means that for all $n \in \mathbb{N}_+$, $\Diamond\!\!\!\!\blacklozenge\chi \in R^n(\diamond)$. Thus, no element from $\mathcal{F}_{\Diamond\!\!\!\!\blacklozenge\psi}$ ever occurs during the run $\mathfrak{R}$, and so this is not an accepting run, which yields a contradiction. The remaining cases are proven analogously.

(R) By Definition 5 (5) neither of these formulas: $\Diamond\!\!\!\!\blacklozenge\chi$, $\Diamond\!\!\!\!\blacklozenge\chi$, $\Diamond\!\!\!\!\blacklozenge\chi$, $\Diamond\!\!\!\!\blacklozenge\chi$ is included in $R^r(\theta)$, so the implication is vacuously satisfied for this case.

case 3: $\psi = \ominus\chi$ or $\psi = \oplus\chi$ **(K)** By Definition 4 (1) neither of these formulas: $\ominus\chi$, $\oplus\chi$ is included in K, so the implication is vacuously satisfied for this case.

(R) Assume that $\ominus\chi \in R^r(\theta)$. By Definition 5 (1), $\chi \in R^r(\theta-1)$. By the inductive hypothesis we get $\mathfrak{M}, \langle r,\theta-1\rangle \Vdash \chi$. From the definition of $R_{\ominus}$ it follows that $\langle r,\theta\rangle R_{\ominus}\langle r,\theta-1\rangle$, hence by Definition 2 we get $\mathfrak{M}, \langle r,\theta\rangle \Vdash \ominus\chi$. The proof of the other case proceeds analogically.

case 4: $\psi = \uparrow\chi$ **(K)** By Definition 4 (1) $\uparrow\chi \notin K$, so the implication is vacuously satisfied for this case.

(R) Assume that $\Uparrow\chi \in R^r(\theta)$. Since $\mathfrak{R}$ is an accepting run, R^r and R^{r+1} satisfy the conditions from Definition 8. By the first condition of this definition we obtain $\chi \in R^{r+1}(\theta)$. By the inductive hypothesis it follows that $\mathfrak{M}, \langle r+1, \theta\rangle \Vdash \chi$. From the definition of $R_{\Uparrow}$ we know that $\langle r, \theta\rangle R_{\Uparrow}\langle r+1, \theta\rangle$, and so by Definition 2 we get $\mathfrak{M}, \langle r, \theta\rangle \Vdash \Uparrow\chi$.

case 5: $\psi = \Downarrow\chi$ **(K)** By Definition 4 (1), $\Downarrow\chi \notin K$, so the implication is vacuously satisfied for this case.

(R) Assume that $\Downarrow\chi \in R^r(\theta)$. Two cases need to be considered. 1) $r = 1$ Since $\mathfrak{R}$ is an accepting run, the kernel K and R^1 satisfy the conditions from Definition 7. By the second condition of this definition we obtain $\chi \in K$. By the inductive hypothesis it follows that $\mathfrak{M}, \langle 0, 0\rangle \Vdash \chi$. From the definition of $R_{\Downarrow}$ we know that $\langle 1, \theta\rangle R_{\Downarrow}\langle 0, 0\rangle$, hence, Definition 2 we get $\mathfrak{M}, \langle r, \theta\rangle \Vdash \Downarrow\chi$. 2) $r > 1$ Since $\mathfrak{R}$ is an accepting run, R^{r-1} and R^r satisfy the conditions from Definition 8. By the second condition of this definition we obtain $\chi \in R^{r-1}(\theta)$. By the inductive hypothesis it follows that $\mathfrak{M}, \langle r-1, \theta\rangle \Vdash \chi$. From the definition of $R_{\Downarrow}$ we know that $\langle r, \theta\rangle R_{\Downarrow}\langle r-1, \theta\rangle$, and so by Definition 2 we get $\mathfrak{M}, \langle r, \theta\rangle \Vdash \Downarrow\chi$.

case 6: $\psi = \Updownarrow\chi$ or $\psi = \Updownarrow\chi$ **(K)** By Definition 4 (1), $\Updownarrow\chi \notin K$ and $\Updownarrow\chi \notin K$, so the implication is vacuously satisfied for this case.

(R) Assume that $\Updownarrow\chi \in R^r(\theta)$. By Definition 5 (3) there exist $n > 0$ and $m < n$ such that $\Uparrow^m\chi \in R(\theta - n)$ or $\Downarrow^m\chi \in R(\theta - n)$. Without loss of generality suppose that the former is the case. By applying Definition 8 (1) m times we obtain $\chi \in R^{r+m}(\theta)$. By the inductive hypothesis we obtain $\mathfrak{M}, \langle r+m, \theta-n\rangle \Vdash \chi$. By the definition of $R_{\Updownarrow}$ it holds that $\langle r, \theta\rangle R_{\Updownarrow}\langle r+m, \theta-n\rangle$, so by Definition 2 we obtain $\mathfrak{M}, \langle r, \theta\rangle \Vdash \Updownarrow\chi$. The proof for the other case proceeds analogically.

case 7: $\psi = \Updownarrow\chi$ **(K)** By Definition 4 (1) $\Updownarrow\chi \notin K$, so the implication is vacuously satisfied for this case.

(R) Assume that $\Updownarrow\chi \in R^r(\theta)$. Two cases need to be considered. 1) $r = 1$ Since $\mathfrak{R}$ is an accepting run, then K and $R^1(\theta)$ satisfy the conditions from Definition 7. By the third condition of this definition $\Updownarrow\chi \notin R^1(\theta)$, so the implication is vacuously satisfied for this case. 2) $r > 1$. Since $\mathfrak{R}$ is an accepting run, R^{r-1} and R^r satisfy the conditions from Definition 8. By the fourth condition of this definition we obtain $\chi \in R^{r-1}(\theta-1)$ or $\chi \in R^{r-1}(\theta)$ or $\chi \in R^{r-1}(\theta+1)$ or $\Updownarrow\chi \in R^{r-1}(\theta-1)$ or $\Updownarrow\chi \in R^{r-1}(\theta)$, or $\Updownarrow\chi \in R^{r-1}(\theta+1)$. If the first disjunct holds, then by the inductive hypothesis we have $\mathfrak{M}, \langle r-1, \theta-1\rangle \Vdash \chi$. By the definition of $R_{\Updownarrow}$ it follows that $\langle r, \theta\rangle R_{\Updownarrow}\langle r-1, \theta-1\rangle$, so by Definition 2 would get $\mathfrak{M}, \langle r, \theta\rangle \Vdash \Updownarrow\chi$. The argument is similar if the second or the third disjunct is true. If the fourth disjunct holds, i.e., $\Updownarrow\chi \in R^{r-1}(\theta-1)$, we proceed in the same way as for the case $\Updownarrow\chi \in R^r(\theta)$. Since by Definition 7 (3), for all $\theta \in \mathsf{ang}$, $\Updownarrow\chi \notin R^1(\theta)$, it follows that there must exist $0 < n < r$ and $0 < m < n$, such that $\theta - m \in \mathsf{ang}$ and $\chi \in R^{r-n}(\theta - m)$ or $\theta + m \in \mathsf{ang}$ and $\chi \in R^{r-n}(\theta + m)$. Without loss of generality assume that the former is the case. By the inductive hypothesis, $\mathfrak{M}, \langle r-n, \theta-m\rangle \Vdash \chi$. From the definition of $R_{\Updownarrow}$ we know that $\langle r, \theta\rangle R_{\Updownarrow}\langle r-n, \theta-m\rangle$, hence, by Definition 2, we get $\mathfrak{M}, \langle r, \theta\rangle \Vdash \Updownarrow\chi$.

case 8: $\psi = \Updownarrow\chi$ **(K)** By Definition 4 (1) $\Updownarrow\chi \notin K$, so the implication is vacuously satisfied for this case.

(R) For the sake of contradiction suppose that $\Updownarrow\chi \in R^r(\theta)$ and $\mathfrak{M}, \langle r, \theta\rangle \nVdash \Updownarrow\chi$. By Definition 2 it means that for all $\langle r', \theta'\rangle$ such that $\langle r, \theta\rangle R_{\Updownarrow}\langle r', \theta'\rangle$ it holds that $\mathfrak{M}, \langle r', \theta'\rangle \nVdash \chi$. By the inductive hypothesis for all such r' and θ' we have $\chi \notin R^{r'}(\theta')$. (Note that by the definition of $R_{\Updownarrow}$ there exists $n_0 \in \mathbb{N}_+$, $n_0 \leq 360$, such that for any $m \geq n_0$ and any $\theta' \in \mathsf{ang}$ we have $\langle r, \theta\rangle R_{\Updownarrow}\langle r+m, \theta'\rangle$, and thus $\psi \notin R^{r+m}(\theta')$.)

Since $\mathfrak{R}$ is an accepting run, then by Definition 8 (3), for each $n \in \mathbb{N}_+$ there exists $\theta \in \mathsf{ang}$ such that $\diamondsuit\!\!\!\!\uparrow\chi \in R^{r+n}(\theta)$. Indeed, since by the assumption neither of the first three cases of Definition 8 (3) holds, then one of the last three cases must hold. This observation applies to subsequent rings. Observe also that by Definition 8 (3), for each two consecutive R^n, R^{n+1} in $\mathfrak{R}$ if $\diamondsuit\!\!\!\!\uparrow\chi \in R^{n+1}(\theta)$ for some $\theta \in \mathsf{ang}$, then $\diamondsuit\!\!\!\!\uparrow\chi \in R^n(\theta-1)$, $\diamondsuit\!\!\!\!\uparrow\chi \in R^n(\theta)$, and $\diamondsuit\!\!\!\!\uparrow\chi \in R^n(\theta+1)$. Let $\mathsf{card}\left(R^n_\xi\right) = |\{\theta \in \mathsf{ang} \mid \xi \in R^n(\theta)\}|$. Note that for any $n \in \mathbb{N}_+$, if $1 \leq \mathsf{card}\left(R^{n+1}_{\diamondsuit\!\!\!\!\uparrow\chi}\right) < 360$, then $\mathsf{card}\left(R^{n+1}_{\diamondsuit\!\!\!\!\uparrow\chi}\right) < \mathsf{card}\left(R^{n}_{\diamondsuit\!\!\!\!\uparrow\chi}\right)$. A direct consequence of this fact is that if for some $n \in \mathbb{N}_+$ and $\theta \in \mathsf{ang}$, $\diamondsuit\!\!\!\!\uparrow\chi \in R^{n+360}(\theta)$, then for all $\gamma \in \mathsf{ang}$ $\diamondsuit\!\!\!\!\uparrow\chi \in R^n(\gamma)$. Recall that by the assumption, for each $n \geq r$ there exists $\theta \in \mathsf{ang}$ such that $\diamondsuit\!\!\!\!\uparrow\chi \in R^n(\theta)$, which means that for all $n \in \mathbb{N}_+$ there exists $\theta \in \mathsf{ang}$ such that $\diamondsuit\!\!\!\!\uparrow\chi \in R^{n+360}(\theta)$, and thus for all $\theta \in \mathsf{ang}$ it holds that $\diamondsuit\!\!\!\!\uparrow\chi \in R^n(\theta)$. Consequently, for all $m \geq n_0$ and all $\theta \in \mathsf{ang}$ we have $\diamondsuit\!\!\!\!\uparrow\chi \in R^m(\theta)$ and $\chi \notin R^{n_0}(\theta)$. It means that for each $\theta \in \mathsf{ang}$, each final state from $\mathcal{F}^\theta_{\diamondsuit\!\!\!\!\uparrow\chi}$ occurs at most finitely many times during $\mathfrak{R}$ (precisely, at most n_0 times), which contradicts the assumption that $\mathfrak{R}$ is an accepting run.

(1 $\Leftarrow$ 2) To finish the proof of the right-to-left implication it remains to show that $\mathfrak{R}$ is an accepting run of $G_{\varphi,K}$ on w, i.e., that the following hold:

1. $R^1 \in Q_0$;
2. for all $r \in \mathbb{N}_+$ we have $R^{r+1} \in \delta(R^r, \sigma^{r+1})$;
3. for every set in $\mathcal{F}$ there are infinitely many elements of $\mathfrak{R}$ belonging to this set.

Ad. 1 We need to check whether R^1 is a successor of the kernel K, i.e., whether the conditions from Definitions 6 and 7 are satisfied. We will start from the former.

Condition 1. Assume that $\diamondsuit\!\!\!\!\blacktriangle\psi \in K$. By the definition of K it means that $\mathfrak{M}, \langle 0,0\rangle \Vdash \diamondsuit\!\!\!\!\blacktriangle\psi$. If there exists no $\theta \in \mathsf{ang}$ such that $\langle 0,0\rangle R_{\diamondsuit\!\!\!\!\blacktriangle}\langle 1,\theta\rangle$ and $\mathfrak{M}, \langle 1,\theta\rangle \Vdash \psi$, then by the definition of $R^r(\diamond)$ we obtain that $\diamondsuit\!\!\!\!\blacktriangle\psi \in R^1(\diamond)$.

Condition 2. Assume that $\neg\diamondsuit\!\!\!\!\blacktriangle\psi \in K$. By the definition of K it means that $\mathfrak{M}, \langle 0,0\rangle \Vdash \neg\diamondsuit\!\!\!\!\blacktriangle\psi$. Since $\mathfrak{M}$ is a model, then by Definition 2 we know that for all $\theta \in \mathsf{ang}$ such that $\langle 0,0\rangle R_{\diamondsuit\!\!\!\!\blacktriangle}\langle 1,\theta\rangle$ it holds that $\mathfrak{M}, \langle 1,\theta\rangle \Vdash \neg\psi$. Therefore, by the definition of $R^r(\theta)$ we obtain that $\neg\psi \in R^1(\theta)$ for all $\theta \in \mathsf{ang}$. Moreover, by the definition of $R^r(\diamond)$ we get $\neg\diamondsuit\!\!\!\!\blacktriangle\psi \in R^1(\diamond)$.

Now let's proceed to the conditions from Definition 7.

Condition 1. Assume that $\uparrow\!\!\!\!\uparrow\psi \in K$. By the definition of K it means that $\mathfrak{M}, \langle 0,0\rangle \Vdash \uparrow\!\!\!\!\uparrow\psi$. Since $\mathfrak{M}$ is a model, then by Definition 2 we know that there exists $\theta \in \mathsf{ang}$ such that $\mathfrak{M}, \langle 1,\theta\rangle \Vdash \psi$. By the definition of $R^r(\theta)$ we infer that $\psi \in R^1(\theta)$.

Condition 2. Assume that $\psi \in K$. By the definition of K it means that $\mathfrak{M}, \langle 0,0\rangle \Vdash \psi$. By the definition of $R_{\downarrow\!\!\!\!\downarrow}$ we know that for all $\theta \in \mathsf{ang}$ it holds that $\langle 1,\theta\rangle R_{\downarrow\!\!\!\!\downarrow}\langle 0,0\rangle$. Since $\mathfrak{M}$ is a model, then by Definition 2 we know that for all $\theta \in \mathsf{ang}$ $\mathfrak{M}, \langle 1,\theta\rangle \Vdash \downarrow\!\!\!\!\downarrow\psi$. By the definition of $R^r(\theta)$ we infer that for all $\theta \in \mathsf{ang}$ $\downarrow\!\!\!\!\downarrow\psi \in R^1(\theta)$.

Condition 3. Observe that by the definition of $R_{\diamondsuit\!\!\!\!\downarrow}$, for all $\theta \in \mathsf{ang}$ cells of the form $\langle 1,\theta\rangle$ have no $R_{\diamondsuit\!\!\!\!\downarrow}$-successors. Since $\mathfrak{M}$ is a model, it follows that for all $\theta \in \mathsf{ang}$ $\mathfrak{M}, \langle 1,\theta\rangle \nVdash \diamondsuit\!\!\!\!\downarrow\psi$. By the definition of $R^r(\theta)$, for all $\theta \in \mathsf{ang}$ $\diamondsuit\!\!\!\!\downarrow\psi \notin R^1(\theta)$.

Ad. 2 We need to check if R^{r+1} is indeed a δ-successor of R^r on the word w. First observe that by the definition of $\sigma^r(\theta)$ and $R^r(\theta)$, for all $\theta \in \mathsf{ang}$, $\sigma^{r+1}(\theta) = R^{r+1}(\theta) \cap \mathrm{PROP}(\varphi)$. Next, we need to verify whether the conditions from Definitions 6 and 8 are satisfied. The former are proven analogously to the same conditions for K, so let's proceed to the conditions from Definition 8.

Condition 1 and 2. These conditions are proven analogously to condition 2 from the previous point.

Condition 3. Assume that $\diamondsuit\!\!\uparrow\psi \in R^r(\theta)$ for some $\theta \in \mathsf{ang}$. From the definition of $R^r(\theta)$ we know that $\mathfrak{M}, \langle r, \theta\rangle \Vdash \diamondsuit\!\!\uparrow\psi$. Since $\mathfrak{M}$ is a model, then by Definition 2 there exist $r' \in \mathbb{N}_+$ and $\theta' \in \mathsf{ang}$ such that $\langle r, \theta\rangle R_{\diamondsuit\!\!\uparrow}\langle r', \theta'\rangle$ and $\mathfrak{M}, \langle r', \theta'\rangle \Vdash \psi$. By the definition of $R_{\diamondsuit\!\!\uparrow}$ we have $r' > r$ and $|\theta - \theta'| \leq |r - r'|$. If $r' = r + 1$, then $|\theta - \theta'| \leq 1$, so $\theta' = \theta - 1$ or $\theta' = \theta$ or $\theta' = \theta + 1$. In such a case, by the definition of $R^r(\theta)$, $\psi \in R^{r+1}(\theta - 1)$ or $\psi \in R^{r+1}(\theta)$ or $\psi \in R^{r+1}(\theta + 1)$. If $r' > r + 1$, then we need to consider 3 cases. 1) $\theta' - \theta = r' - r$. It follows that $\theta' - (\theta+1) = r' - (r+1)$, so by the definition of $R_{\diamondsuit\!\!\uparrow}$, $\langle r+1, \theta+1\rangle R_{\diamondsuit\!\!\uparrow}\langle r', \theta'\rangle$ and thus, by Definition 2, $\mathfrak{M}, \langle r + 1, \theta + 1\rangle \Vdash \diamondsuit\!\!\uparrow\psi$. Consequently, by the definition of $R^r(\theta)$, $\psi \in R^{r+1}(\theta + 1)$. 2) $|\theta' - \theta| < r' - r$. It follows that $\theta' - \theta \leq r' - (r + 1)$, so by the definition of $R_{\diamondsuit\!\!\uparrow}$ $\langle r + 1, \theta\rangle R_{\diamondsuit\!\!\uparrow}\langle r', \theta'\rangle$ and thus, by Definition 2, $\mathfrak{M}, \langle r + 1, \theta\rangle \Vdash \diamondsuit\!\!\uparrow\psi$. Consequently, by the definition of $R^r(\theta)$, $\psi \in R^{r+1}(\theta)$. 3) $\theta - \theta' = r' - r$. It follows that $(\theta - 1) - \theta' = r' - (r + 1)$, so by the definition of $R_{\diamondsuit\!\!\uparrow}$, $\langle r + 1, \theta - 1\rangle R_{\diamondsuit\!\!\uparrow}\langle r', \theta'\rangle$ and by Definition 2, $\mathfrak{M}, \langle r + 1, \theta - 1\rangle \Vdash \diamondsuit\!\!\uparrow\psi$. Hence, by the definition of $R^r(\theta)$, $\psi \in R^{r+1}(\theta - 1)$.

Condition 4. This condition is proven analogously to condition 3.

Ad. 3 First, let's consider a set $\mathcal{F}_{\diamondsuit\psi}$ for some $\diamondsuit\psi \in \mathsf{ncl}(\varphi)$. If $\diamondsuit\psi \notin K$, then, by the definition of $R(\diamond)$, for all $n \in \mathbb{N}_+$ we have $\diamondsuit\psi \notin R^n(\diamond)$. It means that for all $n \in \mathbb{N}_+$, $R^n \in \mathcal{F}_{\diamondsuit\psi}$, so elements of the set $\mathcal{F}_{\diamondsuit\psi}$ occur inifinitely many times during the run $\mathfrak{R}$. Assume, then, that $\diamondsuit\psi \in K$. By the definition of K it means that $\mathfrak{M}, \langle 0, 0\rangle \Vdash \diamondsuit\psi$. Since $\mathfrak{M}$ is a model, then by Definition 2 there exist the least $r \in \mathbb{N}_+$ and $\theta \in \mathsf{ang}$ such that $\langle 0, 0\rangle R_{\diamondsuit}\langle r, \theta\rangle$ and $\mathfrak{M}, \langle r, \theta\rangle \Vdash \psi$. By the definition of $R^r(\theta)$ we obtain $\psi \in R^r(\theta)$. By the definition of $R^r(\diamond)$ it follows that for all $n \geq r$, $\diamondsuit\psi \notin R^n(\diamond)$. Consequently, for all $n \geq r$ $R^n \in \mathcal{F}_{\diamondsuit}$, so infinitely many elements of $\mathfrak{R}$ belong to $\mathcal{F}_{\diamondsuit}$.

Now, let's consider $\mathcal{F}^{\theta}_{\diamondsuit\!\!\uparrow\psi}$. If there exists no $r \in \mathbb{N}_+$ such that $\diamondsuit\!\!\uparrow\psi \in R^r(\theta)$, then for all $n \in \mathbb{N}_+$ we have $\diamondsuit\!\!\uparrow\psi \notin R^n(\theta)$, so for all $n \in \mathbb{N}_+$, $R^n \in \mathcal{F}^{\theta}_{\diamondsuit\!\!\uparrow\psi}$. Consequently, elements of the set $\mathcal{F}^{\theta}_{\diamondsuit\!\!\uparrow\psi}$ occur infinitely many times during the run. Assume, then, that there exists $r \in \mathbb{N}_+$ such that $\diamondsuit\!\!\uparrow\psi \in R^r(\theta)$. By the definition of $R^r(\theta)$ it means that $\mathfrak{M}, \langle r, \theta\rangle \Vdash \diamondsuit\!\!\uparrow\psi$. Since $\mathfrak{M}$ is a model, then by Definition 2 there exist $r' \in \mathbb{N}_+$ and $\theta' \in \mathsf{ang}$ such that $\langle r, \theta\rangle R_{\diamondsuit\!\!\uparrow}\langle r', \theta'\rangle$ and $\mathfrak{M}, \langle r', \theta'\rangle \Vdash \psi$. By the definition of $R^r(\theta)$ we obtain $\psi \in R^{r'}(\theta')$. There are either finitely, or infinitely many $n \in \mathbb{N}_+$ such that $\diamondsuit\!\!\uparrow\psi \in R^n(\theta)$. If the former is the case, it means that $|\{n \in \mathbb{N}_+ \mid \diamondsuit\!\!\uparrow\psi \notin R^n(\theta)\}| = \infty$. Since $\{R^n \mid \diamondsuit\!\!\uparrow\psi \notin R^n(\theta)\} \subseteq \mathcal{F}^{\theta}_{\diamondsuit\!\!\uparrow\psi}$, infinitely many elements of $\mathfrak{R}$ belong to $\mathcal{F}^{\theta}_{\diamondsuit\!\!\uparrow\psi}$. If, on the other hand, there are infinitely many $n \in \mathbb{N}_+$ such that $\diamondsuit\!\!\uparrow\psi \in R^n(\theta)$, then by the earlier argument, for each $n \in \mathbb{N}_+$ such that $\diamondsuit\!\!\uparrow\psi \in R^n(\theta)$ there exists $m > n$ and $\theta' \in \mathsf{ang}$ such that $\psi \in R^m(\theta')$. By the unboundedness of $\mathbb{N}_+$ we get $|\{m \in \mathbb{N}_+ \mid \psi \in R^m(\theta')$ for some $\theta' \in \mathsf{ang}\}| = \infty$. Since $\{R^m \mid \psi \in R^m(\theta')$ for some $\theta' \in \mathsf{ang}\} \subseteq \mathcal{F}^{\theta}_{\diamondsuit\!\!\uparrow\psi}$, infinitely many elements of $\mathfrak{R}$ are from $\mathcal{F}^{\theta}_{\diamondsuit\!\!\uparrow\psi}$. ◀

Customizing BPMN Diagrams Using Timelines

Carlo Combi
Department of Computer Science, University of Verona, Italy
carlo.combi@univr.it

Barbara Oliboni
Department of Computer Science, University of Verona, Italy
barbara.oliboni@univr.it

Pietro Sala
Department of Computer Science, University of Verona, Italy
pietro.sala@univr.it

Abstract

BPMN (Business Process Model and Notation) is widely used standard modeling technique for representing Business Processes by using diagrams, but lacks in some aspects. Representing execution-dependent and time-dependent decisions in BPMN Diagrams may be a daunting challenge [11]. In many cases such constraints are omitted in order to preserve the simplicity and the readability of the process model. However, for purposes such as compliance checking, process mining, and verification, formalizing such constraints could be very useful. In this paper, we propose a novel approach for annotating BPMN Diagrams with *Temporal Synchronization Rules* borrowed from the timeline-based planning field. We discuss the expressivity of the proposed approach and show that it is able to capture a lot of complex temporally-related constraints without affecting the structure of BPMN diagrams. Finally, we provide a mapping from annotated BPMN diagrams to timeline-based planning problems that allows one to take advantage of the last twenty years of theoretical and practical developments in the field.

2012 ACM Subject Classification Applied computing → Business process modeling; Applied computing

Keywords and phrases Business Processes, BPMN, Timelines, Temporal Constraints

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.5

Related Version A full version of the paper is available at `http://profs.scienze.univr.it/~sala/BPMNTimelinesExtended.pdf`.

Funding *Pietro Sala*: acknowledges the financial support from the Italian INdAM-GNCS project 2019 "Formal methods for combined verification".

1 Introduction

Nowadays Process-Aware Information Systems (PAISs) have become the cornerstone for organizing activities in most realities, ranging from large private companies (operating in logistics, manufactoring, avionics, and so on) to healthcare institutions [26]. Business Process Management deals with many important aspects such as analysis, modeling, execution, and monitoring of Business Processes [21].

In this context, BPMN (Business Process Model and Notation) [28] is the standard for representing and managing business processes, but it lacks in some aspects such as the specification of (i) temporal constraints [11, 29], (ii) resources availability [12], and (iii) external data affecting decisions [31].

As pointed out by many applications, time-awareness is a crucial property of business processes in most domains and especially in the healthcare one [20, 29]. However, BPMN does not directly allow the specification of time constraints in process diagrams, despite the fact that they affect the real process flow in many aspects such as choices to be made at given

decision points, event handling, task durations, resource allocation and so on. This limitation has been considered in the literature in different ways. A possible approach is to extend BPMN with constructs borrowed from workflows and simple temporal networks fields [11, 29]. Another approach consists of translating BPMN diagrams into logical or automata-based formalisms [13, 22] and then expressing constraints by means of the considered formalisms.

Moreover, BPMN does not allow the representation of resource availability and external data affecting decisions, even if these aspects are crucial in managing process execution and outcome [11]. A further issue to be considered is that resources and data values change over time. As an example, in the healthcare domain, resource availability with respect to blood analysis may be affected by the time of the day (i.e., morning, afternoon, evening, and night) and the current load of the lab (i.e., the number of undergoing analyses). Time of the day and current load may influence the whole time required for getting results of blood tests. An example taking into account external data affecting decisions is related to shifts in the systolic blood pressure values of a patient undergoing a surgical procedure. Significant differences in pressure values in last 5 hours may force the anaesthetist to administer a local sedation in place of a total one for safety reasons [12].

For the sake of clarity and conciseness of BPMN diagrams, often the formal specification of these aspects are intentionally neglected and left to the following implementation by specific software tools.

In this paper, we propose an approach, residing in between the two aforementioned ones, that allows the annotation of BPMN diagrams based on *Temporal Synchronization Rules* of *Timeline-based planning* [27]. We also show that this simple language may naturally express the specification of (i) temporal constraints, (ii) resource availability, and (iii) external data affecting decisions. Moreover, the proposed approach allows one to constrain the execution of the process (e.g., a decision in an exclusive-gateway) according to the aforementioned specifications. Then, another contribution of this work is a complete mapping of our timeline-annotated BPMN diagrams into a timeline-based planning problem, that is, given a set of state variables and a set of synchronization rules on them, find a consistent execution where all the synchronization rules are satisfied [27]. The translation step suffices for our verification purpose, since various tools for satisfiability of timeline-based planning have been proposed in the last decade [2, 5, 6].

The advantages of our proposal are manyfold:

1. The proposed approach allows us to express complex temporal constraints even if they involve some external data or resources.
2. The temporal behaviour of data and resources may be regulated with the same machinery (i.e., state variables).
3. Our approach allows composability. As a matter of fact, resources/data may be updated/removed/inserted, as well as temporal constraints on the execution of the business process, by simply modifying the relative temporal synchronization rules/state variables.
4. The process diagram is not affected at all and it may be seen through a layered perspective: (a) at the highest level, the original BPMN diagram provides a general idea of how activities are organized; (b) at an intermediate level, temporal synchronization rules, possibly involving one or more external entities, detail how the execution is temporally constrained and how some decision points are affected by (the temporal evolution of) data/resources and/or by some previous temporal behavior of the process; (c) finally, at the lowest level, the state variables regulate the evolutions of the involved data/resources.

The power and generality of this approach come at the price that the definition of a set of temporal constraints in the form of temporal synchronization rules and state variables could be inconsistent (i.e., every possible execution of the diagram combined with every possible consistent evolution of data/resources violates at least one temporal synchronization rule).

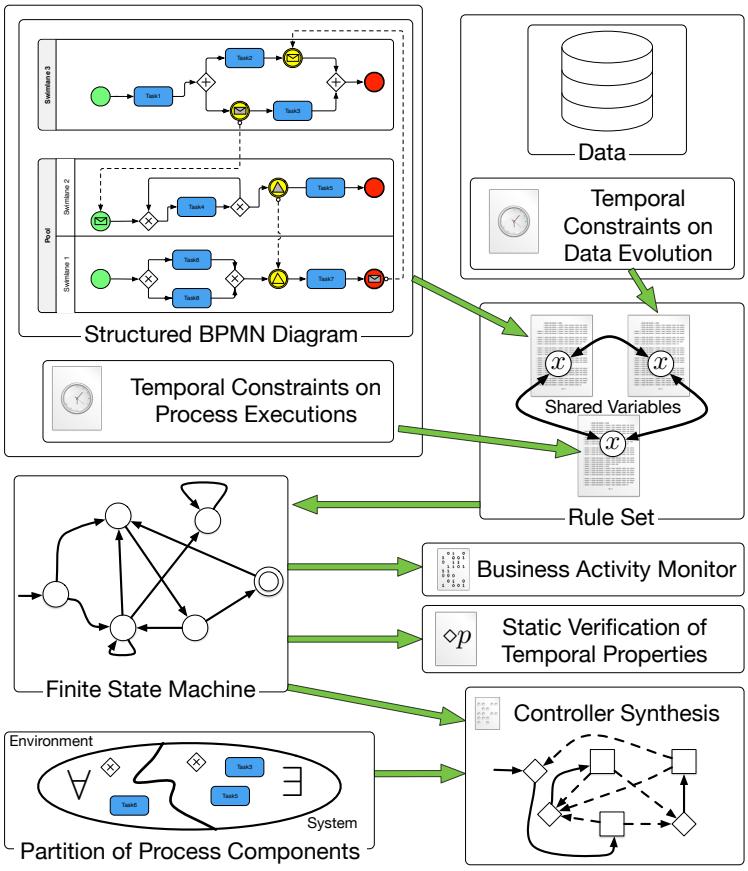


Figure 1 A pipeline for integrating timeline-based planning and BPMN diagrams.

In this paper, we will focus only on structured BPMN diagrams, and thus from now on we will call them just diagrams. A diagram is said to be well-structured if every node with multiple outgoing edges, i.e., a split node, has a corresponding node with multiple incoming edges, i.e., a join node, such that the set of nodes delimited by the split and the join nodes form a Single-Entry-Single-Exit (SESE) region, and these regions within the process are properly nested [15, 19]. In this way a SESE region is any area within a process delimited by a single entry edge and a single exit edge.

The paper is organized as follows. Sec. 2 gives an overall description of the proposed approach. Sec. 3 provides an example of a real-world process in the healthcare domain, which features non-trivial temporal constraints. Sec. 4 recalls the basic concepts and notation of timelines and timeline-based planning, together with some recent results in the field. Sec. 5 shows some meaningful temporal constraints that may be enforced by means of timeline annotations in BPMN diagrams in a straightforward way, without compromising the overall readability of the diagram. Sec. 6 describes how the proposed approach allows the specification of constraints involving data, resources, and decisions. Sec. 7 summarizes the contribution of the paper and sketches some lines for future work.

2 Enriching BPMN with Timelines: the Big Picture

In this section we give an overview of the proposed approach, which is graphically summarized in Fig. 1. BPMN diagrams are often used for modelling business processes, where there are often time-critical, resource-critical, and data-critical situations. Usually business processes

are underspecified w.r.t. such requirements for preserving their readability and conciseness. In this simplified form they cannot be directly managed by a suitable algorithm for controlling the whole process at runtime and/or for performing qualitative/quantitative static analysis. On the other hand, forcing the representation of such requirements by enriching the diagram will compromise the readability of the diagram itself.

In order to overcome such trade-off, our proposal consists of keeping the original diagram and annotating it by using a set of constraints, namely *temporal synchronization rules*, borrowed from the timeline-based planning domain. As we will detail in Sec. 5, such rules are able to express in a concise and clear way temporal constraints that would otherwise be captured by a complex combination of throw/catch events and event-based gateways [11]. In [9], we will provide a way to translate structured BPMN processes into a set of rules representing all and only the possible correct executions of the process. Such mapping is crucial because, as shown in Fig. 1, it allows the representation of both requirements and process as a set of rules.

In Fig. 1, we suppose to have a process that makes use of some data, and constraints on such data must be taken into account. For instance, let us assume that the diagram represents a medical guideline in which the decision on the exclusive gateway is driven by the value of the patient blood pressure. It is straightforward to see that such value cannot increase/decrease too fast in a short amount of time and it would be desirable to force such constraint in order to prune irrealistic behaviors of the process in subsequent analysis. In this paper, we assume that constraints on data may be captured by a suitable set of temporal synchronization rules. As shown in Fig. 1, as a first step, the diagram, temporal constraints, and data constraints are translated into sets of rules in an independent manner. The union of such obtained rules (the Rule Set in Fig. 1) represents the whole description of the considered process. As mentioned before, the translations are pairwise independent but, as we will observe in Sec. 6, rules in different sets may "communicate" via shared variables. For instance, rules representing the diagram may involve the variable representing the pressure, whose behaviour is encoded by other rules coming from data constraints. Another example may be represented by the fact that a given temporal constraint imposes that the execution of two specific tasks must be non-overlapping (since they use the same shared resource), no matter how they are arranged in the diagram (e.g., they may appear in parallel branches). It is easy to see that such approach fosters modularity in the design of every component. As a matter of fact, we may change constraints on the behavior of data, without affecting the diagram, or we may change the diagram without impacting on related temporal constraints.

As depicted in Fig. 1, the whole set of rules is translated into a Finite State Machine (FSM), whose language represents all the possible correct executions of the considered diagram w.r.t. to temporal/data constraints. The FSM may be used for performing a plethora of process-related analyses. In Fig. 1, we just provide three of them. (i) FSM may be translated into an algorithm that may be used at runtime for monitoring the correct execution of the process by means of alerts/execeptions pointing out the violation of a given constraint [16]. (ii) On the FSM we may perform static verification of qualitative/quantitative properties, expressed in temporal logics such as LTL or CTL [18], by using one of the many well-established tools available [8, 17]. (iii) Supposing to be in a scenario where some process elements are under the control of the environment (e.g., medical guidelines). By means of the FSM we may synthesize, if it exists, a controller that "drives" the system-controlled elements (i.e., the process elements which are not controlled by the environment) in a way that the correct termination of the process is ensured, no matter how the environment behaves on its set of elements [25, 30].

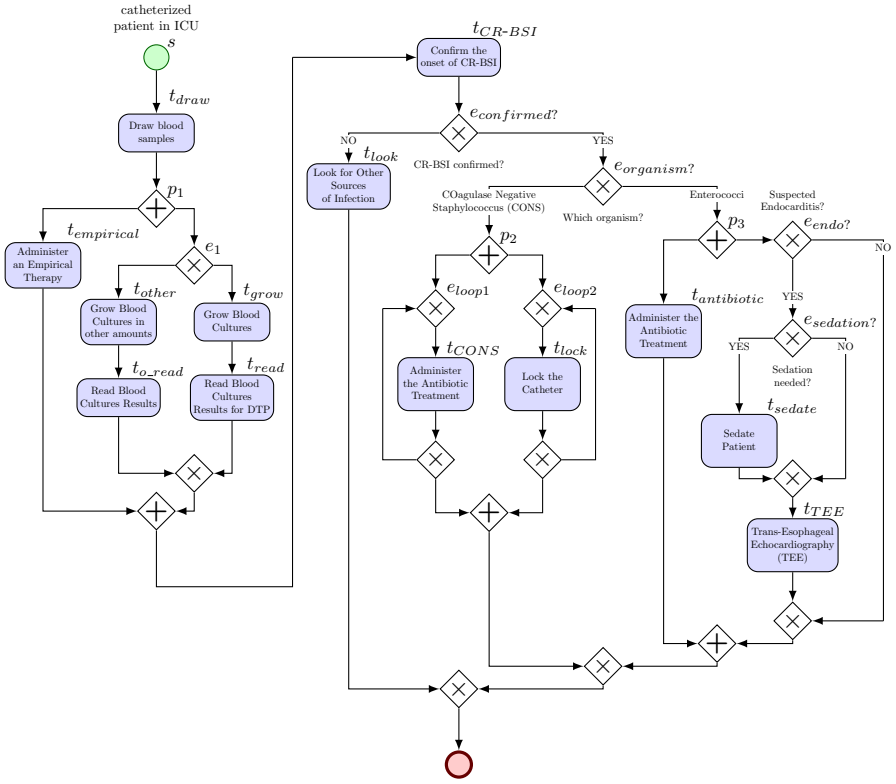


Figure 2 A BPMN Diagram representing CR-BSIs treatment.

3 A motivating example

In this section, we introduce a clinical process model and describe some time-related decisions and constrains that can be considered. The Business Process model, represented in Fig. 2 as a BPMN Diagram, is a process for the treatment of Catheter-Related Bloodstream Infections (CR-BSIs). Vascular catheters are vital for treating ill patients in critical situations. Their main drawback is represented by the concrete possibility of a patogens colonization of their injection site. This may lead patients to develop severe bloodstream infections.

Clinical guidelines for preventing such infections have been proposed and applied. Most of them usually rely on temporal constraints for their applicability [4]. The BPMN diagram in Fig. 2 shows the process for detecting and treating CR-BSIs according to the well-known Infectious Diseases Society of America (IDSA) practice guideline [23]. The guideline includes blood and/or catheter cultures activities for supporting the diagnosis of CR-BSI. In particular, clinicians first draw *simultaneously* two blood samples to be cultured, one from the catheter suspected to be the source of the infection and, the other, from a peripheral vein. We call the first sample LS (local sample) and the second one PS (peripheral sample), respectively. These operations are included in the first process activity of Fig. 2, i.e. *Draw blood samples*. The considered activity takes a t_{draw} time to be completed.

After the first activity, physicians *Administer an empirical therapy* to the patient until the diagnosis of CR-BSI is confirmed. Among the criteria for confirming or not a CR-BSI, we considered the Differential Time to Positivity (DTP), which measures the difference between the time when LS becomes positive w.r.t. a certain micro-organism, and the time when PS becomes positive for the same micro-organism. If such difference exceeds a certain threshold (DTP), then the CR-BSI is confirmed.

In the process of Fig. 2, we considered only two of the possible micro-organisms that may be detected in a CR-BSI infection: Coagulase-negative Staphylococci and Enterococcus spp.

- In case of *Coagulase-negative Staphylococci (CONS)*, the patient is treated with antibiotic or heparin lock therapy. Such therapy consists of alternating between catheter locks and an antimicrobial therapy. In general, such phases have equal duration in order to prevent clot formations. These activities are represented in Fig. 2 by means of the process region related to gateway *p2*, composed of *Administer Antibiotic Treatment* and *Lock Catheter* activities.
- In case of *Enterococcus spp*, the patient is treated by administering Vancomycin. Unfortunately this case is often associated with endocarditis. This means that physicians may choose to perform a Trans-Esophageal Echocardiography (TEE) for detecting the issue. TEE must be performed not before five and up to seven days from the time when CR-BSI has been confirmed. These activities are represented in Fig. 2 by means of the process region related to gateway *p3*.

Summing up, even in this over-simplified representation of a real-world clinical scenario, we need to specify time-related constraints, which cannot be captured by using BPMN without compromising the process model readability. Examples of these time-related constraints are:

- Duration-Induced-Decision (*DID*). Durations and interleaving of given events/tasks preceding a decision point (i.e., an exclusive gateway), determine the choice to be made, and thus the path to follow. In process of Fig. 2, time durations *LS* and *PS* and their related DTP determine which branch of *CR-BSI confirmed?* will be taken.
- Disjoint-Parallel-Tasks (*DPT*). In this case, we consider tasks which may be executed without a given order, but their execution needs to be disjoint for some reasons (e.g., the preemption of a mutually exclusive resource). In the treatment of CONS, *Administer Antibiotic Treatement* and *Lock Catheter* must be executed in a non-overlapping way. Moreover, since in Fig. 2 both activities belong to a loop, they may be executed multiple times.
- Relative-Time-Constraint (*RTC*). Time durations of two given tasks, or the difference between their endpoints are constrained by specified bounds. In process of Fig. 2, the difference between the beginning of the *TEE* activity, and the end of the *CR-BSI* activity must be between five and seven days.

4 A formal account of Timelines

In this section we introduce the basic concepts of timelines and of timelines-based planning [27]. In [9] it is provided an informal explanation, together with a small example, of how the whole timelines-based machinery works. In the following, we use the notation introduced in [7]. We start by introducing the notion of *state variable*.

▶ **Definition 1** (state variable). *A state variable sv is a triple* $sv = (\mathbf{V}_{sv}, \mathbf{\Delta}_{sv}, \mathbf{D}_{sv})$ *where:*

- $\mathbf{V}_{sv}$ *is the finite domain of the state variable sv;*
- $\mathbf{\Delta}_{sv} : \mathbf{V}_{sv} \to 2^{\mathbf{V}_{sv}}$ *is the transition function, which maps each value* $v \in \mathbf{V}_{sv}$ *to the set of values that may be taken by sv immediately after sv has taken value v;*
- $\mathbf{D}_{sv} : \mathbf{V}_{sv} \to \mathbb{N} \times \mathbb{N} \cup \{+\infty\}$ *is a function that maps each* $v \in \mathbf{V}_{sv}$ *to an interval, i.e., a pair of values* $[d^{sv=v}_{\min}, d^{sv=v}_{\max}]$ *with* $0 < d^{sv=v}_{\min} \leq d^{sv=v}_{\max}$*, which represent respectively the minimum and the maximum duration of an interval over which sv takes value v.*

Table 1 A set of useful atoms conjunctions and their interval based interpretation.

shorthand	meaning	translation
$x\langle M\rangle y$	x **meets** y	$x \leq^{e,s}_{[0,0]} y$
$x\langle B\rangle y$	x **begins** y	$x \leq^{s,s}_{[0,0]} y \wedge x \leq^{e,e}_{[1,+\infty)} y$
$x\langle D\rangle y$	x **during** y	$y \leq^{s,s}_{[1,+\infty)} x \wedge x \leq^{e,e}_{[1,+\infty)} y$
$x\langle F\rangle y$	x **finishes** y	$y \leq^{s,s}_{[1,+\infty)} x \wedge x \leq^{e,e}_{[0,0]} y$
$x\langle O\rangle y$	x **overlaps** y	$x \leq^{s,s}_{[1,+\infty)} y \wedge x \leq^{e,e}_{[1,+\infty)} y \wedge y \leq^{s,e}_{[1,+\infty)} x$
$x \subset_{BD} y$	(x **begins** y) $\vee$ (x **during** y)	$y \leq^{s,s}_{[0,+\infty)} x \wedge x \leq^{e,e}_{[1,+\infty)} y$
$x \subseteq y$	(x **begins** y) $\vee$ (x **finishes** y) $\vee$(x **during** y) $\vee$ ($x = y$)	$y \leq^{s,s}_{[0,+\infty)} x \wedge x \leq^{e,e}_{[0,+\infty)} y$
$x \cap_{BMO} y$	(x **begins** y) $\vee$ (x **meets** y) $\vee$(x **overlaps** y)	$x \leq^{s,s}_{[0,+\infty)} y \wedge y \leq^{s,e}_{[0,+\infty)} x \wedge x \leq^{e,e}_{[1,+\infty)} y$
$x = y$	$x = y$	$x \leq^{s,s}_{[0,0]} y \wedge x \leq^{e,e}_{[0,0]} y$

Given a state variable sv, a *timeline for* sv is a sequence $\mathbf{T}_{sv}$ of pairs called *tokens* which consider functions $\mathbf{\Delta}_{sv}$ and $\mathbf{D}_{sv}$. Formally:

▶ **Definition 2** (token). *A token for a state variable* $sv = (\mathbf{V}_{sv}, \mathbf{\Delta}_{sv}, \mathbf{D}_{sv})$ *is a tuple* $\tau = \langle v, d\rangle$ *where* $v \in \mathbf{V}_{sv}$ *and* $d \in \mathbf{D}_{sv}(v)$.

It is worth noticing that in a token the duration d belongs to the allowed durations for the value v.

▶ **Definition 3** (timeline). *A timeline for a state variable* $sv = (\mathbf{V}_{sv}, \mathbf{\Delta}_{sv}, \mathbf{D}_{sv})$ *is a finite sequence* $\mathbf{T}_{sv} = \langle \tau_1, \ldots \tau_k\rangle$ *of tokens for* sv *such that for every* $1 \leq i < k$ *we have* $v_{i+1} \in \mathbf{\Delta}_{sv}(v_i)$.

Given a token $\tau = \langle v, d\rangle$, we denote with $value(\tau)$ its value (i.e., $value(\tau) = v$). Notice that the value of sv in two consecutive tokens within a timeline do not need to be different as it depends on how $\mathbf{\Delta}_{sv}$ is defined. Given a timeline $\mathbf{T}_{sv}$, we denote with $|\mathbf{T}_{sv}|$ its length. Moreover we will use an array-like notation for specific tokens in the sequence. Formally, if $\mathbf{T}_{sv} = \langle \tau_1, \ldots \tau_k\rangle$ we have $\mathbf{T}_{sv}[i] = \tau_i$ for every $1 \leq i \leq k$. In a timeline $\mathbf{T}_{sv} = \langle \tau_1, \ldots \tau_k\rangle$ for sv for every $1 \leq i \leq k$ we define $\mathbf{s}\_time(\mathbf{T}_{sv}, i)$ as $\mathbf{s}\_time(\mathbf{T}_{sv}, i) = \sum_{1 \leq j < i} d_j$ and $\mathbf{e}\_time(\mathbf{T}_{sv}, i)$ as $\mathbf{e}\_time(\mathbf{T}_{sv}, i) = \sum_{1 \leq j \leq i} d_j$. In the following we will often refer to specific sets of timelines instead of single ones. To this purpose, given a set of timelines $\Gamma = \{\mathbf{T}_{sv_1}, \ldots, \mathbf{T}_{sv_n}\}$, we will say that Γ is *repetition-free* if and only if $sv_i \neq sv_j$ for every $1 \leq i \neq j \leq n$. From now on we will assume every set of timelines to be repetition-free. Synchronization among timelines in the same set is given by means of a set of *Temporal Synchronization Rules*, TS-RULES for short. TS-RULES relate tokens, possibly belonging to different timelines, through temporal relations among intervals called *atoms*. Let $\Sigma = \{x, y, z, \ldots\}$ a set of *token names* (i.e., variables ranging over tokens):

▶ **Definition 4** (atom). *An atom is a clause of the form* $x \leq^{\circ,\bullet}_{I} y$ *where* $\circ, \bullet \in \{\mathbf{s}, \mathbf{e}\}$ *and* $I \in \{[l, u], [l, +\infty) : l, u \in \mathbb{N}, l \leq u\}$.

In the above definition **s** (resp., **e**) refers to the start (resp., end) time of tokens x and/or y. By means of *conjuctions* of atoms we may express all the possible Allen's interval relations [1] between two tokens, and some disjunctions of them. In particular, we will use the shorthands reported in Table 1. Tokens appear in conjunctions which are existentially closed for all but one distinguished variable.

▶ **Definition 5** (existential x-free conjunction). *Given a token name x, an existential x-free conjunction is a formula $\mathcal{E}$ of the form*

$$\mathcal{E} = \exists x_1[sv_1 = v_1] \dots \exists x_h[sv_h = v_h](A_1 \wedge \dots \wedge A_m)$$

where for every $1 \leq i \leq h$ we have $v_i \in \mathbf{V}_{sv_i}$ and $x_i \neq x$, moreover for every $1 \leq j \leq m$ A_j is an atom of the form $\overline{x}^1_j \leq^{\circ_j,\bullet_j}_{I_j} \overline{x}^2_j$ where $\overline{x}^1_j, \overline{x}^2_j \in \{x_1, \dots, x_h\} \cup \{x\}$.

Informally, in an existential x-free conjunction, a variable in the atom is existentially closed or equal to the unique free variable x. Moreover, we will say that an existential x-free conjunction $\mathcal{E}$ is an *existentially closed conjunction* if and only if for every $1 \leq j \leq m$, A_j is an atom of the form $\overline{x}^1_j \leq^{\circ_j,\bullet_j}_{I_j} \overline{x}^2_j$ where $\overline{x}^1_j, \overline{x}^2_j \in \{x_1, \dots, x_h\}$ (i.e., $\mathcal{E}$ does not feature any free-variable). From now on we will treat the case of x-free conjunction which are not existentially closed. Existentially closed conjunctions may be seen as a special case of x-free ones and thus we will omit them for the sake of brevity. Moreover, given an x-free conjunction $\mathcal{E} = \exists x_1[sv_1 = v_1] \dots \exists x_h[sv_h = v_h](A_1 \wedge \dots \wedge A_m)$ we define $SVar(\mathcal{E}) = \{sv_1, \dots, sv_h\}$ as the set of state variables in its existential preamble. Analogously, we define $TNames(\mathcal{E}) = \{x_1, \dots, x_h, x_{h+1}\}$ assuming without loss of generality that x_{h+1} is the free variable x.

Since the token names $TNames(\mathcal{E})$ are exactly $h+1$ (all the existentially quantified ones plus the free one x), we may have that $SVar(\mathcal{E}) \leq h$ because it is absolutely allowed that two distinct token names are bound to the same state variable.

Semantics for x-free conjunctions $\mathcal{E} = \exists x_1[sv_1 = v_1] \dots \exists x_h[sv_h = v_h](A_1 \wedge \dots \wedge A_m)$ is given in terms of a set of timelines $\Gamma = \{\mathbf{T}_{\overline{sv}_1}, \dots, \mathbf{T}_{\overline{sv}_n}\}$ such that $SVar(\mathcal{E}) \subseteq \{\overline{sv}_1, \dots, \overline{sv}_n\}$, a state variable sv_{h+1} in $\{\overline{sv}_1, \dots, \overline{sv}_n\}$ (i.e, $\mathbf{T}_{sv_{h+1}}$ is the timeline that will be associated to x), and a function $f : TNames(\mathcal{E}) \to \mathbb{N}$. In such setting we will have that $\Gamma, sv_{h+1}, f \models \mathcal{E}$ if and only if the following conditions hold:

- for every $1 \leq i \leq h+1$ we have $|\mathbf{T}_{sv_i}| \geq f(x_i)$;
- for every $1 \leq i \leq h$ we have $value(\mathbf{T}_{sv_i}[f(x_i)]) = v_i$
- for every $1 \leq j \leq m$ let $A_j = x_{i_j} \leq^{\circ_j,\bullet_j}_{[l_j,u_j]} x_{i'_j}$ (resp., $A_j = x_{i_j} \leq^{\circ_j,\bullet_j}_{[l_j,+\infty)} x_{i'_j}$) for some $1 \leq i_j, i'_j \leq h+1$, then

$$l_j \leq \bullet_j\_time(\mathbf{T}_{i'_j}, f(x_{i'_j})) - \circ_j\_time(\mathbf{T}_{i_j}, f(x_{i_j})) \leq u_j$$

(resp., $l_j \leq \bullet_j\_time(\mathbf{T}_{i'_j}, f(x_{i'_j})) - \circ_j\_time(\mathbf{T}_{i_j}, f(x_{i_j}))$).

Now we are ready to introduce TS-RULES.

▶ **Definition 6** (temporal synchronization rule). *A temporal synchronization rule $\mathcal{R}$ is a formula which has one of the following two forms:*

- *(trigger rule) $\mathcal{R} = x[sv = v] \to \mathcal{E}_1 \vee \dots \vee \mathcal{E}_n$, where for every $1 \leq i \leq n$ we have that $\mathcal{E}_i$ is an existential x-free conjunction;*
- *(triggerless rule) $\mathcal{R} = \mathcal{E}_1 \vee \dots \vee \mathcal{E}_n$ where for every $1 \leq i \leq n$ we have that $\mathcal{E}_i$ is an existentially closed conjunction.*

For the sake of clarity we will provide only the semantics of trigger rules, since triggerless ones are a simplified version of them. Given a trigger rule $\mathcal{R} = x[sv = v] \to \mathcal{E}_1 \vee \dots \vee \mathcal{E}_h$, its semantics is given by means of a set of timelines $\Gamma = \{\mathbf{T}_{sv_1}, \dots, \mathbf{T}_{sv_n}\}$, such that $\{sv_1, \dots, sv_n\} \supseteq \bigcup_{i=1}^{h} SVar(\mathcal{E}_i) \cup \{sv\}$ in such a case we say that Γ is a *candidate* for $\mathcal{R}$.

▶ **Definition 7** (semantics of trigger rules). *Given a trigger rule* $\mathcal{R} = x[sv = v] \rightarrow \mathcal{E}_1 \vee \ldots \vee \mathcal{E}_h$ *and a candidate* $\Gamma = \{\mathbf{T}_{sv_1}, \ldots, \mathbf{T}_{sv_n}\}$ *for it,* Γ *satisfies* $\mathcal{R}$*, written* $\Gamma \models \mathcal{R}$*, if and only if for every* $1 \leq i \leq |\mathbf{T}_{sv}|$*, if* $\mathbf{T}_{sv}[i] = v$ *then there exists* $1 \leq j \leq h$ *and a function* $f : TNames(\mathcal{E}_j) \rightarrow \mathbb{N}$*, for which* $f(x) = i$ *and* $\Gamma, sv, f \models \mathcal{E}_j$*.*

The timelines-based planning problem is defined as follows.

▶ **Definition 8** (timelines-based planning problem). *Given a set of TS-RULES* $\mathbf{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_p\}$ *the timelines-based planning problem, TPP for short, for* $\mathbf{R}$ *consists of determining whether or not there exists a set of timelines* $\Gamma = \{\mathbf{T}_{sv_1}, \ldots, \mathbf{T}_{sv_n}\}$ *such that* $\Gamma \models \mathcal{R}_i$ *for every* $1 \leq i \leq p$*.*

5 Annotating BPMN Diagrams with Timelines

In this section we describe in more details our approach, which consists of annotating BPMN diagrams with temporal synchronization rules. The proposed annotation is able to enrich the description of process execution by maintaining the diagram as simple as possible. In our proposal, we use a synchronization rule based notation, which allows us to easily handle temporal constraints represented by means of timelines. We would like to point out that in our approach each set Γ is associated with a possible instance of the process (i.e., Γ may be seen as the whole process log for a given process instance), while state variables together with TS-RULES abstract away from single instances and represent constraints on such instances exactly as the corresponding BPMN process diagram does.

As an example, we consider the BPMN diagram reported in Fig. 2, which has been annotated by means of timelines. In [9] it is provided a formal mapping from diagrams to timelines-based planning problems. It is easy to prove that such mapping guarantees the existence of a bijection between the solutions of the target planning problem and the correct executions of the related process model. In Fig. 3 we show an instance (i.e., an execution) of the considered process. The execution is represented as a set of timelines, one for each BPMN element. In this example tasks and gateway blocks are correctly interleaved.

Tokens on timelines may take two values, *active*, denoted by $\top$, and *not active*, denoted by $\bot$. It means that each token can be seen as an on/off switch. The meaning of these two values is straightforward: *active* means that the process element is currently executed and its duration is represented by means of the duration of the token, and *not active* means that the process element is not executed in the interval of time corresponding to the token. In Fig. 3, when a token is active, it is represented by using the BPMN notation related to the considered element. Otherwise, when the token is not active, it is represented by means of a dashed line. For example, the execution of task *Administer an Empirical Therapy* has a duration of 4 hours and half, as represented in Fig. 3 by using a task-like shape on the $t_{empirical}$ line from 19.00 to 23.30. The execution of task *Administer the Antibiotic Therapy* related to line t_{CONS} is not executed in the 1-hour interval starting at 10 *Jan* 2018 4:00.

In our proposal, we take advantage from the fact that the BPMN diagram is structured, and associate a timeline to each SESE region. The beginning of an active token represents the entry node (gateway) of the SESE region associated to the timeline, and the ending of such token represents its exit gateway. For instance, in Fig. 3 the two executions related to gateway e_{loop2} are represented by the active tokens [10 *Jan* 2018 2:00, 10 *Jan* 2018 5:00] and [10 *Jan* 2018 5:00, 10 *Jan* 2018 9:00] on the relative timeline.

In [9], more details are given about the way the described tokens can be properly constrained for representing correct executions of gateways and tasks, and about the way interleaving may be forced by means of suitable synchronization rules.

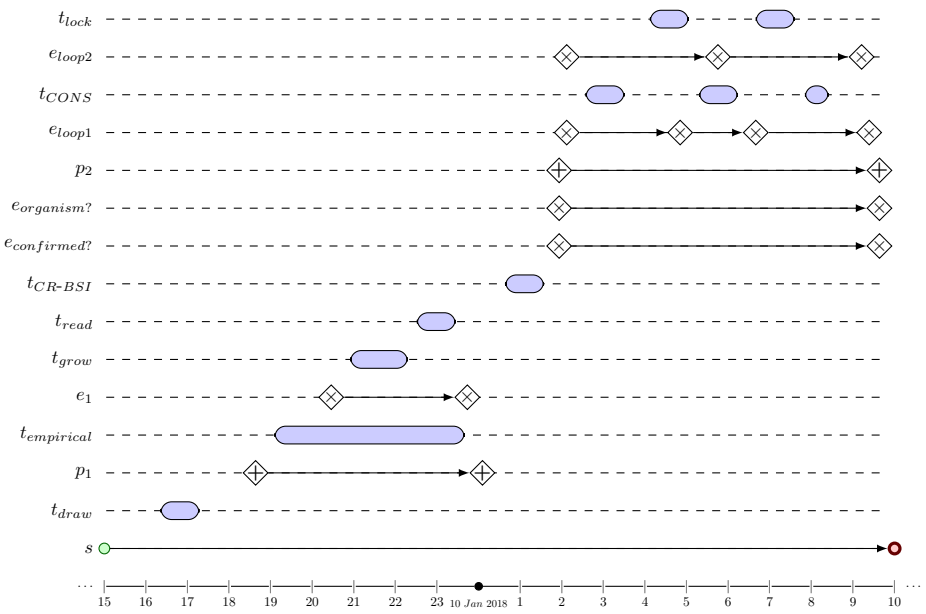


Figure 3 Example of an execution of the business process of Fig. 2, represented as timeline (for the sake of brevity only timelines related to elements involved in the considered execution are shown).

In the following examples we will assume that the presented scenario is taken from timelines representing correct executions of the considered BPMN process. For the process in Fig. 2, a timeline having a token t_{look}, which is active before an active token t_{grow}, is not allowed since the correct execution of the process requires that the execution of task *Look for Other Sources of Infection* related to t_{look} is after the execution of task *Grow Blood Culture* related to t_{grow}. Before providing the rules for the constraints related to the example of Sec. 3, we introduce a (more human-readable) variation on the syntax for TS-RULES. Such syntax seems more suitable for annotating BPMN diagrams. First, instead of anonymous state variable names like $x, y, \ldots$ we will use the element type associated to the state variable and thus we will write something like $task, task', \ldots$ when the state variable is associated to a task, $exclusive, exclusive', \ldots$ when the state variable is associated to a region delimited by an exclusive gateway, and so on. Moreover, we replace $state\text{-}variable = token\text{-}value$ in the quantifications with either $element\text{-}name$ or its overlined version $\overline{element\text{-}name}$ where $element\text{-}name$ is the subscript of the BPMN element associated to state-variable. We will write $element\text{-}name$ if $token\text{-}value = \top$ and $\overline{element\text{-}name}$ if $token\text{-}value = \bot$, respectively. For instance, rule $x[t_{CONS} = \top] \rightarrow \exists y[t_{lock} = \bot](x \subseteq y)$ turns out to be rule $task[CONS] \rightarrow \exists task'[\overline{lock}](task \subseteq task')$ in the new syntax. The DID, DPT and RTC constraints related to the example of Sec. 3 may be expressed as follows.

- Duration-Induced-Decision (DID):

$$\text{C1)}\quad task[grow] \rightarrow \exists exclusive[organism?] \left(\begin{array}{c} task \leq^{s,e}_{[2\ hours, +\infty)} task \wedge \\ task \leq^{s,s}_{[0,+\infty]} exclusive \end{array} \right) \vee (task \leq^{s,e}_{[0,2\ hours]} task)$$

Fig. 4 shows examples of four partial evolutions of timelines $t_{grow}, e_{organism?}$ and t_{look}. These considered scenarios are triggered by the presence of the execution of the task related to t_{grow} (i.e., the rounded rectangle on the bottom dashed line).

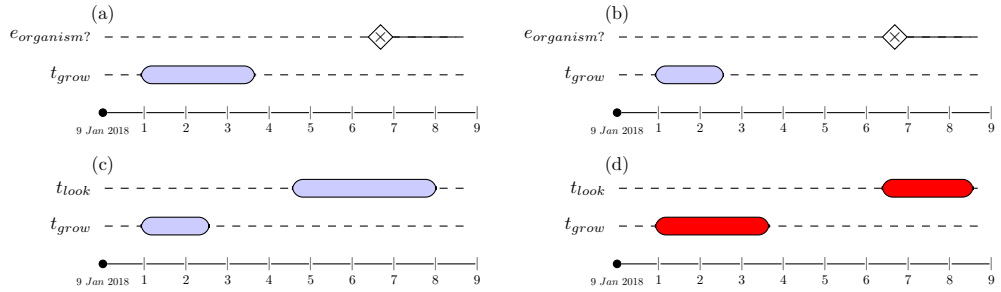


Figure 4 (a), (b), and (c) are examples of executions that fulfill the Duration Induced Decision constraint C1. (d) does not fulfill C1.

Fig. 4(a) represents the case in which the duration of t_{grow} is more than two hours and thus, according to the specified constraint, the YES branch of $e_{confirmed?}$, and the block $e_{organism?}$, must be executed. In this scenario the first disjunction in C1 is fulfilled. When the duration of t_{grow} is less than 2 hours, either YES branch or NO branch of $e_{confirmed?}$ is executed, as depicted in Fig. 4(b) and Fig. 4(c), respectively. In the latter case task related to t_{look} must be executed as correctly depicted in Fig. 4(c).
Example in Fig. 4(d) represents a way to violate constraint C1. In this case, the duration of t_{grow} is greater than 2 hours and the branch NO of $e_{confirmed?}$ is taken by executing t_{look}. This situation violates both disjunctions of C1.

- Disjoint-Parallel-Tasks (DPT):

$$\text{C2)}\quad task[CONS] \rightarrow \exists task'[\overline{lock}](task \subseteq task')$$

Fig. 5 reports examples of two partial evolutions of t_{lock} and t_{CONS} timelines. The intuition behind rule C2 is that if a token on the timeline t_{CONS} is active, then it is contained in a not active token on the timeline t_{lock}.

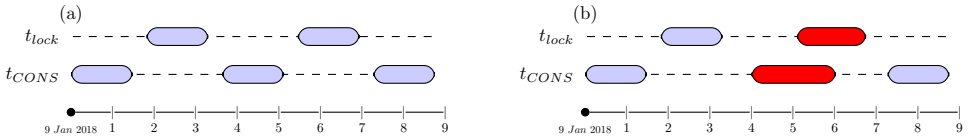


Figure 5 (a) is an example of execution that fulfills the Disjoint-Parallel-Tasks constraint C2. (b) does not fulfill C2.

In Fig. 5(a) an interleaving of tokens in t_{lock} and t_{CONS} that satisfies rule C2 is depicted. In Fig. 5(b) a scenario that violates rule C2 is reported. In this latter case, token [9 *Jan* 2018 4:00, 9 *Jan* 2018 6:00] on timeline t_{CONS} contains the overlap of tokens [9 *Jan* 2018 3:30, 9 *Jan* 2018 5:00] and [9 *Jan* 2018 5:00, 9 *Jan* 2018 6:30] on t_{lock}, and thus it cannot be contained in any token on timeline t_{lock}.

- Relative-Time-Constraint (RTC):

$$\begin{aligned}\text{C3)}\ task[\text{CR-BSI}] \rightarrow\ & \exists task'[\overline{TEE}]\exists task''[TEE](task \subseteq task' \\ & \wedge task'\langle M\rangle task'' \wedge task \leq^{e,s}_{[5\ days, 7\ days]} task'') \\ & \vee \exists task'[\overline{TEE}](task\langle M\rangle task' \wedge task' \leq^{s,e}_{(+\infty,+\infty)} task')\end{aligned}$$

Fig. 6 shows examples of three partial evolutions involving t_{TEE}, $t_{\text{CR-BSI}}$ and t_{CONS} timelines. The scenario reported in Fig. 6(a) fulfills rule C3 since an active token on timeline $t_{\text{CR-BSI}}$ is present, and the next active token on t_{TEE} happens after 6 days.

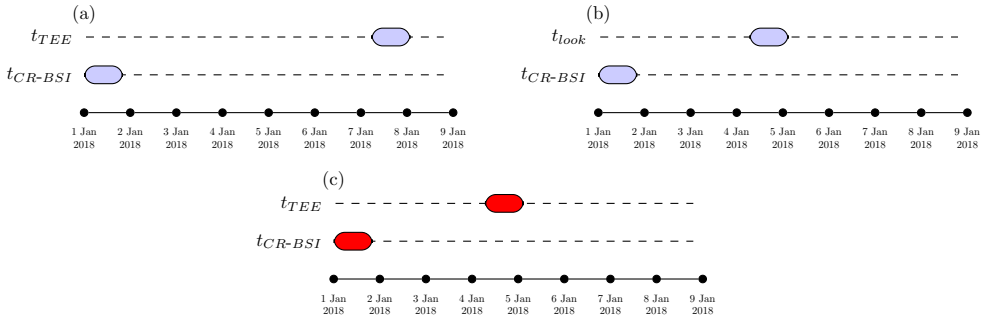


Figure 6 (a) and (b) are examples of executions that fulfill the Relative-Time-Constraint C3. (c) does not fulfill C3.

Also Fig. 6(b) represents a scenario satisfying C3. Assuming that timelines satisfy the correct execution of the process diagram, in this case there is an active token on the timeline t_{look} and thus the NO branch of $e_{confirmed?}$ has been chosen and there are no active tokens on timeline t_{TEE}. This means that the second conjunction of $C3$ is fulfilled. Finally, Fig. 6(c) shows a scenario violating rule C3, since there exists an active token τ on t_{TEE}, which happens after an active token τ' on $t_{CR\text{-}BSI}$, but the distance between the end of τ' and the beginning of τ is less than five days.

TS-RULES allow us to capture different kind of constraints For example, the described temporal constraints may be achieved by suitably adding throw/catch events and event-based gateways to the diagram. However, there are two main drawbacks in this approach:
(i) enforcing such constraints in the diagram may easily make it difficult to read (e.g., see [11] for an example); (ii) modularity is lost forever since some changes in the diagram may change how the constraints are enforced in it.

Moreover, some constraints expressible via TS-RULES may be defined by using *Decision Model and Notation* (DMN) [24]. DMN is a standard notation for modeling decisions, and it is complementary to BPMN. DMN is able to specify conditions on the elements that may change the flow of execution (e.g., exclusive gateways). Our approach can capture DMN sematics in a natural way by introducing additional state variables for data affecting the choice (more on that in Sec. 6) and the related TS-RULES, thus providing a way to check consistency properties between the DMN logic and the process. However, if the choices are inherently depending from the evolution of the data and/or of the flow of the process, TS-RULES explicit such dependence in a more direct and concise way. Finally, TS-RULES are more general since they constrain the flow of execution without the need to be bound to some element in the diagram. For instance, they can force parallel tasks to follow specific patterns as shown by rule C2.

6 Data, Resources, and History-driven gateways

In this section, we illustrate how the proposed approach can be used for expressing data and resource synchronization constraints. Moreover, we introduce a *decision* gateway, based on timelines, for specifying the decision rule about the branch to execute.

In Fig. 7, we report an example of a healthcare process for taking care of severely injured patients [12]. The process of Fig. 7 involves three actors: *paramedics*, *nurses*, and *operating room staff*. Each actor is represented as a swimlane within the pool. *Paramedics* reach the patient and provide transport for her. *Nurses* take care of the patient when she arrives at the hospital, and *operating room staff* (i.e., surgeon and anesthetist), alerted in critical situations, provide emergency surgery.

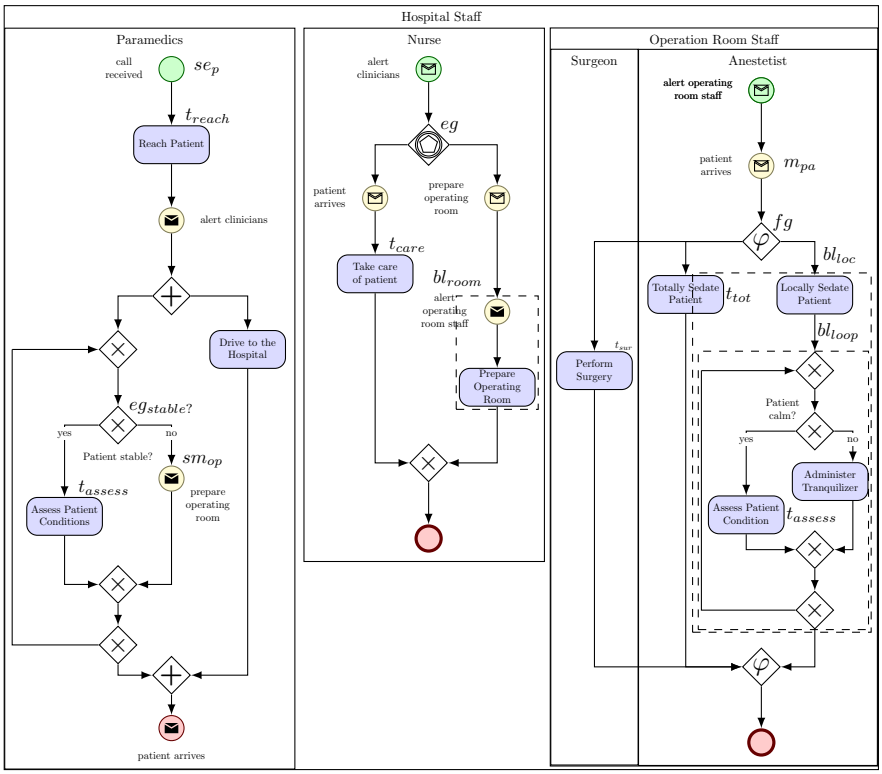


Figure 7 An example of History-driven gateway.

This simple example allows us to introduce the following constraints on data, resources and decisions that may be naturally captured by means of timelines.

- *Enforce parallelization:* this constraint allows us to enforce the simultaneosly execution of two activities beloging to a parallel block. As an example, for specifying that if block bl_{loc} is chosen, then the execution of its internal loop bl_{calm} must be performed for the whole duration of the surgery, we can use the rule: $x[bl_{calm} = \top] \rightarrow \exists y[t_{sur} = \top](y \subseteq x)$. This kind of constraint is symmetrical with respect to the Disjoint Parallel Task constraint described in Sec. 5.
- *Message passing:* BPMN elements like messages, with their possible different semantics, may be easily integrated in our formalism. In this paper, for the sake of space, we only sketch an idea of this kind of constraints, without giving a detailed description and analysis. As an example, during the transport of the patient, paramedics may alert the operation room staff in case the patient situation is getting worse. The aim of the notification alert is requiring the preparation of the operating room. This mechanism is managed by the event-based gateway eg in Fig. 7 in the following way (notice that we consider the messages attached to the gateway as part of it).

$$x[eg = \top] \rightarrow \begin{array}{l} \exists z[se_p = \top]\exists \overline{z}[sm_{op} = \bot]\exists y[t_{care} = \top]\exists \hat{y}[t_{care} = \bot]\exists \check{y}[t_{care} = \bot]\exists \overline{y}[bl_{room} = \bot] \\ \left(z \cap_{BMO} x \wedge \overline{z} \leq^{s,s}_{[0,+\infty)} x \wedge z \leq^{e,e}_{[0,+\infty)} \overline{z} \wedge \hat{y}\langle M\rangle y \wedge y\langle M\rangle \check{y} \wedge \hat{y} \cap_{BMO} x \wedge x \cap_{BMO} \check{y} \wedge y \subseteq x \wedge x \subseteq \overline{y} \right) \vee \\ \exists z[sm_{op} = \top]\exists \overline{z}[se_p = \top]\exists y[bl_{room} = \top]\exists \hat{y}[bl_{room} = \bot]\exists \check{y}[bl_{room} = \bot]\exists \overline{y}[t_{care} = \bot] \\ \left(z \subseteq x \wedge \overline{z} \leq^{s,s}_{[0,+\infty)} x \wedge z \leq^{e,e}_{[0,+\infty)} \overline{z} \wedge \hat{y}\langle M\rangle y \wedge y\langle M\rangle \check{y} \wedge \hat{y} \cap_{BMO} x \wedge x \cap_{BMO} \check{y} \wedge y \subseteq x \wedge x \subseteq \overline{y} \right) \end{array}$$

This proposal is similar to an exclusive gateway with the addition, by means of the $z/\overline{z}$ variables, of a constraint regarding the preemptiveness of messages determining which block will be executed. Managing end events and intermediate message events is slightly

different, since the former can be seen as the end of the related block. For example, in Fig. 7, *patient arrives* is the end of the block se_p. The duration of sm_{op} must be constrained to 1 unit, in order to make it istantaneous by means of rule $x[sm_{op} = \top] \rightarrow x \leq^{s,e}_{[1,1]} x$. Finally, when an intermediate message does not appear as a successor of an event-based gateway, its semantics must be explicitly encoded. This is the case of m_{pa} in Fig. 7, which is mapped to rule $x[m_{pa} = \top] \rightarrow \exists y[se_p = \top](x \leq^{e,e}_{[0,0]} y)$.

- *Resources and roles management:* a very important aspect to consider in managing business processes is related to the definition of roles that are involved in the process execution. BPMN provides swimlanes (within pools) for representing roles (within organizations).
  In the process of Fig. 7 tasks must be performed by the related roles (represented by means of swimlanes), this means that a paramedic cannot perform the surgery, and an anesthetist cannot drive the ambulance. However both a paramedic and an anesthetist may perform task t_{assess}. To force such constraint the rule $x[t_{assess} = \top] \rightarrow \exists y[Paramedic = \top](x = y) \vee \exists y[Anesthetist = \top](x = y)$ can be specified. In this case paramedics and anesthetists represent sets of timelines, one for each resource available in the considered instance. Each of such timelines represents how the specific resource is allocated to each task. The mutual exclusion in the use of resources is guaranteed by the non-overlapping nature of the intervals on the same timeline. The described notation allows us to abstract the number of available resources, since corresponding numbers are inserted at verification time. For example, by instantiating the above rule by using 2 paramedics and 3 anesthetists, we obtain $x[t_{assess} = \top] \rightarrow \exists y[paramedic_1 = \top](x = y) \vee \exists y[paramedic_2 = \top](x = y) \vee \exists y[anesthetist_1 = \top](x = y) \vee \exists y[anesthetist_2 = \top](x = y) \vee \exists y[anesthetist_3 = \top](x = y)$. This allows us to verify quantitative properties related to durations even in presence of multiple instances of the same process, that access the same resources [12].
- *Data driven decisions:* the described timeline-based approach is able to provide a preliminary integration of processes and data. Other proposals presented in literature [10, 14] are more focused on integrating existing formalisms (e.g., Entity-Relation data model), for representing data in BPMN process models.
  The timeline-based approach should not be considered as an alternative for such approaches, but as an annotation working well along with them, by helping in clarifying and verifying properties of data at the time execution of processes.
  As an example, let us suppose that the decision about which branch of $eg_{stable?}$ has to be chosen, is determined by both patient blood pressure (pBP), and patient body temperature (pBT). Let us assume that pBP and pBT are represented by means of the different timelines $(\mathbf{V}_{pBP}, \mathbf{\Delta}_{pBP}, \mathbf{D}_{pBP})$ and $(\mathbf{V}_{pBT}, \mathbf{\Delta}_{pBT}, \mathbf{D}_{pBT})$, respectively. More precisely, $\mathbf{V}_{pBP} = \{70, \ldots, 190, \perp\}$, i.e., on the timeline pBP all the possible ranges of blood pressures plus a disabled value when the pressure is not measured, are represented. The same holds for the body temperature, i.e., $\mathbf{V}_{pBT} = \{29, \ldots 41, \perp\}$. In this case, data may be available only when task t_{assess} is performed, thus the rules $x[t_{assess} = \perp] \rightarrow \exists y[pBP = \perp](y = x)$, $x[t_{assess} = \top] \rightarrow \bigvee_{v \in \{70,\ldots,190\}} \exists y[pBP = v](x = y)$ model this constraint. Similar rules can be specified for constraining pBT.
  Finally, for constraining the gateway sm_{op} to be executed whenever the values of BT and BP exceed certain thresholds, this set of rules can be specified:

$$y[BP = v] \rightarrow \exists x[eg_{stable?} = \top]\exists z[sm_{op} = \top](y \subseteq x \wedge z \subseteq x) \vee \exists x[eg_{stable?} = \perp](y \subseteq x), \text{ with } v > 150$$
$$y[BT = v] \rightarrow \exists x[eg_{stable?} = \top]\exists z[sm_{op} = \top](y \subseteq x \wedge z \subseteq x) \vee \exists x[eg_{stable?} = \perp](y \subseteq x), \text{ with } v > 39$$

  The described example is able to represent the way in which, by means of timeline-based annotation, it is possible to enrich processes with constraints on temporal aspects, roles

and data. Process models are equipped both with the constraints on their execution and with requirements about decisions and durations, without burden the process model.

- *Special behaviors for gateways:* in this work we show how to express BPMN diagram semantics and complex temporal constraints that would involve, if integrated directly in the diagram, complex patterns of throw/catch events as well as event-based gateways. We intentionally did not extend BPMN with some new element, in order to stay within the boundaries of BPMN semantics. However, it is possible to use TS-RULES for extending the standard BPMN notation, by expressing the behavior of complex new elements in a straightforward way. As an example, let us consider gateway fg in Fig. 7. If it is the case that an instance of the process reaches fg, we expect that the patient has to be sedated, either totally or locally, while surgery has to be performed. Thus, in this case, we expect that branch t_{sur} is anyway executed, while choosing exactly one between the branches t_{tot} and bl_{loc}. Moreover, the choice between t_{tot} and bl_{loc} will be dictated by recent results of measurements related to the patient condition. For example, in case that $pBP > 150$ at some time point between 3 hours and the beginning of the surgery, a partial sedation has to be administered, otherwise it is possible to administer the total one. Rules for specifying these expectations are the following:

$$x[fg = \top] \rightarrow \begin{array}{l} \exists y[t_{sur} = \top]\exists z[t_{tot} = \top]\exists w[bl_{loc} = \bot](y \subseteq x \wedge z \subseteq x \wedge x \subseteq w) \vee \\ \exists y[t_{sur} = \top]\exists z[bl_{loc} = \top]\exists w[t_{tot} = \bot](y \subseteq x \wedge z \subseteq x \wedge x \subseteq w) \end{array}$$

$$x[pBP = v] \rightarrow \begin{array}{l} \exists y[fg = \top]\exists z[bl_{loc} = \top](x \leq^{e,s}_{[0,3\ hours]} y \wedge z \subseteq y) \vee \\ \exists y[fg = \bot](x \leq^{e,e}_{[3\ hours,+\infty)} y \wedge y \leq^{s,s}_{[0,+\infty)} x) \end{array} \text{, with } v > 150$$

Summing up, by means of timelines we are able to introduce a BPMN element that behaves like a *conditional parallel* gateway, that is, a parallel gateway which runs all and only the branches that satisfy a certain condition at the precise moment of its execution.

7 Conclusions

In this paper we dealt with issues related to the specification of different kinds of constraints on process models represented by means of BPMN diagrams. We provided a timelines-based approach for expressing admissible executions of a process. Timelines allow us to specify complex constraints possibly related to time, data, and resources, by annotating the BPMN process diagram, without overburden the process diagram itself. Some of the advantages of our proposal are (i) providing a means for specifying complex constraints without extending BPMN; (ii) applying the existing tools for timeline-based planning [2, 5, 6] for verifying *qualitative properties* at design time; (iii) supporting resources optimization in the style of [12], and (iv) checking *quantitative properties* such as the interplay between the number of mutually exclusive resources and the number of process instances that may be completed in a given amount of time. For future work, we plan to apply synchronization rules for querying running processes and monitoring business process activities (Business Activity Monitoring (BAM[3]).

References

1 James F. Allen. Maintaining Knowledge About Temporal Intervals. *Commun. ACM*, 26(11):832–843, November 1983. doi:10.1145/182.358434.

2 Javier Barreiro, Matthew Boyce, Minh Do, Jeremy Frank, Michael Iatauro, Tatiana Kichkaylo, Paul Morris, James Ong, Emilio Remolina, Tristan Smith, and David Smith. EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In *ICKEPS 2012*, 2012.

3 Catriel Beeri, Anat Eyal, Tova Milo, and Alon Pilberg. Query-based monitoring of BPEL business processes. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1122–1124. ACM, 2007.

4 Yuval Shahar Carlo Combi, Elpida Keravnou-Papailiou. *Temporal Information Systems in Medicine*. Springer Science & Business Media, 2010.

5 Amedeo Cesta, Gabriella Cortellessa, Simone Fratini, and Angelo Oddi. Developing an End-to-End Planning Application from a Timeline Representation Framework. In *Twenty-First IAAI Conference*, 2009.

6 S. Chien, D. Tran, G. Rabideau, S.R. Schaffer, D. Mandl, and S. Frye. Timeline-Based Space Operations Scheduling with External Constraints. In *Proc. of the 20th International Conference on Automated Planning and Scheduling*, pages 34–41, 2010.

7 Marta Cialdea Mayer, Andrea Orlandini, and Alessandro Umbrico. Planning and execution with flexible timelines: a formal account. *Acta Informatica*, 53(6):649–680, October 2016. doi:10.1007/s00236-015-0252-z.

8 Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *International Conference on Computer Aided Verification*, pages 359–364. Springer, 2002.

9 Carlo Combi, Barbara Oliboni, and Pietro Sala. Customizing BPMN Diagrams using Timelines (extended version). http://profs.scienze.univr.it/~sala/BPMNTimelinesExtended.pdf.

10 Carlo Combi, Barbara Oliboni, Mathias Weske, and Francesca Zerbato. Conceptual Modeling of Processes and Data: Connecting Different Perspectives. In *ER 2018*, volume 11157 of *LNCS*, pages 236–250, 2018. doi:10.1007/978-3-030-00847-5_18.

11 Carlo Combi, Pietro Sala, and Francesca Zerbato. Driving time-dependent paths in clinical BPMN processes. In *Proceedings of SAC 2017*, pages 743–750, 2017. doi:10.1145/3019612.3019620.

12 Carlo Combi, Pietro Sala, and Francesca Zerbato. A Logical Formalization of Time-Critical Processes with Resources. In *BPM Forum 2018*, volume 329 of *LNBIP*, pages 20–36. Springer, 2018. doi:10.1007/978-3-319-98651-7_2.

13 Giuseppe De Giacomo, Marlon Dumas, Fabrizio Maria Maggi, and Marco Montali. Declarative process modeling in BPMN. In *CAISE*, pages 84–100. Springer, 2015.

14 Riccardo De Masellis, Chiara Di Francescomarino, Chiara Ghidini, Marco Montali, and Sergio Tessaris. Add Data into Business Process Verification: Bridging the Gap between Theory and Practice. In *Proc. of 31st AAAI Conference on Artificial Intelligence*, pages 1091–1099. AAAI Press, 2017. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14627.

15 Marlon Dumas, Luciano García-Bañuelos, and Artem Polyvyanyy. Unraveling Unstructured Process Models. In Jan Mendling, Matthias Weidlich, and Mathias Weske, editors, *Business Process Modeling Notation*, pages 1–7, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

16 Jan-Philipp Friedenstab, Christian Janiesch, Martin Matzner, and Oliver Muller. Extending BPMN for business activity monitoring. In *2012 45th Hawaii International Conference on System Sciences*, pages 4158–4167. IEEE, 2012.

17 Gerard J Holzmann. *The SPIN model checker: Primer and reference manual*, volume 1003. Addison-Wesley Reading, 2004.

18 M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2004.

19 Bartek Kiepuszewski, Arthur H. M. ter Hofstede, and Christoph Bussler. On Structured Workflow Modelling. In Benkt Wangler and Lars Bergman, editors, *Advanced Information Systems Engineering, 12th International Conference CAiSE 2000, Stockholm, Sweden, June 5-9, 2000, Proceedings*, volume 1789 of *Lecture Notes in Computer Science*, pages 431–445. Springer, 2000. doi:10.1007/3-540-45140-4_29.

20 Andreas Lanz, Roberto Posenato, Carlo Combi, and Manfred Reichert. Controlling time-awareness in modularized processes. *Enterprise, Business-Process and Information Systems Modeling*, pages 157–172, 2016.

21 Andreas Lanz, Manfred Reichert, and Barbara Weber. Process time patterns: A formal foundation. *Information Systems*, 57:38–68, 2016. doi:10.1016/j.is.2015.10.002.

22 Fabrizio Maria Maggi, Marco Montali, Michael Westergaard, and Wil M. P. van der Aalst. Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata. In *BPM 2011.*, volume 6896 of *LNCS*, pages 132–147. Springer, 2011. doi:10.1007/978-3-642-23059-2_13.

23 Leonard A Mermel, Barry M Farr, Robert J Sherertz, Issam I Raad, Naomi O'grady, JoAnn S Harris, and Donald E Craven. Guidelines for the management of intravascular catheter-related infections. *Infection Control & Hospital Epidemiology*, 22(4):222–242, 2001.

24 Steven Mertens, Frederik Gailly, and Geert Poels. Enhancing Declarative Process Models with DMN Decision Logic. In Khaled Gaaloul, Rainer Schmidt, Selmin Nurcan, Sérgio Guerreiro, and Qin Ma, editors, *Enterprise, Business-Process and Information Systems Modeling - 16th International Conference, BPMDS 2015, 20th International Conference, EMMSAD 2015, Held at CAiSE 2015, Stockholm, Sweden, June 8-9, 2015, Proceedings*, volume 214 of *Lecture Notes in Business Information Processing*, pages 151–165. Springer, 2015. doi:10.1007/978-3-319-19237-6_10.

25 Angelo Montanari and Pietro Sala. Interval-based Synthesis. In Adriano Peron and Carla Piazza, editors, *Proceedings Fifth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2014, Verona, Italy, September 10-12, 2014.*, volume 161 of *EPTCS*, pages 102–115, 2014. doi:10.4204/EPTCS.161.11.

26 Richard Müller and Andreas Rogge-Solti. BPMN for healthcare processes. In *Proceedings of the 3rd Central-European Workshop on Services and their Composition (ZEUS 2011), Karlsruhe, Germany*, volume 1, 2011.

27 Nicola Muscettola. HSTS: Integrating planning and scheduling. Technical report, Carnegie-Mellon Univ Pittsburgh PA Robotics Inst, 1993.

28 OMG. Business Process Model and Notation (BPMN), Version 2.0, January 2011. URL: http://www.omg.org/spec/BPMN/2.0.

29 Roberto Posenato, Francesca Zerbato, and Carlo Combi. Managing Decision Tasks and Events in Time-Aware Business Process Models. In *BPM 2018*, volume 11080 of *LNCS*, pages 102–118. Springer, 2018. doi:10.1007/978-3-319-98648-7_7.

30 Wolfgang Thomas. On the Synthesis of Strategies in Infinite Games. In *STACS*, pages 1–13, 1995. doi:10.1007/3-540-59042-0_57.

31 Petia Wohed, Wil MP van der Aalst, Marlon Dumas, Arthur HM ter Hofstede, and Nick Russell. On the suitability of BPMN for business process modelling. In *International conference on business process management*, pages 161–176. Springer, 2006.

The Second Order Traffic Fine: Temporal Reasoning in European Transport Regulations

Ana de Almeida Borges
University of Barcelona, Spain
anadealmeidagabriel@ub.edu

Juan José Conejero Rodríguez
University of Barcelona, Spain
juan.conejero@ub.edu

David Fernández-Duque[1]
Ghent University, Belgium
David.FernandezDuque@Ugent.be

Mireia González Bedmar
University of Barcelona, Spain
m.gonzalezbedmar@ub.edu

Joost J. Joosten
University of Barcelona, Spain
jjoosten@ub.edu

Abstract

We argue that European transport regulations can be formalized within the Σ^1_1 fragment of monadic second order logic, and possibly weaker fragments including linear temporal logic. We consider several articles in the regulation to verify these claims.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Higher order logic; Applied computing → Law

Keywords and phrases linear temporal logic, monadic second order logic, formalized law, transport regulations

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.6

Funding *Ana de Almeida Borges*: 2016 DI 032, Generalitat de Catalunya, Departament d'Empresa i Coneixement
Juan José Conejero Rodríguez: RTC-2017-6740-7, Spanish Ministry of Science and Universities
David Fernández-Duque: RTC-2017-6740-7, Spanish Ministry of Science and Universities
Mireia González Bedmar: RTC-2017-6740-7, Spanish Ministry of Science and Universities
Joost J. Joosten: RTC-2017-6740-7, Spanish Ministry of Science and Universities; 2016 DI 032 and 2016 DI 033, Generalitat de Catalunya, Departament d'Empresa i Coneixement; SGR 270, Generallitat de Catalunya; FFI2015-70707P, Spanish Ministry of Economy and Competitiveness

Acknowledgements We are grateful to Albert Atserias and Martín Diéguez for insightful discussions involving Kamp's theorem and its variants.

1 Introduction

The authors of the paper are involved in research projects that collaborate with industry, lawyers and legislators where the main goal is to develop verified legal software. The industrial and social need is evident: various legal decisions are made on the basis of

[1] Corresponding author

algorithmic processing of data in consequence of which individuals can be fined or even sent to jail. Software contains errors, but for our legal context such errors should not be acceptable.

In particular, the above-mentioned projects have as first and main objective to eradicate errors from software that interprets data from tachographs. A tachograph is to a truck what a black-box is to an aeroplane: it registers all kinds of activities from the truck and driver, like speed, movement and others. In practice, a police officer may pull over a truck for an inspection where the tachograph data is read and interpreted by some software. Depending on the verdict of the program, the driver may be instantly fined or sometimes even imprisoned. It is known that many erroneous automated verdicts are issued. This is highly undesirable both from an industrial and from a civil rights perspective. It is here that logic tries to come to the rescue.

The aim of the project is to recast the transport legislation into an unambiguous mathematically formulated language such that proof-checkers may show that the developed code indeed satisfies the legislation. This paradigm allows us to honestly speak of error-free software.[2]

The multi-disciplinary nature of the project poses many challenges. For one, legislation is often intended to leave room for various interpretations and applications of the law. In contrast, mathematical definitions and algorithms are deterministic in nature and disallow ambiguity. The main mitigation of this challenge seems to be the accepted tendency to require unambiguous laws if they should prescribe an algorithm. These laws are written in prose, and albeit technical, it will always leave room for multiple interpretations which sometimes only differ on very subtle yet essential aspects. Here jurisdiction tells us what to do in most cases. Our collaboration with working lawyers has been very interesting in this aspect.

Yet another challenge lies in choosing the right ontology and logico-mathematical framework where to recast the interpreted and disambiguated laws. It is mainly this aspect that is addressed here. In particular, in this paper we will argue that the European regulation can be modelled in linear temporal logics [10], broadly construed as subsystems and extensions of the classical LTL with "until".

To illustrate this claim we identify some passages that may be problematic from a logical perspective, most notably because they contain *prima facie* "impredicative" content: for our purposes, a property is *impredicative* if its definition requires genuine quantification over the set of all subsets of $\mathbb{N}$. This terminology is inspired from Weyl's predicative mathematics which does not accept the powerset axiom for infinite sets [11]. Nevertheless, all laws we consider will fall in the Σ^1_1 fragment of monadic second-order logic, and model-checking formulas in this fragment can be reduced to satisfiability of first-order formulas. Thus we argue that such laws are not ideal from a computational perspective, but even in the worst case scenario, checking the compliance of a law can be reduced to a well-understood problem.

2 European Transport Law

In this section we discuss some passages from the European transport regulation [3] and why they pose a challenge in terms of logical modelling. Our analysis will be based on the following excerpts, which we have found to be problematic from a logical perspective.

[2] Of course, it has its subtleties. Software will be as good as the specification, which may contain errors. Also, we must trust the small kernel of the proof-checker, apart from the hardware and middleware involved. There is also the consistency assumption of the underlying type-theory. A further methodological objection may be that the formalisations and proofs are not easily human-readable.

§4(h) "regular weekly rest period" means any period of rest of at least 45 hours.

§4(i) "a week" means the period of time between 00.00 on a Monday and 24.00 on the following Sunday.

§8.6. In any two consecutive weeks, a driver shall take at least:

- two regular weekly rest periods, or
- one regular weekly rest period and one reduced weekly rest period of at least 24 hours. However, the reduction shall be compensated by an equivalent period of rest taken en bloc before the end of the third week following the week in question.

A weekly rest period shall start no later than at the end of six 24-hour periods from the end of the previous weekly rest period.

§8.7. Any rest taken as compensation for a reduced weekly rest period shall be attached to another rest period of at least nine hours.

§8.9. A weekly rest period that falls in two weeks may be counted in either week, but not in both.

▶ Remark 1. There are additional regulations regarding *daily* rest periods. For the sake of exposition we will only consider the above-mentioned regulations, but in all constructions and examples it should be noted that the driver would additionally have to rest daily in order to fully comply with the law. We will not discuss daily resting periods further in this text.

2.1 Placement of weekly rest periods

Let us consider a case implied by Article §8 of the Regulation, depicted in Figure 1. Each letter-divided segment denotes a week and the smaller segments denote a day, with time flowing from left to right. Furthermore, each serpentine line denotes weekly rest periods of 68 hours except the last one, which lasts only 45 hours.

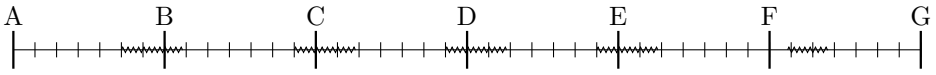


Figure 1 Six consecutive *weeks* and five weekly rest periods (serpentine lines) taken by a hypothetical driver.

Figure 1 represents the activities of a driver who starts resting Saturday at 00:00h and retakes their activity on Monday at 20:00h. Then, until the fourth week, the driver periodically start his weekly rest on Sunday at 00:00h and retake their activity on Tuesday at 20:00h. During the sixth week they rest 45 hours, from Monday at 20:00h to Wednesday at 17:00h.

Since all except the last of these weekly rest periods fall between two weeks, it is reasonable to want to find a procedure that will determine whether there exists a way of counting each of them within one week or the other as per §8.9, so that the situation becomes legal.

In our simple example, the segment FG has a fixed rest period of 45 hours. In the remaining weeks we have to choose where to assign the resting periods,[3] but it is evident that we cannot arrange them in a way that makes the whole interval AG legal. One might argue that this situation is a bit controversial, given that all other articles exposed above except §8.9 are complied beyond their minimum requirements.

[3] We cannot assign parts of this periods to different weeks, since this would give rise to two consecutive reduced weekly rest periods and thus violate §8.6.

Here, the Regulation does not pose a logical problem, nor is it inconsistently worded. But logic is not entirely unrelated to this issue. The complexity that results from §8.9 generates a potential combinatorics problem. As an example, we could encounter situations which follow the structure from Figure 1 with many more occurrences of the *in between* segments. Verifying the legality of the situation could, in principle, require checking a large number of possible assignments of rest intervals to weeks. This non-locality feature has been discussed and formalised in Coq [2].

In a first attempt to formally represent §8.9 we may think in a second-order setting: we could model weekly rest periods as pairs of points indicating the start and end of the rest. Thus §8.9 could be a formula asserting the existence of a function that would model the assignment of weekly rest periods into weeks. Such a formalization would require a second-order existential quantifier (and, in fact, would even fall outside of monadic second order logic, which does not allow for function quantification). In the following section we discuss this issue in some detail.

2.2 Timing of compensations

The second potential source of problems comes from the compensation mechanism of §8.6. To illustrate it, we construct weeks A, B, and $C_i, 1 \leq i \leq n$ such that the sequences $[A, C_1, \ldots, C_n]$ and $[C_1, \ldots, C_n, B]$ are both legal, but the full sequence $[A, C_1, \ldots, C_n, B]$ is *not*. The question then arises: *where is the illegality?* It is in the combination between A and B, where A and B can be arbitrarily far apart from each other. Clearly this is not a good feature for a law.

Throughout this subsection, line segments represent weeks, and the numbers attached to them represent the number of hours rested during each week. In Figure 2, the first and last segments represent the weeks A and B we mentioned before.

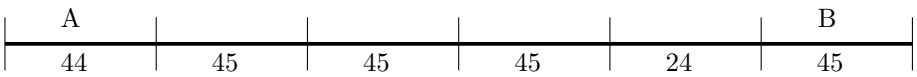


Figure 2 Illegal interval of six consecutive weeks performed by a hypothetical driver.

As shown in Figure 3, if we do not consider the last week, the remaining interval is rendered legal by the law, for we can assume that the hours to be compensated will be incorporated in the week we omitted.

Figure 3 First five weeks of the example represented in Figure 2, together with a possible sixth week that would make the whole interval legal.

Similarly, if we remove week A from the example of Figure 2, the resulting interval (represented in Figure 4) is also legal, since we can assume that the compensation for the fourth week takes place in the weeks outside our interval.

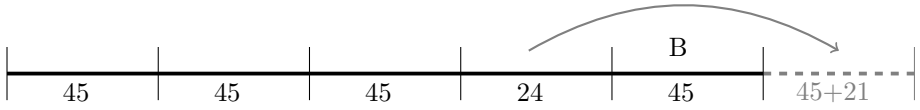


Figure 4 Last five weeks of the interval represented in Figure 2, together with a possible sixth week that would make the whole interval legal.

However, the interval of Figure 2 is illegal, as Figure 5 illustrates. This is because after compensating the first week according to article §8.6, we still have to compensate one hour, but we cannot allocate it within any of the three following weeks without having two consecutive reduced weekly rest periods.

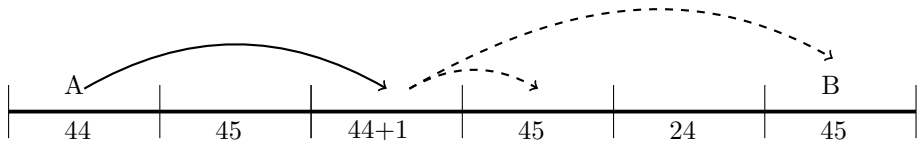


Figure 5 The same interval of Figure 2, with an attempt to assign compensations (dashed lines) that ultimately fails.

This example can be generalized to ensure that that A and B are n weeks apart. The corresponding interval (illustrated in Figure 6) has a similar structure to the one we have treated. The first week has a 44 hour weekly rest period, and all the following weeks have 45 hour weekly rest periods except for the penultimate one, which has a 24 hour weekly rest period.

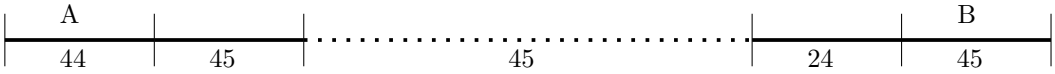


Figure 6 General example of an illegal interval that is legal when week A or week B is erased.

In this situation if we omit one of the weeks A or B the remaining interval will be legal, but the interval as it stands is illegal.

Indeed, a literal reading of the law would require three functions c_1, c_2, c_3 where given a week W, $c_i(W)$ is a (possibly empty) interval that occurs i weeks later and is used to compensate a reduced weekly rest. Once again such functions are not, properly speaking, objects of monadic second order logic, but we will show that they can be represented as such.

In the rest of the article we will show that these properties can indeed be represented in monadic second order logic, and explore whether simpler representations are possible. Before we do so, let us review the logical frameworks we will work with.

3 Temporal Logics

In this section we review the temporal logics that we consider. These are all subsystems of either linear temporal logic LTL or monadic second-order logic MSO interpreted over the natural numbers; as we discuss below, we regard the latter as a temporal logic in view of Kamp's theorem and extensions.

3.1 Linear Temporal Logic

Linear temporal logic is based on the language $\mathcal{L}_{\Box\,\mathsf{U}}$ given by the following grammar:

$$\varphi, \psi \;:=\; \bot \;\mid\; P \;\mid\; \varphi \to \psi \;\mid\; \bigcirc\varphi \;\mid\; \Box\varphi \;\mid\; \varphi\,\mathsf{U}\,\psi,$$

where P is an element of a countable set $\mathbb{P}$ of predicate symbols. We will consider the U-free fragment $\mathcal{L}_{\Box}$, the $\Box$-free fragment $\mathcal{L}_{\mathsf{U}}$ and the language whose only tense is $\bigcirc$, $\mathcal{L}_{\bigcirc}$. As usual, $\bigcirc$ is read as "next", $\Box$ is read as "henceforth", and U as "until". We define other Booleans and $\Diamond$ as abbreviations in the standard way. Note that $\mathcal{L}_{\mathsf{U}}$ is expressively equivalent to $\mathcal{L}_{\Box\,\mathsf{U}}$, so we will seldom work over the full language. We could additionally consider past tenses, but they do not add expressive power to $\mathcal{L}_{\mathsf{U}}$ in models with a starting point (although there are issues with succinctness which we briefly discuss).

The articles we consider also require some counting, but this can be dealt with using the following abbreviations, where $n, m \in \mathbb{N}$. Below, an empty disjunction should be read as $\bot$ and an empty conjunction as $\top$.

- $\bigcirc^0\varphi := \varphi$ and $\bigcirc^{n+1}\varphi = \bigcirc\bigcirc^n\varphi$;
- $\Diamond^{<n}\varphi = \bigvee_{i=0}^{n-1} \bigcirc^n\varphi$ and $\Box^{<n}\varphi = \bigwedge_{i=0}^{n-1} \bigcirc^n\varphi$.

Variants with $\leq n$ instead of $<n$ are defined by reading $\leq n$ as $<n+1$.

Given any formula φ and a set $\Theta \subseteq \{\bigcirc, \Box, \mathsf{U}\}$, we define the Θ-*depth* of φ (in symbols, $\mathrm{dpt}_\Theta(\varphi)$) to be the nesting depth of tenses in Θ, defined in a standard way. If $\Theta = \{\vartheta\}$ we write ϑ-*depth* and $\mathrm{dpt}_\vartheta(\cdot)$ instead of Θ-depth and $\mathrm{dpt}_\Theta(\cdot)$, and if $\Theta = \{\bigcirc, \Box, \mathsf{U}\}$ we write *temporal depth* and $\mathrm{dpt}(\cdot)$ instead of Θ-*depth* and $\mathrm{dpt}_\Theta(\cdot)$. As a general rule we consider the $\bigcirc$-depth to be a negligible complexity measure with respect to the depths of other tenses.

We will always interpret formulas of $\mathcal{L}$ over the structure $(\mathbb{N}, S)$, where $S(n) = n+1$. Hence, for our purposes an LTL *model* $\mathcal{M}$ is merely a function $\cdot^{\mathcal{M}} \colon \mathbb{P} \to 2^{\mathbb{N}}$. We define the satisfaction relation $\models$ inductively by

1. $(\mathcal{M}, n) \models P$ iff $n \in P^{\mathcal{M}}$
2. $(\mathcal{M}, n) \not\models \bot$
3. $(\mathcal{M}, n) \models \varphi \to \psi$ iff $(\mathcal{M}, n) \not\models \varphi$ or $(\mathcal{M}, n) \models \psi$
4. $(\mathcal{M}, n) \models \bigcirc\varphi$ iff $\big(\mathcal{M}, S(n)\big) \models \varphi$
5. $(\mathcal{M}, n) \models \Box\varphi$ iff for all $k \geq 0$ we have that $\big(\mathcal{M}, S^k(n)\big) \models \varphi$
6. $(\mathcal{M}, n) \models \varphi\,\mathsf{U}\,\psi$ iff there exists $k \geq 0$ such that $\big(\mathcal{M}, S^k(n)\big) \models \psi$ and $\forall\, i \in [0, k)$, $\big(\mathcal{M}, S^i(n)\big) \models \varphi$

As usual, a formula φ is *satisfiable* over a set of models Ω if there is $\mathcal{M} \in \Omega$ and $n \in \mathbb{N}$ so that $(\mathcal{M}, n) \models \varphi$, and *valid* on Ω if, for every $\mathcal{M} \in \Omega$ and $n \in \mathbb{N}$, $(\mathcal{M}, n) \models \varphi$.

3.2 Monadic Second-Order Logic

The syntax of monadic second-order logic is defined as follows. First, define a *term* to be given by the grammar

$$t \;:=\; 0 \;\mid\; x \;\mid\; S(t),$$

where x belongs to some fixed set of first-order variables $\mathbb{V}$. Then, the language $\mathcal{L}^2_\forall$ is defined by the grammar

$$\varphi, \psi \;:=\; \bot \;\mid\; P(t) \;\mid\; t < s \;\mid\; \varphi \to \psi \;\mid\; \forall\, x\, \varphi \;\mid\; \forall\, P\, \varphi$$

where x is a variable, t and s terms, and $P \in \mathbb{P}$. Once again we define other Booleans and $\exists$ as standard abbreviations, and define $\mathcal{L}^1_\forall$ to be the sub-language of $\mathcal{L}^2_\forall$ that does not allow quantifiers over elements of $\mathbb{P}$.

The language $\mathcal{L}^2_\forall$ is interpreted over models $\cdot^{\mathcal{M}} \colon \mathbb{V} \cup \mathbb{P} \to \mathbb{N} \cup 2^{\mathbb{N}}$ such that $x^{\mathcal{M}} \in \mathbb{N}$ if x is a variable and $P^{\mathcal{M}} \subseteq \mathbb{N}$ if P is a predicate symbol. For a variable x and $n \in \mathbb{N}$ let $\mathcal{M}[x/n]$ be the model that is the same as $\mathcal{M}$ except that $x^{\mathcal{M}[x/n]} = n$, and for $P \in \mathbb{P}$ and $A \subseteq \mathbb{N}$ define $\mathcal{M}[P/A]$ analogously. Extend $\cdot^{\mathcal{M}}$ to terms by defining recursively $0^{\mathcal{M}} = 0$ and $(S(t))^{\mathcal{M}} = t^{\mathcal{M}} + 1$. The satisfaction relation is then defined as follows:

1. $\mathcal{M} \not\models \bot$
2. $\mathcal{M} \models P(t)$ iff $t^{\mathcal{M}} \in P^{\mathcal{M}}$
3. $\mathcal{M} \models \varphi \to \psi$ iff $\mathcal{M} \not\models \varphi$ or $\mathcal{M} \models \psi$
4. $\mathcal{M} \models \forall x \varphi$ iff for all $n \in \mathbb{N}$, $\mathcal{M}[x/n] \models \varphi$
5. $\mathcal{M} \models \forall P \varphi$ iff for all $A \subseteq \mathbb{N}$, $\mathcal{M}[P/A] \models \varphi$

Satisfiability and *validity* are defined as before. MSO denotes the language $\mathcal{L}^2_\forall$ endowed with these semantics, and MFO denotes MSO restricted to $\mathcal{L}^1_\forall$. In order to unify our semantics for temporal logics and MSO, we regard an LTL model $\mathcal{M}$ as an MSO model by setting $x^{\mathcal{M}} = 0$ for all variables, and similarly regard an MSO model as an LTL model by restricting the domain to $\mathbb{P}$.

We say that a set Ω of models is *definable* in a language $\mathcal{L} \subseteq \mathcal{L}_{\Box \mathsf{U}}$ if there is φ in $\mathcal{L}$ such that for any model $\mathcal{M}$ we have $(\mathcal{M}, 0) \models \varphi$ if and only if $\mathcal{M} \in \Omega$. Similarly, Ω is definable in $\mathcal{L} \subseteq \mathcal{L}^2_\forall$ if there is φ in $\mathcal{L}$ such that for any model $\mathcal{M}$, $\mathcal{M} \models \varphi$ if and only if $\mathcal{M} \in \Omega$. With this in mind, we may regard MFO as a temporal logic in terms of the following.

▶ **Theorem 2** (Kamp [8]). *Let Ω be a set of* LTL *models. Then, Ω is definable in $\mathcal{L}_{\mathsf{U}}$ if and only if it is definable in $\mathcal{L}^1_\forall$.*

There are also known extensions of LTL which are expressively equivalent to full MSO [6], but for our purposes the presentation as MSO is more convenient than such extensions. On the other hand, we will go back and forth between LTL and MFO depending on which is more convenient for the application at hand.

4 Expressibility

In this section we show how the legal articles we have considered could be represented within monadic second order logic. It is crucial to stress that the articles allow for some interpretation and thus certain elements may admit readings different from those we propose. We will also make a few simplifying assumptions for the sake of exposition. From discussions with legal experts we believe that our interpretations are reasonable modulo the aforementioned simplifying assumptions.

We assume that each natural number represents one hour, although we remark that tachograph data is processed[4] minute by minute and this would be the suitable resolution for actual implementations. Each moment in time (each hour in our presentation) is labelled by an activity of the driver: these activities are *driving, resting, availability, other work* and *unknown*.[5]

[4] Actually, tachograph data is recorded second by second and legally interpreted minute by minute. The law prescribes how minutes should be labelled depending on the tachograph data. We refer the reader to [4] for details and for various mathematical problems with this labelling.

[5] The value of *unknown* is not prescribed by the law, but it is implemented in various systems for obvious reasons.

Since the articles we analyse in this paper only involve rest periods, we consider a predicate symbol `Rest`. We also introduce a predicate symbol `Week` which holds on the first hour of each Monday. This condition can be treated model-theoretically – i.e. models are assumed to be equipped with a correct valuation for `Week` – or syntactically by the $\mathcal{L}_\Box$ axiom

$$\texttt{Week} \wedge \Box\big(\texttt{Week} \to (\bigcirc\Box^{<167}\neg\texttt{Week} \wedge \bigcirc^{168}\texttt{Week})\big)$$

(assuming that the model begins on the first hour of a Monday). LTL models satisfying this formula at zero are called *weekly models.* With this in mind, we proceed to illustrate how the legislation could be formalized. However, since we want to isolate possible sources of impredicativity, we will work with simplified variants of the legislation that are more suitable for expository purposes.

4.1 Article §8.9

Article §8.6 requires that each two week period be assigned two rest periods with some additional constraints, and §8.9 indicates how rest periods should be assigned to specific weeks. Our goal in this subsection is to explore the possible impredicativity arising from the assignment itself, independently of the additional conditions of §8.6. Every week should contain at least one 24 hour rest period, but this by itself would not be sufficient to comply with §8.6. On the other hand, a driver resting 45 hours each week would comply with §8.6, so this would be a sufficient, but not necessary, condition for compliance. In order to not commit to either condition, we will consider the following general property: when is it that each week can be assigned a rest period of at least d hours, so that each rest period intersects the week it is assigned to? This simplified condition is already *prima facie* impredicative, as it requires a function mapping rest intervals to weeks. Thus it may be surprising that it can actually be defined in first order logic (and hence in LTL).

▶ **Theorem 3.** *Given $d \in [2, 85]$ there is an $\mathcal{L}^1_\forall$-formula $\varphi = \varphi_d \in \mathcal{L}^1_\forall$ such that given any* LTL *model $\mathcal{M}$, $\mathcal{M} \models \varphi$ if and only if there is an assignment of weekly rest periods such that every week is assigned a rest period of length at least d.*

Proof. In this proof we will assume that variables range over weeks. It is clear that using our fundamental ontology this can be established in first order logic, as a week can be identified with its starting point, which is already marked by the predicate `Week`. Let $E(x)$ be a formula which holds if and only if x is a week with an *early* rest period (of length at least d) which means that it overlaps with the previous week, $L(x)$ a formula that holds if x contains a *late* rest period overlapping with the following week, and $I(x)$ be a formula that holds if and only if x is a week with an *internal* rest period disjoint from (but possibly contiguous with) any early or late rest periods in the week x.

Clearly E, I, L are first-order definable (although their definition depends on d). The condition $d \leq 85$ ensures that if $E(x) \wedge L(x)$ holds then the week x contains disjoint early and late rest periods.[6] Define $\check{E}(x) = E(x) \wedge \neg I(x) \wedge \neg L(x)$, and define $\check{I}(x)$, $\check{L}(x)$ analogously.

Then set

$$\varphi = \forall\, x\, \big(E(x) \vee I(x) \vee L(x)\big) \;\wedge\; \forall\, x\, \forall\, y\, \big(x < y \wedge \check{L}(x) \wedge \check{E}(y) \to \exists\, z \in (x, y)\ I(z)\big).$$

[6] If $E(x)$ then there are at most $d - 1 = 84$ hours in x and likewise for $L(x)$. In a week there are $7 \times 24 = 84 \times 2$ hours.

We claim that φ holds if and only if there is an assignment such that each week is assigned one rest period of length at least d. First assume that such an assignment exists. Clearly $\forall x \left(E(x) \vee I(x) \vee L(x)\right)$ holds, since if x were a counterexample no rest period could be assigned to the week of x.

Now, suppose that $x < y$ are such that $\check{L}(x) \wedge \check{E}(y)$, and choose x, y such that $y - x$ is minimal. Note that x is assigned to its late rest period (as this is the only one available) and y is assigned to its early rest period. It follows that there is a least $z \in (x, y]$ that is not assigned to its late rest period. By minimality $z - 1$ is assigned to its late rest period, hence z cannot be assigned to its early rest period. However, z must be assigned to *some* rest period by assumption, and since this rest period is neither early nor late, the week of z must contain some internal rest period, and $I(z)$ holds.

Now assume that φ holds and define an assignment recursively as follows. Let R be a rest period and suppose that all earlier rest periods have been assigned to some week. If R is internal, assign it to its current week. If R is late for week w and w has not been assigned a rest period, assign R to w. Otherwise, assign R to $w + 1$.

We prove by induction that every week is assigned to some rest period. Fix y and assume that all earlier weeks have been assigned to some period. We may assume that $E(y) \vee I(y) \vee L(y)$ holds, as otherwise φ automatically fails.

If $I(y)$ holds then the week of y has a rest period assigned to it. If $L(y)$ holds then the late rest period of y is assigned to it, unless an earlier one was already assigned to it. So we are left with the hypothetical case where $\check{E}(y)$ holds, and the early rest period of y has been assigned to $y - 1$. Let $x < y$ be minimal with the property that every $z \in [x, y)$ has had its late rest period assigned to it. First note that $E(x)$ fails, since otherwise $x > 0$ and either $x - 1$ has had its late rest period assigned to it, contradicting the minimality of x, or else the early rest period of x would have been assigned to the week of x by our recursion. Note also that $I(z)$ fails for all $z \in [x, y)$, since any internal rest period is automatically assigned to the current week. We conclude that $\check{L}(x)$ holds and $I(z)$ fails for all $z \in (x, y)$, thus φ fails. ◀

4.2 Article §8.6

Now that we have seen that the possibility of assigning rest periods is not itself impredicative, we isolate the compensation mechanism from the rest assignments and analyse it in a similar fashion. In order to do this, we work with *simple* models defined as the set of models $\mathcal{M}$ satisfying the following conditions.

- $\mathcal{M}$ is a weekly model.
- $(\mathcal{M}, 0) \models \Diamond\Box\texttt{Rest} \wedge \Box(\texttt{Week} \wedge \texttt{Rest} \to \Box\texttt{Rest})$.
- Given a week W, $W \cap \texttt{Rest}^{\mathcal{M}}$ is an interval.
- There are never more than 6×24 hours between two consecutive rest periods.

The idea is that all rest periods are internal (the first hour of a week is never spent resting), and so every week can unambiguously be assigned a rest period, until the driver "retires" and rests on all subsequent moments. We impose this condition to clarify that our constructions do not require "immortal" drivers. As stated previously, additional daily rest periods are needed to fully comply with regulations, but these will be ignored for the sake of exposition.

We claim that Article §8.6 admits a formalization in $\mathcal{L}^2_\forall$ by a Σ^1_1 formula over the class of simple models. Let R be a variable meant to denote the union of all continuous rest periods of more than nine hours and C_1, C_2, and C_3 be variables meant to denote periods of compensation: C_1 compensates the previous weekly rest, C_2 compensates the weekly rest of two weeks ago, and C_3 compensates the weekly rest of 3 weeks ago. If W is a week, let $S(W)$ be the successor week to W. We express §8.6 by a formula $\psi_{\S 8.6} := \exists R \, \exists C_1 \, \exists C_2 \, \exists C_3 \, \psi^0_{\S 8.6}$, where $\psi^0_{\S 8.6}$ expresses a conjunction of the following conditions:

- $C_i \cap C_j = \varnothing$ if $1 \leq i < j \leq 3$.
- $\bigcup_{i=1}^{3} C_i \subseteq R$.
- Given a week W, $R \cap W$ is an interval of length at least 24.
- Given a week W and $i \in \{1,2,3\}$, $C_i \cap W$ is an interval. Moreover, if $C_i \cap S^i(W) \neq \varnothing$, then $C_j \cap S^j(W) = \varnothing$ for all $j \in \{1,2,3\} \setminus \{i\}$.
- Given a week W,

$$\left| \left((R \setminus \bigcup_{i=1}^{3} C_i) \cap W \right) \cup \bigcup_{i=1}^{3} \left(C_i \cap S^i(W) \right) \right| \geq 45.$$

It should be clear that each of these conditions is first order definable, hence $\psi_{\S 8.6}$ is Σ^1_1. Moreover, some inspection shows that over the set of simple models, $\psi_{\S 8.6}$ coincides with §8.6. We conclude that §8.6 admits a Σ^1_1 formalization over the set of simple models, as claimed.

▶ Remark 4. It is also possible to formalize §8.6 over the class of all weekly models using a similar Σ^1_1 formula. We restrict our attention to simple models only because the general formalization would be more cumbersome and no more illuminating.

5 Stratified Bisimulations

In this section we present a version of stratified bisimulations for $\mathcal{L}_{\mathsf{U}}$ proposed by Kurtonina and de Rijke [9]. Since all languages we consider contain Booleans and $\bigcirc$, it is convenient to begin with a "basic" notion of bisimulation for this language.

▶ **Definition 5.** *Given $k \geq 0$ and two LTL models $\mathcal{M}$ and $\mathcal{N}$, a binary relation $Z \subseteq \mathbb{N}^2$ is a k-$\bigcirc$-bisimulation (between $\mathcal{M}$ and $\mathcal{N}$) if whenever $x\ Z\ y$, $P \in \mathbb{P}$, and $j \leq k$, we have $x + j \in P^{\mathcal{M}}$ iff $y + j \in P^{\mathcal{N}}$.*

We will use bounded $\bigcirc$-bisimulations as a basis to define bounded bisimulations for more powerful languages.

▶ **Definition 6.** *Fix $k \geq 0$ and two LTL models $\mathcal{M}$ and $\mathcal{N}$. Let $\vec{Z} = (Z_i)_{i=0}^{\infty}$ be a sequence such that for all $i \in \mathbb{N}$, Z_i is a k-$\bigcirc$-bisimulation and $Z_{i+1} \subseteq Z_i$.*

1. *$\vec{Z}$ is a k-$\Box$-bisimulation (between $\mathcal{M}$ and $\mathcal{N}$) if whenever $x\ Z_{i+1}\ y$:*
   - **Forth $\Box$.** *For all $x' \geq x$ there exists $y' \geq y$ such that $x'\ Z_i\ y'$.*
   - **Back $\Box$.** *For all $y' \geq y$ there exists $x' \geq x$ such that $x'\ Z_i\ y'$.*
2. *$\vec{Z}$ is a k-U-bisimulation (between $\mathcal{M}$ and $\mathcal{N}$) if whenever $x\ Z_{i+1}\ y$:*
   - **Forth U.** *For all $x' \geq x$ there exists $y' \geq y$ and a function $\xi\colon [y, y'] \to [x, x']$ such that every $z \in [y, y']$ satisfies $\xi(z)\ Z_i\ z$ and $\xi(z) = x'$ if and only if $z = y'$.*
   - **Back U.** *For all $y' \geq y$ there exists $x' \geq x$ and a function $\eta\colon [x, x'] \to [y, y']$ such that every $z \in [x, x']$ satisfies $z\ Z_i\ \eta(z)$ and $\eta(z) = y'$ if and only if $z = x'$.*

Stratified bisimulations are an essential tool in proving inexpressivity or succinctness results, given that they preserve the truth of formulas of small enough nesting depth.

▶ **Lemma 7** ([9]).

1. *Given two LTL models $\mathcal{M}$ and $\mathcal{N}$ and a k-$\Box$-bisimulation $\vec{Z}$ between them, for all formulas $\varphi \in \mathcal{L}_{\Box}$ and for all $(x, y) \in Z_i$, if φ has $\bigcirc$-depth at most k and $\Box$-depth at most i then $(\mathcal{M}, x) \models \varphi$ iff $(\mathcal{N}, y) \models \varphi$.*
2. *Given two LTL models $\mathcal{M}$ and $\mathcal{N}$ and a k-U-bisimulation $\vec{Z}$ between them, for all formulas $\varphi \in \mathcal{L}_{\mathsf{U}}$ and for all $(x, y) \in Z_i$, if φ has $\bigcirc$-depth at most k and U-depth at most i then $(\mathcal{M}, x) \models \varphi$ iff $(\mathcal{N}, y) \models \varphi$.*

In the next section we use Lemma 7 to show that certain legal properties we have considered are hard or impossible to define in fragments of linear temporal logic.

6 Non-expressibility

We have seen that Articles §8.9 and §8.6 are expressible in MFO and MSO, respectively. We will see that they are not expressible in $\mathcal{L}_{\Box}$ and that §8.6 is not reasonably expressible in $\mathcal{L}_{\mathsf{U}}$, in the sense that any formula expressing it (if it exists) would require very large U-depth. For this, we use constructions similar to the examples given in Section 2. However, since these constructions will be somewhat more elaborate, we settle some notation first.

Say that a model $\mathcal{M}$ is *eventually resting* if there is some m such that for all $n > m$ and all $P \in \mathbb{P}$, $n \in P^{\mathcal{M}}$ iff $P = \texttt{Rest}$. The *end* of an eventually resting model is the least such value of m which is also a multiple of 168 (i.e., a whole number of weeks). A week-long model is an eventually resting models whose end is 168. We define the *concatenation* of two eventually resting models $\mathcal{A}, \mathcal{B}$, denoted $\mathcal{A} \mid \mathcal{B}$, as follows. Let m be the end of $\mathcal{A}$. Then, for a predicate symbol P and $n \in \mathbb{N}$, we set

$$n \in P^{\mathcal{A}|\mathcal{B}} \Leftrightarrow \begin{cases} n \in P^{\mathcal{A}} & \text{if } n \leq m \\ n - m \in P^{\mathcal{B}} & \text{if } n > m. \end{cases}$$

If k is a natural number then $\mathcal{A}^k$ denotes k concatenated copies of $\mathcal{A}$. If $n \in [24, 168)$, then n denotes a week with one weekly resting period of n; we assume that these weekly periods fall in the middle of each week without overlapping with other weeks, with the details being non-essential. However, we do assume that any two instances of the week represented by n are identical.

It will be convenient to represent a given moment in time both by the number of hours t since the beginning of time, and by $168w + h$, where w is the number of weeks since the beginning of time, and $h < 168$ is the number of hours since the beginning of that week.

6.1 Article §8.9

We have seen that the possibility of assigning weekly rest periods to each week is first order definable. One may then ask if $\mathcal{L}_{\Box}$ suffices to define it, and the answer is negative. We prove this via the following construction.

▶ **Definition 8.** *Fix $d \in [24, 84]$. Define the following week-long models:*

- *E is a model whose first $\lfloor d/2 \rfloor$ hours are resting.*
- *I is a model whose hours $(\lfloor d/2 \rfloor + 1, \lfloor d/2 \rfloor + d)$ are resting.*
- *L is a model whose last $\lceil d/2 \rceil$ hours are resting.*
- *Concatenations of letters denote unions of resting hours, i.e., EL denotes a week with a beginning and an end rest period.*

Then, for each $n \in \mathbb{N}$, define the eventually resting models $\mathcal{A}_n = (L \mid EL^n \mid EIL \mid EL^n \mid E)^{n+1}$ and $\overline{\mathcal{A}}_n = L \mid EL^n \mid E \mid \mathcal{A}_n$.

Given $d \in [24, 84]$ and a model $\mathcal{M}$, we say that $\mathcal{M}$ *admits a weekly rest assignment* if it is possible that each week is assigned a weekly rest period of length at least d.

▶ **Lemma 9.** *The model $\mathcal{A}_n$ admits a weekly rest assignment but $\overline{\mathcal{A}}_n$ does not.*

Proof. It is easy to see that $\mathcal{A}_n$ satisfies the formula φ_d of Theorem 3 and that $\overline{\mathcal{A}}_n$ does not. ◀

▶ **Lemma 10.** *There is a bounded $168n$-$\Box$-bisimulation $\vec{Z}$ between $\mathcal{A}_n$ and $\overline{\mathcal{A}}_n$ such that $0\ Z_n\ 0$.*

Proof sketch. Define $r := 2n+3$ and for $x = 168w + h, y = 168v + \ell \in \mathbb{N}$, let $x\ Z_i\ y$ if $h = \ell$ and one of the following holds:
A1. $x = y = 0$ and $i \leq n$,
A2. $0 < v$, $\max\{w, v - n - 2\} \leq (n-i)r$ and $v \equiv w + n + 2 \pmod{r}$, or
A3. $v = w + n + 2$.
Then $\vec{Z}$ is a bisimulation (see Appendix A) and $0\ Z_n\ 0$. ◀

▶ **Theorem 11.** *Given $d \in [24, 84]$, there is no $\mathcal{L}_\Box$ formula φ such for every model $\mathcal{M}$, $\mathcal{M} \models \varphi$ if and only if $\mathcal{M}$ admits a weekly rest assignment.*

Proof. Suppose that $\varphi \in \mathcal{L}_\Box$ is such a formula. Let $d_\bigcirc$ and $d_\Box$ be its $\bigcirc$-depth and $\Box$-depth, respectively. Choose n such that $d_\bigcirc \leq 168n$ and $d_\Box \leq n$. Then by Lemmas 7 and 10, $(\mathcal{A}_n, 0) \models \varphi$ iff $(\overline{\mathcal{A}}_n, 0) \models \varphi$. But, according to Lemma 9, $\mathcal{A}_n$ admits a weekly rest assignment, while $\overline{A}_n$ does not. ◀

6.2 Article §8.6

Our goal now is to show that Article §8.6 is not expressible in $\mathcal{L}_\Box$, and that it needs a formula with a large U-depth to express it in $\mathcal{L}_\mathsf{U}$. As before, we start by defining a model that complies with the article, and one that doesn't, and then prove that they are bisimilar.

▶ **Definition 12.** *For each $n \in \mathbb{N}$, define simple models*

$$\mathcal{B}_n = (44 \mid 45^n \mid 46 \mid 45^n)^n \mid 24 \mid 45 \mid 24$$

and $\overline{\mathcal{B}}_n = 44 \mid 45^n \mid \mathcal{B}_n$.

▶ **Lemma 13.** *Given $n \in \mathbb{N}$, $\mathcal{B}_n \models \psi_{\S 8.6}$ but $\overline{\mathcal{B}}_n \not\models \psi_{\S 8.6}$.*

Proof. In $\mathcal{B}_n$, the first week's missing hour can be compensated on the third week. This creates a chain reaction of compensations, as the third week also needs to be compensated (because it's interpreted as a reduced rest of 44 hours together with a compensation of 1 hour). However, it is always possible to compensate either two weeks after, or on the week of 46 hours, if it is close enough. It is thus never necessary to use up hours from the second block of n 45 hour rest weeks, which are all regular rest periods. This process happens n times, until we reach the last three weeks of the model. Two of them need to be compensated, but it is possible to do so using the unlimited hours of rest available after the end.

Consider now $\overline{\mathcal{B}}_n$. The 24 hour weeks near the end of the model cannot be used to compensate previous weeks, since 24 is the minimum allowed weekly rest. The last 45 hour week cannot be used to compensate previous weeks either, because then there would be more than one consecutive week with no regular rest period. Thus, we erase the last three weeks from consideration. There are $m := 2n^2 + 3n + 1$ weeks in the rest of the model, $2n^2 + n$ of which have 45 rest hours, $n + 1$ of which have 44 rest hours, and n of which have 46 hours, for a total of $45m - 1$ rest hours. Thus there are not enough rest hours to distribute among the period such that each week is assigned 45 hours of weekly rest. ◀

▶ **Lemma 14.** *There is a bounded $168n$-$\Box$-bisimulation $\vec{Z}$ between $\mathcal{B}_n$ and $\overline{\mathcal{B}}_n$ such that $0\ Z_n\ 0$.*

Proof. The stratified bisimulation and the proof are analogous to those used in the proof of Lemma 10. ◀

▶ **Theorem 15.** *There is no $\mathcal{L}_\Box$-formula equivalent to $\psi_{\S 8.6}$ over the class of simple models.*

Proof. Suppose that $\psi \in \mathcal{L}_{\Box}$ is a formula expressing Article §8.6 with $\bigcirc$-depth $d_{\bigcirc}$ and $\Box$-depth $d_{\Box}$. Choose n big enough to ensure that $d_{\bigcirc} \leq 168n$ and $d_{\Box} \leq n$, and let $\vec{Z}$ be the bisimulation of Lemma 14. Then by Lemma 7, $(\mathcal{B}_n, 0) \models \psi$ iff $(\overline{\mathcal{B}}_n, 0) \models \psi$. This contradicts Lemma 13. ◀

Now we show that any formula of $\mathcal{L}_{\mathsf{U}}$ requires nesting depth 20 of U.

▶ **Definition 16.** *For $n \in \mathbb{N}$, define models $\mathcal{C}_n = (44 \mid 45^{2n+1})^{21} \mid 66 \mid 24 \mid 45 \mid 24$ and $\overline{\mathcal{C}}_n = (44 \mid 45^{2n+1}) \mid \mathcal{C}_n$.*

▶ **Lemma 17.** *Given $n \in \mathbb{N}$, $\mathcal{C}_n \models \psi_{\S 8.6}$ but $\overline{\mathcal{C}}_n \not\models \psi_{\S 8.6}$.*

Proof. First we see that $\mathcal{C}_n \models \psi_{\S 8.6}$. Intuitively, even weeks are compensated two weeks later, and the size of the compensation increases by one every $2n+2$ weeks. Thus for example one hour of week 0 is compensated by one hour of week 2, which is compensated by one hour of week 4, and so on until we reach week $2n+2$. Note however that this week only has 44 hours of rest and has used one hour to compensate the previous week, so we need to compensate two hours of rest. This is compensated by two hours on week $2n+4$, and so on until we reach the third 44 hour rest. Since two hours of this rest are used to compensate a previous week, now three hours need to be compensated, and so on. On week $21(2n+2)$ we use 21 hours to compensate, which is the maximum allowed given that each week requires a 24 hour rest period. As before, the last $24 \mid 45 \mid 24$ block cannot be used to compensate, but can be compensated with the following unlimited rest.

More formally, every week w numbered $2k$ (including week zero) will be reduced and compensated by week $2k+2$, up to and including week $21(2n+2)$. The amount of the compensation is the unique $i > 0$ such that $(i-1)(2n+2) \leq w < i(2n+2)$.

As in $\overline{\mathcal{B}_n}$, the $24 \mid 45 \mid 24$ block at the end of $\overline{\mathcal{C}}_n$ cannot be used to compensate previous weeks (see the proof of Lemma 13). There are $m := 22(2n+2)+1$ remaining weeks in $\overline{\mathcal{C}}_n$, of which $22(2n+1)$ have 45 resting hours, 22 have 44 resting hours, and 1 has 66 resting hours, for a total of $45m-1$ resting hours. Thus there are not enough resting hours to distribute among the weeks. ◀

▶ **Lemma 18.** *There is a bounded $168n$-U-bisimulation $\vec{Z}$ between $\mathcal{C}_n$ and $\overline{\mathcal{C}}_n$ such that $0\ Z_{20}\ 0$.*

Proof sketch. Let $r := 2n+2$. For $168w+h \in \mathcal{C}_n$ and $168v+\ell \in \overline{\mathcal{C}}_n$, set $168w+h\ Z_i\ 168v+\ell$ if $h = \ell$ and one of the following holds:
C1. $\max\{w+r, v\} < (21-i)r$ and $w \equiv v \pmod r$;
C2. $v = w+r$.
Then, $\vec{Z}$ is a stratified bisimulation (see Appendix A) and $0\ Z_{20}\ 0$ by C1. ◀

▶ **Theorem 19.** *Any $\mathcal{L}_{\mathsf{U}}$ formula equivalent to $\psi_{\S 8.6}$ has U-depth at least 20.*

Proof. Suppose that $\psi \in \mathcal{L}_{\mathsf{U}}$ is a formula expressing Article §8.6 with $\bigcirc$-depth d and U-depth less than 20. Choose n big enough to ensure that $d \leq n$, and let $\vec{Z}$ be the bisimulation of Lemma 18. Then by Lemma 7, $(\mathcal{C}_n, 0) \models \psi \iff (\overline{\mathcal{C}}_n, 0) \models \psi$. This contradicts Lemma 17. ◀

▶ Remark 20. One can ask how Theorem 19 would differ if we included "since" in the language. In this case, $(\mathcal{C}_n, 0)$ and $(\overline{\mathcal{C}}_n, 0)$ are only about 10-bisimilar. However, the nesting depth of 20 is determined only by the resolution of our models. If instead we used a minute-wise

resolution (which, as we have mentioned, is the resolution required by the law itself), we could stretch this to 20×60 by replacing the 44 hour reduced weekly rests by $44 : 59$ reduced weekly rests. Thus any LTL definition of $\psi_{\S 8.6}$ would have to exploit the temporal resolution in an essential way, making it arguably unnatural.

7 Concluding Remarks

We have shown that the Σ_1^1 fragment of monadic temporal logic is sufficient for formalizing even the most problematic passages we have found in our study of European transport regulations. The upshot is that evaluating whether a given truck driver's record complies with regulations can then be transformed into a model-checking problem over this fragment. Moreover, truth of Σ_1^1 MSO formulas is equivalent to validity for MSO, and via Kamp's theorem we may further reduce it to validity of LTL formulas, for which many algorithms and solvers are already available. Nevertheless, validity in LTL is PSPACE-complete, and moreover the translation of MSO into LTL is non-elementary in the worst case, so this approach is not ideal from a complexity perspective.

On the other hand, LTL is indeed suitable for formalizing portions of the regulation, and in this case the model-checking problem (over deterministic models) is polynomial [5]. In fact, the advantage of having such a general tool available can be viewed as an argument to use "sugared" versions of LTL (say, with counting modalities) in the design of future – and revision of current – laws.

Indeed, consider the following variant of §8.6:

- In every two consecutive weeks, the driver must take two weekly rest periods, at least one of which is regular.
- In every four consecutive weeks, the sum of the weekly rest periods must be of at least 180 hours.

This version of the article can be easily checked to be definable by a not-too-large LTL formula and maintain the spirit of the original, as drivers are required to compensate reduced rest periods within the following three weeks.

A second issue concerns the use of classical logic. This is especially relevant when the law is ambiguous or contradictory, or driving records are incomplete. Up to now our team has found classical logic to be sufficient for our intended applications, but it is possible that some non-classical temporal logic (as in e.g. [1, 7]) will turn out to be the "right" foundation for modelling these regulations.

References

1 J. Boudou, M. Diéguez, and D. Fernández-Duque. A Decidable Intuitionistic Temporal Logic. In *26th EACSL Annual Conference on Computer Science Logic (CSL)*, pages 14:1–14:17, 2017.

2 J. del Castillo Tierz. When the laws of logic meet the logic of laws. Master's thesis, University of Barcelona, Barcelona, 2018. URL: `http://diposit.ub.edu/dspace/handle/2445/133778`.

3 European Parliament and Council of the European Union. Regulation (EC) No 561/2006 of the European Parliament and of the Council of 15 March 2006 on the harmonisation of certain social legislation relating to road transport and amending council regulations (EEC) No 3821/85 and (EC) No 2135/98 and repealing council regulation (EEC) No 3820/85 (text with EEA relevance) - declaration. Official Journal of the European Union, 2006. URL: `https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32006R0561`.

4 D. Fernández-Duque, M. González Bedmar, D. Sousa, J.J. Joosten, and G. Errezil Alberdi. To Drive or Not to Drive: A Formal Analysis of Requirements (51) and (52) from Regulation (EU) 2016/799. Submitted, 2019.

5 D. Harel, J. Tiuryn, and D. Kozen. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000.

6 J.G. Henriksen and P.S. Thiagarajan. Dynamic Linear Time Temporal Logic. *Annals of Pure and Applied Logic*, 96(1-3):187–207, 1999. doi:10.1016/S0168-0072(98)00039-6.

7 N. Kamide and H. Wansing. A Paraconsistent Linear-time Temporal Logic. *Fundamenta Informaticae*, 106(1):1–23, 2011. doi:10.3233/FI-2011-374.

8 H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, UCLA, 1968.

9 N. Kurtonina and M. de Rijke. Bisimulations for Temporal Logic. *Journal of Logic, Language and Information*, 6(4):403–425, 1997.

10 O. Lichtenstein and A. Pnueli. Propositional Temporal Logics: Decidability and Completeness. *Logic Jounal of the IGPL*, 8(1):55–85, 2000.

11 H. Weyl. *The Continuum: A Critical Examination of the Foundation of Analysis*. Mineola: Dover, 1994.

A Bisimulation proofs

Recall the models $\mathcal{A}_n = (L \mid EL^n \mid EIL \mid EL^n \mid E)^{n+1}$ and $\overline{\mathcal{A}}_n = L \mid EL^n \mid E \mid \mathcal{A}_n$.

▶ **Lemma 10.** *There is a bounded* $168n$*-*$\Box$*-bisimulation* $\vec{Z}$ *between* $\mathcal{A}_n$ *and* $\overline{\mathcal{A}}_n$ *such that* $0\ Z_n\ 0$.

Proof. Define $r := 2n+3$ and for $x = 168w + h, y = 168v + \ell \in \mathbb{N}$, let $x\ Z_i\ y$ if $h = \ell$ and one of the following holds:

A1. $x = y = 0$ and $i \leq n$,

A2. $0 < v$, $\max\{w, v-n-2\} \leq (n-i)r$ and $v \equiv w + n + 2 \pmod r$, or

A3. $v = w + n + 2$.

We need to show that $\vec{Z}$ is a stratified bisimulation.

It is clear that $Z_{i+1} \subseteq Z_i$. Assume that $x\ Z_i\ y$ and write $x = 168w + h$, $y = 168v + \ell$; note that by definition we must have $h = \ell$. If $i = 0$, then some inspection shows that x and y share the same formulas of the form $\bigcirc^j p$ with $j \leq 168n$, since the current and subsequent n weeks are of the same form. It is sufficient to check this for Z_0 because it contains all the Z_i.

Otherwise, change variables so that $x \sim_{i+1} y$; we check that the required clauses hold.

Forth $\Box$. Let $x' \geq x$ and write $x' = 168w' + h'$. We claim that there is v' such that $168v' + h' \geq 168v + h$ and $168w' + h'\ Z_i\ 168v' + h'$. If $168(w' + n + 2) + h' \geq 168v + h$ we may take $v' = w' + n + 2$, and the bisimulation holds by A3. Otherwise, we have $v \geq w' + n + 2 \geq w + n + 2$, where the first inequality is strict unless $h' < h$, in which case the second inequality must be strict. Hence x, y do not satisfy A1 nor A3 and thus $\max\{w, v-n-2\} \leq (n-i-1)r$. Take $v' \in (v, v+r]$ with $v' \equiv w' + n + 2 \pmod r$ and set $y' = 168v' + h'$. Note that $w' + n + 2 \leq v < (n-i-1)r + n + 2$ yields $w' \leq (n-i)r$, while $v' \leq v + r \leq (n-i-1)r + r = (n-i)r$, and thus $v - n - 2 \leq (n-i)r$ as well. Thus $x'\ Z_i\ y'$ by A2.

Back $\Box$. Let $y' \geq y$ and write $y' = 168v' + h'$. As before, we claim that there is w' such that $168w' + h' \geq 168w + h$ and $168w' + h'\ Z_i\ 168v' + h'$. If $168(v' - n - 2) + h' \geq 168w + h$ we may take $w' = v' - n - 2$, and the result follows by A3. Otherwise, we have $w \geq v' - n - 2 \geq v - n - 2$ with one inequality being strict , so that x, y do not satisfy A3. If x, y satisfy A2, then $\max\{w, v-n-2\} \leq (n-i-1)r$. If x, y satisfy A1, we have that $w = v = 0$ and $i + 1 \leq n$, so that $\max\{w, v-n-2\} = 0 \leq (n-i-1)r$ as well. Take $w' \in (w, w+r]$ with $w' + n + 2 \equiv v' \pmod r$ and set $x' = 168w' + h'$. It is not hard to check that $\max\{w', v'-n-2\} \leq (n-i)r$, and thus $x'\ Z_i\ y'$ by A2. ◀

Recall the models $\mathcal{C}_n = (44 \mid 45^{2n+1})^{21} \mid 66 \mid 24 \mid 45 \mid 24$ and $\overline{\mathcal{C}}_n = (44 \mid 45^{2n+1}) \mid \mathcal{C}_n$.

▶ **Lemma 18.** *There is a bounded* $168n$*-U-bisimulation* $\vec{Z}$ *between* $\mathcal{C}_n$ *and* $\overline{\mathcal{C}}_n$ *such that* $0\ Z_{20}\ 0$.

Proof. Recall that we defined $r := 2n+2$ and for $168w + h \in \mathcal{C}_n$ and $168v + \ell \in \overline{\mathcal{C}}_n$, we set $168w + h\ Z_i\ 168v + \ell$ if $h = \ell$ and one of the following holds:
C1. $\max\{w + r, v\} < (21 - i)r$ and $w \equiv v \pmod r$, or
C2. $v = w + r$.
We need to show that $\vec{Z}$ is a stratified bisimulation. To see this, assume that $x\ Z_i\ y$ and write $x = 168w + h$, $y = 168v + \ell$. Note that we must have $h = \ell$. If $i = 0$ then some inspection shows that x and y share the same formulas of the form $\bigcirc^j p$ with $j \leq 168n$, as the current and subsequent n weeks are of the same form. It is sufficient to check this for Z_0 because it contains all the Z_i.

If $i > 0$, change variables so that $x \sim_{i+1} y$; we check that the required clauses hold.

Forth U. Let $x' \geq x$ and write $x' = 168w' + h'$. Consider two cases. First assume that $w' \leq w + r$. Set $y' = y + (x' - x)$ and for $z \in [y, y']$ set $\xi(z) = x + (z - y)$. It is then not hard to check that if $x\ Z_{i+1}\ y$ by C1 then $\xi(z)\ Z_i\ z$ by C1, and similarly if $x\ Z_{i+1}\ y$ by C2 then $\xi(z)\ Z_i\ z$ by C2. The other required properties of ξ are easy to check, so that ξ witnesses Forth U.

Otherwise $w' > w + r$. We claim that there is v' such that $168v' + h' \geq 168v + h$ and $168w' + h'\ Z_i\ 168v' + h'$. If $168(w' + r) + h' \geq 168v + h$ we may take $v' = w' + r$. Otherwise, we have $v \geq w' + r > w + r$ so that x, y do not satisfy C2 and thus $\max\{w + r, v\} \leq (21 - i - 1)r$. Take $v' \in (v, v + r]$ with $v' \equiv w' \pmod r$ and set $y' = 168v' + h'$; from $(21 - i - 1)r \geq v \geq w' + r$ and $v' \leq v + r \leq (21 - i)r$ we obtain $x'\ Z_i\ y'$ by C1.

We now construct the function $\xi\colon [y, y'] \to [x, x']$. First define $\xi(y') = x'$. For $z = 168u + t \in [y, y')$, we consider two cases. If $168(u-r)+t \in \big[x, x'\big)$ take $\xi(z) = 168(u-r)+t$, which in view of C2 satisfies all desired properties. Otherwise, $168(u - r) + t \notin [x, x')$, and choose $d \in (0, r]$ such that $w + d \equiv u \pmod r$, then set $\xi(z) = 168(w + d) + t$. The assumption that $w' > w + r$ yields $\xi(z) \in [x, x']$. It remains to show that $\xi(z)\ Z_i\ z$, for which it suffices to check that $\max\{w + d + r, u\} < (21 - i)r$.

If $168(u-r)+t < x$ then since $z \geq y$, either $u > v$ and hence $v < u \leq w+r$, or else $u = v$ and $t \geq h$, so that forcibly $u - r < w$ and thus $v < w + r$. But $v < w + r$ together with $168v+h'\ Z_{i+1}\ 168w+h$ means that C1 holds so that $\max\{w+r, v\} < (21-i-1)r$. Thus we have $u-r \leq w < (21-i-1)r$ so that $u < (21-i)r$. Similarly $w+d \leq w+r < (21-i-1)r$ yields $w + d + r < (21 - i)r$.

Otherwise $168(u - r) + t \geq x'$. But then since $z < y'$, either $u = v'$ and hence $t < h'$, so that $v' - r = u - r > w'$; or else $u < v'$ and $v' - r > u - r \geq w'$. Thus $w' + r \neq v'$, which together with $168w' + h'\ Z_i\ 168v' + h'$ yields $\max\{w' + r, v'\} < (21 - i)r$. From $u \leq v' < (21 - i)r$ and $w + d + r \leq (w + r) + r < w' + r < (21 - i)r$ we obtain $\max\{w + d + r, u\} < (21 - i)r$, as needed.

Back U. This is essentially symmetric and we omit it. ◀

Two-Dimensional Rule Language for Querying Sensor Log Data: A Framework and Use Cases

Sebastian Brandt
Siemens CT, München, Germany
sebastian-philipp.brandt@siemens.com

Diego Calvanese
KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
calvanese@inf.unibz.it

Elem Güzel Kalaycı
KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
kalayci@inf.unibz.it

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

Benjamin Mörzinger
Technische Universität Wien, Austria
moerzinger@ift.at

Vladislav Ryzhikov
Department of Computer Science and Information Systems, Birkbeck, University of London, UK
vlad@dcs.bbk.ac.uk

Guohui Xiao
KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
xiao@inf.unibz.it

Michael Zakharyaschev
Department of Computer Science and Information Systems, Birkbeck, University of London, UK
Faculty of Computer Science, National Research University Higher School of Economics, Moscow, Russia
michael@dcs.bbk.ac.uk

Abstract

Motivated by two industrial use cases that involve detecting events of interest in (asynchronous) time series from sensors in manufacturing rigs and gas turbines, we design an expressive rule language *DslD* equipped with interval aggregate functions (such as weighted average over a time interval), Allen's interval relations and various metric constructs. We demonstrate how to model events in the uses cases in terms of *DslD* programs. We show that answering *DslD* queries in our use cases can be reduced to evaluating SQL queries. Our experiments with the use cases, carried out on the Apache Spark system, show that such SQL queries scale well on large real-world datasets.

2012 ACM Subject Classification Computing methodologies → Ontology engineering; Computing methodologies → Temporal reasoning; Theory of computation → Modal and temporal logics

Keywords and phrases Ontology-based data access, temporal logic, sensor log data

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.7

Funding This work was supported by the EPSRC UK grant EP/S032282. The work of M. Zakharyaschev was carried out while visiting the National Research University Higher School of Economics and supported by the Russian Science Foundation under the grant 17-11-01294.

1 Introduction

Our concern in this paper is spotting events a domain expert is interested in among time-stamped sensor log data stored in a database. The amount of sensor data produced in all areas of modern life is proliferating with a tremendous speed, probably even faster than the volume of social media data. In the industrial sector, in particular manufacturing, sensor data is crucial for monitoring, control, various types of analytics (descriptive, predictive and prescriptive), decision-making as well as research aiming to improve processes and products.

Sensor data is key to the "fourth industrial revolution" in global manufacturing [30, 22], which is witnessed by the German Industrie 4.0 programme and the smart manufacturing initiatives in the United States.

The scenario we are interested in is querying historical (rather than streaming) sensor data stored in a database with the aim of detecting temporal events that can help to explain, predict and improve the behaviour of the system in question. A typical example (cf. [17, 29, 18, 6]) is querying data from gas turbine sensors measuring the active power, rotor speed, blade temperature, etc. and stored at Siemens remote diagnostic centres, where service engineers are looking for events such as

active power trip, which happens when the active power was above 1.5MW for a period of at least 10 seconds, maximum 3 seconds after which there was a period of at least one minute where the active power was below 0.15MW; to avoid short-term fluctuations, the value of active power should be smoothed out by taking the simple moving average of the raw power measurements over the last 5 seconds.

However, accessing data generated by sensors and finding the events of interest is hard because the engineers and researchers with domain expertise necessary to interpret the data usually lack the knowledge required to independently interact with databases and formalise temporal queries such as the one in the example above. Following the current workflow at Siemens, the engineer usually asks an IT specialist to produce a custom-made script (in a proprietary signal processing language) such as

```
message("active power trip") =
      $t1: eval(>, simpleMovingAverage(#activePower, 5s), 1.5):
             for(>= 10s)
           &&
             eval(<, simpleMovingAverage(#activePower, 5s), 0.15):
               start(after[0s, 3s] $t1:end):
                 for(>= 1m);
```

and run it over the sensor data.

A fundamentally different approach is offered by ontology-based data access and management [26, 23, 32] (OBDA, for short), a semantic technology that has been developed over the past decade with the aim of facilitating access to various types of data. The role of ontologies in OBDA is threefold: to integrate distributed and heterogeneous data sources, to enrich incomplete data with background knowledge, and to provide a user-friendly and familiar vocabulary for querying. Unfortunately, however, the existing standard ontology languages are not able to represent temporal events, let alone those that depend on aggregated numerical data. Extending the OBDA paradigm to temporal data has recently become an active research area in semantic technologies [25, 16, 10, 3, 5, 19, 2, 9, 15]. For example, the active power trip event (without smoothing the active power measurements) was captured in the language *datalogMTL*, a combination of metric temporal logic *MTL* with datalog [6]; see formula (2) in Section 3. *MTL* for signal monitoring was also used in [24].

One of the aims of this paper is to design an ontology language that could capture interval aggregate functions such as simple moving average together with various metric constructs. Although aggregates in the context of OBDA have been considered [8, 19, 21], they were allowed only in queries, and so again an IT middleman may be needed to assist the user. In description logics, aggregation features were introduced over concrete domains, but reasoning with them is often undecidable [4]. A first-order aggregate logic was investigated in [13]. For datalog with monotonic aggregates, see [28] and references therein.

The second real-world use case motivating our language concerns the experimental investigation of drilling processes [14]. In manufacturing research, large numbers of experimental drilling processes, such as the one in Fig. 1, are executed using a wide range of different tools

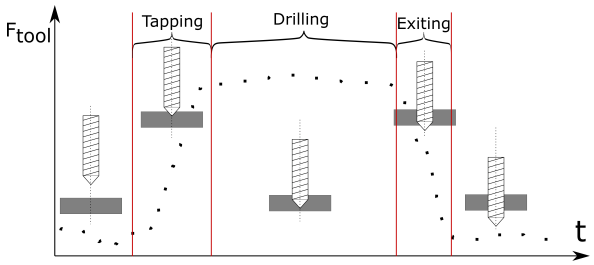


Figure 1 A simplified course of force measurements for a drilling process.

and process parameters. Sensors are used to surveil these processes and generate data, which is then analysed in order to identify, e.g., ways to reduce tool wear, maximise productivity and ensure product quality. To achieve these goals, researchers define intervals of interest (say, processes leading to quality anomalies or those that were interrupted by tool breaks) within such readings and use them to select data for further analysis.

An essential difference from the turbine use case is that, in contrast to the active power trip event described as a sequence of timestamp-value pairs satisfying explicit numerical bounds on their values, now we are looking for certain sequences of *shapes* such as an interval when the force is increasing (tapping), followed by an interval when the force is stable (drilling) and then by an interval when the force is decreasing (exiting). The duration of each of these intervals and the force values may vary widely due to different combinations of workpiece materials, tools, process parameters and tool wear.

To tackle this problem, we further extend our language with the Allen interval relations [1] such as *after* (A), *begins* (B), etc., using which we can capture the normal drilling event with the following rule:

$$\mathit{NormalDrilling}(\boldsymbol{x}) \leftarrow \mathit{Tapping}(\boldsymbol{x}_1),\ \mathit{Drilling}(\boldsymbol{x}_2),\ \mathit{Exiting}(\boldsymbol{x}_3),\\ \boldsymbol{x}_1\ \mathsf{A}\ \boldsymbol{x}_2,\quad \boldsymbol{x}_2\ \mathsf{A}\ \boldsymbol{x}_3,\quad \boldsymbol{x}\ \mathsf{is}\ (\boldsymbol{x}_1 \uplus \boldsymbol{x}_3 \uplus \boldsymbol{x}_3),$$

where $\boldsymbol{x}$ and the $\boldsymbol{x}_i$ are intervals over the real numbers, $\uplus$ returns the union of intervals in case it is also connected, that is, an interval, and the predicates $\mathit{Tapping}(\boldsymbol{x}_1)$, $\mathit{Drilling}(\boldsymbol{x}_2)$ and $\mathit{Exiting}(\boldsymbol{x}_3)$ involve complex interval aggregation to smooth out the fluctuating measurements of force and ensure that it is increasing, stable and decreasing, respectively. An OBDA ontology language, combining datalog with a fragment of the Halpern-Shoham logic *HS* [12], which uses modal operators interpreted by Allen's relations, was introduced in [20, 7].

In this paper, motivated by the sufficiently representative use cases outlined above, we propose a framework of rule-based languages for ontology-mediated queries over sensor log data stored in a database. The framework, provisionally called *DslD* (*datalog for sensor log data*), contains unary predicates for representing events over temporal intervals and binary predicates for capturing numerical sensor measurements over temporal intervals and also the results of aggregation. Thus, *DslD* is essentially two-dimensional, with a temporal dimension comprising intervals over the real numbers $\mathbb{R}$ and a (measurement) value dimension $\mathbb{R}$. *DslD* also contains built-in predicates for expressing quantitative constraints on intervals and values, and the Allen interval relations for representing qualitative constraints. Finally, *DslD* allows one to define built-in interval aggregation operators that, given a (functional) relation, say, representing measurements, returns their moving or weighted average, compute its coalescing (maximal intervals with the same value), etc. Our goal in this paper is to check experimentally, using the two real-world scenarios and data, whether ontology-mediated queries formulated in *DslD* can be evaluated efficiently by standard database engines.

The structure of the paper is as follows. In Section 2, we define the syntax and procedural semantics of *DslD*. In Sections 3 and 4, we write *DslD* programs for the turbine and drilling use cases. In Section 5, we show how to reduce answering queries mediated by *DslD* programs from our use cases to evaluation of SQL queries. In Section 6, we report on testing such SQL translations in our use cases. Finally, we conclude and describe future work in Section 7.

2 Syntax and Semantics of *DslD*

We begin by defining the syntax of the ontology language *DslD*, *datalog for sensor log data*, which is designed to represent data and knowledge about events in sensor-based systems. As the main temporal entity we take the notion of *interval* over the set $(\mathbb{R}, <)$ of real numbers. More precisely, an *interval*, ι, is any nonempty subset of $\mathbb{R}$ of the form $\langle t_1, t_2\rangle$, where $t_1, t_2 \in \mathbb{R} \cup \{-\infty, \infty\}$, "$\langle$" is "(" or "[" and "$\rangle$" is ")" or "]". Note that we admit *punctual* intervals of the form $[t, t]$. We denote by $\mathsf{int}_{\mathbb{R}}$ the set of all intervals over $\mathbb{R}$ and by $|\iota|$ the *length* of a bounded interval ι. Sensor measurements are deemed to be real numbers. We write $R(\iota, v)$ to say that $v \in \mathbb{R}$ is the *value* measured by sensor R over the interval $\iota \in \mathsf{int}_{\mathbb{R}}$, and we write $A(\iota)$ to say that event A occurs in the interval ι. Thus, *DslD* has unary and binary predicate symbols only. Let $A_1, A_2, \dots$ be a list of unary predicate symbols and $R_1, R_2, \dots$ a list of binary predicate symbols in *DslD*; we refer to them as *signature predicates*.

Interval and Value Terms. We distinguish between two sorts of variables and terms. The variables $\boldsymbol{x}, \boldsymbol{y}, \dots$ of sort *interval* range over $\mathsf{int}_{\mathbb{R}}$, while the variables $x, y, \dots$ of sort *value* range over $\mathbb{R}$. *Constants* of sort interval are concrete intervals with rational end-points (from $\mathbb{Q} \cup \{-\infty, \infty\}$). The language contains various (partial) *functions* of sort interval, which include the intersection $\Cap$ and union $\Cup$ of intervals defined by taking

$$\iota \Cap \iota' = \begin{cases} \iota \cap \iota', & \text{if } \iota \cap \iota' \neq \emptyset, \\ \text{undefined}, & \text{otherwise;} \end{cases} \qquad \iota \Cup \iota' = \begin{cases} \iota \cup \iota', & \text{if } \iota \cap \iota' \neq \emptyset, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

The language also includes the length function $|\cdot|$, which is undefined for unbounded intervals; the functions lshift_r and rshift_r, for $r \in \mathbb{Q}$, that shift the left and, respectively, right end of a given interval by r: for example, lshift_{-1} maps every interval $\iota = \langle b, e\rangle$ to $\mathit{lshift}_{-1}(\iota) = \langle b - 1, e\rangle$; the functions lext_r and rext_r that extend the left and, respectively, right end of $\iota = \langle b, e\rangle$ to an interval $\langle b, b + r\rangle$ and, respectively, $\langle e - r, e\rangle$; and possibly other useful operations on intervals. *Terms* of sort interval are built from variables and constants of sort interval using these functions. Likewise, *constants* of sort value are rational numbers; *functions* of sort value include standard $x + y$, $x - y$, $x \times y$, $|x|$, etc.; *terms* of sort value are built from variables and constants of sort value using functions of sort value.

Bullt-in Predicates. Apart from the signature predicates, the language of *DslD* contains a wide range of *built-in predicates*. To begin with, we have the standard $=, \neq, <, \leq$, etc. over $\mathbb{R}$. Next, *DslD* features binary predicates for the Allen interval relations [1] over $\mathsf{int}_{\mathbb{R}}$: *after* ($\mathsf{A}$), *begins* ($\mathsf{B}$), *ends* ($\mathsf{E}$), *during* ($\mathsf{D}$), *later* ($\mathsf{L}$), *overlaps* ($\mathsf{O}$) and their inverses $\bar{\mathsf{A}}$, $\bar{\mathsf{B}}$, etc. Finally, *DslD* has unary *discretisation predicates* $D^{\langle\rangle}_{s,b}$, for $s, b \in \mathbb{Q}$, which are particularly useful with aggregation: the interpretation of $D^{\langle\rangle}_{s,b}$ comprises all intervals of the form $\langle s + ib, s + (i+1)b\rangle$, for $i \in \mathbb{Z}$; see examples in Section 4.

Aggregation Functionals. An *aggregation functional* is a function over binary relations R on $\mathsf{int}_{\mathbb{R}} \times \mathbb{R}$. We illustrate the notion on a number of examples, where the restriction of a relation R to interval $\iota \in \mathsf{int}_{\mathbb{R}}$ will be denoted by $R_\iota = \{(\iota', v') \in R \mid \iota' \cap \iota \neq \emptyset\}$. Now, consider

$$\mathit{max}(R) = \big\{\big(\iota, \max_{(\iota',v')\in R_\iota} v'\big) \mid \iota \in \mathsf{int}_{\mathbb{R}} \text{ and } \iota \subseteq \mathit{dom}\, R\big\},$$

where $\mathit{dom}\, R = \bigcup_{(\iota,v)\in R} \iota$ is the *domain* of R. This aggregation functional returns a relation with the maximum value of R on any subinterval of its domain. Another typical example is the *simple moving average*:

$$\mathit{sma}(R) = \big\{\big(\iota, \frac{1}{|R_\iota|} \sum_{(\iota',v')\in R_\iota} v'\big) \mid \iota \in \mathsf{int}_{\mathbb{R}}, \iota \subseteq \mathit{dom}\, R \text{ and } R_\iota \text{ is finite}\big\}.$$

To define other aggregation functionals, we require the following definition: we say that R is a *functional relation* if $v_1 = v_2$, for any $(\iota_1, v_1), (\iota_2, v_2) \in R$ with $\iota_1 \cap \iota_2 \neq \emptyset$. In this case, R gives rise to the function $f_R\colon\ \mathit{dom}\, R \to \mathbb{R}$ defined by taking

$$f_R(t) = v, \text{ for all } t \in \iota \text{ with } (\iota, v) \in R.$$

The *weighted average* is defined on functional relations R as

$$\mathit{wavg}(R) = \big\{\big(\iota, \frac{1}{|\iota|} \int_\iota f_R(x)\, dx\big) \mid \iota \in \mathsf{int}_{\mathbb{R}} \text{ is bounded and } \iota \subseteq \mathit{dom}\, R\big\},$$

and $\mathit{wavg}(R) = \emptyset$ for non-functional relations R. Note that, since $\mathit{wavg}(R)$ is defined on all subintervals of $\mathit{dom}\, R$, it is *not* a functional relation in general.

Another important aggregation functional is *coalescing*, which is defined on functional relations R by taking

$$\mathit{coalesce}(R) = \big\{(\iota, v) \mid \iota \in \mathsf{int}_{\mathbb{R}} \text{ is maximal with } \iota \subseteq \mathit{dom}\, R \text{ and } f_R(x) = v, \text{ for all } x \in \iota\big\},$$

and $\mathit{coalesce}(R) = \emptyset$ for non-functional R. Note that $\mathit{coalesce}(R)$ is a functional relation and $f_R = f_{\mathit{coalesce}(R)}$. For unary predicates A over $\mathsf{int}_{\mathbb{R}}$, we define coalescing by taking $\mathit{coalesce}(A) = \{\iota \mid (\iota, 0) \in \mathit{coalesce}(A \times \{0\})\}$.

***DsIID* Programs and Data Instances.** There are four types of atoms in *DsIID*. By a *pure atom* we mean a formula of the form $A(\boldsymbol{x})$ and $R(\boldsymbol{x}, y)$, for signature predicates A and R, interval variable $\boldsymbol{x}$ and value variable y. A *built-in atom* is any well-formed atomic formula with a built-in predicate and appropriately typed interval and value terms. A *binding* atom is an expression $\boldsymbol{x}\,\mathsf{is}\,\boldsymbol{t}$ or $x\,\mathsf{is}\,t$, where $\boldsymbol{x}$ is a variable and $\boldsymbol{t}$ a term of sort interval, and x a variable and t a term of sort value. *Aggregate atoms* are defined recursively as

$$\mathit{agg}\{(\boldsymbol{x}, y) \mid \alpha_1, \ldots, \alpha_k\}, \quad \mathit{coalesce}\{\boldsymbol{x} \mid \alpha_1, \ldots, \alpha_k\},$$

where agg is an aggregation functional and the α_i are (pure, built-in, binding or aggregate) atoms.

A *rule* in *DsIID* is an expression of the form

$$\alpha \leftarrow \beta_1, \ldots, \beta_n, \tag{1}$$

where α is a *pure atom* and the β_i are atoms. As usual in datalog, we call α the *head* of the rule and $\beta_1, \dots, \beta_n$ its *body*. A *DslD* program, Π, is a finite set of rules. *Data atoms* take the form $A(\varrho)$ and $R(\varrho, c)$, where ϱ is a constant of type interval and c a constant of type value. A *data instance*, $\mathcal{D}$, is a finite set of data atoms.

Note that we do not impose the standard Datalog safety conditions, so rules such as $A(\boldsymbol{x}) \leftarrow B(\boldsymbol{y})$ are allowed. This leads to the problem that all the intervals with ends from the data instance $\mathcal{D}$ will be returned as a result of the query $A(\boldsymbol{x})$ over $\mathcal{D}$, if any assertion $B(\iota)$ is in $\mathcal{D}$, according to the semantics below. However, all the rules in our use cases described below are safe in the sense the problem above does not occur.

Semantics of *DslD* Programs. We next define the procedural semantics of *DslD* programs. By an *assignment*, $\mathfrak{s}$, we understand a map from interval variables $\boldsymbol{x}$ to $\mathsf{int}_{\mathbb{R}}$ and from value variables x to $\mathbb{R}$. The definition of $\mathfrak{s}$ is inductively extended to terms in a standard way: for instance, $\mathfrak{s}(c) = c$, for any constant c. Note, however, that the extended $\mathfrak{s}$ is in general a partial function: for example, $\mathfrak{s}([0,1] \uplus [2,3])$ is undefined.

Let Π be a *DslD* program and $\mathcal{D}$ a data instance. We first set

$$A_i^0 \;=\; \big\{\, \iota \in \mathsf{int}_{\mathbb{R}} \mid A_i(\iota) \in \mathcal{D} \,\big\}, \quad R_i^0 \;=\; \big\{\, (\iota, \upsilon) \in \mathsf{int}_{\mathbb{R}} \times \mathbb{R} \mid R_i(\iota, \upsilon) \in \mathcal{D} \,\big\}, \quad i \geq 1,$$

and $\mathcal{I}^0 = (A_1^0, \dots, R_1^0, \dots)$. Suppose next inductively that $\mathcal{I}^\xi = (A_1^\xi, \dots, R_1^\xi, \dots)$, for an ordinal ξ, and $\mathfrak{s}$ is a substitution. We define a truth-relation $\mathcal{I}^\xi, \mathfrak{s} \models \alpha$ as follow:

- $\mathcal{I}^\xi, \mathfrak{s} \models A_i(\boldsymbol{x})$ iff $\mathfrak{s}(\boldsymbol{x}) \in A_i^\xi$;
- $\mathcal{I}^\xi, \mathfrak{s} \models R_i(\boldsymbol{x}, y)$ iff $(\mathfrak{s}(\boldsymbol{x}), \mathfrak{s}(y)) \in R_i^\xi$;
- $\mathcal{I}^\xi, \mathfrak{s} \models (\boldsymbol{t} \neq \boldsymbol{t'})$ if $\mathfrak{s}(\boldsymbol{t})$ and $\mathfrak{s}(\boldsymbol{t'})$ are both defined with $\mathfrak{s}(\boldsymbol{t}) \neq \mathfrak{s}(\boldsymbol{t'})$, and similarly for other types of built-in atom;
- $\mathcal{I}^\xi, \mathfrak{s} \models \boldsymbol{x}\,\mathsf{is}\,\boldsymbol{t}$ if $\mathfrak{s}(\boldsymbol{x})$ and $\mathfrak{s}(\boldsymbol{t})$ are both defined with $\mathfrak{s}(\boldsymbol{x}) = \mathfrak{s}(\boldsymbol{t})$, and similarly for $x\,\mathsf{is}\,t$;
- $\mathcal{I}^\xi, \mathfrak{s} \models D^{\langle\rangle}_{s,b}(\boldsymbol{t})$ iff $\mathfrak{s}(\boldsymbol{t}) = \langle s + ib, s + (i+1)b \rangle$, for some $i \in \mathbb{Z}$;
- $\mathcal{I}^\xi, \mathfrak{s} \models \mathit{agg}\{(\boldsymbol{x}, y) \mid \alpha_1, \dots, \alpha_k\}$ iff $(\mathfrak{s}(\boldsymbol{x}), \mathfrak{s}(y)) \in \mathit{agg}(R)$, where

$$R = \big\{(\mathfrak{s}'(\boldsymbol{x}), \mathfrak{s}'(y)) \mid \text{ there is } \mathfrak{s}' \text{ such that } \mathcal{I}^\xi, \mathfrak{s}' \models \alpha_j \text{ for all } j, 1 \leq j \leq k\big\};$$

- $\mathcal{I}^\xi, \mathfrak{s} \models \mathit{coalesce}\{\boldsymbol{x} \mid \alpha_1, \dots, \alpha_k\}$ iff $\mathfrak{s}(\boldsymbol{x}) \in \mathit{coalesce}(A)$, where

$$A = \big\{\mathfrak{s}'(\boldsymbol{x}) \mid \text{ there is } \mathfrak{s}' \text{ such that } \mathcal{I}^\xi, \mathfrak{s}' \models \alpha_j \text{ for all } j, 1 \leq j \leq k\big\}.$$

Given $\mathcal{I}^\xi$, we define $\mathcal{I}^{\xi+1} = (A_1^{\xi+1}, \dots, R_1^{\xi+1}, \dots)$ as follows. For each A_i, we set $A_i^{\xi+1}$ to be A_i^ξ together with all $\mathfrak{s}(\boldsymbol{x})$ such that Π contains $A_i(\boldsymbol{x}) \leftarrow \beta_1, \dots, \beta_n$ and $\mathcal{I}^\xi, \mathfrak{s} \models \beta_j$ for all j, $1 \leq j \leq n$; and analogously for $R_i^{\xi+1}$. For a limit ordinal ζ, we set $\mathcal{I}^\zeta = (A_1^\zeta, \dots, R_1^\zeta, \dots)$, where $A_i^\zeta = \bigcup_{\xi<\zeta} A_i^\xi$ and $R_i^\zeta = \bigcup_{\xi<\zeta} R_i^\xi$. The construction is terminated when $\mathcal{I}^\xi = \mathcal{I}^{\xi+1}$, which should happen for some ξ not exceeding the smallest ordinal, $2^{\aleph_0+}$, whose cardinality is greater than $2^{\aleph_0}$.

▶ **Example 1.** Consider the data instance $\mathcal{D} = \{A([0,1])\}$ and a *DslD* program Π with the following rules:

$$A(\boldsymbol{x}) \leftarrow A(\boldsymbol{y}),\ \boldsymbol{x}\,\mathsf{is}\,\mathit{rshift}_1(\boldsymbol{y}),$$
$$B(\boldsymbol{x}) \leftarrow \mathit{coalesce}\{\boldsymbol{x} \mid A(\boldsymbol{x})\},$$
$$B(\boldsymbol{x}) \leftarrow B(\boldsymbol{y}),\ \boldsymbol{x}\,\mathsf{is}\,\mathit{lshift}_{-1}(\boldsymbol{y}).$$

In this case, $\mathcal{I}^\omega$ comprises $A([0,n])$, for $0 < n$, $B([m,n])$, for $m < 0 < n$, and $B([0,\infty))$; $\mathcal{I}^{\omega\cdot 2}$ also contains $B([m,\infty))$, for $m < 0$, and $\mathcal{I}^{\omega\cdot 2} = \mathcal{I}^{\omega\cdot 2+1}$.

Ontology-Mediated Queries. By a *conjunctive query* (CQ) we mean a first-order formula $\boldsymbol{q}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) = \exists \boldsymbol{y}_1, \dots, \boldsymbol{y}_k, y_1, \dots y_k \, \Phi$, where Φ is a conjunction of pure atoms with signature predicates whose variables are among the $\boldsymbol{x}_i$, $\boldsymbol{y}_i$, y_i. Using the tuple $\bar{\boldsymbol{x}} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$, we denote this CQ by $\boldsymbol{q}(\bar{\boldsymbol{x}})$ and call $(\Pi, \boldsymbol{q}(\bar{\boldsymbol{x}}))$ an *ontology-mediated query* (OMQ). A *certain answer* to the OMQ $(\Pi, \boldsymbol{q}(\bar{\boldsymbol{x}}))$ over the data instance $\mathcal{D}$ is any tuple $\bar{\boldsymbol{a}} = (\iota_1, \dots, \iota_n)$ such that the ends of the intervals ι_i occur in $\mathcal{D}$ and $\mathcal{I}^{2^{\aleph_0+}}, \mathfrak{s} \models \boldsymbol{q}(\bar{\boldsymbol{x}})$, where $\mathfrak{s}$ maps the variables in $\bar{\boldsymbol{x}}$ to the respective constants in $\bar{\boldsymbol{a}}$.

We leave the investigation of semantical and computational properties of answering *DslD* OMQs to future work, focusing below on representing and querying events in our use cases, where *DslD* programs do not involve recursion.

3 The Gas Turbine Use Case

We illustrate the expressive power of *DslD* and its semantics by representing and querying the active power trip event formulated in Section 1. We assume that the turbine sensors measure various parameters such as the active power, blade temperature, etc. asynchronously, and that the measurement data is stored in a database as relations $\mathit{ActivePower}([t, t'), v)$ stating that the measured value of the active power over the interval $[t, t')$ was v.

The active power trip event was considered in [6], where the raw measurement data was first aggregated to avoid short-term fluctuations and then pre-processed by defining temporal concepts (unary predicates) "active power below 0.15MW" and "active power above 1.5MW" by means of fairly complex SQL views. After that, the active power trip was captured by means of the following *MTL* rule, where, for example, $\boxminus_{[b,e]}\varphi$ (or $\diamondminus_{[b,e]}\varphi$) is true at moment t if and only if φ is true at every (respectively, some) moment in the interval $[t-b, t-e]$:

$$\mathit{ActivePowerTrip} \leftarrow \boxminus_{[0,1m]} \mathit{ActivePowerBelow0.15} \wedge \diamondminus_{[60s,63s]} \boxminus_{[0,10s]} \mathit{ActivePowerAbove1.5}. \quad (2)$$

Unlike the one-dimensional metric temporal logic *MTL*, the language *DslD* is two-dimensional and allows us to aggregate the raw data by taking, as required, the simple moving average of the raw power measurements over the last 5 seconds and define the active power trip in a *DslD* program Π_{apt} using the following two rules:

$$\begin{aligned}
&\mathit{ActivePowerTrip}(\boldsymbol{w}) \leftarrow \\
&\qquad \boldsymbol{z} \text{ is } \mathit{lshift}_{1m}(\boldsymbol{x}),\ |\boldsymbol{x}| \geq 1m,\ \mathit{coalesce}\{\boldsymbol{x} \mid y < 0.15,\ \mathit{Power}(\boldsymbol{x}, y)\}, \\
&\qquad \mathit{coalesce}\{\boldsymbol{y} \mid \boldsymbol{y} \text{ is } \mathit{lshift}_{60s}(\mathit{rshift}_{63s}(\boldsymbol{u})), \\
&\qquad\qquad \boldsymbol{u} \text{ is } \mathit{lshift}_{10s}(\boldsymbol{x}),\ |\boldsymbol{x}| \geq 10s,\ \mathit{coalesce}\{\boldsymbol{x} \mid y > 1.5,\ \mathit{Power}(\boldsymbol{x}, y)\}\}, \\
&\qquad \boldsymbol{w} \text{ is } \boldsymbol{z} \frown \boldsymbol{y}, \\
&\mathit{Power}(\boldsymbol{x}, v) \leftarrow \mathit{ActivePower}(\boldsymbol{x}, v'),\ \boldsymbol{y} \text{ is } \mathit{rext}_{5s}(\boldsymbol{x}),\ \mathit{sma}\{(\boldsymbol{y}, v) \mid \mathit{ActivePower}(\boldsymbol{y}, v)\}. \quad (3)
\end{aligned}$$

Note that, for example, the two $\boldsymbol{x}$ in the *coalesce* atoms in the first rule are in fact independent because the second is within the scope of another *coalesce* and does not belong to its head. Thus, the second rule says that the value of *Power* in an interval $\boldsymbol{x}$, for which the data contains a value of *ActivePower*, is defined as the simple moving average of *ActivePower* over the 5s window spanning to the past from the right end of $\boldsymbol{x}$. For example, consider a data instance with the measurements

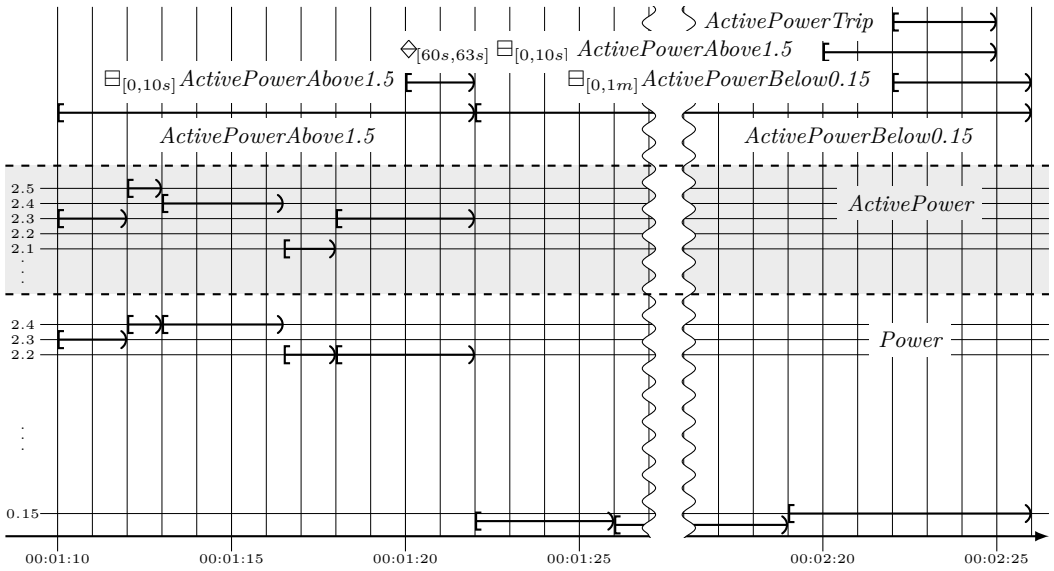


Figure 2 A data instance and the interpretation of Π_{apt} with the respective subformulas of (2).

$$\mathit{ActivePower}([00{:}01{:}10, 00{:}01{:}12), 2.3), \mathit{ActivePower}([00{:}01{:}12, 00{:}01{:}13), 2.5), \dots$$

shown in the middle part of Fig. 2. In this case, the interpretation $\mathcal{I}^{2^{\aleph_0+}}$ will contain

$$\mathit{Power}([00{:}01{:}10, 00{:}01{:}12), 2.3), \mathit{Power}([00{:}01{:}12, 00{:}01{:}13), 2.4), \dots$$

In the first rule of Π_{apt}, variable $\boldsymbol{u}$ representing $\boxminus_{[0,10s]}\mathit{ActivePowerAbove1.5}$ will be instantiated by $[00{:}01{:}20, 00{:}01{:}22)$; variable $\boldsymbol{y}$ corresponding to $\diamondminus_{[60s,63s]}\boxminus_{[0,10s]}\mathit{ActivePowerAbove1.5}$ will be instantiated by $[00{:}02{:}20, 00{:}02{:}25)$, while $\boldsymbol{z}$ for $\boxminus_{[0,1m]}\mathit{ActivePowerBelow0.15}$ by $[00{:}02{:}22, 00{:}02{:}26)$. As a result, the atom $\mathit{ActivePowerTrip}(\boldsymbol{w})$ will be true in the interval $\boldsymbol{w} = [00{:}02{:}22, 00{:}02{:}25)$, exactly where $\mathit{ActivePowerTrip}$ defined by (2) is true.

4 The Drilling Rigs Use Case

Next, we consider several types of drilling process and model them in *DslD*.

Drilling Process. We begin with the process shown in Fig. 1, which can be defined as a sequence of three sub-processes following each other. First, the drill makes contact with the workpiece, and the borehole is created (tapping). In this phase, the diameter increases, which results in increasing process forces. Then, the diameter, and therefore the process forces stay almost constant (drilling), until the drill breaks through on the other side of the plate, which results in a decrease of process forces (exiting).

Figure 3 shows sensor data from force measurements sampled at 2kHz. This signal, however, does not resemble the anticipated shape, which is based on the expectations of domain experts. These expectations are concerned with a particular low-frequency phenomenon. Higher frequency signals within the sampled data are therefore considered

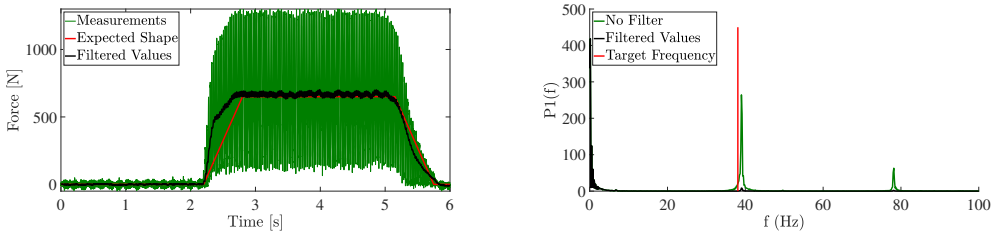


Figure 3 Filtered data compared with expected shapes for time constants determined based on analytical estimation (left) and respective spectrum before and after aggregation (right).

noise and need to be filtered accordingly. A simple version of such a filter is moving average [27]. The parameters of this filter can be determined based on domain knowledge, which is also illustrated in Fig. 3. The drilling process is captured by the following rules:

$$\begin{aligned}
\mathit{AvgForce}(\boldsymbol{x}, y) &\leftarrow \mathit{wavg}\{(\boldsymbol{x}, y) \mid \mathit{Force}(\boldsymbol{x}, y)\}, \\
\mathit{DAvgForce}(\boldsymbol{x}, y) &\leftarrow \mathit{wavg}\{(\boldsymbol{x}, y) \mid \mathit{Force}(\boldsymbol{x}, y)\}, D^{[,)}_{0,13s}(\boldsymbol{x}), && (4)\\
\mathit{ForceDelta}(\boldsymbol{x}, z) &\leftarrow \mathit{DAvgForce}(\boldsymbol{x}, y), \mathit{DAvgForce}(\boldsymbol{x}', y'), \boldsymbol{x}\ \mathsf{A}\ \boldsymbol{x}', z\ \mathsf{is}\ (y - y'), && (5)\\
\mathit{IncForce}(\boldsymbol{x}) &\leftarrow \mathit{coalesce}\{\boldsymbol{x} \mid \mathit{ForceDelta}(\boldsymbol{x}, d), d > 0.1\}, \\
\mathit{ConstForce}(\boldsymbol{x}) &\leftarrow \mathit{coalesce}\{\boldsymbol{x} \mid \mathit{ForceDelta}(\boldsymbol{x}, d), |d| \leq 0.1\}, \\
\mathit{DecForce}(\boldsymbol{x}) &\leftarrow \mathit{coalesce}\{\boldsymbol{x} \mid \mathit{ForceDelta}(\boldsymbol{x}, d), d < -0.1\}, \\
\mathit{SimpleDrilling}(\boldsymbol{x}) &\leftarrow \mathit{IncForce}(\boldsymbol{x}_1), \mathit{max}\{(\boldsymbol{x}_1, y_1) \mid \mathit{AvgForce}(\boldsymbol{x}_1, y_1)\},\ \boldsymbol{x}_1\ \mathsf{A}\ \boldsymbol{x}_2, \\
&\quad \mathit{ConstForce}(\boldsymbol{x}_2), \mathit{max}\{(\boldsymbol{x}_2, y_2) \mid \mathit{AvgForce}(\boldsymbol{x}_2, y_2)\},\ \boldsymbol{x}_2\ \mathsf{A}\ \boldsymbol{x}_3, \\
&\quad \mathit{DecForce}(\boldsymbol{x}_3), \mathit{max}\{(\boldsymbol{x}_3, y_3) \mid \mathit{AvgForce}(\boldsymbol{x}_3, y_3)\}, \\
&\quad |y_1 - y_2| < y_2 \times 0.05,\ |y_2 - y_3| < y_3 \times 0.05, \\
&\quad \boldsymbol{x}\ \mathsf{is}\ (\boldsymbol{x}_1 \uplus \boldsymbol{x}_2 \uplus \boldsymbol{x}_3). && (6)
\end{aligned}$$

Smooth Drilling. Drilling processes can be further classified based on other characteristics of the force measurements. Throughout such a drilling process, however, anomalies can occur, the simplest of which is the presence of significant peaks that might come, for example, from material inhomogeneities. In this sense, a *smooth drilling process* can be identified by comparing the maximum and minimum values within the simple drilling event:

$$\begin{aligned}
\mathit{Drilling}(\boldsymbol{x}) &\leftarrow \mathit{ConstForce}(\boldsymbol{x}), \\
\mathit{SmoothDrilling}(\boldsymbol{x}) &\leftarrow \mathit{SimpleDrilling}(\boldsymbol{x}),\ \mathit{Drilling}(\boldsymbol{x}_1),\ \boldsymbol{x}_1 \subseteq \boldsymbol{x},\ \mathit{AvgForce}(\boldsymbol{x}_1, y), \\
&\quad \mathit{max}\{(\boldsymbol{x}_1, y_1) \mid \mathit{AvgForce}(\boldsymbol{x}_1, y_1)\},\ y_1 - y \leq 0.1 \times y, \\
&\quad \mathit{min}\{(\boldsymbol{x}_1, y_2) \mid \mathit{AvgForce}(\boldsymbol{x}_1, y_2)\},\ y - y_2 \leq 0.1 \times y.
\end{aligned}$$

Unstable Drilling. The *unstable drilling process* event can be classified based on the standard deviation of the force readings. Even though they might look similar based on their average process forces in each phase, unstable processes will have significantly higher standard deviation compared to the stable ones. Again, the threshold that would be used here has to be determined based on domain knowledge:

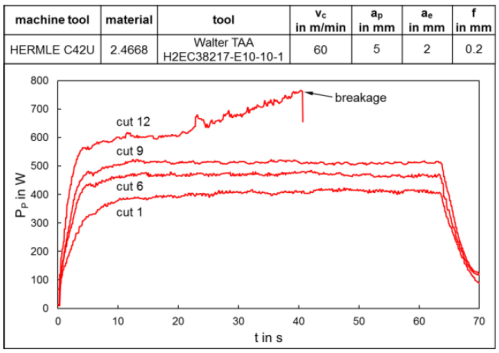


Figure 4 Electrical power readings for a sequence of similar cuts until a tool break occurred [11].

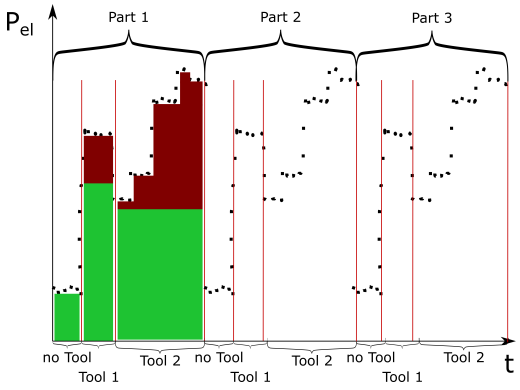


Figure 5 Total electrical power input (P_{el}) for a production machine while manufacturing three similar parts.

$$\mathit{SmoothDrilling}(\boldsymbol{x}) \leftarrow \mathit{SimpleDrilling}(\boldsymbol{x}),\ \mathit{Drilling}(\boldsymbol{x}_1),\ \boldsymbol{x}_1 \subseteq \boldsymbol{x},\ \mathit{AvgForce}(\boldsymbol{x}_1, y),\\ \mathit{sdev}\{(\boldsymbol{x}_1, y_1) \mid \mathit{Force}(\boldsymbol{x}_1, y_1)\},\ |y_1 - y| \leq 0.1 \times y,$$

where *sdev* is an aggregation functional computing the standard deviation.

Tool Break. Tool breaks happen regularly in industrial production processes. Example power readings for such an event are depicted in Fig. 4. In this example, power is directly related to apparent process forces. As a consequence, force readings would differ from the displayed power readings only by a constant conversion factor. To identify tool breaks, we simply find any patterns that match the concept *simple drilling* but are shorter than the expected duration:

$$\mathit{ToolBreak}(\boldsymbol{x}) \leftarrow \mathit{SimpleDrilling}(\boldsymbol{x}),\ \mathit{ExpectedDrillingTime}(\boldsymbol{x}, y),\ |\boldsymbol{x}| \leq y,$$

where $\mathit{ExpectedDrillingTime}(\boldsymbol{x}, y)$ specifies the expected time of drilling for given drill model, material thickness, etc. and is defined by means of mappings.

Energy Per Tool. Closely related to tool wear is the remaining tool lifetime. One way to determine this measure from sensor data is the amount of accumulated electrical energy that was used to drive a given tool. To illustrate, consider Fig. 5 showing power measurements for a production machine. Within the observed window, three similar parts (having the same geometric features) are manufactured using two different tools. To calculate the energy per tool, the power measurements need to be integrated over the intervals in which the respective tool was active. Before doing so, however, it is necessary to deduct the base load of the machine. The base load is defined as the machine's (electrical) power demand without material removal, which can be due to auxiliary systems or power loss in bearings. We use the following *DslD* rules:

$$\begin{aligned}
&\mathit{Pbase}(\boldsymbol{x}, v) \leftarrow \mathit{SimpleDrilling}(\boldsymbol{y}), \boldsymbol{x}\ \mathsf{is}\ \mathit{lext}_{-10s}(\boldsymbol{y}), \mathit{wavg}\{(\boldsymbol{x}, v) \mid \mathit{Power}(\boldsymbol{x}, v)\},\\
&\mathit{Ptool}(\boldsymbol{x}, u) \leftarrow \mathit{wavg}\{(\boldsymbol{x}, v) \mid \mathit{Power}(\boldsymbol{x}, v)\}, \mathit{Pbase}(\boldsymbol{x}', v'), \boldsymbol{x} \subseteq \boldsymbol{x}', u\ \mathsf{is}\ v - v',\\
&\mathit{Etool}(\boldsymbol{x}, v) \leftarrow \mathit{int}\{(\boldsymbol{x}, v) \mid \mathit{Ptool}(\boldsymbol{x}, v)\}, \mathit{SimpleDrilling}(\boldsymbol{x}),
\end{aligned}$$

where, for a functional relation R, aggregation functional $\mathit{int}(R)$ is defined as

$$\mathit{int}(R) = \Big\{\big(\iota, \int_\iota f_R(x)\,dx\big) \mid \iota \in \mathsf{int}_{\mathbb{R}} \text{ is bounded and } \iota \subseteq \mathit{dom}\,R\Big\}.$$

5 Evaluating *DslD* by SQL

In this section, we show that answering *DslD* OMQs in our use cases can be reduced to evaluation of SQL queries. First of all, we observe that our *DslD* programs are non-recursive. Our query answering algorithm for *DslD* is an extension of the classical algorithm for evaluating non-recursive datalog programs; see, e.g., [31]. In a nutshell, the algorithm introduces views for all head predicates, and computes these views in a bottom-up fashion following the dependency relation. For each unary predicate A, we create a view `A` with columns `begin` and `end`; for each binary predicate R, we create a view `R` with columns `begin`, `end` and `value`. When translating non-recursive *DslD* to SQL, the Allen relations and metric constructs can be implemented in a straightforward way. It is much more challenging to deal with aggregation atoms.

We illustrate our translation with examples. To start, we show how to deal with the simple drilling use case. First, we compute the predicate *DAvgForce* in rule (4), which depends on the view `Force(begin, end, value)`. We need to deal with the discretisation predicate $D^{[,)}_{0,13s}$, which "splits" the rows of the `Force` view into windows of 13s. For convenience, we assume that we have an auxiliary table `nums(id)` storing enough numbers 0, 1, ... The following view `D_force` discretises `Force`. In the resulting view, each row contains a split interval [`begin`, `end`), which is contained in the window [`w_begin`, `w_end`):

```
CREATE VIEW D_force AS (
   SELECT GREATEST(begin, nums.id * 13) AS begin,
          LEAST((nums.id + 1) * 13, end) AS end,
          Force.value AS value,
          nums.id * 13 AS w_begin, (nums.id + 1)* 13 AS w_end
   FROM Force, nums
   WHERE begin / 13 <= nums.id  AND nums.id <= end / 13
);
```

The usage of the auxiliary table `nums` is not always necessary, since it can be generated on the fly with a function like `generate_series`. Alternatively, one can also simulate this join by user-defined functions.

Then, we are ready to define the view `AvgForceP` by grouping the rows of `D_force` according to their windows and computing the weighted average by means of the built-in aggregate function `SUM`:

```
CREATE VIEW AvgForceP AS (
   SELECT MIN(begin) AS begin, MAX(end) AS end,
       SUM(value * (end - begin)) / SUM(end - begin) AS value
   FROM D_force
   GROUP BY w_begin, w_end
);
```

Note that, in general, many aggregation functionals (such as *coalesce* in rule (6)) cannot be defined directly by means of built-in aggregate functions in SQL, and one needs to introduce user-defined aggregate functions (UDAFs) according to the SQL engine, e.g., Apache Spark[1] and MS SQL Server[2]. With UDAFs, the coalesce operator can be implemented using, e.g., the standard algorithm [33].

[1] https://spark.apache.org/docs/latest/api/java/org/apache/spark/sql/expressions/UserDefinedAggregateFunction.html

[2] https://docs.microsoft.com/en-us/sql/relational-databases/clr-integration-database-objects-user-defined-functions/clr-user-defined-aggregates

Next, for the predicate `ForceDelta`, we introduce the view capturing rule (5):

```
CREATE VIEW ForceDelta AS (
    SELECT a1.begin, a1.end, a2.value - a1.value AS value
    FROM AvgForceP a1, AvgForceP a2
    WHERE a1.end = a2.begin
);
```

Other predicates in this simple drilling use case can be computed in a similar way. Eventually, the view `SimpleDrilling` will compute all instances of the predicate *SimpleDrilling*.

Finally, we consider the turbine use case. The major technical difference is the handling of windows. For rule (3), which does not have a discretisation predicate in the body, we can define time-based windows for each row relying on the range-based windows in the standard SQL query language. In this example, assuming that the data is provided in table `tb_power(ts, value)`, the time-based window can be defined as follows:

```
CREATE VIEW SmaPower AS (
    SELECT LAG(ts,1) OVER (ORDER BY ts) AS begin,
           ts AS end, AVG(value) OVER  W AS avg
    FROM tb_power
    WINDOW W AS
     (ORDER BY ts RANGE BETWEEN 5 PRECEDING AND 0 FOLLOWING)
);
```

While the approach described above works for the aggregation functionals, built-in and discretisation predicates, and the rules involved in our use cases, we leave the general algorithm for non-recursive *DslD* OMQs to further investigation. (We conjecture that query answering for arbitrary *DslD* OMQs is undecidable.)

6 Evaluation

We evaluated the SQL translations of the *DslD* OMQs defined in our two use cases over large amounts of data. For the turbine use case, we have 4 days of power data in the form of (timestamp, value) pairs for one running turbine. We replicated this sample to imitate the data for one turbine over 10 different periods ranging from 32 to 320 months (from 0.23 GB to 2.3 GB). For the drilling use case, we collected 2.6 GB of real data from a manufacturing company. This data contains force and power measurements associated with timestamps from one drilling tool. We ran our experiments on an AWS server with an Intel Xeon Platinum-8175 processor having 8 logical cores at 2.5 GHz and 64 GB of RAM. The SQL queries were executed on Apache Spark 2.4.0.

Figure 6 illustrates the execution times for the SQL translation of the active power trip OMQ in the turbine use case (Section 3), while Figure 7 shows the times for OMQs in the drilling use case (Section 4). Both sets of results from the two use cases show that the execution times scale linearly over monotonically increasing data. As expected, in the cases where we can benefit from discretisation predicates (as in the drilling use case *DslD* OMQs) the computation load reduces significantly. On the other hand, the active power trip program requires row by row windowing, and this leads to additional work load compared to the cases involving discretisation predicates.

7 Conclusion

As shown by the two use cases considered in this paper, engineers analysing the behaviour of industrial systems by detecting events in sensor log data are facing a challenging task of representing those events in terms of the existing query languages: the queries have complex structure with numerous complex subqueries. The OBDA paradigm would drastically simplify

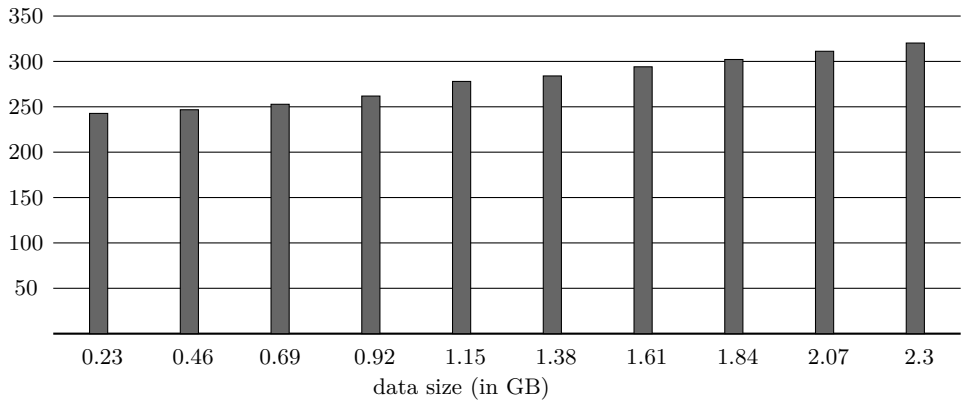


Figure 6 Running times (in seconds) of the active power trip OMQ.

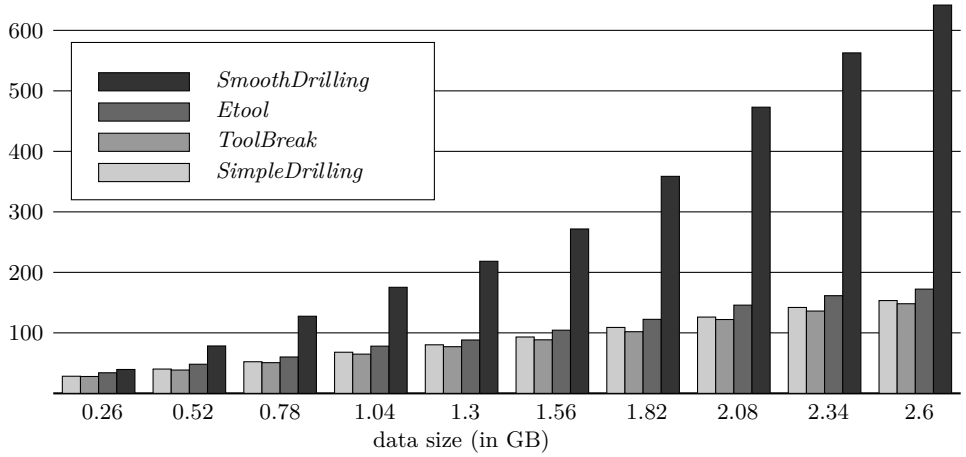


Figure 7 Running times (in seconds) for evaluation of the drilling OMQs.

the engineers' work if the events in question could be captured in ontologies rather than queries. Based on the use cases, we proposed a suitable OBDA ontology language *DslD*, featuring aggregate functionals, Allen's interval relations and various metric constructs. We showed that the non-recursive fragment of *DslD* is enough to capture the events in our cases and can be translated into sufficiently efficient SQL queries, which we tested on real-world data. We achieved good performance results with large amounts of data.

Encouraged by the satisfaction of the involved engineers, we are planning to do a proper user study to see how well the engineers can use this language. Further, an investigation of theoretical properties of the proposed language and optimisation techniques will be conducted.

References

1 J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Commun. ACM*, 26(11):832–843, 1983. doi:10.1145/182.358434.

2 A. Artale, R. Kontchakov, A. Kovtunova, V. Ryzhikov, F. Wolter, and M. Zakharyaschev. First-Order Rewritability of Temporal Ontology-Mediated Queries. In *Proc. of the 24th Int. Joint Conf. on Artificial Intelligence, IJCAI 2015*, pages 2706–2712. IJCAI/AAAI, 2015. URL: http://ijcai.org/papers15/Abstracts/IJCAI15-383.html.

3 F. Baader, S. Borgwardt, and M. Lippmann. Temporalizing Ontology-Based Data Access. In *Proc. of the 24th Int. Conf. on Automated Deduction, CADE-24*, volume 7898 of *LNCS*, pages 330–344. Springer, 2013. doi:10.1007/978-3-642-38574-2_23.

4 F. Baader and U. Sattler. Description logics with aggregates and concrete domains. *Inf. Syst.*, 28(8):979–1004, 2003. doi:10.1016/S0306-4379(03)00003-6.

5 S. Borgwardt, M. Lippmann, and V. Thost. Temporal Query Answering in the Description Logic DL-Lite. In *Proc. of the 9th Int. Symposium on Frontiers of Combining Systems, FroCoS'13*, volume 8152 of *LNCS*, pages 165–180. Springer, 2013. doi:10.1007/978-3-642-40885-4_11.

6 S. Brandt, E. G. Kalaycı, V. Ryzhikov, G. Xiao, and M. Zakharyaschev. Querying Log Data with Metric Temporal Logic. *J. Artif. Intell. Res.*, 62:829–877, 2018. doi:10.1613/jair.1.11229.

7 D. Bresolin, A. Kurucz, E. Muñoz-Velasco, V. Ryzhikov, G. Sciavicco, and M. Zakharyaschev. Horn Fragments of the Halpern-Shoham Interval Temporal Logic. *ACM Trans. Comput. Log.*, 18(3):22:1–22:39, 2017. doi:10.1145/3105909.

8 D. Calvanese, E. Kharlamov, W. Nutt, and C. Thorne. Aggregate queries over ontologies. In *Proc. of the 2nd Int. Workshop on Ontologies and Information Systems for the Semantic Web, ONISW 2008*, pages 97–104, 2008. doi:10.1145/1458484.1458500.

9 V. Gutiérrez-Basulto, J. C. Jung, and R. Kontchakov. Temporalized EL Ontologies for Accessing Temporal Data: Complexity of Atomic Queries. In *Proc. of the 25th Int. Joint Conf. on Artificial Intelligence, IJCAI 2016*, pages 1102–1108. IJCAI/AAAI, 2016. URL: https://www.ijcai.org/Proceedings/16/Papers/160.pdf.

10 V. Gutiérrez-Basulto and S. Klarman. Towards a Unifying Approach to Representing and Querying Temporal Data in Description Logics. In *Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems, RR 2012*, volume 7497 of *LNCS*, pages 90–105. Springer, 2012. doi:10.1007/978-3-642-33203-6_8.

11 M. Hacksteiner, F. Duer, D. Finkeldei, M. Obermair, and F. Bleicher. Monitoring of process power and energy during machining. In *Proc. of the 13th Int. Conf. on High Speed Machining*, 2016.

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

13 L. Hella, L. Libkin, J. Nurmonen, and L. Wong. Logics with Aggregate Operators. In *Proc. of the 14th Annual IEEE Symposium on Logic in Computer Science, LICS 1999*, pages 35–44. IEEE Computer Society, 1999. doi:10.1109/LICS.1999.782583.

14 R. Hussein, A. Sadek, M. A. Elbestawi, and M. H. Attia. Low-frequency vibration-assisted drilling of hybrid CFRP/Ti6Al4V stacked material. *Int. J. Adv. Manuf. Technol.*, 98:2801–2817, 2018. doi:10.1007/s00170-018-2410-2.

15 E. G. Kalaycı, S. Brandt, D. Calvanese, V. Ryzhikov, G. Xiao, and M. Zakharyaschev. Ontology-based Access to Temporal Data With Ontop: a Framework Proposal. *Int. J. Appl. Math. Comput. Sci.*, 29(1):17–30, 2019. doi:10.2478/amcs-2019-0002.

16 E. G. Kalaycı, G. Xiao, V. Ryzhikov, T. E. Kalaycı, and D. Calvanese. Ontop-temporal: A Tool for Ontology-based Query Answering over Temporal Data. In *Proc. of the 27th ACM Int. Conf. on Information and Knowledge Management, CIKM'18*, pages 1927–1930. ACM, 2018. doi:10.1145/3269206.3269230.

17 E. Kharlamov, T. Mailis, G. Mehdi, C. Neuenstadt, Ö. L. Özçep, M. Roshchin, N. Solomakhina, A. Soylu, C. Svingos, S. Brandt, M. Giese, Y. E. Ioannidis, S. Lamparter, R. Möller, Y. Kotidis, and A. Waaler. Semantic access to streaming and static data at Siemens. *J. Web Semant.*, 44:54–74, 2017. doi:10.1016/j.websem.2017.02.001.

18 E. Kharlamov, G. Mehdi, O. Savkovic, G. Xiao, E. G. Kalaycı, and M. Roshchin. Semantically-enhanced rule-based diagnostics for industrial Internet of Things: The SDRL language and case study for Siemens trains and turbines. *J. Web Semant.*, 56:11–29, 2018. doi:10.1016/j.websem.2018.10.004.

19 S. Klarman and T. Meyer. Querying Temporal Databases via OWL 2 QL. In *Proc. of the 8th Int. Conf. on Web Reasoning and Rule Systems, RR 2014*, volume 8741 of *LNCS*, pages 92–107. Springer, 2014. doi:10.1007/978-3-319-11113-1_7.

20 R. Kontchakov, L. Pandolfo, L. Pulina, V. Ryzhikov, and M. Zakharyaschev. Temporal and Spatial OBDA with Many-Dimensional Halpern-Shoham Logic. In *Proc. of the 25th Int. Joint Conf. on artificial Intelligence, IJCAI-16*, pages 1160–1166. IJCAI/AAAI, 2016. URL: https://www.ijcai.org/Proceedings/16/Papers/168.pdf.

21 E. V. Kostylev and J. L. Reutter. Complexity of answering counting aggregate queries over DL-Lite. *J. Web Semant.*, 33:94–111, 2015. doi:10.1016/j.websem.2015.05.003.

22 A. Kusiak. Smart Manufacturing. *Int. J. Prod. Res.*, 56(1–2):508–517, 2017. doi:10.1080/00207543.2017.1351644.

23 M. Lenzerini. Managing Data through the Lens of an Ontology. *AI Magazine*, 39(2):65–74, 2018. doi:10.1609/aimag.v39i2.2802.

24 O. Maler and D. Nickovic. Monitoring Temporal Properties of Continuous Signals. In *Proc. of Joint Int. Confs. on Formal Modeling and Analysis of Timed Systmes, FORMATS 2004, and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004*, volume 3253 of *LNCS*, pages 152–166. Springer, 2004. doi:10.1007/978-3-540-30206-3_12.

25 Ö. L. Özçep and R. Möller. Ontology Based Data Access on Temporal and Streaming Data. In *The 10th Int. Summer School on Reasoning Web, RW 2014*, volume 8714 of *LNCS*, pages 279–312. Springer, 2014. doi:10.1007/978-3-319-10587-1_7.

26 A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking Data to Ontologies. *J. Data Semantics*, X:133–173, 2008. doi:10.1007/978-3-540-77688-8_5.

27 T. A. Runkler. Datenvorverarbeitung. In *Data Mining*, pages 21–34. Vieweg+Teubner, Wiesbaden, 2010. doi:10.1007/978-3-8348-9353-6.

28 A. Shkapsky, M. Yang, and C. Zaniolo. Optimizing recursive queries with monotonic aggregates in DeALS. In *Proc. of the 31st IEEE Int. Conf. on Data Engineering, ICDE 2015*, pages 867–878. IEEE Computer Society, 2015. doi:10.1109/ICDE.2015.7113340.

29 A. Soylu, E. Kharlamov, D. Zheleznyakov, E. Jiménez-Ruiz, M. Giese, M. G. Skjæveland, D. Hovland, R. Schlatte, S. Brandt, H. Lie, and I. Horrocks. OptiqueVQS: A visual query system over ontologies for industry. *Semantic Web*, 9(5):627–660, 2018. doi:10.3233/SW-180293.

30 K. D. Thoben, S. A. Wiesner, and T. Wuest. "Industrie 4.0" and Smart Manufacturing – A Review of Research Issues and Application Examples. *Int. J. Autom. Technol.*, 11:4–16, 2017. doi:10.20965/ijat.2017.p0004.

31 J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I.* Computer Science Press, 1988.

32 G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev. Ontology-Based Data Access: A Survey. In *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence, IJCAI 2018*, pages 5511–5519. ijcai.org, 2018. doi:10.24963/ijcai.2018/777.

33 X. Zhou, F. Wang, and C. Zaniolo. Efficient Temporal Coalescing Query Support in Relational Database Systems. In *Proc. of the 17th Int. Conf. on Database and Expert Systems Applications, DEXA 2006*, volume 4080 of *LNCS*, pages 676–686. Springer, 2006. doi:10.1007/11827405_66.

Time-Aware Probabilistic Knowledge Graphs

Melisachew Wudage Chekol
Data and Web Science Group, University of Mannheim, Germany
mel@informatik.uni-mannheim.de

Heiner Stuckenschmidt
Data and Web Science Group, University of Mannheim, Germany
heiner@informatik.uni-mannheim.de

Abstract

The emergence of open information extraction as a tool for constructing and expanding knowledge graphs has aided the growth of temporal data, for instance, YAGO, NELL and Wikidata. While YAGO and Wikidata maintain the valid time of facts, NELL records the time point at which a fact is retrieved from some Web corpora. Collectively, these knowledge graphs (KG) store facts extracted from Wikipedia and other sources. Due to the imprecise nature of the extraction tools that are used to build and expand KG, such as NELL, the facts in the KG are weighted (a confidence value representing the correctness of a fact). Additionally, NELL can be considered as a transaction time KG because every fact is associated with extraction date. On the other hand, YAGO and Wikidata use the valid time model because they maintain facts together with their validity time (temporal scope). In this paper, we propose a bitemporal model (that combines transaction and valid time models) for maintaining and querying bitemporal probabilistic knowledge graphs. We study coalescing and scalability of marginal and MAP inference. Moreover, we show that complexity of reasoning tasks in atemporal probabilistic KG carry over to the bitemporal setting. Finally, we report our evaluation results of the proposed model.

2012 ACM Subject Classification Information systems → Web Ontology Language (OWL); Computing methodologies → Probabilistic reasoning; Computing methodologies → Temporal reasoning

Keywords and phrases temporal, probabilistic, knowledge graph, OWL-RL

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.8

1 Introduction

Temporal databases have been studied extensively [44, 15, 31]. Recently, support for temporal data from database vendors, such as Teradata, Oracle DB, IBM DB2, PostgreSQL, and so on, has been growing. On the other hand, probabilistic temporal databases have been given little attention [32, 12, 11, 8] and even less for probabilistic temporal knowledge graphs [6, 13]. Several Web knowledge graphs, such as YAGO [23], Wikidata [43], NELL [3], and DBpedia [1], already contain temporal data (each fact is associated with a valid time). In particular, NELL contains temporal probabilistic data where each fact is associated with a transaction time (the time when a fact is extracted and stored in a knowledge graph). Besides, the emergence of open information extraction as a tool for constructing and expanding knowledge graphs (KG) has aided the growth of temporal data [36, 35, 41]. In addition to valid time support, maintaining the transaction time of facts is relevant because never-ending open information extraction systems such MinIE [17], NELL, Google's Knowledge Vault [10], Microsoft's Satori continuously learn new facts from the Web and it is important to record the date on which a fact is learned. This is evident from the NELL knowledge graph as they keep track of learned dates in terms of iterations. Hence, transaction times represent the date on which a fact is extracted or recorded. The valid time indicates the time period on which a fact is considered valid or true. Furthermore, when such facts are learned/extracted from open text,

they are associated with some confidence score. We need a model that supports these three desirable aspects (transaction time, validity time, and confidence scores). Thus, in this work, we propose a framework for representing and querying bitemporal probabilistic KG.

Most existing KG contain schema that can be represented by lightweight ontology languages such as the OWL profiles (for instance, NELL's schema can be captured by the web ontology rule language called OWL RL [26]). In this paper, we study a bitemporal probabilistic extension of OWL RL as modeling language for KG.

Probabilistic graphical models have been widely used to reason about facts extracted at Web scale using a combination of hand-crafted and extracted inference rules [37]. In particular, Markov logic networks (MLN) can be used to deal with temporal relations in open information extraction [28] or checking the consistency of knowledge bases [6, 7]. MLN extends first order logic with uncertainty by attaching weights to formulas. In MLN, there are two important reasoning tasks, marginal and maximum a-posteriori (MAP) inference, in MLN. The former computes the probability of a set of random variables (temporal facts in our setting) whereas the later computes the most probable and consistent world (temporal knowledge graph). Since marginal inference does not scale well, in this work, we present a novel approximate algorithm to compute the marginal distributions of temporal queries efficiently.

In bitemporal probabilistic KG, it is necessary to remove redundant facts, known as deduplication, so as to avoid errors in query answers and reduce the size of the graph. Hence, we embark upon a challenging problem in coalescing under uncertainty. It amounts to merging facts with identical non-temporal arguments and adjacent or overlapping time-intervals. For instance, consider the deterministic facts $\langle r(a, b, 2, 5), 0.8\rangle$ and $\langle s(a, b, 4, 7), 0.7\rangle$ as well as the axiom $r \sqsubseteq s$, clearly the two facts cannot be coalesced with the well known sort-merge approach [2]. Hence, we need to first perform inference using the axiom so that we utilize sort-merge to get the coalesced fact $s(a, b, 2, 7)$. However, we still need to determine the weight of the coalesced fact. For this task, we propose a number of approaches including a rule-based algorithm for coalescing that uses marginal inference to determine the weight of the coalesced fact.

Overall, the contributions of this paper are: (i) we propose a bitemporal model for Web knowledge graphs by considering OWL RL as an atemporal ontology language, (ii) we extend it (bitemporal KG) using MLN for modeling uncertainty, (iii) we address temporal coalescing (both in data and queries) in a probabilistic setting, (iv) we present a novel N-hop based approximate algorithm for marginal inference, (v) we show that the bitemporal scoping of probabilistic facts does not introduce any complexity overhead, and (vi) we provide an empirical evaluation of the proposed approach over the Wikidata KG.

1.1 Motivating Example

To motivate the purpose of this study we rely on two prominent knowledge graphs: NELL and Wikidata. NELL records the extraction dates of facts as shown in Table 1. For instance, it is extracted or recorded on 09/02/2017 that Fernando Torres plays for the Chelsea football club with a confidence of 96.9%. This shows that NELL's representation model resembles that of transaction time from relational databases. On the other hand, Wikidata scopes temporally some of the facts, for instance, with 100% confidence Fernando Torres played for Atletico Madrid from 2001 to 2007 before rejoining them on 2016 and he is still playing for them. This shows that Wikidata uses the valid time model for modeling temporal information. However, Wikidata does not record the extraction date of facts, conversely, NELL does not maintain the valid time of facts. Thus, what is missing is a model which combines both

Table 1 NELL and Wikidata representations of the career of Fernando Torres.

NELL: learned fact	Date learned	Confidence
$(torres, type, athlete)$	12/01/2010	100.0
$(torres, stadium, anfield)$	10/11/2015	100.0
$(torres, playsfor, chelsea)$	09/02/2017	96.9
$(torres, playsfor, spain)$	08/08/2011	87.5
Wikidata: extracted fact	Valid time	Confidence
$(torres, playsfor, atleticoMadrid)$	$[2001, 2007)$	100.0
$(torres, playsfor, liverpool)$	$[2007, 2011)$	100.0
$(torres, playsfor, chelsea)$	$[2011, 2015)$	100.0
$(torres, playsfor, atleticoMadrid)$	$[2016, now)$	100.0

transaction time and valid time (i.e., bitemporal). For instance, the fact that "Fernando Torres played for Chelsea football club from 2011 to 2015" is extracted on 09/02/2017, can be modeled as a bitemporal fact as shown below:
($torres$, $playsfor$, $chelsea$, [2011,2015), $[09/02/2017, UC)$, 96.9), where UC is short for until changed – when the end date of the transaction or recording date is unknown.

In addition to temporal probabilistic facts, a KG can contain (temporal) inference rules, for instance, some of the deduction rules learned from the NELL KG using ProbFOIL$^+$ [33] are shown below: (i) if an athlete led a sports team, then she probably plays for that team; (ii) if an athlete led a sports team and her team plays against another team, then she probably plays for that team; and (iii) if an athlete plays some sport and a team plays that sport, then that athlete probably plays for that team. These rules can be converted into temporal inference rules by adding temporal variables to the predicates and using a numerical predicate that tests interval overlap.

	Inference rule	Weight
$i.$	$athleteledsportsteam(a, b) \rightarrow playsfor(a, b)$	0.93
$ii.$	$athleteledsportsteam(a, y), teamplaysagainstteam(b, y) \rightarrow playsfor(a, b)$	0.96
$iii.$	$athleteplayssport(a, y), teamplayssport(b, y) \rightarrow playsfor(a, b)$	0.93

In order to perform reasoning tasks over probabilistic temporal facts and temporalized inference rules (see Section 4), in this study, we propose a bitemporal model for probabilistic knowledge graphs.

2 Background

2.1 OWL RL

A knowledge graph (KG) is a set of triples that can be be encoded in the W3C standard RDF data model [22]. Let I and L be two disjoint sets denoting the set of IRIs (identifying resources) and literals (character strings or some other type of data), respectively. We abbreviate the union of these sets ($\mathsf{I} \cup \mathsf{L}$) as IL. A triple has the form $(s, r, o) \in \mathsf{I} \times \mathsf{I} \times \mathsf{IL}$ where s is the *subject*, r is the *predicate* or relation, and o is the *object* of the triple. A KG can be extended with temporal information by labeling each triple in the graph with a temporal element. For instance, the temporal element can represent the time period in which the triple is valid, i.e., the *valid time* of the triple [29, 18, 20]. KG often contain an ontology or schema that is encoded in some lightweight language such as OWL RL.

OWL RL is a tractable rule-based ontology language. It prohibits the use of disjunctions of classes and existential quantification for superclass expression to enable polynomial time reasoning [26]. We use description logic for concisely expressing the syntax of OWL RL

axioms. We denote a set of concept names by $\mathsf{N_C} \subseteq \mathsf{I}$; roles names by $\mathsf{N_R} \subseteq \mathsf{I}$; individual names by $\mathsf{N_I} \subseteq \mathsf{IL}$; and their union by N i.e., $\mathsf{N} \subseteq \mathsf{N_C} \cup \mathsf{N_R} \cup \mathsf{N_I}$. We do not consider datatypes for the sake of space, however, our work can be easily extended (for instance by following the work in [5]). Additionally, we use the following notations through out the paper: a and b denote *instance or individual names* in $\mathsf{N_I}$; A and B denote *class or concept names* in $\mathsf{N_C}$; C (resp. D) is a *complex class expression* denoting *subclass expression* (resp. *superclass expression*); p, p^-, r, r_1, r_2 are *role names* in $\mathsf{N_R}$; s and s_i denote axioms or assertions. An OWL RL knowledge graph contains a set of axioms. The syntax of which is given by the following grammar:

$$
\begin{aligned}
\mathit{Axiom}, \mathsf{s} &::= C \sqsubseteq D \mid r_1 \sqsubseteq r_2 \mid r_1 \circ r_2 \sqsubseteq r \mid A(a) \mid r(a,b) \\
C &::= A \mid \bot \mid \{a\} \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \\
D &::= A \mid \bot \mid \neg D \mid D \sqcap D \mid \forall r.D \mid \exists r.\{a\} \mid \leqslant 1\ r.C \mid \leqslant 0\ r.C \\
r, r_1, r_2 &::= p \mid p^-
\end{aligned}
$$

We refer to axioms of the form $A(a)$ and $r(a,b)$ *instance assertions* (facts) and those that appear under the subsumption ($\sqsubseteq$) relation (for instance $r_1 \sqsubseteq r_2$) are called *inclusion axioms*. The assertion $A(a)$ (resp. $r(a,b)$) can be written in RDF syntax as $(a, \texttt{type}, A)$ (resp. (a, r, b)). In this paper, we consider a temporal extension of OWL RL where the instance assertions are temporal and the inclusion axioms are atemporal. We refer to temporal instance assertions as *temporal facts*. In order to facilitate the transformation of OWL RL axioms into first-order formulas, we consider the following normal forms [26]:

$$
\begin{array}{lllll}
C(a) & r(a,b) & A \sqsubseteq C & A \sqcap B \sqsubseteq C & r_1 \sqsubseteq r_2 \\
A \sqsubseteq \{a\} & A \sqsubseteq \leqslant 1\ r.C & A \sqsubseteq \forall r.C & \{a\} \sqsubseteq C & r_1 \circ r_2 \sqsubseteq r \quad r_1^- \sqsubseteq r_2 \quad \mathit{dis}(r_1, r_2)
\end{array}
$$

The axiom $\mathit{dis}(r_1, r_2)$ denotes that r_1 and r_2 are disjoint. Note that the axiom $r_1^- \sqsubseteq r_2$ is subsumed by the axiom $r_1 \sqsubseteq r_2$. An OWL RL KG is in *normal* form if its axioms are normalized. Throughout this paper, we assume that the axioms of KG are normalized. The normal forms are obtained by leveraging structural transformation. Although, the OWL specification provides a partial axiomatization of the OWL RDF-based semantics using a fixed set of OWL RL/RDF rules[1]. In this study, we focus on the OWL direct semantics and translate the OWL RL inference rules into first-order formulas. However, our approach is also applicable to OWL RDF-based semantics. A probabilistic extension of an OWL RL KG can be efficiently modeled using Markov logic network. Although, in this paper, we use Markov logic network, our approach allows to easily adapt other probabilistic modeling frameworks such as ProbLog and probabilistic soft logic.

2.2 Markov Logic Network

A Markov logic network (MLN) combines Markov networks and first-order logic (FOL) by attaching weights to first-order formulas. An MLN program $\mathcal{L}$ is a set of pairs $\mathcal{L} = (f_i, w_i)$ where f_i is a FOL formula and w_i is a real number representing its weight [34]. In this paper, we use the Horn fragment of FOL which efficiently represents OWL RL inference rules. For brevity, we will drop $\forall$ quantifier from all the formulas. The probabilistic facts and rules given in the motivating example can be seen as an MLN program.

[1] https://www.w3.org/TR/owl2-profiles/

Together with a set of constants $\mathcal{C}$, an MLN defines a Markov network $M_{\mathcal{L},\mathcal{C}}$, where $M_{\mathcal{L},\mathcal{C}}$ contains one node for each possible grounding of each predicate appearing in $\mathcal{L}$. The value of the node is 1 if the ground predicate is true, and 0 otherwise. The probability distribution over a possible world x, specified by a ground Markov network $M_{\mathcal{L},\mathcal{C}}$, is:

$$P(X = x) = Z^{-1}\exp\Big(\sum\nolimits_i w_i n(f_i, x)\Big),$$

where $n(f_i, x)$ is the number of true groundings of f_i in x. The groundings of a formula are formed simply by replacing its variables with constants in all possible ways. A ground MLN can be turned into a factor graph. A *factor graph* is a set of factors $\Phi = \{\phi_1, \ldots, \phi_n\}$ where each factor ϕ is a function $\phi(X)$ over a set of random variables X. A factorization of a function g over the variables X is given by: $g(X) = \prod_{i=1}^{n} \phi_i(X_i)$. We convert a ground MLN into a factor graph by using the following: each ground atom $p_i(a)$ in an MLN becomes a random variable X_i, and each ground formula (f_i, w_i) becomes a factor $\phi(X_i)$ which has a value e^{w_i} if the formula is true and 1 otherwise. Such a factor graph determines a probability distribution over X,

$$P(X = x) = Z^{-1}\prod\nolimits_i \phi(X_i) = Z^{-1}\exp\Big(\sum\nolimits_i w_i n_i(f_i, x)\Big).$$

There are two important reasoning tasks in MLN. The first one is called *marginal* inference which is the the task of computing the probability of a set of variables given evidence. The complexity of this problem is known to be #P-hard. The second one is *maximum a-posteriori* (MAP) inference which is the task of finding *the most probable state* of the world, i.e., finding a complete assignment to all ground atoms which maximizes the probability. This problem is known to be NP-hard. We will study these inference tasks for bitemporal probabilistic KG but first we present bitemporal KG.

3 Bitemporal Knowledge Graphs

A bitemporal KG is an extension of a conventional KG by adding temporal elements to each instance assertion in the graph (similar to bitemporal databases [27]). A fact in a bitemporal KG is timestamped with time intervals that represent the fact's *valid time* and *transaction time*. A valid time is the time period in which a fact is considered true or valid. Transaction time is the time when a fact is added to a KG. Thus, a bitemporal KG needs domains for two temporal universes, the valid time universe and the transaction time universe, and it may be desirable or convenient to restrict them to some subset of T. Therefore, let $\mathsf{T_v} \subseteq \mathsf{T}$ denote the valid time universe of a bitemporal KG, and $\mathsf{T_t} \subseteq \mathsf{T}$ denote its transaction time universe. We consider $\mathsf{T} = \mathsf{T_v} \cup \mathsf{T_t}$ to be a discrete time domain which is a linearly ordered finite sequence of *time points*; for instance, days, minutes, or milliseconds. The finite domain assumption ensures that there are finitely many possible worlds in the probabilistic extension. A *time interval* is an ordered pair $[t_b, t_e)$ of time points, with $t_b \leq t_e$ and $t_b, t_e \in \mathsf{T}$, which denotes the closed-open interval of time points from t_b to t_e[2]. We will work with the interval-based temporal domain to define our data model.

[2] It is possible to extend to other interval based representations such as $[t_b, t_e]$, left and right closed interval.

▶ **Definition 1** (Bitemporal KG). *A bitemporal KG is a tuple $G = \langle \mathcal{S}, \mathcal{A} \rangle$ where $\mathcal{S}$ is the atemporal component representing the schema part (in OWL RL) and $\mathcal{A}$ is a set of OWL RL instance assertions in which each assertion $A(a)$ (resp. $r(a,b)$) in the graph is associated with a valid time $[v_b, v_e) \in \mathsf{T_v}$ and transaction time $[t_b, t_e) \in \mathsf{T_t}$, i.e., $g = A(a, [v_b, v_e), [t_b, t_e))$ (resp. $g = r(a, b, [v_b, v_e), [t_b, t_e))$). We refer to g as a bitemporal fact and write it as a first order predicate $\text{CA}(a, A, v_b, v_e, t_b, t_e)$ or $\text{RA}(a, r, b, v_b, v_e, t_b, t_e)$ with CA for concept assertion and RA for role assertion.*

Right-unlimited time intervals are expressed as $[t_b, UC)$ for transaction time and $[v_b, now)$ for valid time, where UC is short for Until Changed and now denotes the current time instance. Note that both UC and now are replaced by the current time during reasoning. For the sake of presentation, we remove the day and month of a given date and write just the years for both valid and transaction time intervals.

▶ **Example 1.** *We convert some of the facts of the KG in Table 1 into bitemporal facts as shown below. This is the same as aligning (or loosely merging) the NELL and Wikidata KG for the task of KG completion or bitemporal KG construction.*

$ra(torres, playsfor, chelsea, 2011, 2015, 2017, UC)$
$ra(torres, stadium, anfield, 2007, 2011, 2015, UC)$
$ra(torres, playsfor, atleticoMadrid, 2001, 2007, 2016, UC)$

Bitemporal KG expansion. Relying on the above example, we can use a rule-based approach for KG expansion. For instance, using the NELL KG, we can extend Wikidata with the help of rules (a rule per relation) of the following form:

$$playsfor(x, y, t_b, t_e), playsfor(x, y, v_b, v_e) \rightarrow playsfor(x, y, v_b, v_e, t_b, t_e).$$

It is possible to use more complex rules that include predicates which test the semantic similarity of relation names. However, this is beyond the scope of this work.

For a bitemporal KG G, its *snapshot* at a valid time v and a transaction time t is the graph $G(v,t)$ (the non-temporal KG): $G(v,t) = \{A(a) \mid A(a, [v,v), [t,t)) \in G\} \cup \{r(a,b) \mid r(a,b,[v,v),[t,t)) \in G\}$. The atemporal or non-temporal KG associated with a bitemporal KG is $u(G) = \bigcup_{v,t} G(v,t)$, the union of the graphs $G(v,t)$. Relying on this characterization, we define temporal entailment.

▶ **Definition 2** (Bitemporal entailment). *We define temporal entailment as follows: for two bitemporal knowledge graphs G_1 and G_2, $G_1 \models_{v,t} G_2$ if and only if $G_1(v,t) \models G_2(v,t)$ for each v and t; $\models_{v,t}$ denotes bitemporal entailment and $\models$ is the standard OWL RL entailment [26].*

Alternatively, bitemporal entailment can be reduced into temporal entailment defined in [20]. For two bitemporal graphs G_1 and G_2, let the valid graphs be $G_1(v) = \{A(a, [t_b, t_e)) | A(a, [v, v), [t_b, t_e)) \in G\}$ and similarly $G_2(v)$; and transaction graphs be $G_1(t) = \{A(a, [v_b, v_e)) | A(a, [v_b, v_e), [t, t)) \in G\}$ and similarly $G_2(t)$. $G_1 \models G_2$ iff $G_1(v) \models_t G_2(v)$ and $G_1(t) \models_v G_2(t)$ where $\models_t$ and $\models_v$ denote temporal entailment.

▶ **Definition 3.** *For a bitemporal KG G, a translation of the normal forms of OWL RL axioms into FOL predicates is given by a bijective function φ as shown.*

$$
\begin{array}{lcl}
A \sqsubseteq C & \mapsto & \text{SC}(A, C) \\
A \sqcap B \sqsubseteq C & \mapsto & \text{INT}(A, B, C) \\
A \sqsubseteq \{a\} & \mapsto & \text{SC}(A, a) \\
A \sqsubseteq \forall r.C & \mapsto & \text{ALL}(A, r, C) \\
A \sqsubseteq \leqslant 1 r.C & \mapsto & \text{ATMOST}(A, r, C) \\
r_1 \sqsubseteq r_2 & \mapsto & \text{SP}(r_1, r_2) \\
r_1 \circ r_2 \sqsubseteq r & \mapsto & \text{RCOMP}(r_1, r_2, r)
\end{array}
\qquad
\begin{array}{lcl}
r_1^- \sqsubseteq r_2 & \mapsto & \text{INV}(r_1, r_2) \\
dis(r_1, r_2) & \mapsto & \text{DIS}(r_1, r_2) \\
A(a) & \mapsto & \text{CA}(a, A) \\
\{a\} \sqsubseteq C & \mapsto & \text{CA}(a, C) \\
r(a, b) & \mapsto & \text{RA}(a, r, b) \\
a \in \mathsf{N_I} & \mapsto & \text{NOM}(a)
\end{array}
$$

We propose a rule-based instance retrieval in which we compute all entailments of the form $\text{CA}(a, A, v_b, v_e, t_b, t_e)$ and $\text{RA}(a, r, b, v_b, v_e, t_b, t_e)$ for a given KG. For this task, we use the temporal OWL RL inference rules shown in Figure 1. These rules are applied repeatedly, to a given KG, until no new conclusion can be made. It is worth noting that the inference rules do not materialize assertions that can be deduced if the KG is inconsistent. Nevertheless, inconsistencies lead to entailments of the form $\text{CA}(a, \bot, v_b, v_e, t_b, t_e)$ for some constant a.

▶ **Lemma 1.** *Let G be a bitemporal KG with an OWL RL schema, its closure can be computed, by applying the rules in Figure 1 until closure, in polynomial time.*

The above lemma follows from the results in OWL RL [26]. Note that in this work, we use FOL syntax of KG axioms, assertions and inference rules for brevity. However, it is also possible to use *reification* in order to represent bitemporal KG using the RDF syntax. Alternatively, more compact representations based on RDR (reification done right) can be utilized [21]. In addition, a more detailed discussion, on the syntax of temporal KG, can be found in [4]. As an example, the first temporal fact in Example 1, can be represented in RDR syntax as follows:

```
<<torres playsfor chelsea>> beginValid 2011 ;
                            endValid 2015 ;
                            beginTrans 2017 ;
                            endTrans UC .
```

In the above syntax, beginValid, endValid, beginTrans, and endTrans are vocabularies denoting the start and end points of valid and transaction time intervals. A query language for bitemporal KG can be defined by extending query languages for valid time KG such as SPARQ-LTL [4]. For the sake of space, we do not present a query language for bitemporal KG.

4 Bitemporal Probabilistic Knowledge Graphs

In this section, we propose a data model for bitemporal probabilistic KG.

4.1 Data Model

A bitemporal probabilistic KG contains a set of temporally annotated facts each with a confidence weight. A formal definition is provided below.

▶ **Definition 4.** *A bitemporal probabilistic KG is a tuple $K = (\mathcal{S}, \mathcal{A}, \mathcal{X} \cup \mathcal{R})$ where $\mathcal{S}$ is a set of OWL RL axioms, $\mathcal{A} = \{(g, w_1), \ldots, (g_n, w_n)\}$ is a set of bitemporal facts in which $g_i \in \mathcal{A}$ has a confidence w_i; $\mathcal{R}$ is temporal OWL RL inference rules given in Figure 1, each rule r_i is deterministic and has weight $w_i = \infty$; and $\mathcal{X} = \{(f_1, w_1), \ldots, (f_m, w_m)\}$ is a set of OWL RL axioms representing background knowledge where each w_j denotes the weight of formula f_i; and $\mathcal{F} = \mathcal{X} \cup \mathcal{R}$.*

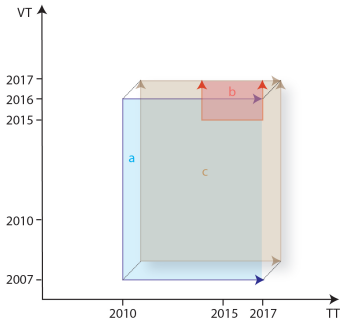
Given a bitemporal probabilistic KG $K = (\mathcal{S}, \mathcal{A}, \mathcal{X} \cup \mathcal{R})$, the probability of a possible world $K' = (\mathcal{S}', \mathcal{A}', \mathcal{X} \cup \mathcal{R}) \subseteq K$ is given by the following log-linear distribution:

$$P(K') = Z^{-1} \exp\left(\sum_{(f_i, w_i) \in \mathcal{X}' : K \models_{v,t} f_i} w_i n(f_i, K') \right), \quad Z = \sum_{K_j \subseteq K} \exp\left(\sum_i w_i n(f_i, K_j) \right);$$

where $n(f_i, K')$ is the number of groundings of the formula f_i that evaluate to true in the world K', Z is the normalization constant, and $\models_{v,t}$ denotes bitemporal entailment.

$$
\begin{aligned}
(r_1)\ & \text{SC}(a,b), \text{CA}(x,a,v,t) \rightarrow \text{CA}(x,b,v,t).\\
(r_2)\ & \text{INT}(a,b,c), \text{CA}(x,a,v^1,t^1), \text{CA}(x,b,v^2,t^2), \text{OV}(v^1,v^2,t^1,t^2) \rightarrow \text{CA}(x,c,v,t).\\
(r_3)\ & \text{ALL}(a,r,c), \text{CA}(x,a,v^1,t^1), \text{RA}(x,r,y,v^2,t^2), \text{OV}(v^1,v^2,t^1,t^2) \rightarrow \text{CA}(y,c,v,t).\\
(r_4)\ & \text{ATMOST}(a,r,c), \text{CA}(x,a,v^1,t^1), \text{RA}(x,r,y,v^2,t^2),\\
& \text{CA}(y,c,v^3,t^3), \text{RA}(y,r,z,v^4,t^4), \text{CA}(z,c,v^5,t^5), \text{OV}(v^1,t^1,\ldots,t^5,t^5) \rightarrow \mathit{equal}(y,z,v,t).\\
(r_5)\ & \text{SP}(r,s), \text{RA}(x,r,y,v,t) \rightarrow \text{RA}(x,s,y,v,t).\\
(r_6)\ & \text{RCOMP}(r,s,t), \text{RA}(x,r,y,v^1,t^1), \text{RA}(y,s,z,v^2,t^2), \text{OV}(v^1,t^1,v^2,t^2) \rightarrow \text{RA}(x,t,z,v,t).\\
(r_7)\ & \text{INV}(r,s), \text{RA}(y,r,x,v,t) \rightarrow \text{RA}(x,s,y,v,t).\\
(r_8)\ & \text{DIS}(r,s), \text{RA}(a,r,b,v^1,t^1), \text{RA}(a,s,b,v^2,t^2), \text{OV}(v^1,t^1,v^2,t^2) \rightarrow \text{CA}(a,\bot,v^1,t^1).\\
(r_9)\ & \text{CA}(x,a,v,t) \rightarrow \text{CA}(x,\top,v,t).\\
(r_{10})\ & \text{CA}(x,a,v,t), \text{NOM}(a) \rightarrow \text{CA}(a,x,v,t).\\
(r_{11})\ & \text{CA}(x,a,v,t), \text{NOM}(a) \rightarrow \text{NOM}(x).\\
(r_{12})\ & \text{RA}(x,r,y), \text{CA}(z,y,v,t), \text{NOM}(y) \rightarrow \text{RA}(x,r,z,v,t).\\
(r_{13})\ & \neg\text{CA}(x,\bot,v,t).
\end{aligned}
$$

Figure 1 *Notation*: $v^i = [v^i_b, v^i_e), v = [v_b, v_e), t^i = [t^i_b, t^i_e)$, and $t = [t_b, t_e)$. A temporalized variant of OWL RL inference rules that we denote by $\mathcal{R}$. They are partially drived from the materialization calculus in [26]. All of the formulas are universally quantified over all the variables. OV is short for overlaps, it checks if a given set of intervals are overlapping. $\top$ and $\bot$ are constant symbols representing top and bottom concepts. Note that rule r_{13} does not belong to the inference rules for OWL RL. This rule takes the notion of inconsistency into account.



a $\langle \mathit{livesIn}(\mathit{John}, \mathit{Paris}, [2007, 2016), [2010, \mathit{now})), 0.9\rangle$
b $\langle \mathit{livesIn}(\mathit{John}, \mathit{Paris}, [2015, \mathit{now}), [2015, \mathit{now})), 0.5\rangle$
c $\langle \mathit{livesIn}(\mathit{John}, \mathit{Paris}, [2007, \mathit{now}), [2010, \mathit{now})), ?\rangle$

Figure 2 A graphical representation of coalescing bitemporal probabilistic facts. The X-axis represents transaction time (TT) and the Y-axis is valid time (VT). The goal is to determine the weight "?" of the coalesced fact **c**.

An important problem in temporal databases is coalescing, we investigate this problem in a probabilistic setting for bitemporal KG.

4.2 Coalescing and Deduplication

Coalescing is a technique used in temporal databases for duplicate elimination [9]. Coalescing is the process of merging bitemporal facts with identical non-temporal arguments and adjacent or overlapping time-intervals. Hence, it is important for deduplication. In other words, coalescing is useful for: reducing the size of probabilistic temporal KG, and preventing incorrect answers in query evaluation. For instance, consider the query "*does John live in*

Paris from 2014 to 2017?" on the temporal facts of Figure 2 before coalescing, i.e., (a) and (b). The result is no, however, the same query on the coalesced fact (c) (brown part of the figure), returns yes. Uncoalesced facts can arise in various cases: during query evaluation via projection or union operations, by not enforcing coalescing in update or insertion operations, and through information extraction from diverse sources or accuracy of the extractor.

A bitemporal KG K is called duplicate-free, if for all pairs of facts $\text{RA}(a, r, b, v_b, v_e, t_b, t_e)$, $\text{RA}(a, r, b, v'_b, v'_e, t'_b, t'_e) \in K$, it holds that: $[v_b, v_e) \cap [v'_b, v'_e) = \emptyset$ and $[t_b, t_e) \cap [t'_b, t'_e) = \emptyset$. In other words, if the non-temporal terms of two temporal facts are the same, then their temporal terms must be disjoint (non-overlapping). Coalescing is challenging in a probabilistic setting, because it is not clear what should be the weight of the new (coalesced) fact. In order to coalesce bitemporal facts, either the valid time or transaction time intervals must be overlapping. The weight of the coalesced fact can be computed using a number of approaches that are shown below. For the sake of space, we will not present a comparison of the approaches. Instead, for our purpose, we use a rule-based approach that uses probabilistic inference in order to compute the exact weight of a coalesced fact.

1. Max: $w_3 = max(w_1, w_2)$, if two facts can be coalesced, assign the higher weight of the two to the coalesced fact,
2. Average: $w_3 = (w_1 + w_2)/2$,
3. Weighted-Average: $w_3 = (\ell_1 w_1 + \ell_2 w_2)/(\ell_1 + \ell_2)$, where ℓ_1 and ℓ_2 are the lengths of the intervals of fact 1 and fact 2,
4. Min: $w_3 = min(w_1, w_2)$ this is similar to Godel's fuzzy conjunction operator,
5. Lukasiewicz's relaxation: this approach works if the weights are between 0 and 1, $w_3 = max(0, w_1 + w_2 - 1)$, and
6. Rule-based coalescing with marginal inference: in order to coalesce the facts of a bitemporal probabilistic KG K, we can construct Horn rules for each relation in the KG, i.e., for each concept or relation m_i in K, we create a rule of the following form:

$$
\begin{aligned}
(r_{14}) \quad & \text{CA/RA}(x, m_i, y, v_b, v_e, t_b, t_e), \text{CA/RA}(x, m_i, y, v'_b, v'_e, t'_b, t'_e), \\
& v = min(v_b, v'_b), v' = max(v_e, v'_e), v_b \leq v'_e,\ v'_b \leq v_e, \\
& t = min(t_b, t'_b), t' = max(t_e, t'_e), t_b \leq t'_e,\ t'_b \leq t_e \rightarrow \text{CA/RA}(x, m_i, y, v, v', t, t').
\end{aligned}
$$

The expression $v_b \leq v'_e,\ v'_b \leq v_e$ (resp. $t_b \leq t'_e,\ t'_b \leq t_e$) tests temporal overlap of the intervals $[v_b, v_e)$ and $[v'_b, v'_e)$ (resp. $[t_b, t_e]$ and $[t'_b, t'_e]$). Besides, *min*, and *max* are predicates representing minimum, and maximum functions respectively. The weights of the coalesced facts are determined by performing marginal inference on the given KG. Once these weights are determined, the uncoalesced facts are removed from the KG. The approach uses one rule for each relation. If a KG has several thousands of relations, we need the same number of coalescing rules to remove duplicates from the KG. Hence, this operation can be very expensive, however, it is done only once. Our rule-based coalescing procedure is given in Algorithm 1.

▶ **Example 2.** *Consider coalescing the bitemporal probabilistic facts, (a) and (b), shown in Figure 2. This operation merges the two facts into one with the weight of the new fact (c) computed by marginal inference.*

Algorithm 1 Coalescing bitemporal probabilistic KG.

procedure COALESCE(K)
  Input: uncoalecsed bitemporal KG $K = (\mathcal{S}, \mathcal{A}, \mathcal{F})$
  Output: coalesced KG K_c
  $\mathsf{N} \leftarrow$ all the concepts and relations in K
  for each concept and relation $m \in \mathsf{N}$ **do**
    $r_m \leftarrow$ create a rule using (r_{14})
    add r_m to $\mathcal{M}$
  end for
  $\mathcal{F} \leftarrow \mathcal{F} \cup \mathcal{M}$
  repeat
    ground the rules in $\mathcal{F}$ using the assertions in $\mathcal{A}$ (see Algorithm 2)
    $\mathcal{A}' \leftarrow$ gather coalesced/inferred assertions
  until closure
  compute the marginal probabilities of the coalesced facts $\mathcal{A}'$ (Algorithm 2)
  use sort-merge to coalesce assertions in $\mathcal{A}$ and $\mathcal{A}'$ (Bohlen et al. [2])
  delete the uncoalesced ones in $\mathcal{A}$ and $\mathcal{A}$'
  return $K_c = (\mathcal{S}, \mathcal{A} \cup \mathcal{A}', \mathcal{F} \setminus \mathcal{M})$
end procedure

5 Inference

In this section, we present two important reasoning tasks in bitemporal probabilistic KG, namely, marginal and maximum a-posteriori (MAP) inference. We denote the set of constants in a bitemporal probabilistic KG $K = (\mathcal{S}, \mathcal{A}, \mathcal{F})$ by $\mathcal{C}$, $\mathcal{C} \subseteq \mathsf{IL} \cup \mathsf{T}$ as the union of sets of IRIs, literals and time points. The Herbrand base $\mathrm{HB}(\mathcal{F})$ of $\mathcal{F}$ can be constructed by instantiating all the variables in $\mathcal{F}$ using the constants in $\mathcal{C}$ (aka. *grounding* or *expansion*). The function θ, given a finite set $\mathcal{C}$, maps each fact in some bitemporal KG into a subset of the Herbrand base $\mathrm{HB}(\mathcal{F})$ of $\mathcal{F}$ with respect to $\mathcal{C}$. Each subset of the Herbrand base is a Herbrand interpretation specifying which ground atoms are true. A Herbrand interpretation H is a Herbrand model of $\mathcal{F}$, denoted by $\models_H \mathcal{F}$, iff it satisfies all groundings of the formulas in $\mathcal{F}$.

▶ **Definition 5.** *Given $K = (\mathcal{S}, \mathcal{A}, \mathcal{F})$ over a set of constants $\mathcal{C}$ and $\mathrm{HB}(\mathcal{F})$ the Herbrand base of $\mathcal{F}$ with respect to $\mathcal{C}$, $\theta : \mathcal{S} \cup \mathcal{A} \rightarrow \mathrm{HB}(\mathcal{F})$ maps $\mathcal{S} \cup \mathcal{A}$ into subsets of $\mathrm{HB}(\mathcal{F})$ as follows:*

$$\theta(\mathcal{S} \cup \mathcal{A}) = \bigcup\nolimits_{y \in \mathcal{S} \cup \mathcal{A}} \varphi(y),$$

$\varphi()$ maps axioms and assertions into FOL predicates using Definition 5. Besides, we extend θ as follows $\theta(K) = \theta(\mathcal{S} \cup \mathcal{A}) \cup \mathcal{F}$ where θ is a bijective function, it induces a one-to-one correspondence between the Herbrand models of $\mathcal{F}$ and *expanded* KG. As already discussed, φ maps inclusion axioms and instance assertions into first-order predicates. Grounding $\mathcal{F}$ with the constants of a bitemporal probabilistic KG may generate a set of new facts; this results in an *expanded* KG.

▶ **Lemma 2.** *Let $K = (\mathcal{S}, \mathcal{A}, \mathcal{F})$ be a bitemporal probabilistic KG; let $\mathcal{C}$ be a set of constants; and let $HB(\mathcal{F})$ be the Herbrand base of $\mathcal{F}$ with respect to $\mathcal{C}$. We have two cases:*

- *for any $K' \subseteq K$, $K \models_{v,t} K' \Rightarrow \theta(K') \models_H \mathcal{F}$ and*
- *for any $H \subseteq HB$, $H \models_H \mathcal{F} \Rightarrow \theta^{-1}(H) \models_{v,t} K''$ and $K'' \subseteq K$.*

5.1 Marginal Inference

Marginal inference is the task of computing the marginal distributions of a set of random variables (queries). In our setting, given a query q and a bitemporal probabilistic KG K, marginal inference involves in computing the probability of the answers of q over K. In this paper, we are interested in Boolean temporal conjunctive queries.

▶ **Definition 6.** *A Boolean temporal conjunctive query q is a formula of the form*

$$q \leftarrow r_1(\mathbf{x}_1, \mathbf{v}_1, \mathbf{t}_1), \ldots, r_i(\mathbf{x}_i, \mathbf{v}_i, \mathbf{t}_i), \ldots, r_n(\mathbf{x}_n, \mathbf{v}_n, \mathbf{t}_n)$$

where $\mathbf{x}_i$ are atemporal variables or constants that are in r_i,

- *$\mathbf{v}_i$ are valid time variables or time points that are in r_i,*
- *$\mathbf{t}_i$ are transaction time variables or time points that are in r_i, and*
- *r_i denotes a temporal relation $\text{CA}(x_i, y_i, v_b, v_e, t_b, t_e)$ or $\text{RA}(x_i, y_i, v_b, v_e, t_b, t_e)$ with $x_i, y_i \in \mathbf{x}_i$, $v_b, v_e \in \mathbf{v}_i$ and $t_b, t_e \in \mathbf{t}_i$.*

In probabilistic databases and statistical relational learning, often the probabilities of queries are computed by grounding, i.e., by replacing all the variables in the queries using constants in the database. The grounding is used to generate a propositional sentence (lineage of a query) for exact inference or a graphical model for approximate computation. Similarly, Boolean temporal conjunctive queries can be grounded by evaluating queries using the *techniques from temporal databases* [16] and instantiating or replacing all the variables in the queries using their answers. This results in a set of ground queries, the probability of which can be computed using MLN systems or any other approximate inference engine. For Boolean temporal conjunctive queries, it is necessary to take care of the overlap of time intervals during grounding. One solution to this problem is to rewrite queries to take into account overlaps. In other terms, Boolean temporal conjunctive queries require checking interval intersection to determine the overlap of intervals in the query predicates. In order to do this, we rewrite queries as discussed below.

The marginal probability of a query q is obtained by a two-step process: (i) firstly, rewrite the query q, and (ii) secondly, perform marginal computation. To rewrite q we add a function called OV (overlaps) which tests if both valid time and transaction times of relations are overlapping. Formally, the rewriting of a Boolean temporal query q:

$$q \leftarrow r_1(\mathbf{x}_1, \mathbf{v}_1, \mathbf{t}_1), \ldots, r_i(\mathbf{x}_i, \mathbf{v}_i, \mathbf{t}_i), \ldots, r_n(\mathbf{x}_n, \mathbf{v}_n, \mathbf{t}_n),$$

is the following:

$$q_r \leftarrow r_1(\mathbf{x}_1, \mathbf{v}_1, \mathbf{t}_1), \ldots, r_i(\mathbf{x}_i, \mathbf{v}_i, \mathbf{t}_i), \ldots, r_n(\mathbf{x}_n, \mathbf{v}_n, \mathbf{t}_n), \text{OV}(\mathbf{v}_1, \ldots, \mathbf{v}_n), \text{OV}(\mathbf{t}_1, \ldots, \mathbf{t}_n),$$

where $\text{OV}(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ (resp. $\text{OV}(\mathbf{t}_1, \ldots, \mathbf{t}_n)$) is a Boolean function that tests if the valid (resp. transaction) time intervals $\mathbf{v}_1, \ldots, \mathbf{v}_n$ (resp. $\mathbf{t}_1, \ldots, \mathbf{t}_n$) are overlapping.

▶ **Definition 7.** *Given a Boolean temporal conjunctive query q and a bitemporal probabilistic KG $K = (\mathcal{S}, \mathcal{A}, \mathcal{F})$, the probability of q over K is computed using the following:*

$$P(q|\mathcal{S}, \mathcal{A}, \mathcal{F}) = Z^{-1}\exp\Big(\sum_{(f_j, w_j) \in \mathcal{F}: q_r \models_{v,t} f_j} w_j n(f_j, q_r, \mathcal{S}, \mathcal{A})\Big),$$

where q_r is the rewriting of q and $\models_{v,t}$ is bitemporal entailment. Note that computing Z takes exponential time in the worst case.

Algorithm 2 Approximate query probability.

1: **procedure** APPROXPROBABILITY(K,q,N)
2: **Input:** $K = (\mathcal{S}, \mathcal{A}, \mathcal{F})$, query q and hop N
3: **Output:** $P_a(q)$
4: $q_r \leftarrow$ rewrite(q)
5: Schema and assertion tables $T_\mathcal{S}$, $T_\mathcal{A} \leftarrow$ load($\mathcal{S}, \mathcal{A}$),
6: Rules table $T_\mathcal{F} \leftarrow$ load($\mathcal{F}$),
7: **for** $f_i \in T_\mathcal{F}$ **do** $T_K \leftarrow T_\mathcal{S} \bowtie T_\mathcal{A} \bowtie f_i$ **end for**
8: Answers table $T_q \leftarrow$ evaluate query(q_r, T_K)
9: Factor graph, $G_F \leftarrow \emptyset$
10: **for** each inference rule $f_i \in T_\mathcal{F}$ **do**
11: $G_F \leftarrow G_F \cup_B (T_q \bowtie f_i)$
12: **end for**
13: $G_F^N \leftarrow$ extract n-hop subgraph(G_F, T_q, N)
14: $P_a(q) \leftarrow$ compute(G_F^N,T_q)
15: **end procedure**

The complexity of computing the probability of a query is known to be #P-hard in general. The above definition shows that temporal scoping of instance assertions does not increase the complexity (since the OV function can be computed during ground which is before probability computation). This is consistent with the results in temporal databases, as already shown a temporal query has the same properties as that of a first-order query langauge [42].

5.1.1 Approximate Marginal Inference

Since exact marginal probability computation is an expensive operation (see the experimental results in Figure 3), often approximate sampling techniques are used to speed up inference. In general, a large portion of a KG is not relevant, for computing the probability of a given query, because in a Markov network a node/variable is independent of all the other nodes/variables given its neighboring nodes/variables (known as the Markov blanket). More formally, given a query variable x_i, $P(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|\text{MB}(x_i))$ where $\text{MB}(x_i)$ is the Markov blanket of x_i. Thus for a given a query, a part of a KG up to a certain depth N, containing the query node, can be extracted to estimate the probability of that query. This approach is described in Algorithm 2. The algorithm takes as input a KG K, a query q and hop distance N. The atemporal schema and temporal assertions are loaded into relational database tables[3] $T_\mathcal{S}$ and $T_\mathcal{A}$ respectively. Likewise, the inference rules are loaded into the table $T_\mathcal{F}$. After loading the KG, we perform KG expansion by applying the inference rules until closure (line 7). Then after, we evaluate the rewriting q_r of the query q on the expanded KG T_K and keep its answers in a relational table T_q (line 8). In lines 9–12, we create a factor graph G_F by iteratively joining the answers table T_q with that of inference rules; $\cup_B$ denotes a bag unions operation. An N-hop subgraph – all nodes that are N distance away from the query node – G_F^N is extracted from G_F with a depth of N hops from the query nodes T_q (line 13). Finally, in line 14, we compute the approximate probability $P_a(q)$ of q using either exact or approximate (by using a Gibbs sampler) inference. Lifted inference is a focus of recent research which leverages the structure (e.g. using symmetries) of MLN

[3] Note that we exclude the discussion of the schema of the tables and SQL join queries for brevity.

knowledge bases for efficient inference. It is worthwhile comparing lifted inference with that of approximate marginal inference as well as investigating the possibility for a combined approach. We leave out this task as a future work.

5.2 MAP Inference

In bitemporal probabilistic KG, MAP inference is the problem of computing the most probable, consistent, and non-probabilistic bitemporal KG (also known as MAP state). More formally, given a KG $K = (\mathcal{S}, \mathcal{A}, \mathcal{F})$ and a mapping function θ, we denote the MAP problem by $map(\theta(K))$. In order to compute $map(\theta(K))$, we need to translate K with the function θ into an equivalent MLN formalization. Then, the inference rules $\mathcal{F}$ are added to this translation. The MAP state is obtained by using $\theta(K)$ and the grounding of $\mathcal{F}$ as input. Applying the inverse translation function θ^{-1} to the MAP state, yields the most probable bitemporal KG.

▶ **Lemma 3.** *Let $K = (\mathcal{S}, \mathcal{A}, \mathcal{F})$ be a bitemporal probabilistic KG, $\mathcal{C}$ a set of constants, and the Herbrand base $HB(\mathcal{F})$ of $\mathcal{F}$ with respect to $\mathcal{C}$. The MAP state of K is obtained as:*

$$\theta^{-1}(H) = \underset{H \subseteq HB : H \models_H \mathcal{F}}{\arg\max} \left(\sum_{(f, w_j) \in \mathcal{F} : H \models_{v,t} f_j} w_j n(f_i, H) \right)$$

Using the above lemma, it can be proved that computing the most probable and consistent bitemporal KG is NP-hard in general. This is shown by exploiting the correspondence between a bitemporal probabilistic KG and an MLN program, and using the bijective function θ. Membership is shown by mapping MAP inference in a bitemporal probabilistic KG into MAP inference in MLN. Hardness is shown by translating MLN programs into KG. From Lemma 3 and the results in [5] it follows that the problem of computing a MAP state is NP-hard.

6 Empirical Evaluation

We conducted two different kinds of experiments: (i) marginal and (ii) MAP inference. For both experiments, we carried out performance tests in terms of running times and the accuracy of marginal distributions. We run the experiments on a Debian 8 virtual machine with 2.6GHz 3-core Intel Haswell processor, 24 GB of main memory and 1TB disk space.

Tools. We used the following probabilistic reasoners: ProbLog [25], Tuffy [30], and TuffyLite [24]. *ProbLog* is a probabilistic extension of Prolog. Unlike MLN, ProbLog assigns probabilities to clauses (first order Horn formulas) and poses the restriction that these probabilities are mutually independent. It defines a probability distribution over logic programs [25]. Its semantics is defined by the success probability of a query in a randomly sampled program. On the other hand, *TuffyLite and Tuffy* are probabilistic reasoners for MLN programs. TuffyLite is an improved version of Tuffy. Hence, we use it for comparison with ProbLog. Note that we intentionally leave out DeepDive [38] from out experiments, because DeepDive uses the same technique as Tuffy and TuffyLite.

Data. For our experiments, we use the Wikidata KG. In particular, we consider a part of the KG that contains structured temporal information. Hence, we extracted temporal facts for various fluents (time-varying relations) including: *member of sports team*, *educated at*, *occupation*, *spouse*, and so on. Overall, we extracted over 3.7 million temporal facts. All of

these temporal facts are only interval timestampped (valid time). As Wikidata is a valid time KG, we extended it by adding transaction time intervals and probabilities to each fact in order to have a bitemporal probabilistic KG. We randomly generated transaction times and probabilities (> 0.5). Besides, as Wikidata is not complete, it does not contain valid times for all the facts. If the end time point of a fact is missing, we replace it with now.

Temporal rules. We designed 36 different bitemporal probabilistic inference rules (ProbLog definite clauses) based on the fluents of Wikidata. For ProbLog, the probabilities p_i of the rules are generated randomly and are set between 0.5 and 0.99. However, for the MLN solver, TuffyLite, we use the log odds of the probabilities as weights, i.e., $w_i = \ln \frac{p_i}{1-p_i}$.

6.1 Marginal Inference

In this experiment, we test the scalability of marginal inference. The results of the experiment are reported in Figure 3(a). As shown, ProbLog performs very well, it outperforms TuffyLite. The run time of marginal inference is almost constant when the number of facts is less than 20,000. This is because ProbLog uses *knowledge compilation* to speed up inference. The reported runtimes are averaged over 5 runs. For TuffyLite, we test the quality of approximate marginal probabilities, we plot the prior and marginal probabilities of 8 randomly selected queries in Figure 3(b). As it can be seen, the approximate marginals are very close to the prior probabilities with the highest absolute error close to 2% for query q_6.

6.2 MAP Inference

In this experiment, we report the running times of ProbLog on different data sizes. We exclude Tuffy and TuffyLite from this experiment due to scalability issues (after running for more than 24hs both systems did not terminate). ProbLog performs most probable explanation (MPE) inference which is slightly different from MAP inference. The runtimes, averaged over 5 runs, are reported in Figure 3(c). As it can be seen, the runtime of ProbLog does not increase linearly with respect to the size of the input data. This is due to each added incorrect bitemporal fact might be involved in a conflict resulting in a non trivial optimization problem. In order to test the scalability of MPE inference, we increased the data size upto 500,000. As expected, the running times were exceedingly high, i.e., upto several hours. When the datasize is 500,000, we interrupted the evaluation of MPE inference after a runtime of 3.63 hours. In both MAP and marginal inference, scalability is a big problem. We will tackle this problem in the future.

7 Related work

In relational databases, bitemporal databases have been throughly invesitgated [27]. Besides, temporal databases have been extensively studied (see surveys [31, 39]). However, relatively fewer works exist on temporal probabilistic databases [11, 8]. In [11], a relational database is used to model and query temporal data, integrity constraints and deduction rules can also be specified. However, these rules must be deterministic (unweighted) unlike what we do here. On the other hand, contrary to this study where we use a bitemporal model, uncertain spatio-temporal databases focus on stochastically modelling trajectories through space and time (see [14] for instance).

Query evaluation in probabilistic databases is an active area of research [19, 7, 40, 12]. With respect to temporal query evaluation over a valid time probabilistic KG, to the best of our knowledge, there are two important studies [6] and [11]. While the former focuses on

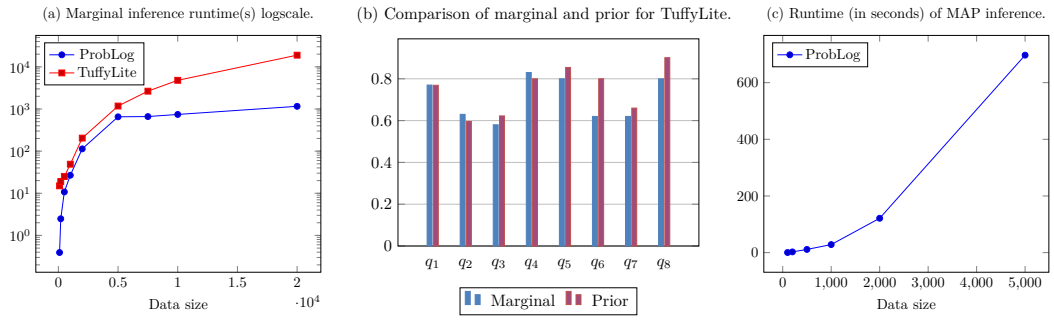


Figure 3 Performance of TuffyLite and ProbLog over fixed query and varying data size.

MAP inference, we study here both marginal and MAP inference in a bitemporal setting, besides, we deal with the problem of temporal coalescing and use a rich schema based on OWL RL. The later deals with marginal inference, the difference with this work are the following: (i) we consider weighted temporal OWL RL inference rules, (ii) we propose coalescing for bitemporal KG, and (iii) we introduce rewriting for coalescing of query answers. In another study [13], the authors proposed an approach for resolving temporal conflicts in RDF knowledge graphs. The idea is to use first-order logic Horn formulas with temporal predicates to express temporal and non-temporal constraints. However, these approaches are limited to a small set of temporal patterns and only allow for uncertainty in facts. Moreover, extending KG using open domain information extraction, will often also lead to uncertainty about the correctness of schema information; a large variety of inference rules and constraints, some of which will be domain specific, can also be the subject of uncertainty.

Multi-temporal RDF database models are first introduced by Grandi [18]. These models allow to capture several aspects of time in data such as validity, efficacy, transaction and so on in a non-probabilistic setting. By contrast, we consider validity and transaction times with a rich probabilistic schema.

8 Conclusion and Outlook

We have proposed a bitemporal model to assert the times in which facts are considered valid and the times when facts are added to a KG. Besides, we studied bitemporal probabilistic KG and proved that standard reasoning tasks such as marginal and MAP inference do not introduce any additional complexity. We have also introduced an efficient algorithm, for marginal inference, based on N-hop graph neighborhoods of query nodes. Furthermore, we have addressed coalescing in a probabilistic setting both in data and during query answering. Our experimental results show that scalability is challenging and approximate techniques with acceptable error margins can be adopted. As a future work, we plan to do further experiments, testing scalability as well as the accuracy of state-of-the-art reasoners and implementation of our approximate marginal inference algorithm.

References

1 Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735, 2007.

2 Michael H. Böhlen, Richard T. Snodgrass, and Michael D. Soo. Coalescing in Temporal Databases. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 180–191, 1996.

3 Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an Architecture for Never-Ending Language Learning. In *AAAI*, volume 5, page 3, 2010.

4 Melisachew Wudage Chekol, Valeria Fionda, and Giuseppe Pirrò. Time Travel Queries in RDF Archives. In *Joint proceedings of the 3rd Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW 2017) and the 4th Workshop on Linked Data Quality (LDQ 2017) co-located with 14th European Semantic Web Conference (ESWC 2017), Portorož, Slovenia, May 28th-29th, 2017.*, pages 28–42, 2017.

5 Melisachew Wudage Chekol, Jakob Huber, Christian Meilicke, and Heiner Stuckenschmidt. Markov Logic Networks with Numerical Constraints. In *ECAI 2016*, pages 1017–1025, 2016.

6 Melisachew Wudage Chekol, Giuseppe Pirrò, Joerg Schoenfisch, and Heiner Stuckenschmidt. Marrying Uncertainty and Time in Knowledge Graphs. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 88–94, 2017.

7 Yang Chen, Daisy Zhe Wang, and Sean Goldberg. ScaLeKB: scalable learning and inference over large knowledge bases. *The VLDB Journal*, 25(6):893–918, 2016.

8 Alex Dekhtyar, Robert Ross, and VS Subrahmanian. Probabilistic temporal databases, I: algebra. *ACM Transactions on Database Systems (TODS)*, 26(1):41–95, 2001.

9 Anton Dignös, Michael H Böhlen, and Johann Gamper. Temporal alignment. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 433–444. ACM, 2012.

10 Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *SIGKDD*, pages 601–610, 2014.

11 Maximilian Dylla, Iris Miliaraki, and Martin Theobald. A temporal-probabilistic database model for information extraction. *Proc. of the VLDB Endowment*, 6(14):1810–1821, 2013.

12 Maximilian Dylla, Iris Miliaraki, and Michael Theobald. Top-k query processing in probabilistic databases with non-materialized views. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 122–133. IEEE, 2013.

13 Maximilian Dylla, Mauro Sozio, and Martin Theobald. Resolving Temporal Conflicts in Inconsistent RDF Knowledge Bases. In *BTW*, pages 474–493, 2011.

14 Tobias Emrich, Hans-Peter Kriegel, Nikos Mamoulis, Matthias Renz, and Andreas Zufle. Querying uncertain spatio-temporal data. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 354–365. IEEE, 2012.

15 Opher Etzion. *Temporal databases: research and practice*, volume 1399. Springer Science & Business Media, 1998.

16 Dengfeng Gao, S Jensen, T Snodgrass, and D Soo. Join operations in temporal databases. *The VLDB Journal – The International Journal on Very Large Data Bases*, 14(1):2–29, 2005.

17 Kiril Gashteovski, Rainer Gemulla, and Luciano Del Corro. Minie: minimizing facts in open information extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2630–2640, 2017.

18 Fabio Grandi. Multi-temporal RDF ontology versioning. In *Proceedings of the 3rd International Workshop on Ontology Dynamics (IWOD-09)*, 2009.

19 Eric Gribkoff and Dan Suciu. SlimShot: In-Database Probabilistic Inference for Knowledge Bases. *PVLDB*, 9(7):552–563, 2016.

20 Claudio Gutierrez, Carlos A Hurtado, and Alejandro Vaisman. Introducing time into RDF. *Knowledge and Data Engineering, IEEE Transactions on*, 19(2):207–218, 2007.

21 Olaf Hartig and Bryan Thompson. Foundations of an alternative approach to reification in RDF. *arXiv preprint*, 2014. `arXiv:1406.3399`.

22 Patrick Hayes. RDF Semantics. W3C Recommendation, 2004 .

23 Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum. YAGO2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web*, pages 229–232. ACM, 2011.

24 Tushar Khot, Niranjan Balasubramanian, Eric Gribkoff, Ashish Sabharwal, Peter Clark, and Oren Etzioni. Markov logic networks for natural language question answering. *arXiv preprint*, 2015. `arXiv:1507.03045`.

25 Angelika Kimmig, Bart Demoen, Luc De Raedt, Vitor Santos Costa, and Ricardo Rocha. On the implementation of the probabilistic logic programming language ProbLog. *Theory and Practice of Logic Programming*, 11(2-3):235–262, 2011.

26 Markus Krötzsch. OWL 2 Profiles: An introduction to lightweight ontology languages. In Thomas Eiter and Thomas Krennwallner, editors, *Proceedings of the 8th Reasoning Web Summer School, Vienna, Austria, September 3–8 2012*, volume 7487 of *LNCS*, pages 112–183. Springer, 2012.

27 Anil Kumar, Vassilis J Tsotras, and Christos Faloutsos. Designing access methods for bitemporal databases. *IEEE Transactions on Knowledge and Data Engineering*, 10(1):1–20, 1998.

28 Xiao Ling and Daniel S Weld. Temporal Information Extraction. In *AAAI*, volume 10, pages 1385–1390, 2010.

29 Boris Motik. Representing and querying validity time in RDF and OWL: A logic-based approach. *J. Web Semantics*, 12:3–21, 2012.

30 Feng Niu, Christopher Ré, AnHai Doan, and Jude Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an RDBMS. *Proc. of the VLDB Endowment*, 4(6):373–384, 2011.

31 Gultekin Ozsoyoglu and Richard T Snodgrass. Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532, 1995.

32 Katerina Papaioannou and Michael Böhlen. TemProRA: Top-k temporal-probabilistic results analysis. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*, pages 1382–1385. IEEE, 2016.

33 Luc De Raedt, Anton Dries, Ingo Thon, Guy Van den Broeck, and Mathias Verbeke. Inducing Probabilistic Relational Rules from Probabilistic Examples. In *IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1835–1843, 2015.

34 Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

35 Anisa Rula, Matteo Palmonari, Andreas Harth, Steffen Stadtmüller, and Andrea Maurino. On the diversity and availability of temporal information in linked open data. *The Semantic Web–ISWC 2012*, pages 492–507, 2012.

36 Anisa Rula, Matteo Palmonari, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Jens Lehmann, and Lorenz Bühmann. Hybrid acquisition of temporal scopes for RDF data. In *European Semantic Web Conference*, pages 488–503. Springer, 2014.

37 Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, and Jesse Davis. Learning first-order horn clauses from web text. In *EMNLP*, pages 1088–1098, 2010.

38 Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. Incremental knowledge base construction using deepdive. *Proceedings of the VLDB Endowment*, 8(11):1310–1321, 2015.

39 Richard T Snodgrass. Temporal databases. In *Theories and methods of spatio-temporal reasoning in geographic space*, pages 22–64. Springer, 1992.

40 Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. Probabilistic databases. *Synthesis Lectures on Data Management*, 3(2):1–180, 2011.

41 Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. Coupled temporal scoping of relational facts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 73–82. ACM, 2012.

42 David Toman. Point vs. interval-based query languages for temporal databases. In *Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 58–67. ACM, 1996 .

43 Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

44 Esteban Zimányi. Temporal aggregates and temporal universal quantification in standard SQL. *ACM SIGMOD Record*, 35(2):16–21, 2006.

Qualitative Reasoning and Data Mining

Yakoub Salhi
CRIL - CNRS & Université d'Artois, Lens, France
salhi@cril.fr

Abstract

In this paper, we introduce a new data mining framework that is based on qualitative reasoning. We consider databases where the item domains are of different types, such as numerical values, time intervals and spatial regions. Then, for the considered tasks, we associate to each item a constraint network in a qualitative formalism representing the relations between all the pairs of objects of the database w.r.t. this item. In this context, the introduced data mining problems consist in discovering qualitative covariations between items. In a sense, our framework can be seen as a generalization of gradual itemset mining. In order to solve the introduced problem, we use a declarative approach based on the satisfiability problem in classical propositional logic (SAT). Indeed, we define SAT encodings where the models represent the desired patterns.

2012 ACM Subject Classification Information systems → Data mining; Information systems → Association rules; Theory of computation → Constraint and logic programming; Computing methodologies → Knowledge representation and reasoning

Keywords and phrases Qualitative Database, Qualitative Pattern Mining, Declarative Approach, SAT Modeling

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.9

1 Introduction

Data mining techniques are applied on different data types, such as transactions, sequences, graphs, texts, etc. In order to consider complex aspects of the real world, it is interesting to extend these techniques for knowledge discovery to new complex data, such as spatio-temporal pieces of information. However, it is important in this context to take into account the simplicity of the pattern structure. Thus, the challenge in this work is to propose a framework that allows us to deal with different complex data types and discovering patterns having a simple structure.

Qualitative reasoning is concerned with facilitating reasoning about complex entities and pieces of information through symbolic representation formalisms. In particular, this kind of reasoning is strongly related to human one and, for instance, it can be used for dealing with pieces of information that come from natural language. In the literature, the qualitative formalisms are widely used for reasoning about two physical entities of the world that are time and space (e.g. see [21]). Indeed, qualitative spatial and temporal reasoning is an important research field in Artificial Intelligence in general, and knowledge representation in particular. The spatial and temporal representation formalisms allow reasoning about configurations by abstracting numerical quantities of space and time thanks to qualitative relations, such as *inside*, *before*, *after*, etc. One of the best known qualitative representation formalisms is the Point Algebra [31], which allows representing and reasoning about the possible relative positions between two points on the timeline. The Interval Algebra [2, 3], for its part, is used for reasoning about the possible positions between two intervals. Furthermore, regarding qualitative spatial reasoning, the Region Connection Calculus RCC8 [25] is one of the most studied formalisms in qualitative reasoning, which concerns topological relations between two spatial regions.

In this work, we propose a framework for data mining using qualitative reasoning, which allows considering different data types, such as numerical values, time intervals and spatial regions. To this end, we first introduce the notion of qualitative database, which is defined by associating to each item a constraint network in a qualitative formalism representing the relations between the pairs of objects of the database w.r.t. this item. Then, we describe data mining tasks for discovering qualitative covariations, called qualitative itemsets, in the previous kind of databases. For instance, the desired patterns can capture pieces of information of the form "a variation of an item a w.r.t. the qualitative relation r_1 is associated with a variation of b w.r.t. the qualitative relation r_2". In a sense, the proposed tasks can be seen as a generalization of those related to gradual itemsets where the qualitative relations that are considered in the extracted patterns are only $\leq$ and $\geq$ on numerical values [7, 10, 11, 20]. We express the interestingness predicate on the qualitative itemsets in a database through two different definitions of support. The first definition takes into consideration a local view by reasoning about the pairs of objects that satisfy the partial order induced by the itemset, while the second is obtained by reasoning about the sequences respecting the previous partial order. These two definitions allow extracting interesting recurrent pieces of information. Finally, we use a declarative and flexible solution for solving the introduced data mining tasks based on the use of the satisfiability problem in classical propositional logic (SAT). Indeed, we define for each task a SAT encoding whose models allow us to obtain all the desired patterns. Thus, we follow in our solution the constraint programming based approach for data mining initiated in [24, 13], which offers a declarative and flexible representation model.

The rest of the paper is organized as follows. After describing related works in Section 2, we introduce in Section 3 the notion of qualitative database. In Section 4, we present the data mining tasks proposed in this work. In Section 5, we describe our SAT-based encodings for solving these tasks, while Section 6 concludes the paper.

2 Related Works

The most related data mining tasks to our framework are those concerned with extracting gradual itemsets [7, 10, 11, 20]. A gradual itemset is a pattern expressing covariations of items having as domains sets of numerical values. For instance, the gradual itemset containg three gradual items $\{sport^{\geq}, weight^{\geq}, diseases^{\leq}\}$ can be used to express the fact "the higher the time of physical activity, the higher the weight loss, and the fewer the number of diseases". The gradual itemset structure allows analyzing numerical data in a simple and intuitive way, since it avoids the quantitative aspects of the considered data.

The data mining framework introduced in this work can be seen as a generalization of that of mining gradual itemsets in the case of numerical data. Indeed, instead of using only the inequality relations $\leq$ and $\geq$, many binary qualitative relations on different data types can be used in our framework, in particular qualitative relations on time intervals and spatial regions.

It is worth noting that we use in our framework measures for determining the quality of a qualitative itemset similar to those proposed in the case of gradual itemsets. In fact, in the same way as in gradual itemset mining, we consider two distinct definitions of support: the first definition considers the pairs of objects that respect the itemset, while the second definition is obtained by reasoning about the length of the sequences that respect the pattern. More precisely, the first definition of support corresponds to the numbers of pairs of objects that satisfy the partial order associated to the pattern, and the second definition corresponds the length of the longest sequences of objects that are ordered using the partial order induced by the pattern.

The use of a declarative approach for data mining was originally proposed in [24] for performing different tasks. Specifically, the authors have demonstrated that constraint programming is an appropriate tool in many respects in itemset mining. One of the main motivations lies in the fact that this framework offers a flexible and generic representation model. Indeed, new constraints often require new implementations for specialized data mining approaches, which can often be integrated in a fairly simple way into declarative frameworks, since it is not needed to change the solving tools. In addition, the continual evolution in the efficiency of tools dedicated to problems that can be used for data mining modeling, like ASP (*Answer Set Programming*), CSP (*Constraint Satisfaction Problem*) and SAT, is a strong argument in favor of using approaches based on these problems. Thus, from this first work, a new line of research has emerged within the data mining community. Indeed, in recent years, many works using CSP and SAT for different data mining tasks have been proposed in the literature (e.g. [13, 16, 12, 30, 19, 9]). In particular, in [8], the authors show that their SAT-based approach achieves better performance than state-of-the-art specialized techniques. In this work, we use a SAT-based approach for solving all the considered data mining tasks. Let us note that a SAT-based approach was recently used for extracting gradual patterns in [22].

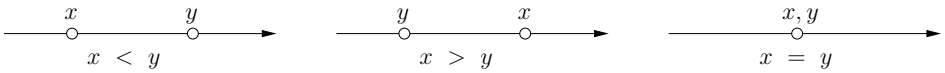
3 Qualitative Database

In this section, we introduce the notion of qualitative database. The main idea consists in associating to each item a constraint network in a qualitative formalism representing the relations between the pairs of objects of the database w.r.t. this item. To illustrate our proposal, we consider three distinct qualitative formalisms for reasoning about time and space, namely Point Algebra [31], Interval Algebra [2, 3] and Region Connection Calculus RCC8 [25].

Given a finite set S, we use $\mathcal{P}(S)$ and $|S|$ to denote respectively the powerset and the cardinality of S. Given a finite set of items $\mathcal{I}$, V_a is used to denote the domain of the item $a \in \mathcal{I}$. The domain of an item can be a numerical value, a temporal interval, a spatial region, etc. Further, we associate to each item a a finite set of qualitative base relations B_a, which consists of *jointly exhaustive* and *pairwise disjoint* relations, i.e., for each $(v, v') \in V_a \times V_a$, there exists exactly one $b \in B_a$ such that $(v, v') \in b$. Further, we only consider the set of qualitative base relations B_a that contains the identity relation $id = \{(v, v') \in V_a \times V_a \mid v = v'\}$, and is closed under the inverse operation $(\cdot)^{-1}$, namely whenever b is in B_a, the inverse $(b)^{-1}$ is also in B_a. A qualitative relation is said to be *universal* if it contains all the base relations.

The *weak composition* of two base relations b and b' in B_a, denoted $b \diamond b'$, is defined as the set of base relations $\{b'' \in B_a \mid \exists (v, v') \in b \ \& \ (v', v'') \in b' \ \& \ (v, v'') \in b''\}$. The weak composition operation is extended to the relations in $\mathcal{P}(B_a)$ as follows: $r \diamond r' = \bigcup_{b \in r, b' \in r'} b \diamond b'$. In this context, it is worth mentioning that the *composition* $\circ$ of two relations is defined as follows: $r \circ r' = \{(v, v') \mid \exists v'', (v, v'') \in r \ \& \ (v'', v') \in r'\}$. In other words, $r \diamond r'$ is the largest set of base relations where each one shares at least one value with $r \circ r'$.

For example, consider the point algebra (PA) qualitative formalism described in Figure 1, which has been mainly used for temporal reasoning. Indeed, PA can be used to encode temporal relations between two points in the timeline. We also describe in Figure 2 the base relations of two other qualitative formalisms: interval algebra (IA) and region connection calculus RCC8. The formalism IA allows encoding relative relations between intervals, while RCC8 allows encoding topological relations between two regions. For instance, the expression $DC(Region1, Region2)$ represents the fact that the two spatial regions $Region1$ and $Region2$ are disconnected.



(a) The base relations of Point Algebra.

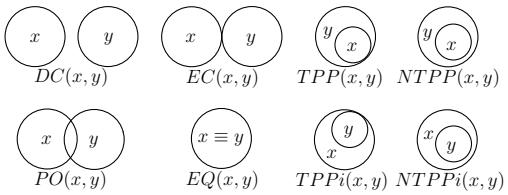
b	$(b)^{-1}$
$<$	$>$
$>$	$<$
$=$	$=$

(b) The inverse table of Point Algebra.

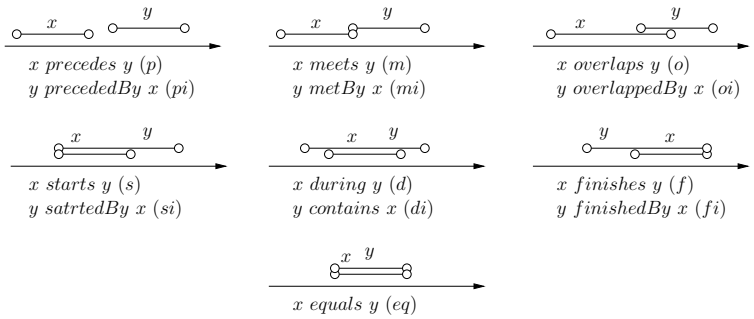
$\diamond$	$<$	$>$	$=$
$<$	$\{<\}$	$\{<,>,=\}$	$\{<\}$
$>$	$\{<,>,=\}$	$\{>\}$	$\{>\}$
$=$	$\{<\}$	$\{>\}$	$\{=\}$

(c) The composition table of Point Algebra.

Figure 1 Point Algebra.



(a) The base relations of RCC8.



(b) The base relations of Interval Algebra.

Figure 2 The qualitative formalisms RCC8 and Interval Algebra.

▶ **Definition 1** (Qualitative Column). *A* q-column *is a structure of the form* $c = (a, \mathcal{O}, R)$*, where* a *is an item, denoted* $item(c)$*,* $\mathcal{O}$ *is a finite non empty set of objects, denoted* $obj(c)$*, and* R *is a mapping form* $\mathcal{O} \times \mathcal{O}$ *to* B_a*, denoted* $rel(c)$*.*

Let us now introduce the notion of qualitative database, which is defined by associating to each item a constraint network in a qualitative formalism representing the relations between the pairs of objects of the database w.r.t. this item.

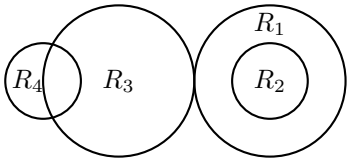
▶ **Definition 2** (Qualitative Database). *A* qualitative database *is a structure of the form* $(\mathcal{O}, \mathcal{I}, \mathcal{C})$*, where* $\mathcal{O}$ *is a finite non empty set of objects,* $\mathcal{I}$ *is a finite non empty set of items and* $\mathcal{C}$ *is a set of q-columns s.t. (i)* $|\mathcal{C}| = |\mathcal{I}|$*, (ii)* $\forall c \in \mathcal{C}$*,* $obj(c) = \mathcal{O}$*, and (iii)* $\forall a \in \mathcal{I}$*, there exists exactly one* $c \in \mathcal{C}$ *s.t.* $item(c) = a$*.*

In the sequel, we sometimes use R_a to denote $rel(c)$ where c is the qualitative column associated to the item a.

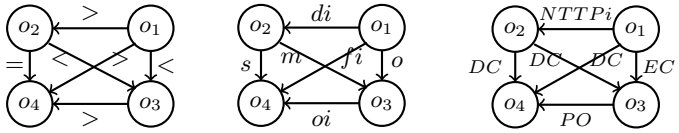
For example, we describe in Figure 3 a qualitative database: we provide in (a) a database using values in item domains, in (b) the concret situation of the considered spatial regions, and in (c) the qualitative database. For instance, the edge between o_1 and o_2 in the left-hand graph represents the qualitative base relation in PA $> (o_1, o_2)$, usually denoted $o_1 > o_2$.

objects	a	b	c
o_1	2	[1,4]	R_1
o_2	1	[2,3]	R_2
o_3	4	[3,6]	R_3
o_4	1	[2,4]	R_4

(a) A database using values in item domains.



(b) A representation of the real situation the regions R_1, R_2, R_3 and R_4.



(c) The qualitative database corresponding to the database in (a).

Figure 3 A Qualitative Database.

4 Mining Qualitative Itemsets

In this section, we introduce data mining tasks for discovering qualitative covariations in qualitative databases. For instance, the patterns in this context can be used to capture pieces of information of the form "a variation of a w.r.t. the qualitative relation r_1 is associated with a variation of b w.r.t. the qualitative relation r_2".

▶ **Definition 3** (Qualitative Itemset). *A qualitative itemset is a finite non empty set of qualitative items I, where a* qualitative item *is a structure of the form a^r where a is an item and $r \subseteq B_a$.*

Let us now describe the partial order on the objects of a database that is induced by a qualitative itemset, and also the notion of ordered sequence that is used for defining the support of a qualitative itemset.

▶ **Definition 4** (Induced Partial Order). *Let $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{C})$ be a qualitative database, $o, o' \in \mathcal{O}$ and $I = \{a_1^{r_1}, \ldots, a_k^{r_k}\}$ a qualitative itemset. Then, we say that o precedes o' w.r.t. I, written $o \preceq_I o'$, if for all $i \in 1..k$, $R_a(o, o') \in r_i$ holds.*

▶ **Definition 5** (Ordered sequence of objects). *Let $\mathcal{D}$ be a qualitative database, $L = \langle o_1, \ldots, o_k \rangle$ a sequence of distinct objects in $\mathcal{D}$ and I a qualitative itemset. We say that L respects I if it is ordered with respect to $\preceq_I$, i.e., $o_i \preceq_I o_{i+1}$ for every $i \in 1..k-1$.*

We here use $\mathcal{L}(\mathcal{D}, I)$ to denote all the sequences of objects occurring in $\mathcal{D}$ that respect the qualitative itemset I.

In the same way as in gradual itemset mining, we express the quality of an itemset in a database through two different definitions of *support*. The first definition captures a local view by taking into consideration the number of pairs that satisfy the partial order induced by the qualitative itemset ($\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{C})$):

$$supp_1(I, \mathcal{D}) = \frac{|\{\{o, o'\} \subseteq \mathcal{O} \mid o \neq o', o \preceq_I o'\}|}{|\mathcal{O}| \cdot (|\mathcal{O}| - 1)/2}.$$

The second definition is obtained by reasoning about the sequences that respect the qualitative itemset. Indeed, it corresponds to the length of the longest sequences that respect the considered itemset:

$$supp_2(I, \mathcal{D}) = \frac{max\{|L| \mid L \in \mathcal{L}(\mathcal{D}, I)\}}{|\mathcal{O}|}.$$

Furthermore, we consider that it is more appropriate to allow the user to select the relations that can be associated to every item in a pattern. For example, it is not interesting to consider the universal or empty relations because they do not describe any variation.

Thus, we define two problems of enumerating qualitative itemsets as follows: given a function f that maps each item a to a subset of relations $f(a) \subseteq \mathcal{P}(B_a)$ which is closed under the inverse operation and the inclusion, and a minimum support threshold v, the problems QIE1 and QIE2 consist in computing respectively the sets of qualitative itemsets $\mathcal{QIE}1(\mathcal{D}, f, v) = \{I \mid supp_1(I, \mathcal{D}) \geq v \ \& \ \forall a^r \in I, r \in f(a)\}$ and $\mathcal{QIE}2(\mathcal{D}, f, v) = \{I \mid supp_2(I, \mathcal{D}) \geq v \ \& \ \forall a^r \in I, r \in f(a)\}$.

Let us consider now two condensed representations, which are similar to those that are widely considered in itemset mining. Before that, we need the following partial order relation. Given two qualitative itemsets I and J, we have $I \sqsubseteq J$ if, $\forall a^r \in I$, $\exists a^{r'} \in J$ s.t. $r' \subseteq r$. Moreover, we have $I \sqsubset J$ if $I \sqsubseteq J$ and $I \neq J$.

▶ **Definition 6** (Closedness). *Let $\mathcal{D}$ be a database and I a qualitative itemset. Then, I is said to be a* closed qualitative itemset *in $\mathcal{D}$ w.r.t. $supp_1$ (resp. $supp_2$) if, for all qualitative itemset J with $I \sqsubset J$, $supp_1(I, \mathcal{D}) > supp_1(J, \mathcal{D})$ (resp. $supp_2(I, \mathcal{D}) > supp_2(J, \mathcal{D})$) holds.*

In other words, a qualitative itemset is closed if there is no more informative qualitative itemset that has the same support.

▶ **Definition 7** (Maximality). *Let $\mathcal{D}$ be a database, v a minimum support threshold and I a qualitative itemset. Then, I is said to be a* maximal qualitative itemset *w.r.t. $supp_1$ (resp. $supp_2$) and the threshold v if, for all qualitative itemset J with $I \sqsubset J$, $supp_1(J, \mathcal{D}) < v$ (resp. $supp_2(J, \mathcal{D}) < v$) holds.*

A qualitative itemset is maximal if there is no more informative qualitative itemset that has a support greater than or equal to the minimum support threshold.

In the context of the condensed representations, one can easily see that we have the following property.

▶ **Proposition 8** (Anti-Monotonicity). *Let $\mathcal{D}$ be a qualitative database and I and J two qualitative itemsets in $\mathcal{D}$. If $I \sqsubseteq J$ then $supp_1(I, \mathcal{D}) \geq supp_1(I, \mathcal{D})$ and $supp_2(I, \mathcal{D}) \geq supp_2(I, \mathcal{D})$.*

Therefore, using the anti-monotonicity property, computing either the closed itemsets or the maximal itemsets in $\mathcal{QIE}1(\mathcal{D}, f, v)$ and $\mathcal{QIE}2(\mathcal{D}, f, v)$ allows getting all the elements of these two sets. Furthermore, the anti-monotonicity property can be used for defining Apriori-like algorithms for solving the problems QIE1 and QIE2 in a fairly simple way. Let us recall that Apriori algorithm was originally proposed in [1] for mining frequent itemsets.

It is worth mentioning that the qualitative relations are not necessarily transitive. For example, we have $1\{<, >\}2\{<, >\}1$ in PA ($x\{<, >\}y$ means that x is different from y) without having $1\{<, >\}1$. This has as a consequence the fact that a sequence respects a qualitative itemset does not implies that its sub-sequences (by avoiding intermediate objects) respect also this pattern. Thus, in order to have transitivity, a solution can consist in restricting

our mining task to the relations that satisfy $\diamond$-*idempotence*: a qualitative relation r is said to be $\diamond$-*idempotent* if $r \diamond r = r$. For example, in PA the $\diamond$-idempotent relations are $\{=\}$, $\{<\}$, $\{<, =\}$, $\{>\}$, $\{>, =\}$ and $\{<, =, >\}$, i.e., all the relations except $\{\}$ and $\{<, >\}$. That being said, we provide in this work general methods for solving QIE1 and QIE2 without considering transitivity.

In order to illustrate the mining tasks described previously, we provide now a simple example. Consider the database described in Table 1. It represents pieces of information related to a set of workers about time at work, productivity and satisfaction degree. For the corresponding qualitative database, we consider interval algebra for time at work, and point algebra for both productivity and satisfaction degree. Moreover, we only consider QIE2 with a support threshold equal to 3 without any restriction on the considered qualitative relations in the patterns on `time`, but we only consider $\{<, \leq, >, \geq\}$ on both `productivity` and `satisfaction`. A first interesting qualitative pattern is $I = \{\texttt{time}^{\{p,o,m\}}, \texttt{productivity}^{\leq}\}$, which has a support equal to 4 since it is satisfied by the sequence $\langle w_1, w_2, w_3, w_4\rangle$. In a sense, it expresses that starting work earlier increase productivity. The pattern I is not closed since it has the same supports as $J = \{\texttt{time}^{\{p,o,m\}}, \texttt{productivity}^{<}\}$. Moreover, J is closed since $J \cup \{\texttt{satisfaction}^{\leq}\}$ and $J \cup \{\texttt{satisfaction}^{\geq}\}$ are respectively 2 and 3. Moreover, $J \cup \{\texttt{satisfaction}^{>}\}$ is a maximal patterns since its support is equal to the fixed threshold and it is not included in any other pattern.

Table 1 A description of a database.

`worker`	`time`	`productivity`	`satisfaction`
w_1	5am to 9am	100	1
w_2	8am to 12am	80	4
w_3	12am to 4pm	60	5
w_4	5pm to 9pm	50	3

5 SAT-based Approach for Enumerating Qualitative Itemsets

In this section, we introduce a SAT-based approach for solving the problems QIE1 and QIE2. We first describe the satisfiability problem in classical propositional logic. We then introduce our SAT encodings for QIE1 and QIE2: the computation of the models of each encoding corresponds to the computation of the desired qualitative itemsets. We here follow the constraint programming based approach for data mining initiated in [24, 13].

5.1 Classical Propositional Logic

We here describe the syntax and the semantics of classical propositional logic. We use Prop to denote the set of propositional variables. The propositional formulas of classical propositional logic (CPL) are built using Prop, the constants $\top$, denoting *true*, and $\bot$, denoting *false*, the unary logical connective $\neg$ and the usual binary connectives $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$. The grammar is defined as follows:

$$\phi ::= p \mid \top \mid \bot \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \phi \leftrightarrow \phi \mid \neg\phi$$

with $p \in$ Prop. The set of propositional formulas is denoted Form. We use the letters p, q, r, s to denote the propositional variables, and the Greek letters ϕ, ψ and χ to denote the propositional formulas. Moreover, given a syntactic object o, we use $Var(o)$ to denote the set of propositional variables occurring in o.

A *Boolean interpretation* $\mathcal{B}$ of a formula ϕ is defined as a function from the set of variables $Var(\phi)$ to $\{0,1\}$ (0 stands for $false$ and 1 for $true$). It is inductively extended to propositional formulas as usual:

$$\begin{array}{ll} \mathcal{B}(\top) = 1 & \mathcal{B}(\bot) = 0 \\ \mathcal{B}(\neg\phi) = 1 - \mathcal{B}(\phi) & \mathcal{B}(\phi \to \psi) = max(1 - \mathcal{B}(\phi), \mathcal{B}(\psi)) \\ \mathcal{B}(\phi \wedge \psi) = min(\mathcal{B}(\phi), \mathcal{B}(\psi)) & \mathcal{B}(\phi \wedge \psi) = min(\mathcal{B}(\phi), \mathcal{B}(\psi)) \end{array}$$

$$\mathcal{B}(\phi \leftrightarrow \psi) = 0 \text{ if } \mathcal{B}(\phi) \neq \mathcal{B}(\psi),\ \mathcal{B}(\phi \leftrightarrow \psi) = 1 \text{ otherwise}$$

A formula ϕ is satisfiable if there exists a Boolean interpretation $\mathcal{B}$ of ϕ such that $\mathcal{B}(\phi) = 1$, and $\mathcal{B}$ is called a *model* of ϕ in this case. We use $Mod(\phi)$ to denote the set of all the models of ϕ.

Consider for instance the formula $(p \wedge q) \leftrightarrow p$, which has exactly three models: $\mathcal{B}_1$ with $\mathcal{B}_1(p) = \mathcal{B}_1(q) = 0$; $\mathcal{B}_2$ with $\mathcal{B}_1(p) = \mathcal{B}_1(q) = 1$; and $\mathcal{B}_3$ with $\mathcal{B}_3(p) = 0$ and $\mathcal{B}_1(q) = 1$.

A propositional formula in *Conjunctive Normal Form* (CNF) is a conjunction of clauses, where a *clause* is a disjunction of literals. It is well-known that every propositional formula can be translated to CNF w.r.t. the satisfiability problem using Tseitin's linear encoding [29]. The problem of determining whether there exists a model that satisfies a given CNF formula, abbreviated as SAT, is one of the most studied NP-complete problems.

A *cardinality constraint* is an inequality of the form $\sum_{i=1}^{n} p_i \geq m$. Several polynomial encodings of this kind of constraints into propositional formulas have been proposed in the literature (e.g. [4, 26, 5]). An *AtMostOne constraint* is a particular case of the form $\sum_{i=1}^{n} p_i \leq 1$, which can be linearly encoded in SAT. For instance, the encoding using sequential counter [26, 23] is defined as follows:

$$(\neg p_1 \vee q_1) \wedge (\neg p_n \vee q_{n-1})$$
$$\bigwedge_{1<i<n} ((\neg p_i \vee q_i) \wedge (\neg q_{i-1} \vee q_i) \wedge (\neg p_i \vee \neg q_{i-1}))$$

where q_i is a fresh propositional variable for $i = 1, \ldots, n-1$.

5.2 A SAT Encoding for QIE1

In this section, we propose a SAT encoding for the problem of enumerating qualitative itemsets QIE1. More precisely, we associate to every instance of QIE1 a propositional formula so that its models allow us to obtain all the corresponding qualitative itemsets.

Let $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{C})$ be a qualitative database, f a function that maps each $a \in \mathcal{I}$ to a subset of $\mathcal{P}(B_a)$ closed under the inverse operation and the inclusion, and v a minimum support threshold. We here use the integer α defined as the value $v \cdot (|\mathcal{D}| \cdot (|\mathcal{D}| - 1)/2)$.

In order to define our encoding, we associate to each pair of an item a and a relation $r \in f(a)$ a distinct propositional variable denoted p_{a^r}. The variable p_{a^r} is used to express the qualitative itemset in the sense that it is true if and only if a^r belongs to the current qualitative itemset. Furthermore, we associate to each ordered pair of different objects (o, o') in $\mathcal{D}$ a distinct propositional variable denoted $q_{(o,o')}$. In the proposed encoding, a variable $q_{(o,o')}$ is true if and only if o precedes o' with respect to the current qualitative itemset. In order not to take into account both symmetric couples of objects in support computation, we also associate a variable denote $s_{\{o,o'\}}$ to each pair of distinct objects $\{o, o'\}$ in $\mathcal{D}$.

The first propositional formula of our encoding for QIE1 allows avoiding the empty itemset by requiring at least one item:

$$\bigvee_{a\in\mathcal{I}} \bigvee_{r\in f(a)} p_{a^r}. \tag{1}$$

Indeed, this formula corresponds to a single clause that expresses that there is at least one variable of the form p_{a^r} assigned to true.

The following conjunction of AtMostOne constraints allows avoiding the association of multiple variations to an item in the same pattern:

$$\bigwedge_{a\in\mathcal{I}} \sum_{r\in f(a)} p_{a^r} \leq 1. \tag{2}$$

More precisely, each AtMostOne constraint is associated to a distinct item and means that there is at most one qualitative relation associated to this item in the pattern.

The following formula allows establishing that each variable $q_{(o,o')}$ is true if and only o precedes o' w.r.t. the qualitative itemset:

$$\bigwedge_{o,o'\in\mathcal{O},o\neq o'} \neg q_{(o,o')} \leftrightarrow \bigvee(\{p_{a^r} \mid a\in\mathcal{I}, r\in(f(a)\setminus\{r'\in f(a) \mid R_a(o,o')\in r'\})\}. \tag{3}$$

We exactly express in the previous formula that $q_{(o,o')}$ is false if and only if there is a qualitative item a^r such that $r(o,o')$ does not hold.

We now introduce the formula that is used for symmetry breaking by considering in the support computation at most one of the couples (o,o') and (o',o):

$$\bigwedge_{o,o'\in\mathcal{O},o\neq o'} s_{\{o,o'\}} \leftrightarrow (q_{(o,o')} \vee q_{(o',o)}). \tag{4}$$

Finally, the following cardinality constraint expresses that support of every qualitative itemset in $\mathcal{D}$ has to be greater than or equal to v:

$$\sum_{o,o'\in\mathcal{O},o\neq o'} s_{\{o,o'\}} \geq \alpha. \tag{5}$$

Let us note that the use of α in the previous constraint is clearly equivalent to the use of v as a minimum support threshold.

We use $\mathcal{ENC}(\mathcal{D},f,v)$ to denote the conjunction of the previous formulas: (1) ∧ (2) ∧ (3) ∧ (4) ∧ (5).

There are three important properties related to our encoding $\mathcal{ENC}(\mathcal{D},f,v)$. First, the soundness property means that every model encodes a frequent qualitative itemset. Second, the completeness property expresses that every frequent qualitative itemset is encoded in a model of the encoding. Third, the non-redundancy property is used to capture the fact that there is a bijective mapping between the set of the models and the set of the frequent qualitative itemsets.

▶ **Proposition 9** (Soundness). *Given an instance $(\mathcal{D},f,v)$ of QIE1, if $\mathcal{B}$ is a model of $\mathcal{ENC}(\mathcal{D},f,v)$ then $I_{\mathcal{B}} = \{a^r \mid \mathcal{B}(p_{a^r}) = 1\} \in \mathcal{QIE}1(\mathcal{D},f,v)$.*

Proof. First, using the formula (1), we clearly have $|I_{\mathcal{B}}| \geq 1$. Then, using (2), we know that an item occurs at most once in every pattern. Moreover, using (3) ∧ (4), we obtain $\{s_{\{o,o'\}} \mid \mathcal{B}(s_{\{o,o'\}}) = 1\} = \{\{o,o'\} \subseteq \mathcal{O} \mid o \neq o', o \preceq_{I_{\mathcal{B}}} o'\}$. Thus, using the cardinality constraint (5), we obtain $|\{s_{\{o,o'\}} \mid \mathcal{B}(s_{\{o,o'\}}) = 1\}| \geq \alpha$ and we have thereby $supp_1(I_{\mathcal{B}},\mathcal{D}) \geq v$. Therefore, $I_{\mathcal{B}}$ belongs to $\mathcal{QIE}1(\mathcal{D},f,v)$. ◀

▶ **Proposition 10** (Completeness). *Given an instance $(\mathcal{D}, f, v)$ of QIE1, if $I \in \mathcal{QIE}1(\mathcal{D}, f, v)$ then there exists a Boolean interpretation $\mathcal{B}_I$ that satisfies the encoding $\mathcal{ENC}(\mathcal{D}, f, v)$, where $I = \{a^r \mid \mathcal{B}_I(p_{a^r}) = 1\}$.*

Proof. Let us define $\mathcal{B}_I$ as follows:
1. for every pair of an item a and a relation $r \in f(a)$, $\mathcal{B}_I(p_{a^r}) = 1$ iff $a^r \in I$;
2. for every ordered pair of distinct objects (o, o'), $\mathcal{B}_I(q_{(o,o')}) = 1$ iff $o \preceq_I o'$;
3. for every pair of distinct objects $\{o, o'\}$, $\mathcal{B}_I(s_{\{o,o'\}}) = 1$ iff $o \preceq_I o'$ or $o' \preceq_I o$.

Using the fact that $|I| \geq 1$, $\mathcal{B}_I$ satisfies (1). Then, using the fact that an item cannot occur more than once in I, $\mathcal{B}_I$ satisfies (2). Further, using the properties 1 and 2 in the definition of $\mathcal{B}_I$, we obtain that $\mathcal{B}_I$ satisfies (3). Using the fact that $\mathcal{B}_I$ satisfies (3) and the property 3 in the definition of $\mathcal{B}_I$, we also obtain that (4) is also satisfied by $\mathcal{B}_I$. Moreover, the formula (5) is satisfied since $supp_1(I, \mathcal{D}) \geq v$. ◀

▶ **Proposition 11** (Non-Redundancy). *Given an instance $(\mathcal{D}, f, v)$ of QIE1, for all two distinct models $\mathcal{B}$ and $\mathcal{B}'$ of $\mathcal{ENC}(\mathcal{D}, f, v)$, $\{a^r \mid \mathcal{B}(p_{a^r}) = 1\} \neq \{a^r \mid \mathcal{B}'(p_{a^r}) = 1\}$ holds.*

Proof. This property is a direct consequence of the fact that we use the equivalence logical connective in the formulas (3) and (4). Indeed, the support is encoded using the variables of the form $q_{(o,o')}$ and $s_{\{o,o'\}}$, and a qualitative itemset cannot have two distinct values for the support. ◀

It is worth noting that having a bijective mapping between the set of the models and the set of the frequent qualitative itemsets allows us to adapt in a fairly simple way our encoding for many variants of QIE1, such as counting the number of patterns.

Let us now introduce the notion of complementary qualitative itemset, which is mainly used for reducing the search space.

▶ **Definition 12** (Complementary Qualitative Itemset). *Let $I = \{a_1^{r_1}, \ldots, a_k^{r_k}\}$ be a qualitative itemset. The complementary of I, denoted I^c, is the qualitative itemset $\{a_1^{(r_1)^{-1}}, \ldots, a_k^{(r_k)^{-1}}\}$.*

We clearly have the following proposition.

▶ **Proposition 13.** *The following two properties are satisfied, for all qualitative database $\mathcal{D}$ and for all qualitative itemset I:*
- $supp_1(I, \mathcal{D}) = supp_1(I^c, \mathcal{D})$
- $supp_2(I, \mathcal{D}) = supp_2(I^c, \mathcal{D})$.

Proposition 13 can be used to avoid unnecessary computations. Indeed, at each found model, we can avoid in the next step both the corresponding qualitative itemset and its complementary itemset. It is worth noting that a similar property is used in the case of gradual patterns [7, 10, 11, 20].

Let us now consider the condensed representations corresponding to the closed and the maximal qualitative itemsets. In order to obtain the closed qualitative itemsets, we first need to conjunctively add to the encoding $\mathcal{ENC}(\mathcal{D}, f, v)$ the following formula:

$$\bigwedge_{a \in \mathcal{I}} \bigwedge_{r \in f(a)} ((\bigwedge_{o,o' \in \mathcal{O}, o \neq o'} (q_{(o,o')} \rightarrow R_a(o, o') \in r)) \rightarrow \bigvee_{r' \subseteq r} p_{a^{r'}}). \tag{6}$$

Indeed, this propositional formula means that, for all qualitative item a^r, if we have $supp_1(I, \mathcal{D}) = supp_1(I \cup \{a^r\}, \mathcal{D})$, then there exists $r' \subseteq r$ such that $a^{r'}$ belongs to I, where I is the qualitative itemset associated to the current model. In other words, it allows making the current qualitative itemset more informative without changing the support.

Then, we add the following formula to express that it is not possible to reduce the size of any relation in the pattern without changing the support:

$$\bigwedge_{a\in\mathcal{I}} \bigwedge_{r\in f(a),|r|>1} (p_{a^r} \rightarrow \bigwedge_{r'\subset r} (\bigvee_{o,o'\in\mathcal{O},o\neq o'} q_{(o,o')} \wedge R_a(o,o') \notin r')). \tag{7}$$

We use $\mathcal{ENC} - \mathcal{C}(\mathcal{D}, f, v)$ to denote the SAT encoding for the problem of enumerating the closed qualitative itemsets: $\mathcal{ENC}(\mathcal{D}, f, v) \wedge (6) \wedge (7)$.

Similarly, to compute the maximal qualitative itemsets, we only need to conjunctively add to $\mathcal{ENC}(\mathcal{D}, f, v)$ the following two formulas:

$$\bigwedge_{a\in\mathcal{I}} \bigwedge_{r\in f(a)} (\sum_{o,o'\in\mathcal{O},o\neq o'} (q_{(o,o')} \wedge R_a(o,o') \in r) \geq \alpha \rightarrow \bigvee_{r'\subseteq r} p_{a^{r'}}) \tag{8}$$

$$\bigwedge_{a\in\mathcal{I}} \bigwedge_{r\in f(a),|r|>1} (p_{a^r} \rightarrow \bigwedge_{r\subset r} \sum_{o,o'\in\mathcal{O},o\neq o'} (q_{(o,o')} \wedge R_a(o,o') \notin r') < \alpha). \tag{9}$$

The formula (8) allows maximizing the size of the current qualitative itemset while keeping the support greater than or equal to v, (9) states that it is not possible to reduce the size of any relation without reducing the support to a value smaller than v. We use $\mathcal{ENC} - \mathcal{M}(\mathcal{D}, f, v)$ to denote the SAT encoding $\mathcal{ENC}(\mathcal{D}, f, v) \wedge (8) \wedge (9)$.

5.3 A SAT Encoding for QIE2

We here propose a SAT encoding for the problem QIE2, which combines formulas defined for QIE1 and new ones that are described in this section.

Let $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{C})$ be a database, f a function that maps each $a \in \mathcal{I}$ to a subset of $\mathcal{P}(B_a)$ closed under the inverse operation and the inclusion, and v a minimum support threshold. We here use the integer β defined as the value $v \cdot |\mathcal{D}|$. We now describe an encoding that allows one to obtain all the elements of $\mathcal{QIE}2(\mathcal{D}, v)$.

In the same way as the previous encoding, we also use in the same way the propositional variables of the forms p_{a^r} and $q_{(o,o')}$: the variables of the form p_{a^r} are used to encode the qualitative itemset, and those of the form $q_{(o,o')}$ to encode its support. Moreover, we associate to each integer $i \in 1..\beta$ and object o in $\mathcal{D}$ a fresh propositional variable t^i_o, which is used to express that the object o is used at the location i in a sequence in $\mathcal{L}(\mathcal{D}, I)$, where I is the current qualitative itemset.

The first formula in our encoding is the conjunction of $(1) \wedge (2) \wedge (3)$ of the previous encoding $\mathcal{ENC}(\mathcal{D}, f, v)$. Indeed, (1) is used to express that every qualitative itemset contains at least one qualitative item, (2) is used to avoid multiple occurrences of an item in the same itemset, and (3) says that $q_{(o,o')}$ is false if and only if there is a qualitative item a^r such that $R_a(o, o') \in r$ does not hold. As a consequence, every model of the previous conjunction encodes a qualitative itemset, where the variables of the form $q_{(o,o')}$ encode the pairs of objects that satisfy the partial order induced by this itemset.

Using the fact that the propositional variables of the form t_o^i are used to build an ordered sequence of objects, the following formula means that an object cannot be used more than once in a sequence:

$$\bigwedge_{o \in \mathcal{O}} \sum_{i=1}^{\beta} t_o^i \leq 1. \tag{10}$$

The following formula says that there is exactly one object at each location:

$$\bigwedge_{i=1}^{\beta} \sum_{o \in \mathcal{O}} t_o^i = 1. \tag{11}$$

Clearly, the previous formula allows us to only consider the qualitative itemsets that have supports greater than or equal to v w.r.t. $supp_2$.

In order to require the ordering induced by the qualitative itemset, the following formula is used to capture the fact that if two objects o and o' occur in successive locations, then the couple (o, o') respects the qualitative itemset, which is expressed by the truth of the variable $q_{(o,o')}$:

$$\bigwedge_{o,o' \in \mathcal{O}, o \neq o'} \bigwedge_{i=1}^{\beta-1} ((t_o^i \wedge t_{o'}^{i+1}) \rightarrow q_{(o,o')}). \tag{12}$$

We use $\mathcal{ENC}2(\mathcal{D}, f, v)$ to denote the encoding that corresponds to the following conjunction: (1) $\wedge$ (2) $\wedge$ (3) $\wedge$ (10) $\wedge$ (11) $\wedge$ (12).

▶ **Proposition 14** (Soundness). *Given an instance $(\mathcal{D}, f, v)$ of QIE2, if $\mathcal{B}$ is a model of $\mathcal{ENC}2(\mathcal{D}, f, v)$ then $I_{\mathcal{B}} = \{a^r \mid \mathcal{B}(p_{a^r}) = 1\} \in \mathcal{QIE}2(\mathcal{D}, f, v)$.*

Proof. The soundness can be shown in the same way as in the case of QIE1. Using (1), we know that $I_{\mathcal{B}}$ contains at least one qualitative item. Then, using (2), each item occurs at most once in every qualitative itemset. Further, using (3), we obtain $a^r \in I_{\mathcal{B}}$ iff, for all $o, o' \in \mathcal{O}$, $\mathcal{B}(q_{(o,o')}) = 1$ iff $R_a(o, o') \in r$. Thus, using (10) $\wedge$ (11) $\wedge$ (12), we know that there exists a sequence $\langle o_1, \ldots, o_\beta \rangle$ which respects $I_{\mathcal{B}}$, where $\mathcal{B}(t_{o_i}^i) = 1$ for $i \in 1..\beta$. As a consequence, $supp_2(I_{\mathcal{B}}, \mathcal{D}) \geq v$ and $I_{\mathcal{B}}$ belongs to $\mathcal{QIE}2(\mathcal{D}, f, v)$ ◀

▶ **Proposition 15** (Completeness). *Given an instance $(\mathcal{D}, f, v)$ of QIE2, if $I \in \mathcal{QIE}2(\mathcal{D}, f, v)$ then there exists a Boolean interpretation $\mathcal{B}_I$ that satisfies the encoding $\mathcal{ENC}2(\mathcal{D}, f, v)$ where $I = \{a^r \mid \mathcal{B}_I(p_{a^r}) = 1\}$.*

Proof. First, given a sequence $s = \langle o_1, \ldots, o_\beta \rangle$ respecting I, we define $\mathcal{B}_I$ as follows:

- for every pair of an item a and a relation $r \in f(a)$, $\mathcal{B}_I(p_{a^r}) = 1$ iff $a^r \in I$;
- for every couple of distinct objects (o, o'), $\mathcal{B}_I(q_{(o,o')}) = 1$ iff $o \preceq_I o'$;
- for every object o and location $i \in 1..\beta$, $\mathcal{B}_I(t_o^i) = 1$ iff $o = o_i$.

For the same reasons described in the proof of Proposition 10, $\mathcal{B}_I$ satisfies (1) $\wedge$ (2) $\wedge$ (3). Then, using the fact that the length of s is β and the objects in this sequence are pairwise distinct, $\mathcal{B}_I$ satisfies also (10) $\wedge$ (11). Finally, using the fact that s respects the partial order induced by I, $\mathcal{B}_I$ satisfies (12). ◀

Contrary to our previous encoding, $\mathcal{ENC}2(\mathcal{D}, f, v)$ does not satisfy the non-redundancy property, since the same qualitative itemset may be associated to distinct sequences. However, this is not a problem for enumerating the qualitative itemsets without redundancy,

because we only need to conjunctively add the negation of the found qualitative itemset at each step instead of the negation of the found model. More precisely, if we found a model representing the qualitative itemset $I = \{a_1^{r_1}, \ldots, a_k^{r_k}\}$, then we conjunctively add the clause $\neg p_{a_1{}^{r_1}} \vee \cdots \vee \neg p_{a_k{}^{r_k}} \vee \bigvee_{a^r \notin I} p_a^r$ to avoid this itemset in the next steps.

In $\mathcal{ENC}2(\mathcal{D}, f, v)$, we use propositional variables that are associated to only β locations, since we aim at computing the qualitative itemsets having supports at least equal to v. However, for computing the closed qualitative itemsets, we need to have the exact value of the support, which means that we have to encode one of the longest sequences in each model of the SAT encoding. In order to avoid this problem, we propose an intermediate solution by restricting $\mathcal{ENC}2(\mathcal{D}, f, v)$ to the closed qualitative itemsets w.r.t. QIE1. In this context, we clearly have the following property.

▶ **Proposition 16.** *Let $\mathcal{D}$ be a qualitative database and I a qualitative itemset. If I is closed in $\mathcal{D}$ w.r.t. $supp_2$, then it is also closed in $\mathcal{D}$ w.r.t. $supp_1$.*

Proof. This property is a direct consequence of the fact that if $supp_2(I, \mathcal{D}) > supp_2(J, \mathcal{D})$, then $supp_1(I, \mathcal{D}) > supp_1(J, \mathcal{D})$ holds for every qualitative itemsets I and J with $I \sqsubset J$. ◀

Thus, the set of closed qualitative itemsets w.r.t. QIE2 is included in that of the qualitative itemsets obtained from the encoding $\mathcal{ENC}2(\mathcal{D}, f, v) \wedge (6) \wedge (7)$. As a consequence, the previous SAT encoding can be used for enumerating all the closed qualitative itemsets w.r.t. QIE2. Indeed, we only need in this context to select the largest patterns w.r.t. $\sqsubseteq$ for every value for the support.

Let us now consider the problem of enumerating the maximal qualitative itemsets. In this context, consider the following formulas:

$$\bigwedge_{a \in \mathcal{I}} \bigwedge_{r \in f(a)} \bigwedge_{o,o' \in \mathcal{O}, o \neq o'} \Big(\bigwedge_{i=1}^{\beta-1} ((t_o^i \wedge t_{o'}^{i+1} \wedge R_a(o,o') \in r) \rightarrow \bigvee_{r' \subseteq r} p_{a^{r'}}) \Big), \tag{13}$$

$$\bigwedge_{a \in \mathcal{I}} \bigwedge_{r \in f(a), |r|>1} (p_{a^r} \rightarrow \bigwedge_{r' \subset r} (\bigvee_{o,o' \in \mathcal{O}, o \neq o'} \bigwedge_{i=1}^{\beta-1} (t_o^i \wedge t_{o'}^{i+1} \wedge R_a(o,o') \notin r')). \tag{14}$$

These two formulas express that, for a sequence of length equal to β, the associated qualitative itemset has to be the largest w.r.t. $\sqsubseteq$. Therefore, in the same way as our encoding for enumerating the closed qualitative itemsets, the encoding $\mathcal{ENC}2(\mathcal{D}, f, v) \wedge (13) \wedge (14)$ allows one to compute a set of patterns that contains all the maximal qualitative itemsets.

It is worth noting that the strategies proposed in [15, 18] for adapting Conflict-Driven Clause-Learning (CDCL) based SAT-solvers to the task of model enumeration can be directly used in the case of our encoding. Furthermore, it is also possible to directly use the decomposition method introduced in [17] for improving the SAT-based approach in solving data mining problems.

6 Conclusion and Perspectives

The first main contribution of this article is a definition of a framework for data mining using qualitative reasoning. This framework allows considering different data types, such as numerical values, time intervals and spatial regions. Moreover, the data mining tasks introduced in this work can be seen as a natural generalization of those related to gradual

itemsets. The second main contribution is our declarative and flexible solution for solving the proposed data mining tasks based on the satisfiability problem in classical propositional logic (SAT): each task is modeled as a propositional formula whose models correspond to the desired patterns.

In our future work, we intend to further study qualitative reasoning in data mining following three main directions: (1) the use of disjunctions of base relations between objects, which allows, for instance, modeling vagueness; (2) considering qualitative formalisms that are not closed under the inverse operation, such as cardinal direction calculus [27, 28]; (3) considering some qualitative relations with arities greater than two in the case of some particular data types (e.g. [14, 6]). Furthermore, we plan to implement the proposed SAT-based methods to provide an experimental study on the use of our framework.

References

1 Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA*, pages 207–216. ACM Press, 1993.

2 James F. Allen. An Interval-Based Representation of Temporal Knowledge. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI '81, Vancouver, BC, Canada*, pages 221–226. William Kaufmann, 1981.

3 James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

4 Olivier Bailleux and Yacine Boufkhad. Efficient CNF Encoding of Boolean Cardinality Constraints. In *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland*, pages 108–122, 2003.

5 Olivier Bailleux, Yacine Boufkhad, and Olivier Roussel. A Translation of Pseudo Boolean Constraints to SAT. *JSAT*, 2(1-4):191–200, 2006.

6 Philippe Balbiani, Jean-François Condotta, and Gérard Ligozat. Reasoning about Cyclic Space: Axiomatic and Computational Aspects. In *Spatial Cognition III, Routes and Navigation, Human Memory and Learning, Spatial Representation and Spatial Learning*, pages 348–371. Springer, 2003.

7 Fernando Berzal, Juan C. Cubero, Daniel Sánchez, María Amparo Vila Miranda, and José-María Serrano. An Alternative Approach to Discover Gradual Dependencies. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 15(5):559–570, 2007.

8 Abdelhamid Boudane, Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. A SAT-Based Approach for Mining Association Rules. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA*, pages 2472–2478. IJCAI/AAAI Press, 2016.

9 Abdelhamid Boudane, Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. SAT-Based Data Mining. *International Journal on Artificial Intelligence Tools*, 27(1):1–24, 2018.

10 Lisa Di-Jorio, Anne Laurent, and Maguelonne Teisseire. Fast extraction of gradual association rules: a heuristic based method. In *CSTST 2008: Proceedings of the 5th International Conference on Soft Computing as Transdisciplinary Science and Technology, Cergy-Pontoise, France*, pages 205–210. ACM, 2008.

11 Lisa Di-Jorio, Anne Laurent, and Maguelonne Teisseire. Mining Frequent Gradual Itemsets from Large Databases. In *Advances in Intelligent Data Analysis VIII, 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France*, pages 297–308. Springer, 2009.

12 Tias Guns, Anton Dries, Siegfried Nijssen, Guido Tack, and Luc De Raedt. MiningZinc: A declarative framework for constraint-based mining. *Artificial Intelligence*, 244:6–29, 2017.

13 Tias Guns, Siegfried Nijssen, and Luc De Raedt. Itemset mining: A constraint programming perspective. *Artificial Intelligence*, 175(12-13):1951–1983, 2011.

14 Amar Isli and Anthony G. Cohn. A new approach to cyclic ordering of 2D orientations using ternary relation algebras. *Artificial Intelligence*, 122(1-2):137–187, 2000.

15 Saïd Jabbour, Jerry Lonlac, Lakhdar Sais, and Yakoub Salhi. Extending modern SAT solvers for models enumeration. In *Proceedings of the 15th IEEE International Conference on Information Reuse and Integration, IRI 2014, Redwood City, CA, USA*, pages 803–810. IEEE Computer Society, 2014.

16 Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. Boolean satisfiability for sequence mining. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA*, pages 649–658. ACM, 2013.

17 Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. Decomposition Based SAT Encodings for Itemset Mining Problems. In *Advances in Knowledge Discovery and Data Mining - 19th Pacific-Asia Conference, PAKDD 2015, Ho Chi Minh City, Vietnam*, pages 662–674. Springer, 2015.

18 Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. On SAT Models Enumeration in Itemset Mining. *CoRR*, abs/1506.02561, 2015. `arXiv:1506.02561`.

19 Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. Mining Top-k motifs with a SAT-based framework. *Artificial Intelligence*, 244:30–47, 2017.

20 Anne Laurent, Marie-Jeanne Lesot, and Maria Rifqi. GRAANK: Exploiting Rank Correlations for Extracting Gradual Itemsets. In *Flexible Query Answering Systems, 8th International Conference, FQAS 2009, Roskilde, Denmark*, pages 382–393. Springer, 2009.

21 Gérard Ligozat. *Qualitative Spatial and Temporal Reasoning*. ISTE. Wiley, 2013.

22 Jerry Lonlac, Saïd Jabbour, Engelbert Mephu Nguifo, Lakhdar Saïs, and Badran Raddaoui. Extracting Frequent Gradual Patterns Using Constraints Modeling. *CoRR*, abs/1903.08452, 2019. `arXiv:1903.08452`.

23 João P. Marques-Silva and Inês Lynce. Towards Robust CNF Encodings of Cardinality Constraints. In *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA*, pages 483–497, 2007.

24 Luc De Raedt, Tias Guns, and Siegfried Nijssen. Constraint Programming for Itemset Mining. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA*, pages 204–212, 2008.

25 David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions and Connection. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92). Cambridge, MA, USA*, 1992.

26 Carsten Sinz. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain*, pages 827–831, 2005.

27 Spiros Skiadopoulos and Manolis Koubarakis. Composing cardinal direction relations. *Artificial Intelligence*, 152(2):143–171, 2004.

28 Spiros Skiadopoulos and Manolis Koubarakis. On the consistency of cardinal direction constraints. *Artificial Intelligence*, 163(1):91–135, 2005.

29 Gregory S. Tseitin. On the complexity of derivations in the propositional calculus. In *Structures in Constructives Mathematics and Mathematical Logic, Part II*, pages 115–125, 1968.

30 Willy Ugarte, Patrice Boizumault, Bruno Crémilleux, Alban Lepailleur, Samir Loudni, Marc Plantevit, Chedy Raïssi, and Arnaud Soulet. Skypattern mining: From pattern condensed representations to dynamic constraint satisfaction problems. *Artificial Intelligence*, 244:48–69, 2017.

31 Marc B. Vilain and Henry A. Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In *Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, PA, USA. Volume 1: Science*, pages 377–382. Morgan Kaufmann, 1986.

Recurrent Neural Networks Applied to GNSS Time Series for Denoising and Prediction

Elena Loli Piccolomini
Department of Computer Science and Engeneering, University of Bologna, Italy

Stefano Gandolfi
Department of Engeneering, University of Bologna, Italy

Luca Poluzzi
Department of Engeneering, University of Bologna, Italy

Luca Tavasci
Department of Engeneering, University of Bologna, Italy

Pasquale Cascarano
Department of Mathematics, University of Bologna, Italy

Andrea Pascucci
Department of Mathematics, University of Bologna, Italy

Abstract

Global Navigation Satellite Systems (GNSS) are systems that continuously acquire data and provide position time series. Many monitoring applications are based on GNSS data and their efficiency depends on the capability in the time series analysis to characterize the signal content and/or to predict incoming coordinates. In this work we propose a suitable Network Architecture, based on Long Short Term Memory Recurrent Neural Networks, to solve two main tasks in GNSS time series analysis: denoising and prediction. We carry out an analysis on a synthetic time series, then we inspect two real different case studies and evaluate the results. We develop a non-deep network that removes almost the 50% of scattering from real GNSS time series and achieves a coordinate prediction with 1.1 millimeters of Mean Squared Error.

2012 ACM Subject Classification General and reference → General conference proceedings; Mathematics of computing → Time series analysis; Computing methodologies → Supervised learning by regression; Information systems → Global positioning systems

Keywords and phrases Deep Neural Networks, Recurrent Neural Networks, Time Series Denoising, Time Series Prediction

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.10

1 Introduction

Nowadays global navigation satellite systems (GNSS), such as the GPS, are widely used tools for many monitoring applications. Thanks to the capability of these systems to provide continuously acquired data, which are converted in three-dimensional coordinates after data processing, the monitoring is usually performed through time series analysis. GNSS monitoring can have different purposes depending on the monitored object and the data processing varies consequently therefore this leads to very different time series in terms of signal to noise ratio and noise characteristics. Moreover, the analysis can have different goals: sometimes a signal-denoising is required in order to allow more accurate characterization of the signals, whereas in other cases the goal is a reliable prediction of the coordinates that will be obtained from the incoming data. For GNSS time series analysis many forecasting models have been proposed, based on ARMA, ARIMA and Kalman methods [2],[20],[16],[18], [17] and many denoising models, based on moving averages, sequential filtering and, recently,

based on deep learning [15],[12],[5]. Deep Neural Networks (DNNs) have been investigated by many researchers during last years due to the development of computational resources (e.g GPUs) and the raise of Big Data, in order to find mathematical models inspired by the information flowing and processing in human brain. DNNs are widely employed to solve different data science tasks such as pattern recognition, classification, regression, anomaly detection, signal-denoising and density estimation. The class of DNNs algorithms provides a family of networks, suited for sequential data processing, called Recurrent Neural Networks (RNNs). Among them Long Short Term Memory (LSTM) model, introduced at the beginning to solve vanishing gradient problems which affect all the RNNs [1],[9],[4], has showed its strength for almost all of the time dependent problems. The LSTM is characterized by a gated structure which allows it to store past sequence features in its memory block, bringing out them in the output and preserving long term dependences. LSTM approach achieves grateful results in speech recognition [7] and [8], hand-writing recognition [6] and recently in Traffic Flow Prediction [19], Real Time Autonomous Vehicle Navigation [13]. The aim of this work is to develop a deep learning method suitable for prediction and denoising of GNSS time series. Our model add to the LSTM layer used in [12] an activation hyperbolic tangent (*tanh*) layer and a Full Connected layer. In Section 2, first we introduce basic notions about Deep Feed Forward Neural Networks, Recurrent Neural Networks and LSTM Network, then we present the proposed network architecture and its forward equations. In Section 3 we present three different time series used for the experiments and finally in Section 4 we discuss the results.

2 Framework

In this section we carry out a brief review about DNNs and their developments. In the first part we introduce Feed Forward Neural Networks (FFNNs) and we point out their structure. In the second part we remark Recurrent Neural Networks (RNNs), a powerful tool for sequential data processing. Finally we provide our model forward equations.

2.1 Feed Forward Neural Networks

The main goal of a FFNN algorithm is to approximate some function f^* in a suitable functional space. A FFNN describes an approximation map, called net, defined as follows:

$$y = f(x, \theta_1, \dots, \theta_k), \tag{1}$$

where x, y are the input and the output of the net, respectively, and $\theta_1, \dots, \theta_k$ are the parameters that the net has to learn respect to a given training set. The map (1) is the composition of k functions and it can be written as follows:

$$f(x, \theta_1, \dots, \theta_k) = f_k \circ f_{k-1} \circ \cdots \circ f_1, \tag{2}$$

setting $x_1 = x$ and $y_j = f_j(x_j, \theta_j)\ \forall j = 1 \dots k$, where $x_j = y_{j-1}\ \forall j = 2 \dots k$. Each of the functions f_k is called the k-th hidden-layer. The depth of a FFNN is described by the parameter k. The universal approximation theorem [11] states that a FFNN with a linear output layer and at least one hidden layer can approximate as good as depth increases any Borel measurable function from a finite dimensional space to another. For this reason deeper FFNNs showed better results in many tasks; the drawback is the computational cost. Hereafter we provide the forward equation of a 1-hidden layer network:

$$x = input \tag{3}$$
$$a = Wx + b \tag{4}$$
$$y = \Phi(a) \tag{5}$$

where W is the weight matrix, b is the bias vector, Φ is the pointwise activation function. The goal is to find the weights and the bias by minimizing a certain loss function over a given training set (e.g the mean squared error). This problem can be handled by using iterative gradient based method such as stochastic gradient descent (SGD) or ADAM method [14], while the Back Propagation Algorithm [3] is used to compute the gradients.

2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a family of Neural Networks suited for time series processing. In RNNs the output at previous time steps affects the output at the current time step and therefore RNNs are able to catch long term dependencies in sequential data. Given a starting hidden layer vector h_0, then for each time step t and for each input vector x_t the forward equations of a general RNN are:

$$h_t = W_{hh}h_{t-1} + U_{xh}x_t + b_h \tag{6}$$
$$y_t = \Phi(h_t) \tag{7}$$

where W_{hh} denotes the hidden-to-hidden weight matrix, U_{xh} the input-to-hidden weight matrix, b_h the bias vector and Φ is a pointwise non-linear activation function. The main difference between FFNN (3)-(5) and RNN (6)-(7) forward equations, is that the latter have a temporal structure. As for FFNNs, gradient based algorithms are used to optimize a loss function respect to the unknown weights and the Back Propagation Through Time (BPTT) algorithm is employed to compute the gradients. One drawback of RNNs is that BPTT algorithm computes gradients that tend to vanish or explode due to the fact that we are composing many times the same non linear function [1],[9]. The most effective model used in practical applications are gated RNNs such as Long Short-Term Memory (LSTM) nets. A LSTM network [10] is made up of LSTM units showed in Figure 1(b). Generally, a LSTM unit is composed of a cell which is able to record the main information over long time intervals [4] and three different gates: the forget gate, the input gate and the output gate. These gates have the task to supervise the flow of information and prevent vanishing gradient problems. The forget gate uses a sigmoid function to decide which information has to be taken into account in the previous cell state. The input gate decides which new information has to be stored in the cell memory. The output gate decides the contents of the output vector. Each gate takes the same input: x_t the current input and h_{t-1} the previous hidden layer vector. The LSTM unit forward equations are presented below:

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f) \tag{8}$$
$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i) \tag{9}$$
$$g_t = tanh(U_g x_t + W_g h_{t-1} + b_g) \tag{10}$$
$$c_t = i_t * g_t + f_t * c_{t-1} \tag{11}$$
$$o_t = \sigma(U_o x_t + W_o h_{t-1} + b_o) \tag{12}$$
$$h_t = tanh(c_t) * o_t \tag{13}$$

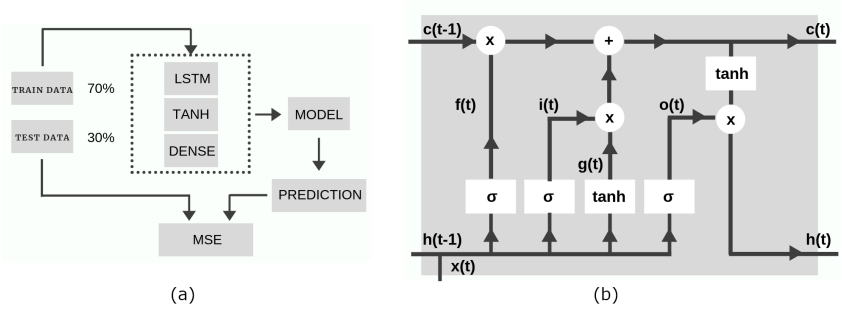


Figure 1 (a) Complete process used in this study. (b) LSTM unit.

where $*$ represents the pointwise product, f_t, i_t, o_t are respectively the forget gate, the input gate and the output gate vectors at time t, c_t is the cell vector at time t and the terms U and W in each equation are the weight matrices.

2.3 Proposed Network Architecture

Given an input sequence x_t, for $t = 1 \dots \mathrm{T}$, and an initial hidden vector h^0, the model proposed in this paper is a RNN model with the following forward equations:

$$h_t^1 = \Phi(x_t, h_{t-1}) \tag{14}$$
$$h_t^2 = tanh(h_t^2) \tag{15}$$
$$y_t = w_{h^2} * h_t^1 + b_{h^2} \tag{16}$$

where $*$ represents the scalar product, Φ is taken as the LSTM unit with the forward equations (8)-(13). We add after the classic LSTM unit: an activation *tanh* layer and then a full connected layer that realizes a dot product between the weights w_{h^2} and the output of the previous layers. We will call our model LSTM-Full in the following. A crucial point of this architecture is the choice of the h_t^1 vector dimension which influences the number of the weights to train. We remark that for each time t, the hidden layer vector h_t^1 has the same dimension. In the following we indicate with H the length of these hidden layer vectors.

3 Materials

Global Navigation Satellite Systems (GNSS) is a technology that uses data sent by artificial satellites to get the position of the receiver. Using the same data but changing the receiver quality and the data processing it is possible to obtain a wide range of accuracy that ranges from 10/15 meters in real-time positioning of a smartphone to a few millimeters (mm) using a geodetic class receiver that acquires data for 24-hour to estimate the daily average position. A GNSS receiver acquires data continuously and after a data processing phase provides a time series of positions useful for geodynamics or geological applications such as tectonic plate motion research, landslides and subsidence studies. We consider three sets of data in order to evaluate the efficiency of the designed LSTM-Full network in GNSS time series analysis. The first time series is a synthetic one, which allows to compare the output of the network to a known "ground truth" and has been created taking into account the well known characteristics of a geodetic time series of daily positions. The second time series derives from a real GNSS permanent station that daily processes the data using a static approach and it is characterized by a comparatively high signal to noise ratio. The third time series comes from a monitoring GNSS station used for early warning monitoring and it is characterized by a low signal compared to the measurements.

3.1 Synthetic Time Series

To create a Synthetic Time Series that simulates properly a real GNSS one, it is important to understand which are the main characteristics of the GNSS positioning. Using GNSS signals a receiver can estimate its position, with different levels of accuracy, measuring the transmitting time of GNSS signals emitted from four or more GNSS satellites and knowing the position of each satellite during the time. The signal crosses the atmosphere with a speed close to the speed of light depending on the physical characteristics of the atmosphere that change continuously. Therefore a GNSS receiver installed on a building roof or on a stable rock can measure during the time some periodical effects due to the seasonal thermal dilatation or solid earth and ocean tides. For all these reasons a GNSS time series can be composed by some periodical terms and, as literature shows, a noisy component which is the sum of a white noise that follows a normal distribution and a random walk noise. The synthetic time series is constructed as follows:

$$x(t) = mt + q + Asin(2\pi ft) \tag{17}$$

and it is characterized by the following parameters: $m = 0, q = 0.01, A = 0.01, f = \frac{1}{365}$ and considering $t \in [1, 4000]$. We assume that x, q, A are expressed in meters (m) whereas t is expressed in days (d) and f is expressed in d^{-1}. In order to represent a realistic case study we add: w gaussian noise, with mean equal to zero and standard deviation $p = 2$ mm and r random walk noise or red noise generated using a normal distribution with mean zero and standard deviation equal to 0.03 mm. Hence we construct the raw time-series as follows:

$$\tilde{x}(t) = x(t) + w + r. \tag{18}$$

It is showed in Figure 2(a) using the green dots. Since in our model we use the activation *tanh* layer it is suggested to scale all the time series values in $[-1, 1]$. This kind of action increases the net performances.

3.2 Real Static Time Series

The first real time series, hereafter called "static" due to the GNSS data processing used, is 11 years long and each time step represents a daily solution. It has three different components: North, East and Up components. The North component is depicted in Figure 3 (a) using green dots. It is for sure the most accurate time series obtainable by GNSS technology because each solution is the mean of the observations recorded with a sampling time of 30 seconds during 24 hours. Since these kind of time series are generally used for tectonic plate motion or landslide monitoring it is important to have accurate and no noise solutions. The analysis that we conduct refers to those applications where the goal is the characterization of the periodical signals and in this case a signal denoising can improve the results.

3.3 Real Kinematic Time Series

The second real GNSS time series, hereafter called "kinematic", derives from higher frequency (1Hz) coordinates solutions, that were estimated using a kinematic approach during the data processing. The time span considered for the test is about 30 days and every sample represents a solution per second. As for the static time series, our data set is made up of North, East and Up components. In Figure 4 (a) we depict the East Component of this time series using the green dots. This is a completely different scenario respect to the previous one

because we have a solution every second and for this reason, the redundancy of the system and the accuracy are lower. This approach can be suitable for early warning applications where it is much more important to have as soon as possible information about unexpected changes respect to the normal movement. Since this kinematic time series is used for structure monitoring such as bridge monitoring or critical landslides where a fast movement of the object can be critical for the safety of life, here the ultimate goal of our analysis is to develop a method capable to give an accurate prediction of the incoming coordinates to improve the capability to detect some anomalies.

4 Results and Discussion

In this section we carry out an objective analysis in order to investigate the properties of the proposed LSTM-Full method and demonstrate its effectiveness. All the experiments are performed on a PC Intel(R) Core(TM) i5-6200U CPU @ 2.30 GHz 2.40GHz with 8.00Gb RAM using Python 3.6 libraries, such as Keras and Pandas. In the first part we analyze the synthetic static GNSS time series depicted as in Section 3.1, then we inspect the two real GNSS time series, described in Section 3.2 and 3.3 respectively.

4.1 Data preparation

Since the LSTM-Full method is a supervised learning algorithm we need to define the form of the input and the form of the label vectors. We can represent the whole raw time series as a sequence of values $\tilde{x}_1, \ldots, \tilde{x}_{\mathrm{T}}$, where T is the number of analysed samples. Then we construct the input as vectors defined as follows:

$$X_i = [\tilde{x}_i, \ldots, \tilde{x}_{i+s}], \quad i \in [1, \mathrm{T} - s], \tag{19}$$

where s is the sliding window parameter representing the length of each vector. We underline that the data set has a time order respect to the variable i and the net has to be fed following this order. We can say that in time series analysis the time is an hidden feature decoded by the data set time order. The label vector is built as a vector y and its components are defined as shown below:

$$y_i = [\tilde{x}_{i+s+1}], \quad i \in [1, \mathrm{T} - s]. \tag{20}$$

In the following analysis the data set is divided in two disjoint subsets: the training set (taken as 70% of the whole data set) and the test set (30% of the whole data set). The first is used to optimize the model parameters, whereas the other is employed to evaluate model accuracy. The Mean Squared Error (MSE) value is used to evaluate the prediction accuracy of the proposed model whereas the standard deviation (STD) is used to evaluate the scatter of the solution. The general mathematical expressions of MSE and STD are:

$$\mathrm{MSE}(u, z) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(u_i - z_i)^2}, \tag{21}$$

$$\mathrm{STD}(v) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(v_i - \tilde{v})^2}, \tag{22}$$

where N is the length of the vectors u,z, M and $\tilde{v}$ are the length and the mean of the vector v, respectively.

Table 1 All the results are computed setting $H = 30$. Columns 1 and 2 show respectively the MSE(x,$\tilde{y}$) and STD(x,$\tilde{y}$) values, where x is the theoretical signal and $\tilde{y}$ is the predicted signal, for different length of the input, both in millimeters. The raw time series has a MSE equal to 2.327 millimeters and a STD equal to 1.329 millimeters. Column 3 shows the computational training time, in seconds, choosing different input size.

s	MSE(x,$\tilde{y}$) (mm)	STD(x,$\tilde{y}$) (mm)	time (s)
7	1.276	0.718	7.862
14	1.212	0.687	9.953
30	1.245	0.693	15.115
183	1.198	0.690	59.482
365	1.206	0.678	105.21

4.2 Synthetic time series analysis

The purposes of this analysis are to evaluate:

- if the model proposed can predict the theoretical time series $x(t)$ defined in (17) using only the raw signal $\tilde{x}(t)$ defined in (18),
- if the model proposed can filter the raw data,
- the sensitivity of the proposed method to the choice of the parameters H (defined in Section 2.3) and s (defined in Section 4.1).

In the following we indicate with $x, \tilde{x}$ and $\tilde{y}$ the theoretical time series, the raw time series and the solution constructed with the LSTM-Full algorithm, respectively. All of them represent a 730 days solution. In the first column of Table 1 we report the results of MSE$(x, \tilde{y})$, over different choices of the sliding window parameter s. The choice of the sliding window parameter is a crucial point and in practice it should be a trade off between the training time and the prediction performance. Here we suggest, when possible, to choose s as the lag with the highest Autocorrelation Function (ACF) value of raw data (see Figure 2(c)); this seems necessary if we want to compute a less scattered solution. In Figure 2(b) (red-dot line) we depict the solution for $s = 365$, since that the time series considered has an annual seasonality. In the second column of Table 1 we report the STD and we observe that the scattering is reduced while the sliding window dimension increases. The best improvements in terms of MSE (48% less than raw data) is reached choosing $s = 183$ whereas the best improvements in denoising is reached choosing $s = 365$ (49% less than raw data). As we expect the computational training time increases when the input size, that is s, increases. In Figure 2(d) we report the ACF plot of the differences between the theoretical time steps and the predicted time steps. The plot highlights that there is no relevant correlation within prediction errors, as a good prediction is expected to be. All the previous results are computed setting the hidden vectors length equal to 30. In Table 2 we report the summary of the analysis carried out choosing different values for the parameter H and setting $s = 365$. As it emerges from the first and the second column of Table 2 the best results in terms of MSE and of STD are obtained setting $H = 5$ (50 % less for MSE and 51 % less for STD respect to raw data). In the third column we report the computational training time that again increases while the parameter H increases.

4.3 Static time series analysis

So far we have proved the effectiveness of the proposed LSTM-Full method on a test problem, we now want to evaluate our model on the real time series described in Section 3.2. Since the ultimate goal for this time series is to remove the noisy components, we set the hidden

Table 2 All the results are computed setting $s = 365$. Columns 1 and 2 show respectively the MSE(x,$\tilde{y}$) and STD(x,$\tilde{y}$) values, where x is the theoretical signal and $\tilde{y}$ is the predicted signal, for different length of H. The raw time series has a MSE equal to 2.327 millimeters and a STD equal to 1.329 millimeters. Column 3 shows the computational training time, in seconds, choosing different values of H.

H	MSE(x,$\tilde{y}$) (mm)	STD(x,$\tilde{y}$) (mm)	time (s)
5	1.169	0.646	109.174
30	1.205	0.678	115.049
100	1.309	0.727	224.441
300	1.262	0.708	2801.995

Table 3 Column 1 shows the STD values for the North Component, in millimeters, for different size of the sliding window parameter. The raw time series have a STD value equal to 0.975 millimeters. Column 2 shows the computational training time in seconds choosing different input size.

s	STD($p_s - \tilde{y_s}$) (mm)	time (s)
7	0.563	8.026
14	0.481	10.153
30	0.432	15.348
183	0.432	66.314
365	0.423	183.304

vectors length parameter H equal to 5. In the following we indicate with $\tilde{x}$ and $\tilde{y_s}$ the raw time steps and the filtered solution respect to the sliding window parameter s, with p_0 and p_s the 5th-degree regression polynomials respect to $\tilde{x}$ and $\tilde{y_s}$. In order to evaluate the scattering of raw data and of $\tilde{y_s}$, we consider the value STD($p_s - \tilde{y_s}$). In the first column of Table 3 we compute the STD values of the filtered time series over different choices of the sliding window parameter s. The best result is achieved using s equal to 365, which is the lag with the highest autocorrelation function value, shown in Figure 3(b). In Figure 3(c) we plot the filtered time series values obtained using $s = 365$ compared with the raw North Component of the static time series. In the second column of Table 3 we report the computational training time, which increases while s increases, as it is for the synthetic case. In Table 4 we report the STD values for the North Component, East Component and Up Component of our real static time series, using a sliding window parameter equal to 365.

4.4 Kinematic time series analysis

In the case of the kinematic time series described in Section 3.3, the aim of the LSTM-Full model is to give an accurate prediction for the incoming time steps, since we want to apply this model for structural monitoring. Since we are not seeking for a filtered solution, we fix the dimension of the LSTM hidden vectors at 100. In order to evaluate the efficiency of the proposed method in this case we use the MSE value between the raw data $\tilde{x}$ and the predicted time steps $\tilde{y}$. The lag with the highest value of the ACF is 86164 (the number of seconds in a sideral day), shown Figure 4 (b), and it is impossible to use it as sliding window parameter, due to the computational cost of the algorithm. In the first column of Table 5 we report the MSE values over different arbitrary size of the sliding window parameter increased until the machine used for this experiment can afford the computational cost. The best results are reached using $s = 60$ or $s = 300$. In Figure 4 (a) and in Figure 4(c) we can see that the predicted time series (red dots) model seems to well reproduce the data (green

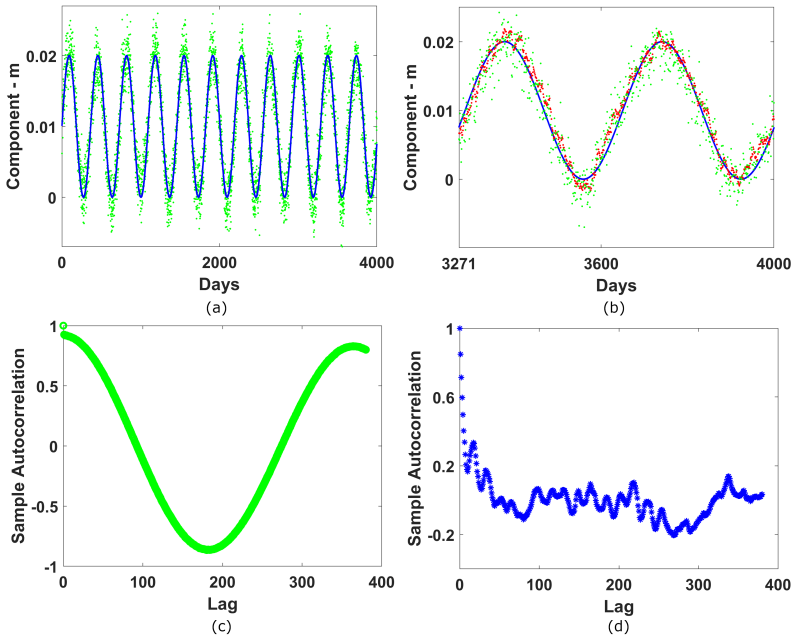


Figure 2 (a) Plot over 4000 days of the raw synthetic time series (green dots) and the theoretical sinusoidal signal (line blue). (b) Predicted time steps over 730 days (red dots) compared with the raw signal (green dots) and the theoretical sinuisoidal signal (blue line).(c) ACF over 400 lag of the raw time series. (d) ACF of the prediction error over 365 lags.

Table 4 In the first column: STD values in millimeters for the North, East and Up component of the real static time series, using a sliding window parameter equal to 365. In the second column: the STD values of the raw signals for each component.

Component	STD($p_s - \tilde{y_s}$) (mm)	STD($p_0 - \tilde{x}$) (mm)
N	0.423	0.975
E	0.559	1.031
U	1.221	2.808

dots). The computational training time, reported in the second column of Table 5, increases while the slinding window parameter increases as for the other case studies. In Table 6 we write down the MSE values for the North, East and Up Component of our time series setting the slinding window parameter equal to 300.

5 Conclusions

The proposed LSTM-Full Network architecture is suitable for denoising and prediction tasks for GNSS time series. Its performance depends on the choice of the sliding window parameter and of the hidden vectors size of the LSTM layer. From the experiments we suggest to choose the sliding window parameter as the lag with the highest value of the autocorrelation function, when the computational resources allow it. The hidden vectors size should be choosen small if the task is denoising otherwise it should be higher to give an accurate prediction on incoming data.

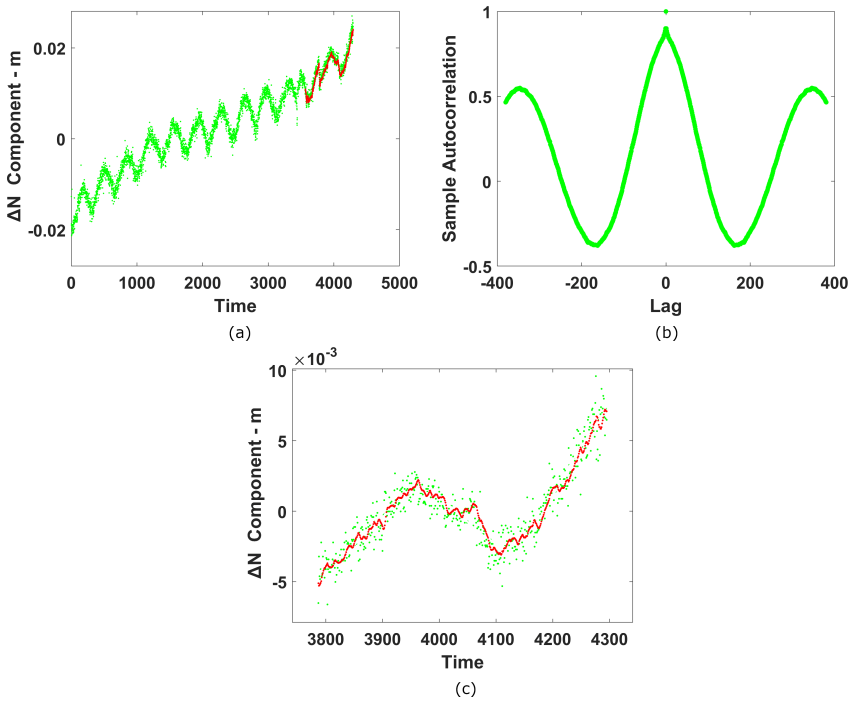


Figure 3 (a) Plot of the raw North Component of the Static Time Series (green dots) and the filtered time series (red dots). (b) ACF over 400 lag of the raw time series. (c) Plot of 509 filtered time steps (red dots), computed choosing s=365, compared with the raw time steps (green dots).

Table 5 Column 1 shows the MSE($\tilde{x}, \tilde{y}$) values for the East Component, in millimeters, for different size of the sliding window parameter. Column 2 shows the computational training time in seconds choosing different input size.

s	MSE($\tilde{x}, \tilde{y}$) (mm)	time (s)
60	1.188	76.28
300	1.184	313.53
1800	1.208	1811.2350
3600	1.220	2872.0116

Table 6 MSE($\tilde{x}, \tilde{y}$) values in millimeters for the North, East and Up component of the real kinematic time series, using a sliding window parameter equal to 300.

Component	MSE($\tilde{x}, \tilde{y}$) (mm)
N	1.741
E	1.188
U	2.497

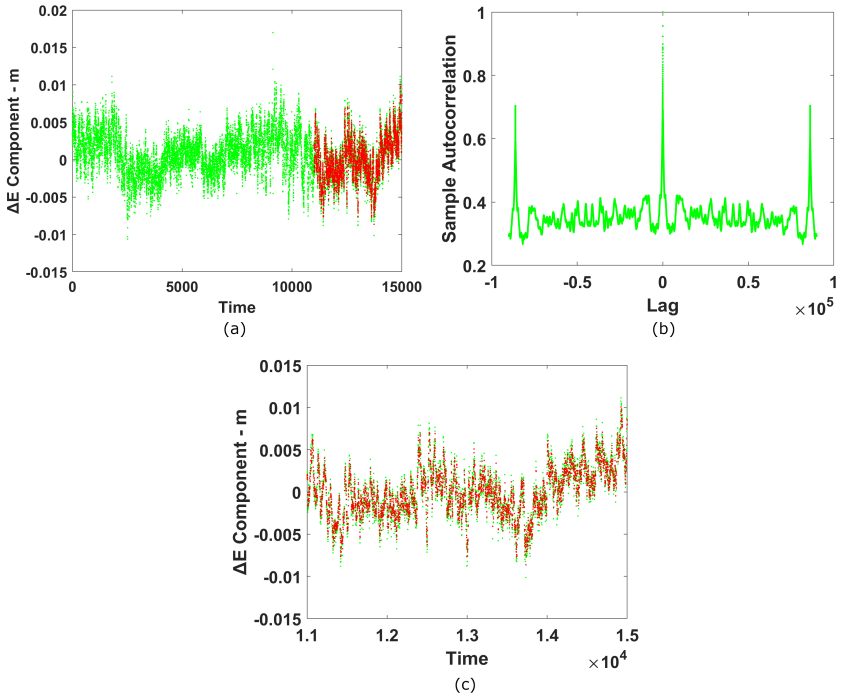


Figure 4 (a) Plot of the raw East Component of the Static Time Series (green dots) and the predicted time series (red dots). (b) ACF over 90000 lag of the raw time series. (c) Plot of the predicted time series (red dots), computed choosing s=300, compared with the raw time steps (green dots).

References

1 Y Bengio, Patrice Simard, and Paolo Frasconi. Learning Long-Term dependencies with Gradient Descent is Difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5:157–66, February 1994. doi:10.1109/72.279181.

2 Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer New York, 1996.

3 David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back Propagating Errors. *Nature*, 323:533–536, October 1986. doi:10.1038/323533a0.

4 Claudio Gallicchio. Short-Term Memory of Deep RNN. In *Proceedings for European Symposium on Artificial Neural Networks*, February 2018.

5 Stefano Gandolfi, Luca Poluzzi, and Luca Tavasci. Structural Monitoring Using GNSS Technology and Sequential Filtering. In *Proceedings for FIG Working Week*, May 2015.

6 Alex Graves. Generating Sequences With Recurrent Neural Networks. *ArXiv: 1308.0850*, August 2013. arXiv:1308.0850.

7 Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2013 - Proceedings*, pages 273–278, December 2013. doi:10.1109/ASRU.2013.6707742.

8 Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech Recognition with Deep Recurrent Neural Networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 38, March 2013. doi:10.1109/ICASSP.2013.6638947.

9 Sepp Hochreiter. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116, April 1998. doi:10.1142/S0218488598000094.

10 Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. *Neural computation*, 9:1735–80, December 1997. doi:10.1162/neco.1997.9.8.1735.

11 K.M. Hornik, M. Stinchcomb, and H. White. Multilayer feedforward networks are universal approximator. *IEEE Transactions on Neural Networks*, 2, January 1989.

12 Changhui Jiang, Shuai Chen, Yuwei Chen, Boya Zhang, Ziyi Feng, Hui Zhou, and Yuming Bo. A MEMS IMU De-Noising Method Using Long Short Term Memory Recurrent Neural Networks (LSTM − RNN). *Sensors*, 18:3470, October 2018. doi:10.3390/s18103470.

13 Hee-Un Kim and Tae-Suk Bae. Deep Learning-Based GNSS Network-Based Real-Time Kinematic Improvement for Autonomous Ground Vehicle Navigation. *Journal of Sensors*, 2019:1–8, March 2019. doi:10.1155/2019/3737265.

14 Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, December 2014.

15 José Lima and J Casaca. Smoothing GNSS Time Series with Asymmetric Simple Moving Averages. *Journal of Civil Engineering and Architecture*, 6, June 2012. doi:10.17265/1934-7359/2012.06.013.

16 X. Luo, M. Mayer, and B. Heck. Analysing Time Series of GNSS Residuals by Means of AR(I)MA Processes. In *VII Hotine-Marussi Symposium on Mathematical Geodesy*, pages 129–134. Springer Berlin Heidelberg, 2012.

17 Xiaoguang Luo. *GPS Stochastic Modelling - Signal Quality Measures and ARMA Processes*. Springer, January 2013. doi:10.1007/978-3-642-34836-5.

18 Ping-Feng Pai and Chih-Sheng Lin. A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega*, 33(6):497–505, 2005. URL: https://EconPapers.repec.org/RePEc:eee:jomega:v:33:y:2005:i:6:p:497-505.

19 Yuelei Xiao and Yang Yin. Hybrid LSTM Neural Network for Short-Term Traffic Flow Prediction. *Information*, 10:105, March 2019. doi:10.3390/info10030105.

20 Jingzhou Xin, Jianting Zhou, Simon Yang, Xiaoqing Li, and Yu Wang. Bridge structure deformation prediction based on GNSS data using Kalman − ARIMA − GARCH model. *Sensors (Basel, Switzerland)*, 18, January 2018. doi:10.3390/s18010298.

From Quantified CTL to QBF

Akash Hossain
IRIF, Univ. Paris Diderot, France
akashoss42@gmail.com

François Laroussinie
IRIF, Univ. Paris Diderot, France
https://www.irif.fr/~francoisl/
francoisl@irif.fr

Abstract

QCTL extends the temporal logic CTL with quantifications over atomic propositions. This extension is known to be very expressive: QCTL allows us to express complex properties over Kripke structures (it is as expressive as MSO). Several semantics exist for the quantifications: here, we work with the *structure semantics*, where the extra propositions label the Kripke structure (and not its execution tree), and the model-checking problem is known to be PSPACE-complete in this framework. We propose a model-checking algorithm for QCTL based on a reduction to QBF. We consider several reduction strategies, and we compare them with a prototype (based on the SMT-solver Z3) on several examples.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Modal and temporal logics; Theory of computation → Verification by model checking

Keywords and phrases Model-checking, Quantified CTL, QBF solvers, SAT based model-checking

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.11

Funding Project FREDDA ANR-17-CE40-0013

1 Introduction

Temporal logics have been introduced in computer science in the late 1970's [14]; they provide a powerful formalism for specifying correctness properties of evolving systems. Various kinds of temporal logics have been defined, with different expressiveness and algorithmic properties. For instance, the *Computation Tree Logic* (CTL) expresses properties of the computation tree of the system under study (time is branching: a state may have several successors), and the *Linear-time Temporal Logic* (LTL) expresses properties of one execution at a time (a system is viewed as a set of executions).

Temporal logics allow *model checking*, i.e. the automatic verification that a finite state system satisfies its expected behavioral specifications [15, 3]. It is well known that CTL model-checking is PTIME-complete and LTL model-checking (based on automata techniques) is PSPACE-complete. Verification tools exist for both logics and model-checking is now commonly used in the design of critical reactive systems. The main limitation to this approach is the state-explosion problem: symbolic techniques (for example with BDD), SAT-based approaches, or partial order reductions have been developed and they are impressively successful. The SAT-based model-checking consists in using SAT-solvers in the decision procedures. It was first developed for bounded model-checking (to search for executions whose length is bounded by some integer, satisfying some temporal property) which can be reduced to some satisfiability problem and then can be solved by a SAT-solver [2]. SAT approaches have also been extended to unbounded verification and combined with other techniques [12]. Many studies have been done in this area, and it is widely considered as an important approach in practice, which complements other symbolic techniques like BDD (see [1] for a survey).

In terms of expressiveness, CTL (or LTL) still has some limitations: in particular, it lacks the ability of *counting*. For instance, it cannot express that an event occurs (at least) at every even position along a path, or that a state has two successors. In order to cope with this, temporal logics have been extended with *propositional quantifiers* [16]: those quantifiers allow for adding fresh atomic propositions in the model before evaluating the truth value of a temporal-logic formula. That a state has at least two successors can then be expressed (in *quantified* CTL, hereafter written QCTL) by saying that it is possible to label the model with atomic proposition p in such a way that there is a successor that is labelled with p and one that is not.

Different semantics for QCTL have been studied in the literature depending on the definition of the labelling: either it refers to the finite-state model – it is the *structure* semantics – or it refers to the execution tree – it is the *tree semantics*. Both semantics are interesting and have been extensively studied [9, 7, 13, 8, 4, 10]. While the tree semantics allow us to use the tree automata techniques to get decision procedures (model-checking and satisfiability are TOWER-complete [10]), the situation is quite different for the structure semantics: in this framework, model-checking is PSPACE-complete and satisfiability is undecidable [7].

In this paper, we focus on the structure semantics, and we propose a model-checking algorithm based on a reduction to QBF (propositional logic augmented with quantifiers): given a Kripke structure $\mathcal{K}$ and a QCTL formula Φ, we show how to build a QBF formula $\widehat{\Phi}^{\mathcal{K}}$ which is valid iff $\mathcal{K} \models \Phi$. It is natural to use QBF quantifiers to deal with propositional quantifiers of QCTL. Of course, QBF-solvers are not as efficient as SAT-solvers, but still much progress has been made (and QBF-solvers have already been considered for model-checking, as in [6]). We propose several reductions depending on the way of dealing with nested temporal modalities, and we have implemented a prototype (based on Z3 SMT-solver [5]) to compare these reductions over several examples. As far as we know, it is the first implementation of a model-checker for QCTL.

Here, our first objective is to use the QBF-solver as a tool to check complex properties over limited size models, and this is therefore different from the classical use of SAT-based techniques which are precisely applied to solve verification problems for very large systems.

The outline of the paper is as follows: we begin with setting up the necessary formalism in order to define QCTL. We then devote Section 3 to the different reductions to QBF. Finally, Section 4 contains several practical results and examples.

2 Definitions

2.1 Kripke structures

Let AP be a set of atomic propositions.

▶ **Definition 1.** *A* Kripke structure *is a tuple $\mathcal{K} = \langle V, E, \ell \rangle$, where V is a finite set of vertices (or states), $E \subseteq V \times V$ is a set of edges (we assume that for any $x \in V$, there exists $x' \in V$ s.t. $(x, x') \in E$), and $\ell \colon V \to 2^{\mathsf{AP}}$ is a labelling function.*

An infinite path (also called an execution) in a Kripke structure is an infinite sequence $\rho = x_0 x_1 x_2 \dots$ such that for any i we have $x_i \in V$ and $(x_i, x_{i+1}) \in E$. We write $\mathsf{Path}^{\omega}_{\mathcal{K}}$ for the set of infinite paths of $\mathcal{K}$ and $\mathsf{Path}^{\omega}_{\mathcal{K}}(x)$ for the set of infinite paths issued from $x \in V$. Given such a path ρ, we use $\rho_{\leq i}$ to denote the i-th prefix $x_0 \dots x_i$, $\rho_{\geq i}$ for the i-th suffix $x_i x_{i+1} \dots$, and $\rho(i)$ for the vertex x_i. The size of $\mathcal{K}$ is $|V| + |E|$.

Given a set $P \subseteq \mathsf{AP}$, two Kripke structures $\mathcal{K} = (V, E, \ell)$ and $\mathcal{K}' = (V', E', \ell')$ are said P-equivalent (denoted by $\mathcal{K} \equiv_P \mathcal{K}'$) if $V = V'$, $E = E'$, and for every $x \in V$ we have: $\ell(x) \cap P = \ell'(x) \cap P$.

2.2 QCTL

This section is devoted to the definition of the logic QCTL, which extends the classical branching-time temporal logic CTL with quantifications over atomic propositions.

▶ **Definition 2.** *The syntax of* QCTL *is defined by the following grammar:*

$$\mathsf{QCTL} \ni \varphi, \psi ::= q \mid \neg\varphi \mid \varphi \vee \psi \mid \mathbf{EX}\varphi \mid \mathbf{E}\varphi\mathbf{U}\psi \mid \mathbf{A}\varphi\mathbf{U}\psi \mid \exists p.\ \varphi$$

where q and p range over AP*.*

QCTL formulas are evaluated over states of Kripke structures:

▶ **Definition 3.** *Let $\mathcal{K} = \langle V, E, \ell\rangle$ be a Kripke structure, and $x \in V$. The semantics of* QCTL *formulas is defined inductively as follows:*

$\mathcal{K}, x \models p$ *iff* $p \in \ell(x)$

$\mathcal{K}, x \models \neg\varphi$ *iff* $\mathcal{K}, x \not\models \varphi$

$\mathcal{K}, x \models \varphi \vee \psi$ *iff* $\mathcal{K}, x \models \varphi$ *or* $\mathcal{K}, x \models \psi$

$\mathcal{K}, x \models \mathbf{EX}\varphi$ *iff* $\exists (x, x') \in E$ *s.t.* $\mathcal{K}, x' \models \varphi$

$\mathcal{K}, x \models \mathbf{E}\varphi\mathbf{U}\psi$ *iff* $\exists \rho \in \mathsf{Path}^{\omega}_{\mathcal{K}}(x), \exists i \geq 0$ *s.t.* $\mathcal{K}, \rho(i) \models \psi$ *and* *for any* $0 \leq j < i$, *we have* $\mathcal{K}, \rho(j) \models \varphi$

$\mathcal{K}, x \models \mathbf{A}\varphi\mathbf{U}\psi$ *iff* $\forall \rho \in \mathsf{Path}^{\omega}_{\mathcal{K}}(x), \exists i \geq 0$ *s.t.* $\mathcal{K}, \rho(i) \models \psi$ *and* *for any* $0 \leq j < i$, *we have* $\mathcal{K}, \rho(j) \models \varphi$

$\mathcal{K}, x \models \exists p.\ \varphi$ *iff* $\exists \mathcal{K}' \equiv_{\mathsf{AP}\setminus\{p\}} \mathcal{K}$ *s.t.* $\mathcal{K}', x \models \varphi$

In the sequel, we use standard abbreviations such as $\top$, $\bot$, $\wedge$, $\Rightarrow$ and $\Leftrightarrow$. We also use the additional (classical) temporal modalities of CTL: $\mathbf{AX}\varphi = \neg\mathbf{EX}\neg\varphi$, $\mathbf{EF}\varphi = \mathbf{E}\top\mathbf{U}\varphi$, $\mathbf{AF}\varphi = \mathbf{A}\top\mathbf{U}\varphi$, $\mathbf{EG}\varphi = \neg\mathbf{AF}\neg\varphi$, $\mathbf{AG}\varphi = \neg\mathbf{EF}\neg\varphi$, $\mathbf{E}\varphi\mathbf{W}\psi = \neg\mathbf{A}\neg\psi\mathbf{U}(\neg\psi \wedge \neg\varphi)$ and $\mathbf{A}\varphi\mathbf{W}\psi = \neg\mathbf{E}\neg\psi\mathbf{U}(\neg\psi \wedge \neg\varphi)$.

Moreover, we use the following abbreviations related to quantifiers over atomic propositions: $\forall p.\ \varphi = \neg\exists p.\ \neg\varphi$, and for a set $P = \{p_1, \ldots, p_k\} \subseteq \mathsf{AP}$, we write $\exists P.\varphi$ for $\exists p_1 \ldots \exists p_k.\varphi$ and $\forall P.\varphi$ for $\forall p_1 \ldots \forall p_k.\varphi$.

The size of a formula $\varphi \in \mathsf{QCTL}$, denoted $|\varphi|$, is defined inductively by: $|q| = 1$, $|\neg\varphi| = |\exists p.\varphi| = |\mathbf{EX}\varphi| = 1 + |\varphi|$, $|\varphi \vee \psi| = |\mathbf{E}\varphi\mathbf{U}\psi| = |\mathbf{A}\varphi\mathbf{U}\psi| = 1 + |\varphi| + |\psi|$. Moreover we use $\mathsf{ht}(\varphi)$ to denote the *temporal height* of φ, that is the maximal number of nested temporal modalities in φ. And given a subformula ψ in Φ, the *temporal depth* of ψ in Φ (denoted $\mathsf{td}_{\Phi}(\psi)$) is the number of temporal modalities having ψ in their scope.

In the following, we denote by $\mathsf{SubF}(\Phi)$ (resp. $\mathsf{SubTF}(\Phi)$) the set of subformulas of Φ (resp. the set of subformulas starting with a temporal modality).

Two QCTL formulas φ and ψ are said to be *equivalent* (written $\varphi \equiv \psi$) iff for any structure $\mathcal{K}$, any state x, we have $\mathcal{K}, x \models \varphi$ iff $\mathcal{K}, x \models \psi$. This equivalence is substitutive.

2.2.1 Discussion on the semantics

The semantics we defined is classically called the *structure semantics*: a formula $\exists p.\varphi$ holds true in a Kripke structure $\mathcal{K}$ iff there exists a p-labelling of the structure $\mathcal{K}$ such that φ is satisfied. Another well-known semantics coexists in the literature for propositional quantifiers,

the *tree semantics*: $\exists p.\ \varphi$ holds true when there exists a p-labelling of the *execution tree* (the infinite unfolding) of the Kripke structure under which φ holds. If, for CTL, interpreting formulas over the structure or the execution tree is equivalent, this is not the case for QCTL. Moreover, these two semantics do not have the same algorithmic properties: if QCTL model-checking and satisfiability are TOWER-complete for the tree semantics (the algorithms are based on tree automata techniques), QCTL model-checking is PSPACE-complete for the structure semantics but satisfiability is undecidable. (see [10] for a survey). Nevertheless, in both semantics, QCTL and QCTL* (the extension of CTL* with quantifications) are equally expressive, and are as expressive [1] as the Monadic Second-Order Logic over the finite structure or the infinite trees (depending on the semantics). Note also that any QCTL formula is equivalent to a formula in Prenex normal form (we will use this result in next sections).

Finally, there is also the *amorphous semantics* [7], where $\exists p.\ \varphi$ holds true at a state s in some Kripke structure $\mathcal{K}$ if, and only if, there exists some Kripke structure $\mathcal{K}'$ with a state s' such that s and s' are bisimilar, and for which there exists a p-labelling making φ hold true at s'. With these semantics, the logic is insensitive to unwinding, and more generally it is bisimulation-invariant (contrary to the two previous semantics, see below).

2.3 Examples of QCTL formulas

QCTL allows us to express complex properties over Kripke structures: for example, we can build a characteristic formula (up to isomorphism) of a structure, we can reduce model-checking problems for multi-player games to QCTL model-checking [11]... Below, we give several examples of counting properties, to illustrate the expressive power of propositional quantifiers.

The first one of the formulas below expresses that there exists a unique reachable state satisfying φ, and the second one states that there exists a unique immediate successor satisfying φ:

$$\mathbf{E}_{=1}\mathbf{F}\varphi \;=\; \mathbf{EF}\varphi \wedge \forall p.\big(\mathbf{EF}(p \wedge \varphi) \Rightarrow \mathbf{AG}(\varphi \Rightarrow p)\big) \tag{1}$$

$$\mathbf{E}_{=1}\mathbf{X}\varphi \;=\; \mathbf{EX}\varphi \wedge \forall p.\ \big(\mathbf{AX}(\varphi \Rightarrow p) \vee \mathbf{AX}(\varphi \Rightarrow \neg p)\big) \tag{2}$$

where we assume that p does not appear in φ. Consider the formula 1: if there were two reachable states satisfying φ, then labelling only one of them with p would falsify the **AG** subformula. For 2, the argument is similar.

The existence of at least k successors satisfying a given property can be expressed with:

$$\mathbf{E}_{\geq k}\mathbf{X}\varphi = \exists P.\ \Big(\bigwedge_{1\leq i\leq k} \mathbf{EX}\big(p_i \wedge \bigwedge_{i'\neq i} \neg p_{i'}\big) \wedge \mathbf{AX}\Big(\big(\bigvee_{1\leq i\leq k} p_i\big) \Rightarrow \varphi\Big)\Big) \tag{3}$$

And we can define $\mathbf{E}_{=k}\mathbf{X}\varphi$ as $\mathbf{E}_{\geq k}\mathbf{X}\varphi \wedge \neg\mathbf{E}_{\geq k+1}\mathbf{X}\varphi$. Note that these examples show why QCTL formulas are not bisimulation-invariant.

When using QCTL to specify properties, one often needs to quantify (existentially or universally) over *one* reachable state we want to mark with a given atomic proposition. To this aim, we add the following abbreviations:

$$\exists^1 p.\varphi = \exists p.\big((\mathbf{E}_{=1}\mathbf{F}\, p) \wedge \varphi\big) \qquad \forall^1 p.\varphi = \forall p.\big((\mathbf{E}_{=1}\mathbf{F}\, p) \Rightarrow \varphi\big)$$

[1] This requires adequate definitions, since a temporal logic formula may only deal with the reachable part of the model, while MSO has a more *global* point of view.

3 Model-checking QCTL

Model-checking QCTL is a PSPACE-complete problem (for detailed results about program complexity and formula complexity, see [10]), and it is NP-complete for the restricted set of formulas of the form $\exists P.\varphi$, with $P \subseteq \mathsf{AP}$ and $\varphi \in \mathsf{CTL}$ [9]. In this section, we give a reduction from the QCTL model-checking problem to QBF.

In the following, we assume a Kripke structure $\mathcal{K} = \langle V, E, \ell \rangle$, an initial state $x_0 \in V$ and a QCTL formula Φ to be fixed. Let V be $\{x_0, \ldots, x_n\}$. We also assume w.l.o.g. that every quantifier $\exists$ and $\forall$ in Φ introduces a fresh atomic proposition, and distinct from the propositions used in $\mathcal{K}$. We use AP_Q^Φ to denote the set of quantified atomic propositions in Φ.

These assumptions allow us to use an alternative notation for the semantics of Φ-subformulas: the truth value of φ will be defined for a state x in $\mathcal{K}$ within an *environment* $\varepsilon : \mathsf{AP}_Q^\Phi \mapsto 2^V$, that is a partial mapping associating a subset of vertices to a proposition in AP_Q^Φ. We use $\mathcal{K}, x \models_\varepsilon \varphi$ to denote that φ holds at x in $\mathcal{K}$ within ε. This ensures that the $\mathcal{K}$'s labelling ℓ is not modified when a subformula is evaluated, only ε is extended with labellings for new quantified propositions. Formally the main changes of the semantics are as follows:

$$\mathcal{K}, x \models_\varepsilon p \text{ iff } \Big((p \in \mathsf{AP}_Q^\Phi \text{ and } x \in \varepsilon(p)) \text{ or } (p \notin \mathsf{AP}_Q^\Phi \text{ and } p \in \ell(x)) \Big)$$

$$\mathcal{K}, x \models_\varepsilon \exists p.\ \varphi \text{ iff } \exists V' \subseteq V \text{ s.t. } \mathcal{K}, x \models_{\varepsilon[p \mapsto V']} \varphi$$

where $\varepsilon[p \mapsto V']$ denotes the mapping which coincides with ε for every proposition in $AP_Q^\Phi \setminus \{p\}$ and associates V' to p.

We use this new notation in order to better distinguish initial $\mathcal{K}$'s propositions and quantified propositions to make proofs simpler. Of course, there is no semantic difference: $\mathcal{K}, x \models \Phi$ iff $\mathcal{K}, x \models_\emptyset \Phi$.

In next sections, we consider general *quantified propositional formulas* (QBF) of the form:

$$\mathsf{QBF} \ni \alpha, \beta ::= q \mid \alpha \wedge \beta \mid \alpha \vee \beta \mid \neg\alpha \mid \alpha \Leftrightarrow \beta \mid \alpha \Rightarrow \beta \mid \exists q.\alpha \mid \forall q.\alpha$$

The formal semantics of a formula α is defined over a Boolean valuation for free variables in α (*i.e.* propositions which are not bound by a quantifier [2]), and it is defined as usual. A formula is said to be closed when it does not contain free variables. In the following, we use the standard notion of validity for closed QBF formulas.

Our aim is then to build a (closed) QBF formula $\widehat{\Phi}^{x_0}$ such that $\widehat{\Phi}^{x_0}$ is valid iff Φ holds true at x_0 in $\mathcal{K}$.

3.1 Overview

We present several reductions of QCTL model-checking problem to QBF validity problem. Given a Φ-subformula φ, a vertex $x \in V$, and a subset $P \subseteq AP_Q^\Phi$, we define a QBF formula $\widehat{\varphi}^{x,P}$ whose variables belong to $\mathsf{AP}_Q^\Phi \times V$ (in the following, we use the notation p^x for $p \in \mathsf{AP}_Q^\Phi$ and $x \in V$). The first two reductions are based on different encodings of temporal modalities, but share a common part given in Figure 1.

[2] We assume w.l.o.g. that every quantifier $\exists$ and $\forall$ introduces a new proposition.

$$\widehat{\neg\varphi}^{x,P} = \neg\widehat{\varphi}^{x,P} \qquad \widehat{\varphi \vee \psi}^{x,P} = \widehat{\varphi}^{x,P} \vee \widehat{\psi}^{x,P} \qquad \widehat{\mathbf{EX}\varphi}^{x,P} = \bigvee_{(x,x')\in E} \widehat{\varphi}^{x',P}$$

$$\widehat{\exists p.\varphi}^{x,P} = \exists p^{x_0} \ldots p^{x_n}.\widehat{\varphi}^{x,P\cup\{p\}} \qquad \widehat{p}^{x,P} = \begin{cases} p^x & \text{if } p \in P \\ \top & \text{if } p \notin P \text{ and } p \in \ell(x) \\ \bot & \text{otherwise} \end{cases}$$

Figure 1 Reduction for basic modalities for reductions UU and FP.

$$\widehat{\mathbf{E}\varphi\mathbf{U}\psi}^{x,P} = \overline{\mathbf{E}\varphi\mathbf{U}\psi}^{x,P,\{x\}} \quad \text{with:}$$

$$\overline{\mathbf{E}\varphi\mathbf{U}\psi}^{x,P,X} = \widehat{\psi}^{x,P} \vee \left(\widehat{\varphi}^{x,P} \wedge \bigvee_{(x,x')\in E s.t. x'\notin X} \overline{\mathbf{E}\varphi\mathbf{U}\psi}^{x',P,X\cup\{x'\}}\right)$$

$$\widehat{\mathbf{A}\varphi\mathbf{U}\psi}^{x,P} = \overline{\mathbf{A}\varphi\mathbf{U}\psi}^{x,P,\{x\}} \quad \text{with:}$$

$$\overline{\mathbf{A}\varphi\mathbf{U}\psi}^{x,P,X} = \begin{cases} \widehat{\psi}^{x,P} & \text{if } \exists (x,x') \in E,\ x' \in X \\ \widehat{\psi}^{x,P} \vee \left(\widehat{\varphi}^{x,P} \wedge \bigwedge_{(x,x')\in E} \overline{\mathbf{A}\varphi\mathbf{U}\psi}^{x',P,X\cup\{x'\}}\right) & \text{otherwise} \end{cases}$$

Figure 2 Reduction for temporal modalities **EU** and **AU** – variant UU.

3.1.1 Unfolding characterization of the until operators

First, we can complete previous construction rules of Figure 1 with those of Figure 2 to get the first method (called UU). This is a naive approach consisting in encoding the temporal modalities as unfoldings of the transition relation. The rules of Figure 2 can be seen as a depth-first way to look for a path satisfying an Until modality (for **EU**) or its negation (for **AU**) among all the simple paths issued from x.

Before stating the correctness of the construction, we need to associate a Boolean valuation v_ε for variables in $\mathsf{AP}_Q^\Phi \times V$ to an environment ε for AP_Q^Φ. We define v_ε as follows: for any $p \in \mathsf{AP}_Q^\Phi$ and $x \in V$, $v_\varepsilon(p^x) = \top$ iff $x \in \varepsilon(p)$.

Now we have the following theorem whose proof is in Appendix A:

▶ **Theorem 4.** *Given a* QCTL *formula* Φ*, a Kripke structure* $\mathcal{K} = \langle V, E, \ell\rangle$*, a state* $x \in V$*, an environment* $\varepsilon : \mathsf{AP}_Q^\Phi \mapsto 2^V$ *and a* Φ*-subformula* φ*, if* $\widehat{\varphi}^{x,\mathsf{dom}(\varepsilon)}$ *is defined inductively w.r.t. the rules of Figures 1 and 2, we have:* $\mathcal{K}, x \models_\varepsilon \varphi$ *iff* $v_\varepsilon \models \widehat{\varphi}^{x,\mathsf{dom}(\varepsilon)}$

It remains to define $\widehat{\Phi}^x$ as $\widehat{\Phi}^{x,\emptyset}$ and we get the reduction: $\mathcal{K}, x_0 \models \Phi$ iff $\widehat{\Phi}^{x_0}$ is valid.

The main drawback of this reduction is the size of the QBF formula: indeed any Until modality may induce a formula whose size is in $O(|V|!)$, and the size of the resulting formula $\widehat{\Phi}^{x_0}$ is then in $O((|\Phi| \cdot |V|!)^{\mathsf{ht}(\Phi)})$. Nevertheless, one can notice that the reduction does not use new quantified propositons to encode the temporal modalities, contrary to other methods we will see later.

3.2 Fixed point characterization of the until operators

Here we present the fixed point method (called FP) for dealing with the modalities **AU** and **EU**. Let φ and ψ be two QCTL-formulas. The idea of the method is to build a QCTL formula that is equivalent to $\mathbf{E}\varphi\mathbf{U}\psi$ (or $\mathbf{A}\varphi\mathbf{U}\psi$), using only the modalities **EX**, **AX** and **AG**.

We first have the following lemma:

▶ **Lemma 5.** *For any* QCTL *formula* $\mathbf{E}\varphi\mathbf{U}\psi$*, we have:*

$$\mathbf{E}\varphi\mathbf{U}\psi \quad \equiv \quad \forall z.\Big(\mathbf{AG}\big(z \Leftrightarrow (\psi \vee (\varphi \wedge \mathbf{EX}\, z))\big) \;\Rightarrow\; z\Big)$$

Proof. Let x be a state in a Kripke structure $\mathcal{K}$. Let θ be the formula $\big(\mathbf{AG}\big(z \Leftrightarrow (\psi \vee (\varphi \wedge \mathbf{EX}\, z))\big) \;\Rightarrow\; z\big)$.
Assume $\mathcal{K}, x \models \mathbf{E}\varphi\mathbf{U}\psi$. We can use the standard characterization of **EU** as fixed point: x belongs to the least fixed point of the equation $Z = \psi \vee (\varphi \wedge \mathbf{EX}\, Z)$ where ψ (resp. φ) is here interpreted as the set of states satisfying ψ (resp. φ). Therefore any z-labelling of reachable states from x [3] corresponding to a fixed point will have the state x labelled. This is precisely what is specified by the QCTL formula.
Now if $\mathcal{K}, x \models \theta$ for every z-labelling corresponding to a fixed point of the previous equation, this is the case for the z-labelling of the states reachable from x and satisfying $\mathbf{E}\varphi\mathbf{U}\psi$, and we deduce $\mathcal{K}, x \models \mathbf{E}\varphi\mathbf{U}\psi$. ◀

And we have the same result for **AU**:

▶ **Lemma 6.** *For any* QCTL *formula* $\mathbf{A}\varphi\mathbf{U}\psi$*, we have:*

$$\mathbf{A}\varphi\mathbf{U}\psi \quad \equiv \quad \forall z.\Big(\mathbf{AG}\big(z \Leftrightarrow (\psi \vee (\varphi \wedge \mathbf{AX}\, z))\big) \;\Rightarrow\; z\Big)$$

Proof. Similar to the proof of Lemma 5. ◀

As a direct consequence, we get the following result:

▶ **Proposition 7.** *For any* QCTL *formula* Φ*, we can build an equivalent* QCTL *formula* $\mathsf{fpc}(\Phi)$ *such that: (1)* $\mathsf{fpc}(\Phi)$ *is built up from atomic propositions, Boolean operators, propositional quantifiers and modalities* **EX** *and* **AG***, and (2) the size of* $\mathsf{fpc}(\Phi)$ *is linear in* $|\Phi|$*.*

Note that the size of $\mathsf{fpc}(\Phi)$ comes from the fact that there is no duplication of subformulas when applying the transformation rules based on equivalences of Lemmas 5 and 6. Moreover, the temporal height of $\mathsf{fpc}(\Phi)$ is smaller than $\mathsf{ht}(\Phi) + 1$ [4].

And to translate $\mathsf{fpc}(\Phi)$ into QBF, it remains to add a single rule [5] to the definitions of Figure 1 to deal with **AG**:

$$\widehat{\mathbf{AG}\varphi}^{x,P} = \bigwedge_{(x,y)\in E^*} \widehat{\varphi}^{\,y,P} \tag{4}$$

Where E^* is the reflexive and transitive closure of E. Then we have:

[3] Labelling other states does not matter.
[4] The temporal height will be increased by 1 if Φ has an until operator whose members are Boolean combinations of atomic propositions.
[5] For **AX** we can either change the rule for **EX** with a disjunction, or express it with **EX** and $\neg$.

▶ **Theorem 8.** *Given a* QCTL *formula* Φ*, a Kripke structure* $\mathcal{K} = \langle V, E, \ell \rangle$*, a state* $x \in V$ *and an environment* $\varepsilon : \mathsf{AP}_Q^\Phi \mapsto 2^V$*, for any* Φ*-subformula* φ*, if* $\widehat{\mathsf{fpc}(\varphi)}^{x,\mathsf{dom}(\varepsilon)}$ *is defined inductively w.r.t. the rules of Figure 1 and the rule 4, we have:*

$$\mathcal{K}, x \models_\varepsilon \varphi \quad \textit{iff} \quad v_\varepsilon \models \widehat{\mathsf{fpc}(\varphi)}^{x,\mathsf{dom}(\varepsilon)}$$

Proof. The proof is a direct consequence of Proposition 7. ◀

With this approach, the size of $\widehat{\mathsf{fpc}(\Phi)}^{x}$ is in $O((|\Phi| \cdot |\mathcal{K}|)^{\mathsf{ht}(\Phi)})$. Indeed, an $\mathbf{EU}$ (or $\mathbf{AU}$) modality gives rise to a QBF formula of size $(|V| + |E|)$. The exponential size comes from the potential nesting of temporal modalities: to avoid it, one could consider the DAG-size of formulas. In the next section, we consider another solution. Note also that the number of propositional variables in the QBF formula is bounded by $|\mathsf{AP}_Q^\Phi| \cdot |V|$.

3.3 Reduction via flat formulas

To avoid the size explosion of $\widehat{\Phi}^x$, one can use an alternative approach for Prenex QCTL formulas. Remember that any QCTL formula can be translated into an equivalent QCTL formula in Prenex normal form whose size is linear in the size of the original formula [10].

In the following, a CTL formula is said to be *basic* when it is of the form $\mathbf{EX}\alpha$, $\mathbf{E}\alpha\mathbf{U}\beta$ or $\mathbf{A}\alpha\mathbf{U}\beta$ where α and β are Boolean combinations of atomic propositions. It is easy to observe that any CTL formula can be translated into a QCTL formula with a temporal height less or equal to 2:

▶ **Proposition 9.** *Any* CTL *formula* Φ *is equivalent to some* QCTL *formula* Ψ *of the form:*

$$\exists\{\kappa_1 \dots \kappa_m\}.\Big(\Phi_0 \wedge \bigwedge_{i=1\dots m} \mathbf{AG}(\kappa_i \Leftrightarrow \theta_i)\Big)$$

where Φ_0 *is a Boolean combination of basic* CTL *formulas and every* θ_i *is a basic* CTL *formula (for any* $1 \leq i \leq m$*). Moreover,* $|\Psi|$ *is in* $O(|\Phi|)$*.*

Proof. Let S_Φ be the set of *temporal* subformulas occurring in Φ at a temporal depth greater or equal to 1. We will prove the proposition by induction over the size of S_Φ. If $|S_\Phi| = 0$, the formula satisfies the property. Now consider a formula with $|S_\Phi| > 0$. Φ must have at least one basic (strict) subformula θ_1. Then Φ is equivalent to the formula $\exists\kappa_1.(\Phi[\theta_1 \leftarrow \kappa_1] \wedge \mathbf{AG}(\kappa_1 \Leftrightarrow \theta_1))$, where κ_1 is a fresh atomic proposition, and $\varphi[\alpha \leftarrow \beta]$ is φ where every occurrence of α is replaced by β. Indeed, any state reachable from the current state x will be labelled by κ_1 iff θ_1 holds true at that state (NB: the states that are not reachable from x do not matter for the truth value of Φ), and this enforces the equivalence. We have $|S_{\Phi[\theta_1 \leftarrow \kappa_1]}| < |S_\Phi|$, thus we can apply induction hypothesis to get:

$$\Phi[\theta_1 \leftarrow \kappa_1] \equiv \exists\{\kappa_2 \dots \kappa_m\}.\Big(\Phi_0 \wedge \bigwedge_{i=2\dots m} \mathbf{AG}(\kappa_i \Leftrightarrow \theta_i)\Big) = \Psi'$$

Where $\kappa_2 \dots \kappa_m$ are fresh atomic propositions, Φ_0 is a Boolean combination of basic CTL formulas and every θ_i is a basic CTL formula. We can conclude that:

$$\begin{aligned}\Phi &\equiv \exists\kappa_1.\Big[\Big(\exists\{\kappa_2 \dots \kappa_m\}.\Big(\Phi_0 \wedge \bigwedge_{i=2\dots m} \mathbf{AG}(\kappa_i \Leftrightarrow \theta_i)\Big)\Big) \wedge \mathbf{AG}(\kappa_1 \Leftrightarrow \theta_1)\Big] \\ &\equiv \exists\{\kappa_1 \dots \kappa_m\}.\Big(\Phi_0 \wedge \bigwedge_{i=1\dots m} \mathbf{AG}(\kappa_i \Leftrightarrow \theta_i)\Big) = \Psi\end{aligned}$$

Note that the last equivalence comes from the fact that no κ_i with $i > 1$ occurs in θ_1. By i.h. the size of Ψ' is linear in $|\Phi[\theta_1 \leftarrow \kappa_1]|$, and the size of $\Phi[\theta_1 \leftarrow \kappa_1]$ is smaller than that of Φ, therefore the size of Ψ is linear in $|\Phi|$. ◀

It is important to note that S_Φ can be described as $\{\varphi_1 \dots \varphi_m\}$, where $\varphi_1 = \theta_1$, and $\varphi_{i+1} = \theta_{i+1}[\kappa_1 \leftarrow \varphi_1, \dots \kappa_i \leftarrow \varphi_i]$. The correspondance $\theta_i - \varphi_i$ will be crucial later on.

In the following, a QCTL formula of the form $\mathcal{Q}.\big(\Phi_0 \wedge \bigwedge_{i=1\dots m} \mathbf{AG}(\kappa_i \Leftrightarrow \theta_i)\big)$, where $\mathcal{Q}$ is a sequence of quantifications, Φ_0 is a Boolean combination of basic CTL formulas, every κ_i is an atomic proposition, and every θ_i (for $i = 1, \dots, m$) is a basic CTL formula, is said to be a *flat* formula.

As a corollary of previous results, we have:

▶ **Proposition 10.** *Any* QCTL *formula* Φ *is equivalent to some flat formula whose size is linear in* $|\Phi|$.

Given a QCTL formula Φ, we use $\mathsf{flat}(\Phi)$ to denote the flat formula equivalent to Φ, obtained by first translating Φ into Prenex normal form, and then transform the CTL subformula as described in Proposition 9.

Applying method FP to some flat formula provides a QBF formula of polynomial size since a flat formula has a temporal height less or equal to 2:

▶ **Corollary 11.** *Given a* QCTL *formula* Φ*, a Kripke structure* $\mathcal{K} = \langle V, E, \ell \rangle$ *and a state* x*, the* QBF *formula* $\widehat{\mathsf{fpc}(\mathsf{flat}(\Phi))}^x$ *obtained by applying the rules of Figure 1 and the rule 4 is valid iff* $\mathcal{K}, x \models \Phi$*. And the size of* $\widehat{\mathsf{fpc}(\mathsf{flat}(\Phi))}^x$ *is in* $O(|V| \cdot (|V| + |E|) \cdot |\Phi|)$.

Therefore this reduction (called FFP) provides a PSPACE algorithm for QCTL model-checking. But there are two disadvantages to this approach. First, putting the formula into Prenex normal form may increase the number of quantified atomic propositions and the number of alternations (which is *in fine* linear in the number of quantifiers in the original formula) [10]. For example, when extracting a quantifier $\forall$ from some **EX** modality, we need to introduce two propositions, this can be seen for the formula $\mathbf{EX}(\forall p.(\mathbf{AX}p \vee \mathbf{AX}\neg p))$ which is translated as:

$$\exists z.\forall p.\forall z'.\Big((\mathbf{EX}(z \wedge z') \Rightarrow \mathbf{AX}(z \Rightarrow z')) \wedge \mathbf{EX}(z \wedge (\mathbf{AX}p \vee \mathbf{AX}\neg p))\Big)$$

where the proposition z is used to mark a state, and z' is used to enforce that only at most one successor is labelled by z. Of course, these two remarks may have a strong impact on the complexity of the decision procedure. Finally, note also that the resulting QBF formula is not in Prenex normal form.

3.4 Variant of FFP

In the previous reduction, the modalities **EU** and **AU** may introduce an alternation of quantifiers: an atomic proposition κ is introduced by an existential quantifier, and then a universal quantifier introduces a variable z to encode the fixed point characterisation of **U**. We propose another reduction in order to avoid this alternation: for this, we will use bit vectors (instead of single Boolean values) associated with every state to encode the distance from the current state to a state satisfying the right-hand side of the Until modality. This reduction produces a prenex QBF formula.

$$\widehat{\varphi \wedge \psi}^{x,P} = \widehat{\varphi}^{x,P} \wedge \widehat{\psi}^{x,P} \qquad \widehat{\varphi \vee \psi}^{x,P} = \widehat{\varphi}^{x,P} \vee \widehat{\psi}^{x,P} \qquad \widehat{\neg p}^{x,P} = \neg \widehat{p}^{x,P}$$

$$\widehat{\mathbf{EX}\varphi}^{x,P} = \bigvee_{(x,y)\in E} \widehat{\varphi}^{y,P} \qquad \widehat{\mathbf{AX}\varphi}^{x,P} = \bigwedge_{(x,y)\in E} \widehat{\varphi}^{y,P} \qquad \widehat{\mathbf{AG}\varphi}^{x,P} = \bigwedge_{(x,y)\in E^*} \widehat{\varphi}^{y,P}$$

For $Q \in \{\exists, \forall\}$ $\quad \widehat{Qp.\varphi}^{x,P} = Qp^{x_0} \dots p^{x_n} \widehat{\varphi}^{x,P\cup\{p\}}$

For $Q \in \{\exists, \forall\}$ $\quad \widehat{Q\dot{\kappa}_i.\varphi}^{x,P} = \begin{cases} Q\overline{\kappa_i}^{x_0} \dots \overline{\kappa_i}^{x_n}.\widehat{\varphi}^{x,P\cup\{\overline{\kappa_i}\}} & \text{if } \dot{\kappa}_i \text{ encodes} \\ & \text{a least fixed point.} \\ Q\kappa_i^{x_0} \dots \kappa_i^{x_n}.\widehat{\varphi}^{x,P\cup\{\kappa_i\}} & \text{otherwise} \end{cases}$

$$\widehat{p}^{x,P} = \begin{cases} p^x & \text{if } p \in P \text{ and } p \text{ is a Boolean var.} \\ [p^x < |V|] & \text{if } p \in P \text{ and } p \text{ is a Bit vect.} \\ \top & \text{if } p \notin P \wedge p \in \ell(x) \\ \bot & \text{otherwise} \end{cases}$$

Figure 3 Transformation rules for method FBV.

First, we consider a QCTL formula Φ under negation normal form (NNF), where the negation is only applied to atomic propositions. This transformation makes that Φ is built from temporal modalities in $S_{tmod} = \{\mathbf{EX}, \mathbf{AX}, \mathbf{EU}, \mathbf{AU}, \mathbf{EW}, \mathbf{AW}\}$

We can then reformulate Proposition 10 as the following proposition, whose proof is in Appendix B:

▶ **Proposition 12.** *Any* QCTL *formula* Φ *is equivalent to some* QCTL *formula* Ψ *in NNF of the form:* $\Psi = \mathcal{Q}\,\exists\{\kappa_1 \dots \kappa_m\}.\Big(\Phi_0 \wedge \bigwedge_{i=1\dots m} \mathbf{AG}(\kappa_i \Rightarrow \theta_i)\Big)$ *where* $\mathcal{Q}$ *is a sequence of quantifications,* Φ_0 *is a* CTL *formula containing only the temporal modalities* **EX**, **AX** *or* **AG** *and whose temporal height is less or equal to 1, and every* θ_i *is a basic* CTL *formula (with* $1 \leq i \leq m$*). Moreover,* $|\Psi|$ *is in* $O(|\Phi|)$*.*

From the previous proposition, we derive a new reduction to QBF (called FBV). For modalities **EW** and **AW**, we use the same encoding as for method FP, except that we use the corresponding Boolean proposition κ_i directly for the greatest fixed point. And for Until-based modalities, corresponding to least fixed points, we will use bit vectors instead of atomic propositions to encode the truth value of **EU** or **AU**: for a formula $\theta_i = \mathbf{E}\varphi\mathbf{U}\psi$, we will consider a bit vector $\overline{\kappa_i}$ of length $\lceil \log(|V|+1) \rceil$ for every state instead of a single Boolean value κ_i. The idea is that in a state x, the value $\overline{\kappa_i}$ encodes in binary the distance (in terms of number of transitions) from x to a state satisfying ψ along a path satisfying φ. And for $\theta_i = \mathbf{A}\varphi\mathbf{U}\psi$, the value $\overline{\kappa_i}$ encodes the maximal distance before a state satisfying ψ (along a path where φ is true). In the following, such a $\overline{\kappa_i}$ associated to a θ_i based on an Until is called an Until-κ. Note that given a bit vector $\overline{\kappa_i}$ and an integer value d encoded in binary, we will use $[\overline{\kappa_i}^y = d]$ and $[\overline{\kappa_i}^y < d]$ to denote the corresponding propositional formulas over $\overline{\kappa_i}^y$.

The new reduction is based on the rewriting rules of Figure 3 for several operators, and we define the reduction for $\mathbf{AG}(\kappa_i \Rightarrow \theta_i)$ for $\theta_i = \mathbf{E}\varphi\mathbf{W}\psi$ or $\theta_i = \mathbf{A}\varphi\mathbf{W}\psi$ as follows:

$$\widehat{\mathbf{AG}(\kappa_i \Rightarrow \mathbf{E}\varphi\mathbf{W}\psi)}^{x,P} = \bigwedge_{(x,y)\in E^*} \Big(\kappa_i^y \Rightarrow \big(\widehat{\psi}^{y,P} \vee (\widehat{\varphi}^{y,P} \wedge \bigvee_{(y,y')\in E} \kappa_i^{y'})\big)\Big) \tag{5}$$

$$\widehat{\mathbf{AG}(\kappa_i \Rightarrow \mathbf{A}\varphi\mathbf{W}\psi)}^{x,P} = \bigwedge_{(x,y)\in E^*} \Big(\kappa_i^y \Rightarrow \big(\widehat{\psi}^{y,P} \vee (\widehat{\varphi}^{y,P} \wedge \bigwedge_{(y,y')\in E} \kappa_i^{y'})\big)\Big) \tag{6}$$

And for $\mathbf{AG}(\overline{\kappa_i} \Rightarrow \theta_i)$ with $\theta_i = \mathbf{E}\varphi\mathbf{U}\psi$ or $\theta_i = \mathbf{A}\varphi\mathbf{U}\psi$, we have:

$$\begin{aligned}\widehat{\mathbf{AG}(\overline{\kappa_i} \Rightarrow \mathbf{E}\varphi\mathbf{U}\psi)}^{x,P} = \bigwedge_{(x,y)\in E^*} &\Big[\Big([\overline{\kappa_i}^y = 0] \Rightarrow \widehat{\psi}^{y,P}\Big) \\ &\wedge \bigwedge_{1\le d<|V|} \Big([\overline{\kappa_i}^y = d] \Rightarrow \big(\widehat{\varphi}^{y,P} \wedge \bigvee_{(y,y')\in E} [\overline{\kappa_i}^{y'} = d-1]\big)\Big)\Big]\end{aligned} \tag{7}$$

where $[\overline{\kappa_i}^y = d]$ and $[\overline{\kappa_i}^y < d]$ with a given value d correspond to propositional formulas over $\overline{\kappa_i}^y$. And for $\mathbf{AG}(\overline{\kappa_i} \Rightarrow \theta_i)$ with $\theta_i = \mathbf{A}\varphi\mathbf{U}\psi$, we have:

$$\begin{aligned}\widehat{\mathbf{AG}(\overline{\kappa_i} \Rightarrow \mathbf{A}\varphi\mathbf{U}\psi)}^{x,P} = \bigwedge_{(x,y)\in E^*} &\Big[\Big([\overline{\kappa_i}^y = 0] \Rightarrow \widehat{\psi}^{y,P}\Big) \\ &\wedge \bigwedge_{1\le d<|V|} \Big([\overline{\kappa_i}^y = d] \Rightarrow \big(\widehat{\varphi}^{y,P} \wedge \bigwedge_{(y,y')\in E} [\overline{\kappa_i}^{y'} < d]\big)\Big)\Big]\end{aligned} \tag{8}$$

Note that in the equivalences 7 and 8, we cannot replace the implication by an equivalence: for a state y s.t. $\widehat{\varphi}^{y,P}$, the existence of some successor y' s.t. $[\overline{\kappa_i}^{y'} < d]$ is not sufficient to imply $[\overline{\kappa_i}^y = d]$. This is why we consider only implications here.

Method FBV is defined from previous rules and we get a Prenex QBF formula:

▶ **Corollary 13.** *Given a* QCTL *formula* Φ*, a Kripke structure* $\mathcal{K} = \langle V, E, \ell\rangle$*, and a state* x*, the Prenex* QBF *formula* $\widehat{\mathsf{flat}(\Phi)}^x$ *obtained by applying the rules of Figure 3 and rules 5, 6, 7, and 8 above, is valid iff* $\mathcal{K}, x \models \Phi$*, and* $|\widehat{\mathsf{flat}(\Phi)}^x|$ *is in* $O(|V| \cdot (|V| + |E|) \cdot |\Phi|)$*.*

This approach allows us to easily adapt the algorithm to bounded model-checking: instead of considering values from 1 to $|V|$ for d in the definition of Until modalities, one can restrict the range to a smaller interval, to get a smaller QBF formula to check. Note also that obtaining a Prenex QBF formula is interesting in practice, because many QBF-solvers require Prenex formulas as inputs.

We can also observe that this reduction proceeds a bit like the classical model-checking algorithm for CTL where for deciding $x \models \Phi$, the truth value of every Φ-subformula is computed in every state of the model. In some case, this may induce additional work compared to methods like FP or UU (for example to decide whether the initial state satisfies a **EU** formula).

3.4.1 Using BitVectors for $\exists^1$ and $\forall^1$

The quantifiers $\exists^1$ and $\forall^1$ are very useful in many specifications. It can be interesting to develop ad-hoc algorithms in order to improve the generated QBF formulas. For the first methods we described, they are translated into propositional formulas (instead of introducing extra quantified atomic propositions as they are formally defined). Another method consists in using bit vectors as in the treatment of Until modalities described above: in this case, these quantifiers introduce a unique bit vector of size $\lceil \log(|V|+1) \rceil$ to store the number of the state

selected by the quantifier (that is the state which will be labelled by the proposition). This method is interesting, since it reduces the number of quantified propositions, it is integrated in the method FBV.

4 Experimental results

In this section, we consider three examples to illustrate the QBF-based model-checking approach for QCTL. The problems we will consider can be solved more efficiently without a QCTL model-checker, but they provide valuable insights on the performance of the reduction strategies, and the properties to be checked cannot be expressed with classical temporal logics. The first example shows that the performances can change significally when we choose different formulas to check, even if they are equivalent. The second problem illustrates how QCTL can naturally express properties related to game theory, and it is a situation where we can observe how bounded model-checking can improve the performance of FBV. In the last example, we check formulas with a large temporal height, thus we can see how much space is gained by flattening the formulas. We wanted to manipulate graphs that could easily be scaled up by tweaking a few parameters, so we chose to use grids.

We have implemented a prototype to try the different reduction strategies [6] . Our tool is available online [7]: given a Kripke structure $\mathcal{K}$ with a state x and a formula Φ, it produces a specification file (corresponding to $\widehat{\Phi}^x$) for the SMT-solver Z3 [5]. The choice of Z3 was motivated by the fact that the generated QBF formulas are not always Prenex, which many QBF-solvers require, unlike Z3.

4.1 k-connectivity

Here, we consider an undirected graph, and we want to check whether there exist (at least) k internally disjoint paths [8] from a vertex x to some vertex y. A classical result in graph theory due to Menger ensures that, given two vertices x and y in a graph G, the minimum number of vertices whose deletion makes that there is no more paths between x and y is equal to the maximum number of internally disjoint paths between these two vertices.

We can encode these two ideas by the following QCTL formulas (interpreted over x):

$$\Phi_k \;=\; \exists p_1 \ldots \exists p_{k-1} \Big(\bigwedge_{1\leq i<k} \mathbf{EX}\big(\mathbf{E}(p_i \wedge \bigwedge_{j\neq i} \neg p_j)\,\mathbf{U}\, y\big) \;\wedge\; \mathbf{EX}\,\mathbf{E}\big(\bigwedge_{1\leq i<k} \neg p_i\big)\,\mathbf{U}\, y\Big) \tag{9}$$

$$\Psi_k \;=\; \forall^1 p_1 \ldots \forall^1 p_{k-1}\, \mathbf{EX}\,\Big(\mathbf{E}\big(\bigwedge_{1\leq i<k} \neg p_i\big)\,\mathbf{U}\, y\Big) \tag{10}$$

Φ_k uses the labelling by the p_i's to mark the internal vertices of k paths between the current position and the vertex y. The modality $\mathbf{EX}$ is used to consider only the intermediate states (and not the starting state). The formula Ψ_k proceeds differently: the idea is to mark exactly $k-1$ states with $p_1, \ldots, p_{k-1}$ and to verify that there still exists at least one path leading to y without going through the states labelled by some p_i.

By Menger's Theorem, we know that these formulas are equivalent over undirected graphs.

---

[6] NB: For reductions FFP and FBV the formula has to be given in Prenex normal form.

[7] `https://www.irif.fr/~francoisl/qctlmc.html`

[8] Two paths $src \leftrightarrow r_1 \leftrightarrow \ldots \leftrightarrow r_k \leftrightarrow dest$ and $src \leftrightarrow r'_1 \leftrightarrow \ldots \leftrightarrow r'_{k'} \leftrightarrow dest$ are internally disjoint iff $r_i \neq r'_j$ for any $1 \leq i \leq k$ and $1 \leq j \leq k'$. And note that with this def., if there is an edge (x, y), there exist k internally disjoint paths from x to y for any k.

We interpret these formulas over Kripke structures $\mathcal{S}_{n,m}$ with $n \geq m$ (see Figure 4) which correspond to two kinds of grids $n \times n$ connected by m edges (these edges are of the form $(q_{i,n}, r_{1,i})$ or $(q_{n,i}, r_{i,1})$. The initial state is $q_{1,1}$ and when evaluating Φ_k or Ψ_k we assume the state $r_{n,n}$ to be labelled by y. In this context, we clearly have that Φ_k and Ψ_k hold for true at $q_{1,1}$ iff $k \leq m$.

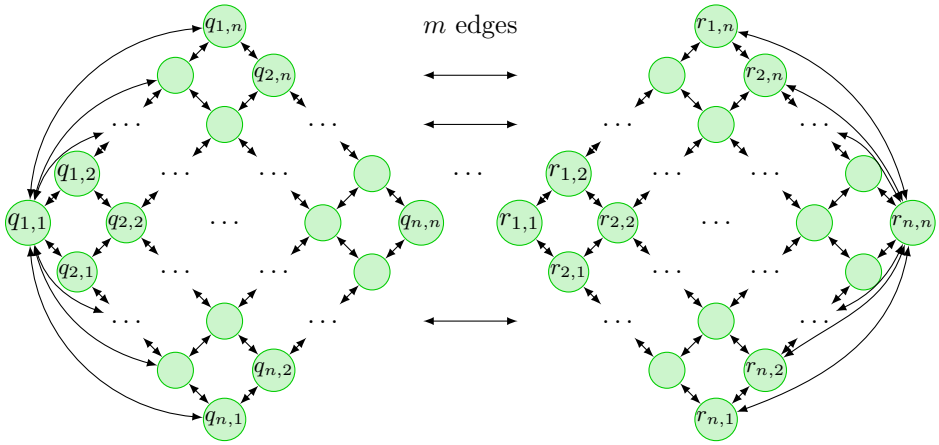


Figure 4 Structure $\mathcal{S}_{n,m}$ for the k-connectivity problem.

Detailed results are presented in Appendix C. The main lessons we can see are:

- Formula Φ_k is much more difficult to verify: the number of temporal modalities is probably one explanation. Another one for methods FP and FFP(based on the fixed point characterization of **U**/**W** modalities) could be an alternation of quantifiers: in Φ_k, there is an existential quantification over the p_is and the **EU**s introduce a universal quantification, but it is not the case for Ψ_k, where there are only universal quantifications for these reductions.
- The reduction FP is the most efficient: it can be used to verify models with more than two thousands states when m is small. The method FFP is also rather efficient, but the flattening seems to be too costly for such a simple formula.
- The reduction UU produces very large QBF formulas, but rather simple to check.

We can generalise the problem by verifying that there exist at least k internally disjoint paths between any pair of reachable vertices x and y in a given structure. These previous formulas can be modified as follows:

$$\Phi^g_k \;=\; \forall^1 y \exists p_1 \ldots \exists p_{k-1}\, \mathbf{AG}\Big(\bigwedge_{1\leq i<k} \mathbf{EX}\big(\mathbf{E}(p_i \wedge \bigwedge_{j\neq i} \neg p_j)\,\mathbf{U}\, y\big) \;\wedge\; \mathbf{EX}\,\mathbf{E}(\bigwedge_{1\leq i<k} \neg p_i)\,\mathbf{U}\, y\Big) \tag{11}$$

$$\Psi^g_k \;=\; \forall^1 y \forall^1 p_1 \ldots \forall^1 p_{k-1} \mathbf{AG}\Big[\,\mathbf{EX}\,\Big(\mathbf{E}(\bigwedge_{1\leq i<k} \neg p_i)\,\mathbf{U}\, y\Big)\Big] \tag{12}$$

In that case, Φ^g_k is useless: too complex to be verified. And the method FP is still the most efficient.

4.2 Nim game

Nim game is a turn-based two-player game. A configuration is a set of heaps of objects and a boolean value indicating whose turn it is. At each turn, a player has to choose one non-empty heap and remove at least one object from it. The aim of each player is to remove

the last object. Given a configuration c and a Player-J with $J \in \{1, 2\}$, we can build a finite Kripke structure $\mathcal{S}_J$, where x_c is a state corresponding to the configuration c ; and use a QCTL formula Φ^J_{win} such that $\mathcal{S}, x_c \models \Phi^J_{\mathsf{win}}$ iff Player-J has a wining strategy from c. Note that there is a simple and well-known criterion over the numbers of objects in each heap to decide who has a winning strategy, but we consider this problem just because it is interesting to illustrate what kind of problem we can solve with QCTL.

Each configuration corresponds to a state in $\mathcal{S}_J$. Every move for Player-$\overline{J}$ from a configuration c to a configuration c' provides a transition $(x_c, x_{c'})$ in $\mathcal{S}_J$. However, a move of Player-J from c to c' is encoded as two transitions $x_c \to x_{c,c'} \to x_{c'}$ where $x_{c,c'}$ is then an intermediary state we use to encode a strategy for Player-J (marking $x_{c,c'}$ by an atomic proposition will correspond to Player-J choosing c' from c). We assume that every state x_c is labelled by t_1 if it's Player-1's turn to play at c, and by t_2 otherwise. Every intermediary state $x_{c,c'}$ is labelled by int. We also label empty configurations by w_1 or w_2, depending on which player played the last move.

Clearly, the size of $\mathcal{S}$ will depend on the number of objects in each set in the initial configuration. The formula Φ^J_{win} depends only on J:

$$\Phi^J_{\mathsf{win}} \;=\; \exists m.\Big(\mathbf{AG}\big(t_J \;\Rightarrow\; \mathbf{EX}m\big) \;\wedge\; \mathbf{AF}\big(w_J \vee (\mathrm{int} \wedge \neg m)\big)\Big)$$

This formula holds true in a state corresponding to some configuration c iff there exists a labelling by m such that every reachable configuration where it's Player-J's turn, has a successor labelled by m (thus a possible choice to do) and every execution from the current state leads to either a winning state for Player-J or a non-selected intermediary state, therefore all outcomes induced by the underlying strategy have to verify $\mathbf{F}w_J$. Note that in this example, the Kripke structure is acyclic (except the self-loops on the ending states).

From detailed results in appendix, we can see:

- One can consider structures with more than 10 thousand states. Note that the number of heaps is important for the size of the model, but the maximal length of a game depends on the number of objects (and is rather small in our examples).
- The most efficient method is FFP (with FP).
- Method FBV is more efficient when we consider bounded model-checking. Note that in this case, the verification may not be complete: if the QBF formula is valid, the property is satisfied by the structure, otherwise, no conclusion can be done, except if we can prove that the chosen bound was big enough to be sure that there is no solution. In our case, we can easily compute the maximal bound: at each turn, a player has to pick at least one object, such a move may give rise to one transition in the model (for the opponent), or two transitions (for the player for whom we look for a strategy). Thus, if there are n objects in the initial configuration, we can choose $\frac{3n}{2}$ for the bound.

4.3 Resources distribution

The last example is as follows: given a Kripke structure $\mathcal{S}$ and two integers k and d, we aim at choosing at most k states (called targets in the following) such that every reachable state (from the initial one) can reach a target in less than d transitions. This problem can be encoded with the following QCTL formula where d modalities $\mathbf{EX}$ are nested:

$$\Phi_{\mathsf{res}} \;=\; \exists^1 c_1 \ldots \exists^1 c_k\, \mathbf{AG}\Big((\bigvee_{1\le i\le k} c_i) \vee \mathbf{EX}\Big((\bigvee_{1\le i\le k} c_i) \vee \Big(\ldots \vee \mathbf{EX}(\bigvee_{1\le i\le k} c_i)\Big)\Big)\Big)$$

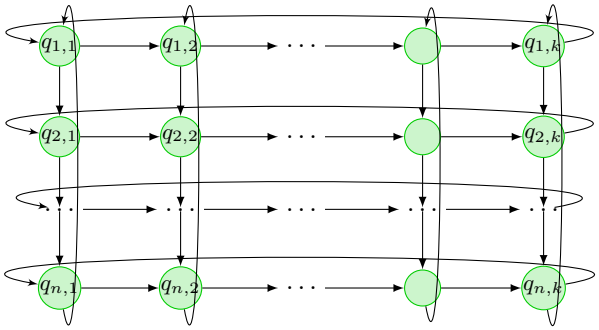


Figure 5 Structure $\mathcal{K}_{n,k}$ for the resources distribution problem.

For experimental results, we consider the grid $\mathcal{K}_{n,m}$ described at Figure 5. Note that for this example, reductions UU and FP are similar because there is no Until in the formula (**AG** is treated as a conjunction). From detailed results in appendix, one can see:

- Only small models have been successfully verified.
- The nesting of **EX**s operators give an advantage to the reduction based on flattening (FBV and FFP): the size of the QBF formula increases more slowly.
- The reduction FBV is the most efficient on this example. Since there is no Until modality (except behind **AG** which is treated separately), the difference with FFP is due to the encoding of $\exists_1$ operator with a unique bit vector in FBV, this choice seems to be more efficient in this example.

5 Conclusion

We have presented several reductions from QCTL model-checking to QBF. This provides a first tool for QCTL model-checking. Of course, this is an ongoing-work, and many improvements are possible: the reduction strategies are still naive and could be significantly improved, and a better understanding of QBF-solvers would also be helpful to produce more efficient formulas (we have not yet tried to normalise formulas in a specific form which is often a crucial aspect in SAT/QBF-solving). Still, these first results are rather interesting and encouraging. They show the importance of writing "good" QCTL formulas for which the solver will be able to provide a result (this problem already exists for classical temporal logics, but it is more significant here due to the complexity induced by the quantifications). The examples also show that there is no "one best strategy": it depends on the structure of the considered formula, and then offering several reduction strategies seems to be necessary in a QBF-based model-checker for QCTL. Finally this work is also interesting because it could easily be adapted for other logics (like Sabotage logics [17]). In particular, we plan to figure how it could be adapted for LTL. In the future, we plan to continue to work on reduction strategies, and to use other QBF-solvers.

References

1 Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. *Advances in Computers*, 58:117–148, 2003.

2 Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic Model Checking without BDDs. In *Tools and Algorithms for Construction and Analysis of Systems, 5th International Conference, TACAS '99, Amsterdam, The Netherlands, March 22-28, 1999, Proceedings*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 1999.

3 Edmund M. Clarke and E. Allen Emerson. Design and Synthesis of Synchronization Skeletons using Branching-Time Temporal Logic. In Dexter C. Kozen, editor, *Proceedings of the 3rd Workshop on Logics of Programs (LOP'81)*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1982.

4 Arnaud Da Costa, François Laroussinie, and Nicolas Markey. Quantified CTL: Expressiveness and Model Checking. In Maciej Koutny and Irek Ulidowski, editors, *Proceedings of the 23rd International Conference on Concurrency Theory (CONCUR'12)*, volume 7454 of *Lecture Notes in Computer Science*, pages 177–192. Springer-Verlag, September 2012.

5 Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Budapest, Hungary, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

6 Nachum Dershowitz, Ziyad Hanna, and Jacob Katz. Bounded Model Checking with QBF. In *Theory and Applications of Satisfiability Testing, 8th International Conference, SAT 2005, St. Andrews, UK, June 19-23, 2005, Proceedings*, volume 3569 of *Lecture Notes in Computer Science*, pages 408–414. Springer, 2005.

7 Tim French. Decidability of Quantified Propositional Branching Time Logics. In Markus Stumptner, Dan Corbett, and Mike Brooks, editors, *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence (AJCAI'01)*, volume 2256 of *Lecture Notes in Computer Science*, pages 165–176. Springer-Verlag, December 2001.

8 Tim French. Quantified Propositional Temporal Logic with Repeating States. In *Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and of the 4th International Conference on Temporal Logic (TIME-ICTL'03)*, pages 155–165. IEEE Comp. Soc. Press, July 2003.

9 Orna Kupferman. Augmenting Branching Temporal Logics with Existential Quantification over Atomic Propositions. In Pierre Wolper, editor, *Proceedings of the 7th International Conference on Computer Aided Verification (CAV'95)*, volume 939 of *Lecture Notes in Computer Science*, pages 325–338. Springer-Verlag, July 1995.

10 François Laroussinie and Nicolas Markey. Quantified CTL: Expressiveness and Complexity. *Logical Methods in Computer Science*, 10(4), 2014.

11 François Laroussinie and Nicolas Markey. Augmenting ATL with strategy contexts. *Inf. Comput.*, 245:98–123, 2015.

12 Kenneth L. McMillan. Applying SAT Methods in Unbounded Symbolic Model Checking. In *Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings*, volume 2404 of *Lecture Notes in Computer Science*, pages 250–264. Springer, 2002.

13 Anindya C. Patthak, Indrajit Bhattacharya, Anirban Dasgupta, Pallab Dasgupta, and P. P. Chakrabarti. Quantified Computation Tree Logic. *Information Processing Letters*, 82(3):123–129, 2002.

14 Amir Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS'77)*, pages 46–57. IEEE Comp. Soc. Press, October-November 1977.

15 Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In Mariangiola Dezani-Ciancaglini and Ugo Montanari, editors, *Proceedings of the 5th International Symposium on Programming (SOP'82)*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag, April 1982.

16 A. Prasad Sistla. *Theoretical Issues in the Design and Verification of Distributed Systems*. PhD thesis, Harvard University, Cambridge, Massachussets, USA, 1983.

17 Johan van Benthem. An Essay on Sabotage and Obstruction. In *Mechanizing Mathematical Reasoning, Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, volume 2605 of *Lecture Notes in Computer Science*, pages 268–276. Springer, 2005.

A Proof of Theorem 4

Proof. We assume Φ to be fixed and we prove the property by structural induction over φ. Boolean operators are omitted.

- $\varphi = p$: if $\mathcal{K}, x \models_\varepsilon p$, then either p is a quantified proposition and it belongs to $\mathsf{dom}(\varepsilon)$ and x belongs to $\varepsilon(p)$ and thus $v_\varepsilon \models p^x$ by def. of v_ε, or p belongs to $\ell(x)$. In both cases we have $v_\varepsilon \models \widehat{p}^{x,\mathsf{dom}(\varepsilon)}$. The converse is similar.
- $\varphi = \mathbf{EX}\psi$: $\mathcal{K}, x \models_\varepsilon \mathbf{EX}\psi$ iff there exists $(x, x') \in E$ s.t. $\mathcal{K}, x' \models_\varepsilon \psi$, iff (by i.h.) there exists $(x, x') \in E$ s.t. $v_\varepsilon \models \widehat{\psi}^{x',\mathsf{dom}(\varepsilon)}$ which is equivalent to $v_\varepsilon \models \widehat{\mathbf{EX}\psi}^{x,\mathsf{dom}(\varepsilon)}$.
- $\varphi = \exists p.\psi$. We have $\mathcal{K}, x \models_\varepsilon \exists p.\psi$ iff there exists $V' \subseteq V$ s.t. $\mathcal{K}, x \models_{\varepsilon[p \mapsto V']} \psi$, iff (i.h.) $v_{\varepsilon[p\mapsto V']} \models \widehat{\psi}^{x,\mathsf{dom}(\varepsilon)\cup\{p\}}$ which is equivalent to $v_\varepsilon \models \exists p^{v_1} \dots p^{v_n}.\widehat{\psi}^{x,\mathsf{dom}(\varepsilon)\cup\{p\}}$ (by def. of v_ε).
- $\varphi = \mathbf{E}\psi_1\mathbf{U}\psi_2$: The definition of $\widehat{\varphi}^{x,\mathsf{dom}(\varepsilon)}$ corresponds to a finite unfolding of the expansion law that characterizes the **EU** modality. Assume $\mathcal{K}, x \models_\varepsilon \varphi$. There exists a path $\rho \in \mathsf{Path}^\omega_{\mathcal{K}}(x)$ and a position $i \geq 0$ s.t. $\rho(i) \models_\varepsilon \psi_2$ and $\rho(k) \models_\varepsilon \psi_1$ for any $0 \leq k < i$. The finite prefix $x = \rho(0)\cdots\rho(k)$ can be assumed to be simple, and then $k < |V|$. By using i.h., we get $v_\varepsilon \models \widehat{\psi_2}^{\rho(i),\mathsf{dom}(\varepsilon)}$, and $v_\varepsilon \models \widehat{\psi_1}^{\rho(k),\mathsf{dom}(\varepsilon)}$ for any $0 \leq k < i$. From this point, the reader can easily verify by induction (starting at i, down to 0) that $v_\varepsilon \models \overline{\mathbf{E}\psi_1\mathbf{U}\psi_2}^{\rho(k),dom(\varepsilon),\{\rho(j)|j\leq k\}}$ for all $k \leq i$. This makes $\widehat{\varphi}^{x,\mathsf{dom}(\varepsilon)}$ to be satisfied by v_ε. Conversely, assume $v_\varepsilon \models \widehat{\varphi}^{x,\mathsf{dom}(\varepsilon)}$. Let us build a finite path $x = x_0, x_1, \cdots x_i$ that satisfies $\psi_1\mathbf{U}\psi_2$. The first vertex is of course x, and we have $v_\varepsilon \models \overline{\mathbf{E}\psi_1\mathbf{U}\psi_2}^{x,dom(\varepsilon),\{x\}}$. Fix i so that for every $0 \leq k \leq i$, x_k is built, $v_\varepsilon \models \overline{\mathbf{E}\psi_1\mathbf{U}\psi_2}^{x_k,dom(\varepsilon),\{x_j|j\leq k\}}$, $v_\varepsilon \not\models \widehat{\psi_2}^{x_k,dom(\varepsilon)}$, and $(x_k, x_{k+1}) \in E$ when $k < i$. Then, there must be $(x_i, y) \in E$ so that $y \notin \{x_j|j \leq i\}$, and $v_\varepsilon \models \overline{\mathbf{E}\psi_1\mathbf{U}\psi_2}^{y,dom(\varepsilon),\{x_j|j\leq i\}\cup\{y\}}$, so we set $x_{i+1} = y$. The sequence eventually stops, because V is finite and the path is simple. If x_i is its last vertex, then we must have $v_\varepsilon \models \widehat{\psi_2}^{x_i,dom(\varepsilon)}$, and $v_\varepsilon \models \widehat{\psi_1}^{x_k,dom(\varepsilon)}$ for every $0 \leq k < i$. By using i.h., we get $\mathcal{K}, x \models_\varepsilon \varphi$.
- $\varphi = \mathbf{A}\psi_1\mathbf{U}\psi_2$: this case is similar to the previous one, except that we have to consider loops. Assume $\mathcal{K}, x \models_\varepsilon \varphi$. Then any path issued from x satisfies $\psi_1\mathbf{U}\psi_2$. If x contains a self-loop, then ψ_2 has to be satisfied at x (this is ensured by the first case in the def. of $\widehat{\varphi}^{x,\mathsf{dom}(\varepsilon)}$). Otherwise we consider all the paths from x: either there is a simple prefix witnessing $\psi_1\mathbf{U}\psi_2$, or there is a loop from some point. In the latter case, one of the state in the loop has to verify ψ_2. In both cases, the definition of $\widehat{\varphi}^{x,\mathsf{dom}(\varepsilon)}$ gives the result. ◀

B Proof of Proposition 12

Proof. Consider w.l.o.g. a QCTL formula Φ in Prenex normal form and NNF. We can define Φ_0 and the basic formulas θ_is approximately as in Proposition 9, except that Φ_0 contains only **EX**, **AX** or **AG** modalities, and every other modality gives rise to some quantified proposition κ and a subformula $\mathbf{AG}(\dots)$ in the main conjunction of Ψ. Every θ_i starts with a modality in S_{tmod}. Let φ_i be the original Φ-subformula associated with θ_i. Note that Ψ is in NNF, and κ_i occurs only once in Ψ in the scope of a negation, and it happens in the

subformula $\mathbf{AG}(\kappa_i \Rightarrow \theta_i)$. We now have to show that Φ is equivalent to Ψ. Consider the formula $\widetilde{\Psi}$ where every $\Rightarrow$ is replaced by $\Leftrightarrow$: by following the same arguments of Proposition 9, we clearly have $\Phi \equiv \widetilde{\Psi}$, and $\widetilde{\Psi} \Rightarrow \Psi$. It remains to prove the opposite direction.

To prove $\Psi \Rightarrow \widetilde{\Psi}$, it is sufficient to show that this is true for the empty $\mathcal{Q}$ (as equivalence is substitutive). Assume $\mathcal{K}, x \models_\varepsilon \Psi$. Then there exists an environment ε' from $\{\kappa_1, \ldots, \kappa_m\}$ to 2^V such that $\mathcal{K}, x \models_{\varepsilon \circ \varepsilon'} \Phi_0 \wedge \bigwedge_i \mathbf{AG}(\kappa_i \Rightarrow \theta_i)$[9]. Now we have:

$$\forall i, \quad \mathcal{K}, x \models_{\varepsilon \circ \varepsilon'} \theta_i \quad \Rightarrow \quad \mathcal{K}, x \models_\varepsilon \varphi_i \quad \text{and} \quad \mathcal{K}, x \models_{\varepsilon \circ \varepsilon'} \Phi_0 \quad \Rightarrow \quad \mathcal{K}, x \models_{\varepsilon \circ \varepsilon'} \widetilde{\Phi_0}$$

Indeed, assume that it is not true and $\mathcal{K}, x \models_{\varepsilon \circ \varepsilon'} \theta_i$ and $\mathcal{K}, x \not\models_\varepsilon \varphi_i$. Consider such a formula φ_i with the smallest temporal height. The only atomic propositions κ_j occurring in θ_i are then associated with some θ_j and φ_j which verify the property and thus any state satisfying such a θ_j, also satisfies φ_j. Therefore any state labelled by such a κ_j is correctly labelled (and satisfies φ_j). And the states that are not labelled by κ_j cannot make θ_i to be wrongly evaluated to true (because κ_j is not in the scope of a negation). Therefore φ_i holds true at x. The same holds for Φ_0 and $\widetilde{\Phi_0}$. As a direct consequence, we have $\Psi \Rightarrow \widetilde{\Psi}$. ◀

C Experimental results

Detailed results for the three examples are presented in this section. In every case, we distinguished the time required to build the QBF formula (the Z3 specification) and the time required to solve it. Times are given in seconds.

C.1 k-connectivity

n, m	$3, 2$	$4, 3$	$4, 3$	$5, 4$	$35, 4$
formula	Ψ_2	Ψ_4	Φ_4	Ψ_5	Ψ_4
# states	18	32	32	50	2450
res	sat	unsat	unsat	unsat	sat
Reduction UU					
Time to build z3 form.	0.01	38.45	70.15	–	–
Size of z3 form.	7682	7922175	29983034	–	–
Time to solve z3 form.	0.04	2.38	8.61	–	–
Reduction FP					
Time to build z3 form.	0	0.03	0.07	0.09	388
Size of z3 form.	1313	9097	11234	27416	38675099
Time to solve z3 form.	0.03	0.06	–	0.16	132.67
Reduction FFP					
Time to build z3 form.	0.03	0.1	0.25	0.1	–
Size of z3 form.	3666	20944	56196	61519	–
Time to solve z3 form.	0.12	0.28	–	57.96	–
Reduction FBV					
Time to build z3 form.	0.01	1	0.23	1.3	–
Size of z3 form.	6363	31243	107120	87681	–
Time to solve z3 form.	40.16	–	43.09	–	–

[9] We define the graph $G_{\varepsilon \circ \varepsilon'}$ of $\varepsilon \circ \varepsilon'$ to be $G_\varepsilon \uplus G_{\varepsilon'}$

n, m	$3, 1$	$3, 3$	$9, 2$	$27, 2$
formula	Ψ_2^g	Ψ_3^g	Ψ_3^g	Ψ_3^g
# states	18	18	162	1458
res	unsat	sat	unsat	unsat
Reduction UU				
Time to build z3 form.	$1,08$	$9,3$	–	–
Size of z3 form.	1195684	8632341	–	–
Time to solve z3 form.	$0,43$	$8,96$	–	–
Reduction FP				
Time to build z3 form.	0.02	$0,04$	$1,94$	$190,94$
Size of z3 form.	13268	18337	1698025	139992889
Time to solve z3 form.	$0,1$	$0,23$	$45,51$	–
Reduction FFP				
Time to build z3 form.	$0,02$	0.04	2.07	192
Size of z3 form.	5180	6877	551447	44643959
Time to solve z3 form.	$0,10$	$35,38$	180	–
Reduction FBV				
Time to build z3 form.	0.1	0.05	0.95	195
Size of z3 form.	6285	8331	726681	59447253
Time to solve z3 form.	$0,35$	–	–	–

C.2 Nim game

	$[2, 2]$	$[3, 2]$	$[4, 5, 2]$	$[3, 4, 5]$	$[2, 3, 4, 4]$	$[5, 4, 3, 6]$	$[2, 4, 8, 14]$
# states	16	31	280	328	398	1595	13556
res	unsat	sat	sat	sat	sat	sat	unsat
Reduction UU							
Time to build z3 form.	$0,00$	$0,00$	$0,09$	$0,31$	$8,99$	–	–
Size of z3 form.	32	105	90095	285505	1011324	–	–
Time to solve z3 form.	$0,02$	$0,03$	$0,09$	$0,10$	$0,39$	–	–
Reduction FP							
Time to build z3 form.	$0,00$	$0,00$	$0,00$	$0,00$	$0,00$	$0,19$	$0,21$
Size of z3 form.	104	216	2283	2698	3225	13865	125486
Time to solve z3 form.	$0,01$	$0,02$	$0,04$	$0,4$	$0,06$	$0,15$	$62,2$
Reduction FFP							
Time to build z3 form.	$0,00$	$0,00$	$0,03$	$0,05$	$0,07$	$0,23$	$6,95$
Size of z3 form.	161	326	3323	3926	4703	19928	178158
Time to solve z3 form.	$0,02$	$0,03$	$0,12$	$0,10$	$0,18$	$0,18$	$26,66$
Reduction FBV							
Time to build z3 form.	$0,00$	$0,02$	$0,79$	$1,05$	$1,88$	$24,18$	–
Size of z3 form.	2468	9371	844412	1171832	1696982	29013387	–
Time to solve z3 form.	$0,05$	$0,12$	$207,6$	$341,6$	$879,8$	–	
Reduction FBV with bounded Until							
Bound for Until:	10	20	20	20	21	28	43
Time to build z3 form.	0	$0,01$	$0,08$	$0,1$	$0,1$	$0,48$	$6,43$
Size of z3 form.	1682	6291	65712	77816	97371	540997	7311052
Time to solve z3 form.	$0,02$	$0,1$	$2,37$	$3,17$	$4,20$	$184,94$	–

C.3 Resources distribution

$n \times m$	10×5	10×5	10×5	10×7	10×10	10×10
k, d	$2, 8$	$4, 4$	$4, 6$	$4, 5$	$2, 3$	$4, 7$
# states	50	50	50	70	100	100
res	sat	unsat	sat	unsat	unsat	?
Reduction UU − FP						
Time to build z3 form.	$0, 03$	$0, 08$	$0, 07$	$0, 10$	$0, 13$	$0, 22$
Size of z3 form.	112155	32357	70757	74417	45905	283907
Time to solve z3 form.	$0, 08$	$274, 00$	$0, 06$	–	$0, 09$	–
Reduction FFP						
Time to build z3 form.	$0, 05$	$0, 08$	$0, 10$	$0, 10$	$0, 17$	$0, 19$
Size of z3 form.	14515	23213	24715	44744	43510	90916
Time to solve z3 form.	$0, 06$	$308, 48$	$0, 09$	–	$0, 10$	–
Reduction FBV						
Time to build z3 form.	$0, 08$	$0, 02$	$0, 03$	$0, 03$	$0, 02$	$0, 09$
Size of z3 form.	9521	7633	11133	13123	7521	25733
Time to solve z3 form.	$0, 05$	$27, 16$	$0, 02$	155	$0, 09$	–

Towards Certified Model Checking for PLTL Using One-Pass Tableaux

Alex Abuin
Ikerlan Technology Research Centre,
P. J. M. Arizmendiarrieta, 2 20500 Arrasate-Mondragón Gipuzkoa, Spain
aabuin@ikerlan.es

Alexander Bolotov
University of Westminster,
W1W 6UW, London, UK
https://www.westminster.ac.uk/about-us/our-people/directory/bolotov-alexander
A.Bolotov@westminster.ac.uk

Unai Díaz de Cerio
Ikerlan Technology Research Centre,
P. J. M. Arizmendiarrieta, 2 20500 Arrasate-Mondragón Gipuzkoa, Spain
udiazcerio@ikerlan.es

Montserrat Hermo
University of the Basque Country,
P. Manuel de Lardizabal, 1, 20018-San Sebastián, Spain
montserrat.hermo@ehu.es

Paqui Lucio
University of the Basque Country,
P. Manuel de Lardizabal, 1, 20018-San Sebastián, Spain
http://www.sc.ehu.es/paqui
paqui.lucio@ehu.eus

Abstract

The standard model checking setup analyses whether the given system specification satisfies a dedicated temporal property of the system, providing a positive answer here or a counter-example. At the same time, it is often useful to have an explicit proof that certifies the satisfiability. This is exactly what the *certified model checking (CMC)* has been introduced for. The paper argues that one-pass (context-based) tableau for PLTL can be efficiently used in the CMC setting, emphasising the following two advantages of this technique. First, the use of the context in which the eventualities occur, forces them to fulfil as soon as possible. Second, a dual to the tableau sequent calculus can be used to formalise the certificates. The combination of the one-pass tableau and the dual sequent calculus enables us to provide not only counter-examples for unsatisfied properties, but also proofs for satisfied properties that can be checked in a proof assistant. In addition, the construction of the tableau is enriched by an embedded solver, to which we dedicate those (propositional) computational tasks that are costly for the tableaux rules applied solely. The combination of the above techniques is particularly helpful to reason about large (system) specifications.

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

Keywords and phrases Temporal logic, fairness, expressiveness, linear-time, Certified model checking

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.12

Funding *Alexander Bolotov*: This author has been partially supported by the UK Knowledge Transfer Partnership KTP011063 Lumina Learning & University of Westminster, and the Spanish Project TIN2017- 86727-C2-2-R, and by the University of the Basque Country under Project LoRea GIU18-182.
Montserrat Hermo: This author has been partially supported by Spanish Project TIN2017-86727-C2-2-R, and by the University of the Basque Country under Project LoRea GIU18-182.
Paqui Lucio: This author has been partially supported by Spanish Project TIN2017-86727-C2-2-R, and by the University of the Basque Country under Project LoRea GIU18-182.

1 Introduction and Problem Setup

Model Checking [10, 31, 11] is an algorithmic method for determining whether a complex hardware or software system satisfies a given property. Many important properties to be verified reflect the system's dynamics and are expressed in some temporal logic. If the property does not hold, the checker returns a counter-example: a trace/model of the system that does not satisfy the property. This counter-model acts as a "certificate" of the failure and its role is to help the user to identify the source of the problem which could be in the system design, in the property, and even in the model checker. However, there are a number of scenarios where another certification is needed. One of these scenarios consists on proving, only once, the correction of the underlying algorithm of the model checker. For this task, interactive proof assistants such as Coq or Isabelle are good tools. They allow us to certify a model checker and even obtain an executable program by the refinement of some extraction mechanism. For instance Amjad [1] described how to code BDD-based symbolic model checking algorithms into an automatic theorem prover. More recently, Esparza et al. [13] have verified an automata-based model checker with Isabelle theorem prover. The second scenario involves the model checker to certify that a particular property is true. For example, the user may deal with a very complex specification and is not sure if the specification is well written. Here it is important to have techniques that provide not only a counter-example, but also certify that the system meets the property. This is exactly what the *certified model checking (CMC)* has been introduced for. In this second scenario a number of techniques have been previously proposed. Some of the techuiques that deal with finite-state systems can be found in [23, 29]. In [23] an automata-theoretic approach to model checking is addressed. In [29] a deductive proof system was introduced for verifying branching time properties expressed in the mu-calculus. For infinite-state systems, Mebsout et al. [28] recently presented a new technique for generating and verifying proof certified in SMT-based model checkers, focusing on proofs of invariant properties. The use of invariants has been exploited in [17, 22], where the proof is generated from the inductive invariant obtained with the k-liveness algorithm [9]. The resulting approach can be implemented as a model checker based on the combination of k-liveness with an engine for invariant properties that is capable of producing inductive invariants. A drawback of this approach is that, although it is very competitive, the task of finding counter-examples and the task of generating proofs (in this case via finding invariants), are very different requiring for the latter the addition of extra mechanisms to the own model checker.

In this paper, we propose an CMC based on dual systems of tableaux and sequent calculus, originally introduced in [14, 15]. It produces certificate proofs, formal proofs in the sequent calculus, and counterexamples - open branches in the tableau. This is also one of the main advantages of our approach: the same reasoning mechanism applies for both tasks - certificates and counterexamples.

The CMC (see Figure 1) differs from the traditional model checking in providing a proof of the satisfiability of the given property, and not only a counter-example. Incorporating the notation of [21] (and slightly modifying it) we represent the methodology of the certified model checking as the following signature:

$$\mathcal{CMC} :: System \times \varphi \longrightarrow \mathbb{B} \times (Proof \mid Counter\text{-}example)$$

where, given a specification of a system, S, and a property, φ, a certified model checking produces a Boolean result, $\mathbb{B}$, indicating whether S satisfies φ, along with

- a proof (or certification), in the positive case, or
- a counter-example, in the negative case.

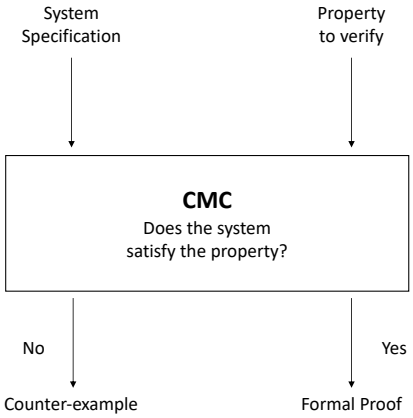


Figure 1 General schema of CMC.

However, we believe that to take the full advantage of CMC, and enabling its industrial application to real systems, we need to ensure that CMC meets the following requirements.

- **(i)** Proofs should be generated automatically.
- **(ii)** An CMC needs to offer a CMC user sufficient information to understand the proof without additional "costs" related to specialist knowledge of the underlying proof technique.
- **(iii)** The presentation of the proof should enable the CMC users to easily navigate through its trace. This becomes particularly important when the system is badly defined – here the navigation through the trace can help to detect errors.
- **(iv)** Finally, when developing a safety critical system following a safety process (e.g. processes of standard ISO26262 [19], IEC61508 [18] or EN50128 [7]), it is mandatory to analyse how a bug in a tool (a model checker in our case) may affect the safety of the developed system. Depending on the level of these effects, some actions are required to increase our confidence in the model checker. One of the mechanisms to increase the probability of finding a failure is to re-evaluate the outputs of the CMC by an independent tool developed by an independent team.

We propose a particular CMC method of realising the CMC philosophy (see Figure 2) meeting the characteristics (i)-(iv) above while maintaining the common (for the traditional Model Checker) functionality. Our method is centered on a context-based temporal one-pass tableaux technique whose performance is optimised by a SAT solver.

Various tableaux techniques have been proposed for a rich variety of temporal logics, linear and branching: *Propositional Linear-Time Temporal Logic* (PLTL); *Computation Tree Logic* (CTL); CTL* which generalizes PLTL and CTL, etc. (an excellent survey can be found in [16]). One of the core ideas of the tableaux methods is to identify eventualities within the given temporal input and to check that they are fulfilled. Traditional tableaux techniques require two phases to perform this test. In the first phase, a graph which gives all possible pre-models for the tableau input, in constructed. In the second phase, for each state, s in this graph, that contains some "eventually φ", a graph-theoretic algorithm looks for a state, reachable from s, that satisfies φ. Note also that the two-pass tableaux methods fail to maintain the classical correspondence between tableaux and sequents that associates a sequent proof with the closed tableau.

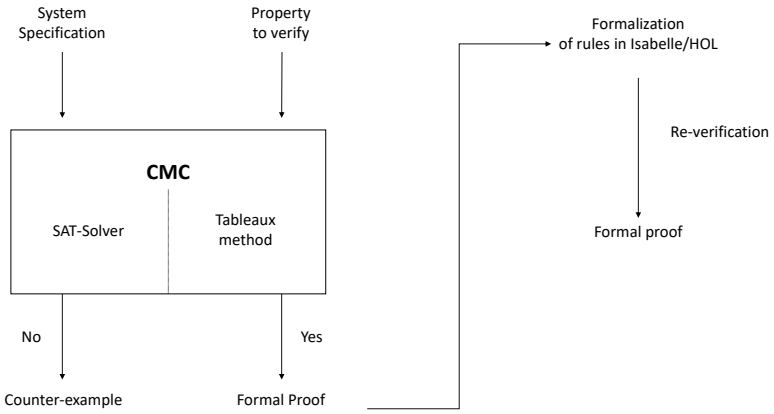


Figure 2 General schema of the proposal.

To avoid the second phase and, hence, to keep the ability of generating (sequent) proofs from tableaux, in [14, 15], dual systems of tableaux and sequents, for PLTL, were presented. Every logic defined in [14, 15] was proved to be sound and complete. In particular, [15] contains a proof that the tableau system we use in the present paper, is a decision method for the full PLTL, i.e. it is sound, refutationally complete, and terminating. The termination property is achieved on the basis of any fair selection strategy. A very similar sequent calculus is presented in [6] (see [15] for more details about the similarities and differences of these systems). The one-pass tableau method [15] has been extended to a concurrent constraint logic in [12], and also to ECTL$^\sharp$ - a branching-time sublogic of CTL* in [5].

The tableau method in [15] makes use of the so-called *context* of an eventuality to force its fulfillment. The context of an eventuality is simply the set of formulae that "accompanies" the eventuality in the label of the node. When a one-pass tableau checks whether a property φ holds or not, it is always able to issue a certificate: either a counter-model or a complete explanation (formal proof) of why φ is true.

We abbreviate the one-pass tableau method as $\tau^{\uparrow}_{pltl}$ and its dual sequent calculus as TTC.

Considering one-pass tableau method as one of the core components of the CMC solution we present in this paper, we argue that it conforms with the properties (i)-(iv) mentioned above. Indeed, the method automatically checks whether the specification of the system satisfies the property (i).[1] If it does not then it provides a counter-example. A counter-example is given by a trace which is intuitively clear (ii). In the case of a positive answer, a relevant proof certifies it. Again, the output proof is represented as a trace enabling an easy navigation through it (iii). Moreover, one of the attractive features of our method is that it forces the eventualities to be fulfilled as soon as possible, thus, the method will potentially generate shorter paths (with fewer nodes) than those produced by non-context based tableaux.

In our proposal, the performance of the one-pass tableau technique is complemented, for efficiency, by the SAT solving. Note that the idea of encoding of transition systems into propositional SAT was first proposed in [20] for AI planning problems, where the authors show that SAT algorithms scale much better on the SAT-encodings than planing algorithms

[1] For the purposes of the paper, we let the specification, S, be in a dedicated form $S = \mathit{Init} \wedge TR$, where Init is a PLTL formula that represents the initial states and TR is the representation of all the transitions allowed, see §3 for details.

on the original graph formulation. Following this success, SAT solvers have been also used in *Bounded Model Checking* (BMC). In both frameworks, a propositional formula is used to encode the input problem. Planning problems deal with finite paths along a finite graph, hence the encoding is complete w.r.t. the original problem. In model checking, paths within the transition systems are, in general, infinite. SAT-based BMC utilises the encoding of the model-checking problems in satisfiability checking, more precisely, the propositional encoding of statements expressing that there exists a length-k path (along the transition system) that does not satisfy a given property. The BMC increases k until either the SAT's answer is "yes" (i.e. a counter-example is found), or the search becomes intractable, or k reaches a certain bound. The original SAT-based BMC algorithm [3], although complete for finite state, is limited in practice to falsification. Many additional strategies have been introduced to make BCM complete, see [4] for a good survey. There is also a large amount of work, starting with [32], on using SAT solving for improving the satisfiability test of the full PLTL. Recent papers [24, 25] use SAT solvers to seek for a model for an input formula. This model is essentially a graph/automaton produced by the first pass of a two-pass tableau method, which should be followed for testing the fulfilment of eventualities. SAT solvers are called for the generation of all (different) successors of every state in the graph. The authors use well-known temporal equivalences (like $p\mathcal{U}q \equiv (q \vee (p \wedge \circ(p\mathcal{U}q)))$ to compute the successors of the given state. Here, the method utilises the renaming of subformulae containing temporal modalities (such as $p\mathcal{U}q$) by fresh propositional variables and utilise the SAT solving to calculate different successor states of each state in the transition system. This prevents the repeated generation of "propositionally equivalent" states. The relevant heuristics for pruning the search-space and on-the-fly mechanism for testing eventuality fulfilment are introduced in the implementation.

Our proposal uses SAT solvers in a similar way, but is very different in the primary goals. In our method, SAT solvers are employed to calculate a set of the "next" states in the tableau: being in the "current state", SAT solver calculates those successor states in the transition systems that satisfy the negation of the tested property. For that, we do not rename all temporal modalities in the label of the tableau node, but only those of the form $\circ\varphi$, where φ only contains classical operators. Moreover, our proposal of using the one-pass tableau technique (helped, for efficiency, by the SAT solver) is complete for deciding (unbounded) model checking problems and works on infinite traces. The most remarkable difference of our proposal (regarding [24, 25]) is related to the fact that we pursue CMC, i.e. we would like to generate proofs in a calculus for PLTL, hence we use the one-pass context-based tableau along with its dual sequent calculus. In addition, the context-based tableau is particularly well suited for dealing with the specifications of transition systems ("always"-formulae); the context plays the role "of forcing eventualities to be fulfilled as soon as possible" and acts as a semantic constraint that prevents the generation of several states (which are generated in the two-pass approach).

In building the proof, we invoke Isabelle/HOL [30] to verify the construction of the tableaux in order to avoid possible errors caused by the implementation (iv). This external validation is carried out by a sequent calculus TTC which is formalized in Isabelle/HOL and is dual to the one-pass tableau method. We also note that the same reasoning mechanism based on the one-pass tableau is applied in our approach to counter-examples and proofs which makes the tool easy to understand. This facilitates the industrial spreading of CMC.

The remaining of the paper is organised as follows. In §2 we review the technique to construct a one-pass tableau $\tau^{\uparrow}_{pltl}$. In §3 we present one-pass tableau based model checking. This follows by the description of the certification utilising Isabelle in §4. Finally, in §5 we summarise the results and provide an account of future work.

2 One-pass Context-based Tableau

To make the paper self-contained and easier to read, in this section we first recall the syntax and semantics of the underlying logic, PLTL and then we will review the one-pass tableau method.

2.1 Syntax and Semantics of PLTL

▶ **Definition 1** (PLTL Language). *The language of PLTL comprises*

- *A set, Prop, of propositional symbols.*
- *Propositional connectives* $\neg, \wedge, \vee$*, and constants* $\mathbf{T}$ *and* $\mathbf{F}$
- *Future-time temporal connectives, "*$\Box$*" (always), "*$\Diamond$*" (eventually) "*$\circ$*" (at the next moment in time), "*$\mathcal{U}$*" (until), and "*$\mathcal{R}$*" (release).*

▶ **Definition 2** ($\mathsf{WFF}_{\mathsf{PLTL}}$). *The set of well-formed formulae of PLTL, denoted by* WFF_{PLTL}*, is inductively defined as the smallest set satisfying the following.*

- *Any element of Prop,* $\mathbf{T}$ *and* $\mathbf{F}$ *are in* WFF_{PLTL}*.*
- *If A and B are in* WFF_{PLTL} *then so are* $\neg A, A \wedge B, A \vee B, \Box A, \Diamond A, A\mathcal{U}B$*, and* $A\mathcal{R}B$*.*

A literal is either a propositional symbol (a positive literal) or the negation of a propositional symbol (a negative literal).

Definition 1 introduces PLTL with the set of temporal operators that are convenient for our representation in the paper. We note that this set is richer than $\{\mathcal{U}, \circ\}$ which is known to be sufficient to represent all other linear-time temporal operators. For example, '$\Diamond\varphi$' can be defined as $\mathbf{T}\mathcal{U}\varphi$ while $\varphi\mathcal{R}\psi$ can be defined via $\mathcal{U}$ as $\neg(\neg\varphi\mathcal{U}\neg\psi)$ (here and in the remaining of the paper φ and ψ are meta-symbols denoting PLTL formulae).

Formulae of the type $\Diamond\varphi$ and $\varphi\mathcal{U}\psi$ are called *eventualities*, and formulae of the type $\Box\varphi$ are called *always-formulae.*

A model, $\mathcal{M} = s_0, s_1, s_2, s_3, \ldots$, for PLTL formulae is a discrete, linear sequence of states, isomorphic to natural numbers, $\mathbb{N}$. Each state, $s_i, 0 \leq i$, is a set of positive literals, which are satisfied at the i-th moment of time. We write $\langle \mathcal{M}, i\rangle \models \varphi$ to indicate that φ is true in the model $\mathcal{M}$ at the (state) index $i \in N$.

Below we inductively define the relation $\models$ which evaluates PLTL formulae in a model M at i-th moment of time. Note that here we follow so called "anchored" version of PLTL that defines the PLTL validity and satisfiability (see below) at the "beginning" of time, the initial state, s_0 of a model.

- $\langle \mathcal{M}, i\rangle \models \mathbf{T}$
- $\langle \mathcal{M}, i\rangle \not\models \mathbf{F}$
- $\langle \mathcal{M}, i\rangle \models \neg\varphi$ iff $\langle \mathcal{M}, i\rangle \not\models \varphi$
- $\langle \mathcal{M}, i\rangle \models \varphi \wedge \psi$ iff $\langle \mathcal{M}, i\rangle \models \varphi$ and $\langle \mathcal{M}, i\rangle \models \psi$
- $\langle \mathcal{M}, i\rangle \models \varphi \vee \psi$ iff $\langle \mathcal{M}, i\rangle \models \varphi$ or $\langle \mathcal{M}, i\rangle \models \psi$
- $\langle \mathcal{M}, i\rangle \models \circ\varphi$ iff $\langle \mathcal{M}, i+1\rangle \models \varphi$
- $\langle \mathcal{M}, i\rangle \models \Box\varphi$ iff $\langle \mathcal{M}, j\rangle \models \varphi$ for every $j \geq i$.
- $\langle \mathcal{M}, i\rangle \models \Diamond\varphi$ iff there exists $j \geq i$ such that $\langle \mathcal{M}, j\rangle \models \varphi$.
- $\langle \mathcal{M}, i\rangle \models \varphi\mathcal{U}\psi$ iff there exists $j \geq i$ such that $\langle \mathcal{M}, j\rangle \models \psi$ and for every k, $i \leq k < j$, we have $\langle \mathcal{M}, k\rangle \models \varphi$.
- $\langle \mathcal{M}, i\rangle \models \varphi\mathcal{R}\psi$ iff for every $j \geq i$, either $\langle \mathcal{M}, j\rangle \models \psi$ or there exists k such that $i \leq k < j$ and $\langle \mathcal{M}, k\rangle \models \varphi$.

▶ **Definition 3** (PLTL satisfiability, validity). *If, for a formula φ, there exists a model, M, such that $(M,0) \models \varphi$ then φ is satisfiable. Formula φ is called valid if it is satisfiable in all models.*

In the remaining of the paper we will use capital letters $\Phi, \Psi, \Delta, \ldots$ to denote sets of PLTL formulae. The semantics above can be extended to sets of formulae in the standard way: given a set of PLTL formulae $\Phi = \gamma_1, \gamma_2, \ldots \gamma_n$, the following holds: $\langle \mathcal{M}, i\rangle \models \Phi$ iff $\langle \mathcal{M}, i\rangle \models \gamma_k$, for all $k,\ 1 \leq k \leq n$.

2.2 Useful PLTL properties

In this section we recall those PLTL syntactic and semantic properties that are useful for our tableau construction. First, the tableau procedure will take as an input PLTL formulae converted to their negated normal forms (NNF).

▶ **Definition 4** (Procedure for obtaining NNF). *For a given PLTL formula, φ, push the negations in φ inward until they are applied only to propositions. This involves applying the standard set of rewrite rules used to obtain NNF in classical logic (including $\neg\mathbf{T} \longrightarrow \mathbf{F}$ and $\neg\mathbf{F} \longrightarrow \mathbf{T}$) with the additional transformations for temporal operators:*

$$\neg\bigcirc\varphi \longrightarrow \bigcirc\neg\varphi \qquad \neg\Box\varphi \longrightarrow \Diamond\neg\varphi \qquad \neg\Diamond\varphi \longrightarrow \Box\neg\varphi$$
$$\neg(\varphi\mathcal{U}\psi) \longrightarrow \neg\varphi\mathcal{R}\neg\psi \qquad \neg(\varphi\mathcal{R}\psi) \longrightarrow \neg\varphi\mathcal{U}\neg\psi$$

The following result [26] can be easily established.

▶ **Proposition 5** (Translation into NNF preserves satisfiability). *For any* PLTL *formula φ, the following holds* $\langle \mathcal{M}, 0\rangle \models \varphi$ *iff* $\langle \mathcal{M}, 0\rangle \models \mathit{NNF}(\varphi)$.

As a simple example, $\mathsf{NNF}(\neg\bigcirc\Box\neg a) = \bigcirc\Diamond a$. We will utilise this in §3.

In what follows we deal with sets of formulae in NNF. Literals and formulae in NNF of the form $\mathbf{F}$, $\mathbf{T}$ and $\bigcirc\varphi$ are called *elementary*, the remaining formulae are subsequently called *non-elementary*. In addition, sets of elementary formulae are also called elementary.

▶ **Definition 6** (Consistent set of PLTL formulae in NNF). *A set of PLTL formulae in NNF is consistent if it does neither contain $\mathbf{F}$, nor $\{\varphi, \mathit{NNF}(\neg\varphi)\}$ for any formula φ. Otherwise it is called inconsistent.*

Note that the above notion of consistency is syntactic. To check whether $\{\varphi, \psi\}$ is inconsistent we test if $\varphi = \mathsf{NNF}(\neg\psi)$. The cost of this check is linear on the length of the formula.

Since our transition system specifications are given by sets of PLTL formulae and PLTL has the *finite model property* [33] we can consider its interpretation over *cyclic* structures. Noting that an infinite sequence $s_0, s_1, \ldots, s_k, \ldots$ induces the successor relation, R, such that $(s_i, s_{i+1}) \in R$ for all $i \in \pi$, we define below the notions of a cyclic sequence, cyclic path and cyclic model.

▶ **Definition 7** (Cyclic Sequence, Cyclic Path). *Let π be a finite sequence of states $\pi = s_0, s_1, \ldots, s_j$. Then*

- *π is cyclic iff there exists $s_i,\ 0 \leq i \leq j$ such that $(s_j, s_i) \in R$.*
- *A sequence $s_i, \ldots, s_j$ is a loop with a cycling element s_i abbreviated as $\langle s_i, \ldots, s_j\rangle^\omega$.*
- *A cyclic path over a cyclic sequence π is an infinite sequence $\xi(\pi) = s_0, s_1, \ldots, s_{i-1}\langle s_i, s_{i+1}, \ldots, s_j\rangle^\omega$.*

▶ **Definition 8** (Cyclic Model). *A model $\mathcal{M}$ is cyclic if it is a cyclic path.*

Now, assuming that PLTL formulae are interpreted over cyclic models, we overview the construction of the one-pass tableau $\tau^{\uparrow}_{pltl}$ applied to some input Σ, in symbols $\tau^{\uparrow}_{pltl}(\Sigma)$, adapting the technique developed in [15]. We will see, in the subsequent sections, the benefits of its main feature – checking the validity of the given input in "one pass", without the second "pass" where an auxiliary graph is built to check if all the eventualities are satisfied or not.

▶ **Definition 9** (Tableau, Consistent Node, Closed branch). *A one-pass tableau for a set of formulae Σ, abbreviated as $\tau^{\uparrow}_{pltl}(\Sigma)$ is a labelled tree T, where nodes are labeled with sets of formulae, such that the following two conditions hold:*

(a) *The root is labelled by the tableau input, Σ.*

(b) *Any other node, m, is labelled with sets of formulae as the result of the application of one of the expansion rules to the parent node, n.*

A node $n \in T$ is consistent, abbreviated as $n_{\top}$, if its label is a consistent set of formulae (see Def. 6), else n is inconsistent, abbreviated as $n_{\bot}$.

If a branch, b, of T, contains an inconsistent node $n_{\bot}$, then b is closed, else b is open.

Informal Introduction to one-pass tableau. In the set of expansion rules, on the top of the standard $\alpha - \beta$ rules, we also have β^{+} rules that are characteristic (and crucial!) for our construction. These rules (which were originally introduced in [14, 15]) reflect our dedicated account of the eventualities, namely, we treat an eventuality as occurring in some *context*. By the context of the selected eventuality, we understand a collection of all other formulae within the label of the node. Subsequently, β^{+} rules use the context to force eventualities to be fulfilled as soon as possible. The tableau expansion rules apply repeatedly until they produce an inconsistent node, $n_{\bot}$, or a node with the labels that already occurred within the given path. In the former case the expansion of the given branch terminates with $n_{\bot}$ as a leaf. In the latter case, a repetitive node in the branch witnesses that the branch is open. Once no more expansion ($\alpha - \beta, \beta^{+}$ type) rules are applicable to the given branch with the last consistent node $n_{\top}$, the expansion rules ensure that its labelling is similar to a "state" in the standard temporal tableau. Then the "next-state" rule applies which generates successors with the labels that are arguments of all $\circ$ modalities and the whole cycle of applying the expansion and the "next-state" rules is repeated until the tableau construction terminates. The nature of our rules ensures that the terminated tableau is either closed, indicating that the input does not have a model, hence unsatisfiable, or open, indicating a model for the tableau input.

2.3 $\tau^{\uparrow}_{pltl}$ Rules

Recall that all formulae in the input of the tableau have been already transformed into their NNF. Presenting α-, β- and "next-state" rules for the construction of the semantic tableaux, we assume that these apply respectively, to α-formulae, β-formulae and "next"-formulae such that α_1 denotes the set of formulae in the conclusion of an α-rule, while β_1, β_2 denote the sets of formulae in the alternative conclusions of a β-rule, and γ_1 denotes the result of "jumping" from a state to a pre-state.

The sets of $\alpha - \beta$ rules are given in Table 1. These are standard in temporal tableaux construction (see, for instance, [2]).

Table 1 $\alpha - \beta$-rules.

	α	α_1
$(\wedge)$	$\varphi \wedge \psi$	φ, ψ
$(\Box)$	$\Box\varphi$	$\varphi, \circ\Box\varphi$

	β	β_1	β_2
$(\mathcal{U})$	$\varphi\mathcal{U}\psi$	ψ	$\varphi, \circ(\varphi\mathcal{U}\psi)$
$(\mathcal{R})$	$\varphi\mathcal{R}\psi$	φ, ψ	$\psi, \circ(\varphi\mathcal{R}\psi)$
$(\vee)$	$\varphi \vee \psi$	φ	ψ
$(\Diamond)$	$\Diamond\varphi$	φ	$\circ\Diamond\varphi$

Table 2 β^+-rules, where

- Δ is a (possibly empty) set (conjunction) of formulae,
- If $\Delta \neq \emptyset$ then Δ' is a set (conjunction) of all elements of Δ except for $\Box$ -formulae and $\neg\Delta'$ is the disjunction of all negated elements of Δ', else $\neg\Delta'$ is $\mathbf{F}$.

	β	β_1	β_2
$(\mathcal{U})^+$	$\Delta, \varphi\mathcal{U}\psi$	Δ, ψ	$\Delta, \varphi, \circ((\varphi \wedge (\mathsf{NNF}(\neg\Delta')))\mathcal{U}\psi)$
$(\Diamond)^+$	$\Delta, \Diamond\varphi$	Δ, φ	$\Delta, \circ((\mathsf{NNF}(\neg\Delta'))\mathcal{U}\varphi)$

β^+ rules (see Table 2) are crucial in the construction of $\tau^{\uparrow}_{pltl}$ as they use the so-called *context*, Δ, to force the eventuality $\varphi\mathcal{U}\psi$ to be fulfilled as soon as possible. We illustrate this concept on the $(\mathcal{U})^+$ rule. When $(\mathcal{U})^+$ is applied to a node labelled by a set of formulae $\{\Delta, \varphi\mathcal{U}\psi\}$, then the context is Δ and the resulting labelling of the next node contains the formula $(\varphi \wedge \neg\Delta)\mathcal{U}\psi$, where $\neg\Delta$ means the disjunction of all negated elements of Δ. Therefore, if ψ is not satisfied, then $\neg\Delta$ also belongs to the label of that node. This means that the context, Δ, of the previous label is not repeated. As Δ is a finite set/conjunction of formulae and $\neg\Delta$ is the finite disjunction of the negations of the formulae in Δ, the $(\mathcal{U})^+$ rule forces at least one formula in Δ to be falsified during the transition from the previous node to the subsequent one, whenever ψ is not satisfied. Note that the $(\mathcal{U})^+$ rule only applies to some selected eventuality, and in this sense, the unique eventuality, which becomes marked. Each application of the $(\mathcal{U})^+$ rule to the set $\{\Delta, \varphi\mathcal{U}\psi\}$ introduces the so-called *next-step variant* $(\varphi \wedge (\mathsf{NNF}(\neg\Delta'))\mathcal{U}\psi$ where Δ', as we know, is a conjunction of all elements of Δ except for $\Box$ -formulae, which keeps the mark for the selected eventuality. Note that any formula of the type $\Box\varphi$, which was a member of Δ, will be repeated forever and if we keep it in Δ' it would appear in the resulting $\mathsf{NNF}(\neg\Delta')$ as a disjunct $\Diamond\neg\varphi$ which would never be satisfied. Hence, any $\Box$- formula can be immediately dropped when we form Δ'. For the soundness of the construction, each node of the tableau must have at most one marked eventuality, i.e. the one to which the $(\mathcal{U})^+$ has been applied. Note that when a node of the tableau does not contain any marked eventuality, then one of them is randomly marked.

The $(\mathcal{U})^+$ rule allows us to avoid the construction of the auxiliary graph of stages (the second pass in two-pass tableau methods) which is used to determine whether all eventualities are satisfied or not.

The other β^+ rule, $(\Diamond)^+$, is obviously derivable from the $(\mathcal{U})^+$ rule. However, we present it as part of the tableau as its application makes proofs more transparent to the reader and the user of the tableau as a model checker. We will explain this in the subsequent sections of the paper. It is worth noting that, because in the β^+ rules, $\neg\Delta' = \mathbf{F}$ whenever $\Delta = \emptyset$, the β rules $(\mathcal{U})$ and $(\Diamond)$ become particular cases of β^+ rules $(\mathcal{U})^+$ and $(\Diamond)^+$ when $\Delta = \emptyset$. In this case the β_2 child of the $(\mathcal{U})^+$ rule is reduced to $\varphi, \circ(\mathbf{F}\mathcal{U}\psi)$, from which we can derive the β_2 child of the $(\mathcal{U})$ rule.

Table 3 The next-state rule (Σ_1 is a set of literals).

	γ	γ_1
$(\circ)$	$\Sigma_1, \circ(\Sigma_2)$	Σ_2

For the formulation of the next-state rule $(\circ)$, we first introduce the following notation. Given a set of formulae, Φ, we denote by $\circ(\Phi)$ the set $\{\circ\varphi \mid \varphi \in \Phi\}$. The next-state rule (see Table 3), is applied to jump from a node labelled by $\Sigma_1, \circ(\Sigma_2)$ to a new state, which is labeled by the set Σ_2

3 One-pass tableau based Model Checking

In this section we explain how the model checking can be performed using the one-pass tableau method described in §2. First, we define the procedure to follow, then we explain the optimization we pursue by embedding a (propositional) solver; finally, we present an example, showing how beneficial it can be for the users of CMC to obtain an explicit and understandable formal proof.

Our model checker receives as its input a specification, S, of the given transition system, and the given property P, both written in the PLTL language. Whereas P is any PLTL formula, transition system specifications are restricted to a sublanguage. Our specification language is inspired in the so-called *constraint style* specification language used in the well-known model checker NuSMV (introduced in [8]). The specification S consists of a set *Init* that is a non-temporal (or classical) formula, and a conjunction (set) TR of formulae of the form $\Box\rho$ where ρ is a boolean combination of literals and $\circ\ell$ where ℓ is a literal. $S = \mathit{Init} \wedge TR$ is the system specification such that

- A state is initial if and only if it satisfies the formula *Init*, and
- Any pair "(current state, next state)" is in the transition relation if, and only if, it satisfies TR.

In Example 10 we provide the *Init* and TR formulae that represent a specific transition system. Then, given a specification S and a property P, the model checker decides whether any model of S satisfies P, by deciding if $S \cup \{\neg P\}$ is unsatisfiable or not. For that, S and $\neg P$ are firstly converted into NNF. The tableau method $\tau^{\uparrow}_{pltl}$ is suitable for deciding the (un)satisfiability problem $\mathsf{NNF}(S \cup \{\neg P\})$. It explicitly tries to generate a model of $\mathsf{NNF}(S \cup \{\neg P\})$. The results are interpreted as follows:

- If a cycle is found in a tableau branch, this branch is open, and represents a model that satisfies the set of formulae with which the tableau has been called. Consequently, this is a counter-model proving that the system S does not satisfy the property P.
- If the tableaux closes, i.e all its leaves are inconsistent sets, it means that the tableaux input is unsatisfiable, hence, all models of S satisfy the property P.

This way of performing model checking is a particular case of the method described in Section 2. It brings us the following benefits: first of all, the tableau is built on-the-fly, allowing structures not to deal with eventualities fulfillment; due to the use of the context, the eventualities are satisfied as soon as possible. In terms of the implementation, all branches are completely independent so they can be parallelised without shared data. Finally, we have a potential memory improvement by only requiring to keep traces (of the branch that we deal at the moment).

On the other hand, a large proportion of the computational effort is spent on classical propositional reasoning. Since the specification, S, of the system is the most determining factor, this is especially inefficient when the specified system is large. However, S involves very

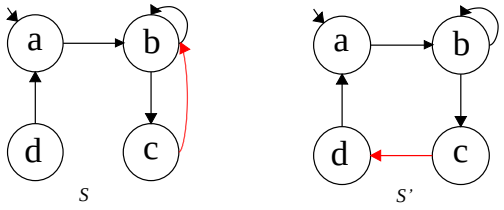


Figure 3 Example showing how the approach can help in detecting errors in system specification.

simple temporal formulae: always-formulae with arguments that are Boolean combinations of literals and literals preceded by "next". TR is a conjunction of the formulae of the form $\Box\rho$, and according to the tableau rules (and semantics), ρ and $\Box\rho$ are maintained in all states. Therefore, renaming in ρ the formulae $\circ\ell$ by fresh literals ℓ', we make ρ purely propositional. Hence, for a more efficient implementation of $\tau^{\uparrow}_{pltl}$, we propose to add a SAT solver to carry out the propositional reasoning in the tableau. Initially, we pass to the solver *Init* and the renamed formulae extracted from TR. Then, the tableau rules work on the temporal formulae (taken from the NNF of the negated property). Subsequently, in every node which is a "state" (i.e. a node labelled by an elementary set of formulae) the SAT solver is called adding to its input all formulae that appear new in the node and are either non-temporal (classical) formulae, or formulae of the type of the TR ones, but adequately translated. The SAT solver returns propositional models (atoms and atoms with prime apostrophe) defining all possible transitions to the next state that do not contradict (yet) $\neg P$. To get each of the next states, the variables are renamed back from ℓ' to $\circ\ell$ and the next-state rule is applied.

▶ **Example 10.** Let us introduce a running example (Figure 3), to illustrate how the tableau method works and the role of the solver, given the specification written in the form $Init \wedge TR$. The *Init* formula of both transition systems in Figure 3 is: $a \wedge \neg b \wedge \neg c \wedge \neg d$. Let us suppose that the transition system S' on the right-hand side of Figure 3 is the system we intend to specify, but a misprint in the system specification produces a wrong specification S in the left-hand side of Figure 3. That is, we mistakenly specify that there is a transition "from c to b" instead of the intended one: "from c to d". The resulting specification is:

$$\begin{aligned} TR = \{\ & \Box\,(a \rightarrow (\circ\neg a \wedge \circ b \wedge \circ\neg c \wedge \circ\neg d)),\\ & \Box\,(b \rightarrow ((\circ\neg a \wedge \circ b \wedge \circ\neg c \wedge \circ\neg d)\ \vee\ (\circ\neg a \wedge \circ\neg b \wedge \circ c \wedge \circ\neg d))),\\ & \Box\,(c \rightarrow (\circ\neg a \wedge \circ b \wedge \circ\neg c \wedge \circ\neg d)),\\ & \Box\,(d \rightarrow (\circ a \wedge \circ\neg b \wedge \circ\neg c \wedge \circ\neg d))\ \} \end{aligned}$$

Now, suppose that we want to check if S satisfies the PLTL formula $\circ\Box\neg a$. Converting its negation to NNF we get the formula $\circ\Diamond a$. Then, we call $\tau^{\uparrow}_{pltl}$ with the following input – the label of the initial node – $TR \cup \{Init, \circ\Diamond a\}$. Thinking on the system S', we expect to get a sequence $\langle a, b, c, d, a\rangle$ as a counter-example, since this sequence is a model of $\circ\Diamond a$ and the correct system S'. However, $S \cup \{\circ\Diamond a\}$ is unsatisfiable and our model checker generates a closed tableau for it.

In Figure 4 we depict a big-step version of that closed tableau. This one represents the different branches of the tableaux enabling an easy follow-up of the runs across the system S. In the rest of this section and in Section 4 we discuss the utility of the closed tableau and its dual sequent proof to find an error when defining S instead of the intended S'. The tableau root, in Figure 4, contains the $Init = a \wedge \neg b \wedge \neg c \wedge \neg d$, the negated property, $\circ\Diamond a$, and TR which represents the system's transitions. The tableau method applies its rules until a state

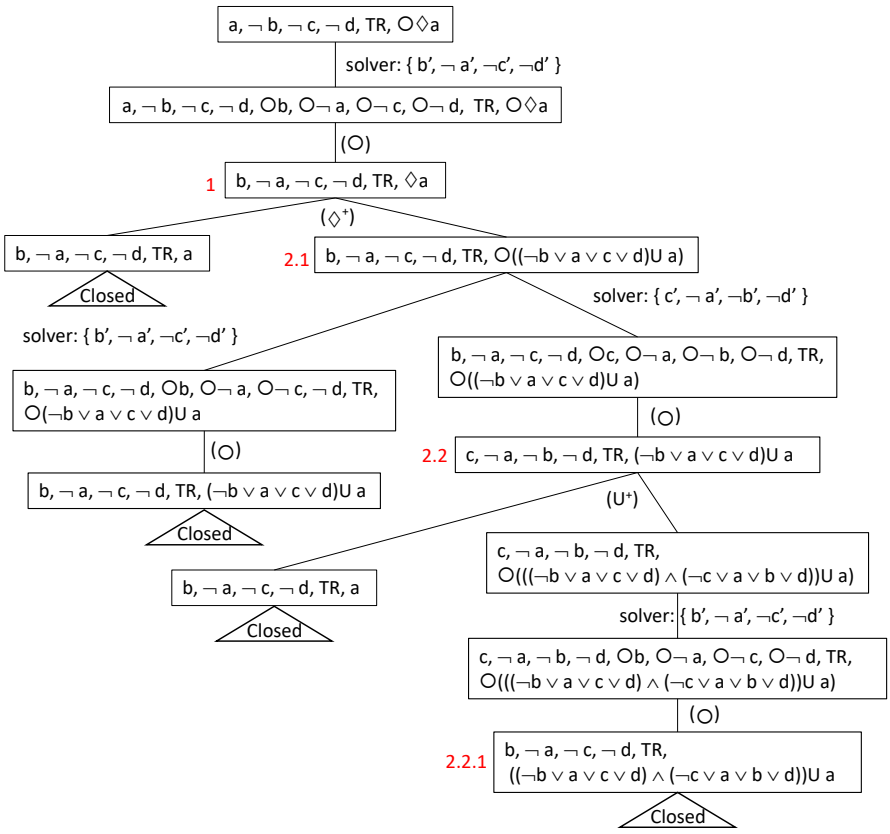


Figure 4 A big-step representation of the closed tableau for $S \cup \{\circ\Diamond a\}$.

is reached. It is now, when the solver is called to work with the propositional translation of TR and the remaining propositional formulae in the current node. This procedure gives us, one by one, the different models that satisfy the (translated) specification TR, and the current state (propositional formulae with which it is called). These models, (built for the formulae - results of the renaming of each "next" formula $\circ l$ by ℓ'), are transformed back to PLTL formulae by re-instantiating the renamed $\circ\ell$. Then, the next-state rule is applied and the tableaux continues working applying the temporal rules to the non-elementary temporal formulae (that come from the negated property). When the tableau is closed, a different model is supplied by the solver, if there is any. For example, for node 2.1 of the tableau, two different propositional models are supplied. First, the solver returns a model in which only b' (that is, $\circ b$) is true. It applies the next rule and all the subsequent branches close. Returning to 2.1., the solver supplies a new propositional model in which it is now c' (i.e. $\circ c$) that is true. In short, the solver is in charge of returning in each state all possible models of the next state (if any). In our running example, if users analyse the tableau they can see the reason for the property not to be fulfilled: the trace $\langle root, 2.1, 2.2, 2.2.1\rangle$ describes the transitions (omitting the negated literals) $a \Rightarrow b \Rightarrow c \Rightarrow b$. By comparing that trace with the intended specification (S') the user will be able to find an error.

4 Isabelle Proofs as User-checkable Certificates

We have codified a version of the calculus TTC in [15] (for formulae in NNF) as an Isabelle/HOL theory ([30]). The file `TTC_Calculus.thy` contains the encoding of the sequent rules that we will explain in this section, and another file `TTC_Soundness.thy` provides the soundness proof of them. On the top of this theory we define (for the running example) a transition system, S, as a collection, TR, of named formulae (in the fixed syntax). Thus, we introduce the (binary) transition relation between states and the formula *Init* specifying the initial states of the system. We automatically prove lemmas asserting that falsity $\mathbf{F}$ is derivable from the union of S and the negation (in NNF) of a fixed property φ. Such a proof object guarantees that φ is satisfied in every run of S, and can be independently and efficiently verified by the interactive theorem-prover Isabelle/HOL, providing a machine-checked certificate. Moreover, when the proof is unexpected, interactive theorem-provers enable to use this certificate to analyze the transition system seeking for specification errors. For that, we automatically generate an Isabelle proof that allows the user to check the successive subgoals of the proof. This Isabelle proof can be generated with different levels of granularity to facilitate different levels of analysis. In this section, we explain the main ingredients of this Isabelle/HOL development. The corresponding files can be download from `http://github.com/alexlesaka/OnePassTableau`.

We use a datatype PLTL_formula to define the syntax of the considered formulae, which are the two boolean constants ($\mathbf{T}$ and $\mathbf{F}$), the atoms (strings preceded by constructor Var, or shortly V), classical connectives (preceded by a dot, to avoid conflicts) of negation ($.\neg$), conjunction ($.\wedge$), disjunction ($.\vee$), and implication ($.\longrightarrow$), along with temporal connectives for next, until, release, eventually and always. For automating the Isabelle proof, we add two extra connectives $\ddot{\mathcal{U}}$ (the until operator surrounded by dieresis) to denote the selected eventuality in goals and $\ddot{\circ}$ to mark the sequents formed by elementary formulae. We define the TTC rules by an inductive binary relation (predicate) TTC_proves, which is denoted "$\vdash$" in infix notation. The first argument of $\vdash$ is a set of PLTL formulae (implemented as an ordered list without repeated elements) and the second is a PLTL formula. By the construction of the calculus, the second argument is always the constant $\mathbf{F}$, but (for clarity) we prefer to keep $\vdash$ as a binary relation, and to explicitly represent that falsehood in the right-hand side of each goal. The Isabelle definition of $\vdash$ includes the two contradiction rules:

$$\begin{array}{ll} \mathit{TTC\_Ctd1}: & \varphi \mathbin{.\in} \Delta \Longrightarrow (\mathsf{NNF\_Neg}\ \varphi) \mathbin{.\in} \Delta \Longrightarrow \Delta \vdash \mathbf{F} \\ \mathit{TTC\_Ctd2}: & \mathbf{F} \mathbin{.\in} \Delta \Longrightarrow \Delta \vdash \mathbf{F} \end{array}$$

where $.\in$ is the user-defined infix operator for member of a list, and the user-defined function NNF_Neg computes the negation normal form of the negation of a given formula, i.e. $(\mathsf{NNF\_Neg}\ \varphi) = \mathsf{NNF}(.\neg\varphi)$. We also encode the traditional rules for classical and temporal connectives:

$$\begin{array}{ll} \mathit{TTC\_T}: & \Delta \vdash \mathbf{F} \Longrightarrow \mathbf{T}\ \#\ \Delta \vdash \mathbf{F} \\ \mathit{TTC\_And}: & \varphi \bullet \psi \bullet \Delta \vdash \mathbf{F} \Longrightarrow (\varphi . \wedge \psi)\ \#\ \Delta \vdash \mathbf{F} \\ \mathit{TTC\_Or}: & \varphi \bullet \Delta \vdash \mathbf{F} \Longrightarrow \psi \bullet \Delta \vdash \mathbf{F} \Longrightarrow (\varphi . \vee \psi)\ \#\ \Delta \vdash \mathbf{F} \\ \mathit{TTC\_Imp}: & (\mathsf{NNF\_Neg}\ \varphi) \bullet \Delta \vdash \mathbf{F} \Longrightarrow \psi \bullet \Delta \vdash \mathbf{F} \Longrightarrow (\varphi . \longrightarrow \psi)\ \#\ \Delta \vdash \mathbf{F} \\ \mathit{TTC\_Alw}: & \varphi \bullet \circ\Box\varphi \bullet \Delta \vdash \mathbf{F} \Longrightarrow (\Box\varphi)\ \#\ \Delta \vdash \mathbf{F} \\ \mathit{TTC\_R}: & \varphi \bullet \psi \bullet \Delta \vdash \mathbf{F} \Longrightarrow \psi \bullet \circ(\varphi \mathcal{R} \psi) \bullet \Delta \vdash \mathbf{F} \Longrightarrow (\varphi \mathcal{R} \psi)\ \#\ \Delta \vdash \mathbf{F} \\ \mathit{TTC\_Evt}: & \varphi \bullet \Delta \vdash \mathbf{F} \Longrightarrow \circ\Diamond\varphi \bullet \Delta \vdash \mathbf{F} \Longrightarrow (\Diamond\varphi)\ \#\ \Delta \vdash \mathbf{F} \\ \mathit{TTC\_U}: & \psi \bullet \Delta \vdash \mathbf{F} \Longrightarrow \varphi \bullet \circ(\varphi \mathcal{U} \psi) \bullet \Delta \vdash \mathbf{F} \Longrightarrow (\varphi \mathcal{U} \psi)\ \#\ \Delta \vdash \mathbf{F} \end{array}$$

where # is the standard *cons* constructor of lists and • is our user-defined operator for insert an element φ in the correct position of an ordered list Δ (if φ is already in Δ, then the result is Δ itself). For automating proofs, we have defined an order on the set of PLTL formulae where the minimal formulae are literals, and formulae with connectives are lexicographic ordered according to the following order on the set of connectives are ordered (from lowest to highest) as follows: $\ddot{}\circ\ddot{}$, $\circ$, $.\wedge$, $.\vee$, $.\longrightarrow$, $\Box$, $\mathcal{R}$, $\Diamond$, $\mathcal{U}$, $\ddot{}\mathcal{U}\ddot{}$. We order the antecedent of the sequent, in decreasing order, at the beginning of any proof using the following extra rule:

$$TTC\_Interchange: \quad (\mathsf{sort}\ \Delta) \vdash \mathbf{F} \Longrightarrow \Delta \vdash \mathbf{F}$$

Then, every application of a rule preserves the order in the generated subgoals by means of the operator • that inserts each formula in the correct place. As a consequence, the first formula of any sequent is a *non-elementary* formula (see Section 2), if there exists at least one. Moreover, the first formula is the eventuality, if there exists at least one, and it is the selected eventuality whenever the selection has already been done. The rules applied to the selected eventuality are:

$$\begin{array}{ll}
TTC_Evt_Plus: & \varphi \bullet \Delta \vdash \mathbf{F} \Longrightarrow \circ((\mathsf{negCtxt}\ \Delta)\mathcal{U}\varphi) \bullet \Delta \vdash \mathbf{F} \\
& \Longrightarrow \Diamond\varphi\ \#\ \Delta \vdash \mathbf{F} \\
TTC_U_Plus: & \psi \bullet \Delta \vdash \mathbf{F} \Longrightarrow \varphi \bullet \circ((\varphi\ .\sqcap.\ (\mathsf{negCtxt}\ \Delta))\ddot{}\mathcal{U}\ddot{}\psi) \bullet \Delta \vdash \mathbf{F} \\
& \Longrightarrow \varphi\mathcal{U}\psi\ \#\ \Delta \vdash \mathbf{F} \\
TTC_U_Sel: & \psi \bullet \Delta \vdash \mathbf{F} \Longrightarrow \varphi \bullet \circ((\varphi\ .\sqcap.\ (\mathsf{negCtxt}\ \Delta))\ddot{}\mathcal{U}\ddot{}\psi) \bullet \Delta \vdash \mathbf{F} \\
& \Longrightarrow \varphi\ddot{}\mathcal{U}\ddot{}\psi\ \#\ \Delta \vdash \mathbf{F}
\end{array}$$

where $(\mathsf{negCtxt}\ \Delta)$ is the negation of the context Δ, that is a disjunction of the negations (in NNF) of all formulae in Δ excepting the formulae of the form $\Box\varphi$. In addition, the operator $.\sqcap.$ is a conjunction up to subsumption, hence we avoid adding subsumed disjunctions, in particular, adding duplicated disjunctions. Note that the premises of *TTC_U_Sel* are really a copy of the premises of *TTC_U_Plus*.Consequently, *TTC_U_Plus* is applied at the first time when the eventuality has been just selected, whereas *TTC_U_Sel* is applied after that, while it is kept selected. When all formulae in the sequent are elementary we apply the following rule:

$$TTC\_Next\_State: \quad (\mathsf{next\_state}\ \Delta) \vdash \mathbf{F} \Longrightarrow \Delta \vdash \mathbf{F}$$

This rule applies when all the formulae in Δ are elementary (see Section 2) and the function next_state has filtered the formulae in Δ starting by the $\circ$ operator removing from them this operator.

The transition system in S on the left-hand of Figure 3 is defined as the list $TR = [T1,\ T2,\ T3,\ T4]$ of PLTL formulae:

$$\begin{array}{lcl}
T1 & = & \Box((V\ a)\ .\longrightarrow\ (\ \circ(V\ b)\ .\wedge\ \circ(\ .\neg\ (V\ a))\ .\wedge\ \circ(\ .\neg\ (V\ c))\ .\wedge\ \circ(\ .\neg\ (V\ d)))\ ) \\
T2 & = & \Box((V\ b)\ .\longrightarrow\ (\ (\circ(V\ b)\ .\wedge\ \circ(\ .\neg\ (V\ a))\ .\wedge\ \circ(\ .\neg\ (V\ c))\ .\wedge\ \circ(\ .\neg\ (V\ d)))\ .\vee \\
 & & \qquad (\circ(V\ c)\ .\wedge\ \circ(\ .\neg\ (V\ a))\ .\wedge\ \circ(\ .\neg\ (V\ b))\ .\wedge\ \circ(\ .\neg\ (V\ d)))\ ) \\
T3 & = & \Box((V\ c)\ .\longrightarrow\ (\ \circ(V\ b)\ .\wedge\ \circ(\ .\neg\ (V\ a))\ .\wedge\ \circ(\ .\neg\ (V\ c))\ .\wedge\ \circ(\ .\neg\ (V\ d)))\ ) \\
T4 & = & \Box((V\ d)\ .\longrightarrow\ (\ \circ(V\ a)\ .\wedge\ \circ(\ .\neg\ (V\ b))\ .\wedge\ \circ(\ .\neg\ (V\ c))\ .\wedge\ \circ(\neg\ (V\ d)))\ )
\end{array}$$

along with $Init = (V\ a)\ .\wedge\ .\neg\ (V\ b)\ .\wedge\ .\neg\ (V\ c)\ .\wedge\ .\neg\ (V\ d)$. Note that in the transition specifications the "next" operator is completely distributed over conjunction and disjunction. Hence, contradictions of the form $\circ(V\ x)$, $\circ.\neg\ (V\ x)$ are detected. Otherwise, the contradiction $V\ x$, $\neg\ (V\ x)$ should be detected at the next state.

We have implemented, with the help of the Eisbach tools [27], two prototypes of automatic solvers: one big-step and one small-step. These two solvers print into a text file the "apply" instructions of the Isabelle proof, at the same time that they prove the lemma. The proof of lemma runningExample_bigStep_proof is given in Figure 5. It proves the property $S \ @ \ [\circ\Diamond(V\ a)] \vdash \mathbf{F}$ by a list of "apply" instructions. This proof can be found in file `ProofGeneration.thy` and it is a big-step proof that enables the user to check the transitions of the system S that are performed when checking whether it satisfies a property φ, that is checking the unsatisfiability of $S \cup \{\neg\varphi\}$, in fact the derivability of the sequent $S, \neg\varphi \vdash \mathbf{F}$.

```
lemma runningExample_bigStep_proof: "S @ [○(◇(V ''a''))] ⊢ F"
apply (rule TTC_Interchange, simp add: S_def TR_def T1_def T2_def T3_def T4_def Init_def) (*1*)
apply one_step_solver                                                                    (*2*)
apply (all ‹rule TTC_Next_State›; simp)                                                  (*3*)
apply (fold T1_def T2_def T3_def T4_def)                                                 (*4*)
apply (simp add: T1_def T2_def T3_def T4_def Init_def)                                   (*5*)
apply one_step_solver                                                                    (*6*)
apply (all ‹rule TTC_Next_State›; simp)                                                  (*7*)
apply (fold T1_def T2_def T3_def T4_def)                                                 (*8*)
apply (simp add: T1_def T2_def T3_def T4_def Init_def)                                   (*9*)
apply one_step_solver                                                                    (*10*)
apply (all ‹rule TTC_Next_State›; simp)                                                  (*11*)
apply (simp add: T1_def T2_def T3_def T4_def Init_def)                                   (*12*)
apply one_step_solver                                                                    (*13*)
apply (all ‹rule TTC_Next_State›; simp)                                                  (*14*)
apply (fold T1_def T2_def T3_def T4_def)                                                 (*15*)
apply (simp add: T1_def T2_def T3_def T4_def Init_def)                                   (*16*)
apply one_step_solver                                                                    (*17*)
  done
```

Figure 5 The big-step lemma proof.

The (proof) method one_step_solver systematically applies the TTC_Calculus rules until we obtain a set of non-proved subgoals with antecedents exclusively formed by the elementary formulae. Hence, the rule TTC_Next_State is applied to all subgoals, which depict (as "Proof state") all subgoals related to a possible next state of the system, that is, after all possible transitions from the current state. For clarity, we fold the transition relations to their names. Hence, after the "apply" in line 3 (Figure 5), the user can see that there is only one subgoal:

$$[\Diamond(V\ a),\ T4,\ T2,\ T3,\ T1,\ .\neg(V\ d),\ .\neg(V\ c),\ .\neg(V\ a),\ (V\ b),\ ] \vdash \mathbf{F}$$

This means the only state that is reachable from the initial one is the state that satisfies b. That corresponds to the node marked with 1 in Figure 4. After the "apply", line 7, there are two subgoals that correspond, respectively, to the nodes marked with 2.1 and 2.2 in Figure 4. Thus, from the state, which satisfies exactly b, the system can reach either the same state again or the state satisfying exactly c. In both cases, the property to check is $(\neg b \vee a \vee c \vee d)\ \mathcal{U}\ a$. The goal 2.1 is proved, whereas the "apply", line 11, (Figure 5) generates the subgoal corresponding to node 2.2 in Figure 4. This subgoal corresponds to the transition from the state that satisfies b to the state that satisfies c. The property to check at this state is $(\neg b \vee a \vee c \vee d)\ \mathcal{U}\ a$. After the "apply" in line 14, the subgoal which corresponds to the node 2.2.1 in 4 is reached; here the property to check is $((\neg b \vee a \vee c \vee d) \wedge (\neg c \vee a \vee b \vee d))\ \mathcal{U}\ a$. The proof of this subgoal completes the proof of the lemma, that has explored the two possible runs $\langle a, b, b\rangle$ and $\langle a, b, c, b\rangle$ where the property $\circ\Diamond a$ is not satisfied. This shows the inability of the transition system to reach the state that satisfies d, which reveals an error in the specification that makes unreachable the state that satisfies a.

The "apply(fold ...)" instructions in Figure 5 are only to show, in subgoals, the names ($T1$, $T2$, $T3$, and $T4$) instead of the corresponding PLTL formulae defining the transitions, for brevity and clarity. After that, we must use "apply(simp add: ...)" to enable the application of the rules.

We have also implemented a method one_step_solver_print which prints into a file of text all applications of the TTC rules that are hidden in the big-step proof. Indeed, it is a printing version of the method one_step_solver. Calling one_step_solver_print, instead of one_step_solver, we can generate a small-step proof (see lemma runningExample_small-Step_proof) for the user wishing to check the Isabelle's subgoals step by step. This proof is the result of the substitution, in the big-step proof, of each of the five occurrences of apply one_step_solver by the list of "apply" instructions of TTC_Calculus rules. The method one_step_solver_print prints such a list, in a text file, while it is solving the goal.

5 Conclusions

In this paper we have presented a novel framework of applying a Certified Model Checking methodology. Our method involves the representation of the system specification, S, in a specific format that reflects the dynamic behaviour of the system. First, it involves the formalization of the "initial conditions", (the *Init* PLTL formula), that specifies the initial states of the model to be build. Second, we use the representation, TR, of the transition relation to build a state space of the system, as a "global invariant". Finally, the property, P, to be checked against the specification S is written as a PLTL formula. This task is performed by a one-pass temporal tableau, $\tau^{\uparrow}_{pltl}$. It takes $S, \neg P$ as its input (converted into the NNF). For an eventuality to be fulfilled, the tableau technique essentially uses its context. Tableau rules that deal with the eventualities in NNF($\neg P$), force their fulfilment "as soon as possible". This process is optimised by a SAT solver, which tackles non-temporal content of the tableau nodes. The tableau method, in one pass, either returns a negative answer, producing a counter-example, thus showing that P is not satisfied by S, or verifies P against the system specification. In the latter case, our method generates an explicit and easily readable evidence in Isabelle/HOL. In this way the tableau result is formally proven and the user can review the test to make sure that everything is working as expected (or that no errors have been made with the specification).

Our future work will cover three directions. First, it is further work on the implementation of and experimentation with the one-pass tableau and the embedded SAT solver. Second, the developed Isabelle automatic solver for proof generation is only an initial prototype and we will improve its efficiency. Finally, note that the one-pass tableau method has been also developed for the branching-time setting, tackling Computation Tree Logic CTL ([6]), widely used in model checking, and for a richer logic, $ECTL^{\sharp}$ ([5]). This will enable us to extend the use of the one-pass tableau as a model checker to the branching-time setting, as the extensions are conceptually intuitive.

References

1 Hasan Amjad. Programming a Symbolic Model Checker in a Fully Expansive Theorem Prover. In *Theorem Proving in Higher Order Logics*, pages 171–187. Springer Berlin Heidelberg, 2003. doi:10.1007/10930755_11.

2 Mordechai Ben-Ari. *Mathematical Logic for Computer Science*. Springer-Verlag, London, 2012. doi:10.1007/978-1-4471-4129-7.

3 Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Masahiro Fujita, and Yunshan Zhu. Symbolic Model Checking Using SAT Procedures Instead of BDDs. In *Proceedings of the 36th Annual ACM/IEEE Design Automation Conference*, DAC '99, pages 317–320, New York, NY, USA, 1999. ACM. doi:10.1145/309847.309942.

4 Armin Biere and Daniel Kröning. SAT-based model checking. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 277–303. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-10575-8_10.

5 Alexander Bolotov, Montserrat Hermo, and Paqui Lucio. Extending Fairness Expressibility of ECTL+: A Tree-Style One-Pass Tableau Approach. In Natasha Alechina, Kjetil Nørvåg, and Wojciech Penczek, editors, *25th International Symposium on Temporal Representation and Reasoning (TIME 2018)*, volume 120 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:22, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TIME.2018.5.

6 Kai Brünnler and Martin Lange. Cut-free sequent systems for temporal logic. *The Journal of Logic and Algebraic Programming*, 76(2):216–225, 2008. doi:10.1016/j.jlap.2008.02.004.

7 CENELEC and EN50128. 50128. *Railway applications-Communication, Signaling and Processing Systems-Software for Railway Control and Protection Systems*, 2011.

8 Alessandro Cimatti, Edmund Clarke, Fausto Giunchiglia, and Marco Roveri. NUSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, March 2000. doi:10.1007/s100090050046.

9 Koen Claessen and Niklas Sörensson. A liveness checking algorithm that counts. In Gianpiero Cabodi and Satnam Singh, editors, *Formal Methods in Computer-Aided Design, FMCAD 2012, Cambridge, UK*, pages 52–59. IEEE, 2012.

10 Edmund M. Clarke and E. Allen Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *Logics of Programs, Workshop, Yorktown Heights, New York, May 1981*, pages 52–71, 1981. doi:10.1007/BFb0025774.

11 Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999.

12 Marco Comini, Laura Titolo, and Alicia Villanueva. Abstract Diagnosis for tccp using a Linear Temporal Logic. *Theory and Practice of Logic Programming*, 14(4-5):787–801, 2014. doi:10.1017/S1471068414000349.

13 Javier Esparza, Peter Lammich, René Neumann, Tobias Nipkow, Alexander Schimpf, and Jan-Georg Smaus. A Fully Verified Executable LTL Model Checker. In *Computer Aided Verification*, pages 463–478. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-39799-8_31.

14 Joxe Gaintzarain, Montserrat Hermo, Paqui Lucio, Marisa Navarro, and Fernando Orejas. A Cut-Free and Invariant-Free Sequent Calculus for PLTL. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 481–495. Springer, 2007. doi:10.1007/978-3-540-74915-8_36.

15 Joxe Gaintzarain, Montserrat Hermo, Paqui Lucio, Marisa Navarro, and Fernando Orejas. Dual Systems of Tableaux and Sequents for PLTL. *The Journal of Logic and Algebraic Programming*, 78(8):701–722, 2009. doi:10.1016/j.jlap.2009.05.001.

16 Rajeev Goré. Tableau Methods for Modal and Temporal Logics. In Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Springer Netherlands, Dordrecht, 1999. doi:10.1007/978-94-017-1754-0_6.

17 Alberto Griggio, Marco Roveri, and Stefano Tonetta. Certifying Proofs for LTL Model Checking. In *Formal Methods in Computer-Aided Design, FMCAD 2018, Austin, USA*, pages 1–9, October 2018. doi:10.23919/FMCAD.2018.8603022.

18 IEC. IEC 61508 Functional safety of electrical/electronic/programmable electronic safety-related systems, 2010.

19 ISO. Road vehicles – Functional safety, 2011.

20 Henry Kautz and Bart Selman. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1194–1201. AAAI Press, 1996. URL: http://dl.acm.org/citation.cfm?id=1864519.1864564.

21 Jörg Kreiker, Andrzej Tarlecki, Moshe Y. Vardi, and Reinhard Wilhelm. Modeling, Analysis, and Verification - The Formal Methods Manifesto 2010 (Dagstuhl Perspectives Workshop 10482). *Dagstuhl Manifestos*, 1(1):21–40, 2011. doi:10.4230/DagMan.1.1.21.

22 Tuomas Kuismin and Keijo Heljanko. Increasing Confidence in Liveness Model Checking Results with Proofs. In Valeria Bertacco and Axel Legay, editors, *Hardware and Software: Verification and Testing*, pages 32–43. Springer International Publishing, 2013.

23 Orna Kupferman and Moshe Y. Vardi. From complementation to certification. *Theoretical Computer Science*, 345(1):83–100, 2005. doi:10.1007/978-3-540-24730-2_43.

24 Jianwen Li, Geguang Pu, Lijun Zhang, Moshe Y Vardi, and Jifeng He. Accelerating LTL satisfiability checking by SAT solvers. *Journal of Logic and Computation*, 28(6):1011–1030, April 2018. doi:10.1093/logcom/exy013.

25 Jianwen Li, Shufang Zhu, Geguang Pu, Lijun Zhang, and Moshe Y. Vardi. SAT-based explicit LTL reasoning and its application to satisfiability checking. *Formal Methods in System Design*, January 2019. doi:10.1007/s10703-018-00326-5.

26 Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, Berlin, Heidelberg, 1992. doi:10.1007/978-1-4612-0931-7.

27 Daniel Matichuk, Toby Murray, and Makarius Wenzel. Eisbach: A Proof Method Language for Isabelle. *Journal of Automated Reasoning*, 56, January 2016. doi:10.1007/s10817-015-9360-2.

28 Alain Mebsout and Cesare Tinelli. Proof Certificates for SMT-based Model Checkers for Infinite-state Systems. In *Proceedings of the 16th Conference on Formal Methods in Computer-Aided Design*, FMCAD '16, pages 117–124, 2016. doi:10.1109/FMCAD.2016.7886669.

29 Kedar S. Namjoshi. Certifying Model Checkers. In *Proceedings of the 13th International Conference on Computer Aided Verification*, CAV '01, pages 2–13. Springer-Verlag, 2001. doi:10.1007/3-540-44585-4_2.

30 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. doi:10.1007/3-540-45949-9.

31 Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In Mariangiola Dezani-Ciancaglini and Ugo Montanari, editors, *International Symposium on Programming*, pages 337–351, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg. doi:10.1007/3-540-11494-7_22.

32 Kristin Y. Rozier and Moshe Y. Vardi. LTL Satisfiability Checking. In Dragan Bošnački and Stefan Edelkamp, editors, *Model Checking Software*, pages 149–167, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. doi:10.1007/978-3-540-73370-6_11.

33 Aravinda P. Sistla and Edmund M. Clarke. The Complexity of Propositional Linear Temporal Logics. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 159–168, New York, NY, USA, 1982. ACM. doi:10.1145/800070.802189.

Minimisation of Models Satisfying CTL Formulas

Serenella Cerrito
IBISC, Univ Evry, Université Paris-Saclay, 91025, Evry, France
serenella.cerrito@univ-evry.fr

Amélie David
IBISC, Univ Evry, Université Paris-Saclay, 91025, Evry, France
amely.david@laposte.net

Valentin Goranko
Stockholm University, Sweden
University of Johannesburg (visiting professorship), South Africa
valentin.goranko@philosophy.su.se

Abstract

We study the problem of minimisation of a given finite pointed Kripke model satisfying a given CTL formula, with the only objective to preserve the satisfaction of that formula in the resulting reduced model. We consider minimisations of the model with respect both to state-based redundancies and formula-based redundancies in that model. We develop a procedure computing all such minimisations, illustrate it with some examples, and provide some complexity analysis for it.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Computing methodologies → Temporal reasoning

Keywords and phrases CTL, model minimisation, bisimulation reduction, tableaux-based reduction

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.13

Funding The work of Valentin Goranko was supported by a research grant 2015-04388 of the Swedish Research Council.

1 Introduction

1.1 The problem of study and our proposal

The Computation Tree Logic CTL ([7], [9]) is one of the most useful and applicable temporal logics in computer science, because of its good balance between expressiveness and computational efficiency of model checking. One of the main problems that arise in its practical use is the *state explosion problem*, which calls for methods for reducing the size of the state transition systems arising when modelling real programs or systems. A lot of research has been done over the past three-four decades in addressing and resolving that problem by applying various techniques, such as bisimulation minimisations, abstraction refinements, BDD-based symbolic representations and symbolic model checking, partial order reductions, SAT-based model checking, etc. (cf [8] for comprehensive and up-to-date accounts of these). Most of these techniques follow the idea of applying minimisations, reductions, or abstractions to the original model, prior to doing model checking of the desired properties in it, by ensuring that the reduced model preserves all relevant properties (e.g., by being bisimulation equivalent to the original one). This approach is certainly very natural and has proved to be practically very useful.

Here, however, we take a somewhat different approach, viz. we study the problem of minimisation of a given finite pointed Kripke model (aka, pointed interpreted transition system) $(\mathcal{M}, s)$ *that is already known to satisfy a given* CTL *formula* θ, with the only objective to preserve the satisfaction of that formula in the minimised model. We argue that this problem is natural and important, too, because the formula θ can be viewed as

a formal specification of all critical features that the system must possess. Then, one may naturally want to synthesise a smallest and simplest possible abstract model of the system that satisfies that specification at its initial state, e.g. in order to facilitate further multiple verifications of various other properties and eventually its practical implementation. For instance, such formulas might be specifications of components of a product transition system, and such product constructions usually produce large redundancies that should preferably be eliminated before the actual implementation.

The main problem of this study is more precisely described as follows. We assume that some pointed Kripke model $(\mathcal{M}, s)$, satisfying a given CTL formula θ is already available, e.g. extracted from a real system or constructed by some of the well-known methods (tableaux, automata, etc, see e.g. [10]). We are then interested in producing a "minimal" such pointed model out of the given one, that still satisfies θ. By "minimal" here we mean a pointed model that cannot be further reduced by means of general and explicitly specified reducing operations, such as identifying states or taking submodels, to an even smaller one that still satisfies θ. We note that a given model satisfying a given formula may not be minimal with respect to that property for at least two different reasons: it may have redundancies caused by bisimilar states, and it may have redundancies with respect to the formula that it must satisfy. Thus, minimizing procedures for both types of redundancies are generally necessary, because most of the currently used methods for constructing satisfying models of CTL formulas (typically, tableaux or automata-based) do not usually produce minimal models in either sense.

Contributions. Our main contribution is the development of a minimization procedure that eliminates both kinds of redundancies. Respectively, our proposal, in a nutshell, is to combine and iterate two reduction procedures:

- ▷ ***Bisimulation reduction procedure,*** based on some of the well-known algorithms, e.g. in [16] or [13]. This procedure eliminates redundancies caused by bisimilar states and works in low polynomial (at most quadratic) time. Note that for our purpose we are only interested in bisimulation reduction with respect to the language of the given formula θ (called θ-bisimilarity in the following).
- ▷ ***Formula-driven reduction procedure,*** based on a tableaux-like construction. It implements two simple minimisation ideas:
  - to satisfy a disjunction, use part of the model to satisfy just one disjunct;
  - select only minimal (irreducible) sets of necessary successors of each state.

Because of the possible choices in both cases above, this procedure branches and eventually may produce several minimisations.

While this work focuses on minimisation of models of CTL formulas, we also consider in passing the simpler case of minimisation of models of formulas of the basic modal logic.

Related work. As the problem is important and very natural, there is much related work, though, up to our knowledge, none of it addresses exactly the same problem or follows the same approach as ours. We give a brief (and, for lack of space, quite incomplete) overview of related approaches to model minimisation, in a roughly chronological order.

Algorithmic bisimulation minimisation of Kripke models (aka, interpreted transition systems) has been explored extensively in the literature, going back to [13] and [16]; see [14] for an overview and references therein. The question of generation of minimal models with respect to bisimulation has been studied e.g. in [3], [2]. In [12] a method is proposed for obtaining a minimal transition system, representing a communicating system given by a set of parallel processes. More related to our work are [6] and [17], which explore *compositional*

minimization. There, a system on which a CTL formula θ needs to be model-checked is taken to be the product of n transitions systems $M_1, .., M_n$. A local model-checking of each M_i allows for the computation of a BDD representing a reduced number of transitions, so to reduce the final global product. Unlike our work, however, bisimulation-based reductions are not taken into account and redundancies caused by disjunctions are not considered. The above approach is then extended in [1], where a notion of formula-dependent state equivalence is proposed. However, again, redundancies caused by disjunctions are ignored, as well as subset inclusion (see Section 3.4).

Mogavero and Murano [15] have proposed a logic extending CTL* and internalizing minimal model construction by means of two minimal model quantifiers, Λ and Ξ. That approach, while thematically closely related, is somewhat orthogonal and incomparable to ours. The main difference is that we do not extend the CTL language to reason about truth in minimal models of formulas, but are interested in the *actual computing* of the minimizations of a model with respect to a formula, which we do purely semantically and constructively. Besides, we consider a stronger notion of minimality, taking into account also bisimulation. Thus, the objectives, approaches and results are quite different. We compare the two approaches with some more details and an example in Section 5.

Bozzelli and Pearce [4] explore the idea of "temporal equilibrium model" of an LTL formula, satisfying minimality requirement with respect to state labels. Cerrito and David [5] investigate the question of bisimulation minimisation of models of the multi-agent extension ATL of CTL.

Structure of the paper. We start with a brief background on the logic CTL and on bisimulation in Section 2. In Section 3 we describe two versions of our minimization procedure and we illustrate it on some examples. Some results about properties of the procedure are established in Section 4. We conclude by indicating some lines of future work in Section 5. A few proofs of auxiliary results are put in a short appendix.

2 Preliminaries

2.1 CTL: syntax and semantics

Here we only provide brief basic preliminaries on CTL. For further details see e.g. [10, Ch.7]. The syntax of CTL is given by the following grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{EX}\,\varphi \mid \mathsf{E}(\varphi\,\mathsf{U}\,\varphi) \mid \mathsf{A}(\varphi\,\mathsf{U}\,\varphi)$$

where $\top$ is the logical constant for truth, Prop is a set of **proposition symbols** and $p \in$ Prop. We also use the following abbreviations: $\mathsf{AX}\,\varphi := \neg\mathsf{EX}\,\neg\varphi$, $\mathsf{EF}\,\varphi := \mathsf{E}(\top\,\mathsf{U}\,\varphi)$, $\mathsf{AF}\,\varphi := \mathsf{A}(\top\,\mathsf{U}\,\varphi)$, $\mathsf{EG}\,\varphi := \neg\mathsf{AF}\,\neg\varphi$ and $\mathsf{AG}\,\varphi := \neg\mathsf{EF}\,\neg\varphi$.

The set of atomic propositions occurring in a formula φ is denoted by $\mathrm{prop}(\varphi)$. The **basic modal logic BML** is the fragment of CTL that does not involve the operator U, i.e. extends propositional logic only with EX.

CTL formulas are interpreted over transition systems.

▶ **Definition 1.** *A **transition system** is a pair $\mathcal{T} = (S, R)$, where S is a nonempty set of **states** and $R \subseteq S \times S$ is a **transition relation** on S. Unless otherwise specified, transition systems will be assumed* serial *(this requirement is typically imposed for models of* CTL *but not for models of* BML*), i.e. for every $s \in S$ there is $s' \in S$ such that $(s, s') \in R$. When a distinguished state $s \in S$ is considered, $(\mathcal{T}, s)$ is called a **rooted (at s) transition system**, or a **pointed transition system**. A **path** in $\mathcal{T}$ is a sequence $\lambda : \mathbb{N} \to S$ such*

that $(\lambda(n), \lambda(n+1)) \in R$ *for every* $n \in \mathbb{N}$. *An* ***interpreted transition system*** *(ITS) over* $\mathcal{T}$ *is a tuple* $\mathcal{M} = (S, R, \mathrm{Prop}, L)$, *where* Prop *is a set of* ***proposition symbols*** *and* $L : S \to \mathcal{P}(\mathrm{Prop})$ *is a* ***state description function*** *defining for every state in* S *the set of atomic propositions true at that state. A* ***rooted (pointed) interpreted transition system*** $(\mathcal{M}, s)$ *is defined accordingly.*

Given an ITS $\mathcal{M} = (S, R, \mathrm{Prop}, L)$ *and any subset* P *of* Prop, *we define the* ***reduction of*** $\mathcal{M}$ ***to*** P *to be the ITS* $\mathcal{M}|_P = (S, R, P, L|_P)$, *where* $L|_P : S \to \mathcal{P}(P)$ *is defined by* $L|_P(s) = L(s) \cap P$ *for every* $s \in S$.

Given an ITS $\mathcal{M} = (S, R, \mathrm{Prop}, L)$, *an ITS* $\mathcal{M}' = (S', R', \mathrm{Prop}, L')$ *is said to be a* ***substructure*** *of* $\mathcal{M}$ *whenever* $S' \subseteq S$, L' *is the restriction of* L *to* S', *and* $R' = R \cap (S' \times S')$. *By an abuse of language, we say that* $\mathcal{M}'$ *is a substructure of* $\mathcal{M}$ *also when* $R' = (R \cap (S' \times S')) \cup \{< t_1, t_1 >, \ldots, < t_n, t_n >\}$ *where* $\{t_1, .., t_n\} \subseteq S'$, *for any* $n \geq 0$. *The ITS* $\mathcal{M}'$ *is said to be a* ***proper substructure*** *of* $\mathcal{M}$ *when* $S' \subset S$.

▶ **Definition 2.** *Let* $\mathcal{M} = (S, R, \mathrm{Prop}, L)$ *be an interpreted transition system,* $s \in S$ *and* φ *a* CTL*-formula.* ***Truth of*** φ ***at*** s ***in*** $\mathcal{M}$, *denoted by* $\mathcal{M}, s \models \varphi$, *is defined inductively on* φ *as follows (we give here only the non-boolean cases):*

- $\mathcal{M}, s \models \mathsf{EX}\,\varphi$ *iff there is a state* s' *such that* $(s, s') \in R$ *and* $\mathcal{M}, s' \models \varphi$.
- $\mathcal{M}, s \models \mathsf{E}(\varphi\,\mathsf{U}\,\psi)$ *iff there is a path* λ *in* $\mathcal{M}$ *starting from* s *and* $i \geq 0$ *such that* $\mathcal{M}, \lambda(i) \models \psi$ *and* $\mathcal{M}, \lambda(j) \models \varphi$ *for every* $j < i$.
- $\mathcal{M}, s \models \mathsf{A}(\varphi\,\mathsf{U}\,\psi)$ *iff for every path* λ *in* $\mathcal{M}$ *starting from* s, *there is* $i \geq 0$ *such that* $\mathcal{M}, \lambda(i) \models \psi$ *and* $\mathcal{M}, \lambda(j) \models \varphi$ *for every* $j < i$.

An ITS $(\mathcal{M}, s)$ *is a* ***pointed model of*** φ *whenever* $\mathcal{M}, s \models \varphi$.

2.2 Types, components, and extended closure of CTL formulas

We use some notions and terminology from the literature on tableaux-based satisfiability decision methods (see e.g. [10, Ch.13]). Formulas of CTL can be classified as: **literals**: $\top$, $\neg\top$, $p, \neg p$, where $p \in \mathrm{Prop}$, **successor formulas**: $\mathsf{EX}\,\varphi$ and $\neg\mathsf{EX}\,\varphi$, **conjunctive formulas** (also called ***α*-formulas**), and **disjunctive formulas** (also called ***β*-formulas**). The formulas in the last three classes have respective **components** that are given by Table 1. For convenience, the tables provide also the components of some defined formulas (e.g. $\mathsf{EF}\,\psi$). It is well-known (cf. [10, Ch.13]) that any conjunctive (resp. disjunctive) formula in the table is equivalent to the conjunction (resp. disjunction) of its components.

Table 1 Types of formulas and their components.

Conjunctive formula	Components
$\neg\neg\varphi$	φ
$\neg(\varphi \vee \psi)$	$\neg\varphi,\ \neg\psi$
$\neg\mathsf{E}(\varphi\,\mathsf{U}\,\psi)$	$\neg\psi,\ \neg\varphi \vee \neg\mathsf{EXE}(\varphi\,\mathsf{U}\,\psi)$
$\neg\mathsf{A}(\varphi\,\mathsf{U}\,\psi)$	$\neg\psi,\ \neg\varphi \vee \neg\mathsf{AXA}(\varphi\,\mathsf{U}\,\psi)$
$\mathsf{EG}\,\varphi$	$\varphi,\ \mathsf{EXEG}\,\varphi$
$\mathsf{AG}\,\varphi$	$\varphi,\ \mathsf{AXAG}\,\varphi$

Disjunctive formula	Components
$\varphi \vee \psi$	$\varphi,\ \psi$
$\mathsf{E}(\varphi\,\mathsf{U}\,\psi)$	$\psi,\ \varphi \wedge \mathsf{EXE}(\varphi\,\mathsf{U}\,\psi)$
$\mathsf{A}(\varphi\,\mathsf{U}\,\psi)$	$\psi,\ \varphi \wedge \mathsf{AXA}(\varphi\,\mathsf{U}\,\psi)$
$\mathsf{EF}\,\psi$	$\psi,\ \mathsf{EXEF}\,\psi$
$\mathsf{AF}\,\psi$	$\psi,\ \mathsf{AXAF}\,\psi$

Successor formula	Components
$\mathsf{EX}\varphi$ (existential successor formula)	φ
$\neg\mathsf{EX}\varphi$ (universal successor formula)	$\neg\varphi$

▶ **Definition 3.** *The **extended closure** of a formula φ is the least set of formulas $\mathsf{ecl}(\varphi)$ such that:*

1. $\varphi \in \mathsf{ecl}(\varphi)$,
2. $\mathsf{ecl}(\varphi)$ *is closed under taking all components of each formula ψ in $\mathsf{ecl}(\varphi)$, i.e., conjunctive, disjunctive and successor components, according to the type of ψ*

For any set of formulas Γ we define $\mathsf{ecl}(\Gamma) := \bigcup\{\mathsf{ecl}(\varphi) \mid \varphi \in \Gamma\}$.

A formula $\mathsf{E}(\varphi \mathsf{U} \psi)$ (in particular, $\mathsf{EF}\,\psi$) is said to be an **existential eventuality** and $\mathsf{A}(\varphi \mathsf{U} \psi)$ (in particular, $\mathsf{AF}\,\psi$) – a **universal eventuality**.

2.3 Bisimulations and invariance

We recall here the well-known notion of bisimilarity of interpreted transition systems (see, for instance, [14] or [10, Ch.3]).

▶ **Definition 4.** *Let $\mathcal{M}_1 = (S_1, R_1, \mathrm{Prop}, L_1)$ and $\mathcal{M}_2 = (S_2, R_2, \mathrm{Prop}, L_2)$ be two interpreted transition systems over the same set of propositions* Prop. *A relation $\beta \subseteq S_1 \times S_2$ is a **bisimulation** between $\mathcal{M}_1$ and $\mathcal{M}_2$, denoted $\mathcal{M}_1 \overset{\beta}{\rightleftarrows} \mathcal{M}_2$, iff for all $s_1 \in S_1$ and $s_2 \in S_2$, $s_1 \beta s_2$ implies:*

1. *Atom Equivalence: $\mathsf{L}_1(s_1) = \mathsf{L}_2(s_2)$;*
2. *Forth condition: For any $r_1 \in S_1$, if $s_1 R_1 r_1$ then there is some $r_2 \in S_2$ such that $s_2 R_2 r_2$ and $r_1 \beta r_2$;*
3. *Back condition: For any $t_2 \in S_2$, if $s_2 R_2 t_2$ then there is some $t_1 \in S_1$ such that $s_1 R_1 t_1$ and $t_1 \beta t_2$.*

*Two states $s_1 \in S_1$ and $s_2 \in S_2$ are **bisimilar** if there is a bisimulation β between $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $s_1 \beta s_2$. We denote that by $(\mathcal{M}_1, s_1) \overset{\beta}{\rightleftarrows} (\mathcal{M}_2, s_2)$ (or, just $(\mathcal{M}_1, s_1) \rightleftarrows (\mathcal{M}_2, s_2)$ when β is inessential) and say that the rooted models $(\mathcal{M}_1, s_1)$ and $(\mathcal{M}_2, s_2)$ are **locally bisimilar**. If there is a bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$ that links every state in S_1 to some state of S_2 and vice versa, we say that $\mathcal{M}_1$ and $\mathcal{M}_2$ are **(globally) bisimilar**.*

The following is a minor adaptation of a well-known result relating bisimulations and logic (see e.g. [18] or [10, Ch.3]). Here bisimulation is between reductions of ITS to a subset P of atomic propositions, thus *Atom Equivalence* is relativised to the propositions in P only.

▶ **Proposition 5** (Relativised bisimulation invariance)**.** *Let φ be a* CTL *formula,* $\mathrm{prop}(\varphi) \subseteq$ Prop, $\mathcal{M}_1$=(S_1,R_1, Prop, L_1) *and* $\mathcal{M}_2$=(S_2, R_2, Prop, L_2), *and $\beta \subseteq S_1 \times S_2$ be a local bisimulation between $(\mathcal{M}_1|_{\mathrm{prop}(\varphi)}, s_1)$ and $(\mathcal{M}_2|_{\mathrm{prop}(\varphi)}, s_2)$. Then $(\mathcal{M}_1|_{\mathrm{prop}(\varphi)}, s_1) \models \varphi$ iff $(\mathcal{M}_2|_{\mathrm{prop}(\varphi)}, s_2) \models \varphi$.*

When $\mathcal{M}_1 \overset{\beta}{\rightleftarrows} \mathcal{M}_2$ and $\mathcal{M}_1 = \mathcal{M}_2 = \mathcal{M}$ we say that **β is a bisimulation in $\mathcal{M}$**. Every such bisimulation is an equivalence relation in $\mathcal{M}$ and therefore generates a quotient-structure from $\mathcal{M}$ which we call the **quotient of $\mathcal{M}$ with respect to β**. It is well-known (see e.g. [11] or [10, Ch.3]) that amongst all bisimulations in $\mathcal{M}$ there is a largest one, $\beta_{\mathcal{M}}$. The quotient of $\mathcal{M}$ with respect to $\beta_{\mathcal{M}}$, hereafter denoted by $\widetilde{\mathcal{M}}$, is called the **bisimulation collapse of $\mathcal{M}$**. Note that every two different states in $\widetilde{\mathcal{M}}$ are non-bisimilar.

All these concepts relativise to reductions of ITS with respect to subsets P of atomic propositions. Note that, the smaller the subset P is, the larger the respective largest bisimulation in $\mathcal{M}|_P$, and therefore the smaller the bisimulation collapse $\widetilde{\mathcal{M}|_P}$. Therefore, when trying to minimize a model of a given formula θ with respect to bisimulations, we will be interested in $\widetilde{\mathcal{M}|_{\mathrm{prop}(\theta)}}$.

3 Model minimisation procedure (MMP)

3.1 Brief informal description

Our main aim is to develop an efficient procedure that minimises – in a sense to be made precise later – any given finite pointed model $(\mathcal{M}, s)$ of a CTL formula θ.

To facilitate and optimise that procedure, we precede it with global model checking in $\mathcal{M}$ of the formulas in the extended closure of θ. Since model checking of CTL formulas is very efficient, viz., bi-linear in both the size of the model and the length of the formula ([7], see also [10, Ch.7]), this preprocessing would not increase the overall complexity of the minimisation procedure.

Now, given $(\mathcal{M}, s)$ and the input formula θ, such that $(\mathcal{M}, s) \models \theta$, by applying global model checking in $\mathcal{M}$ we identify the set $\|\theta\|_{\mathcal{M}}$ of all states in $\mathcal{M}$ satisfying θ. If θ must be satisfied in *the same* (up to bisimulation collapse) state as s in the obtained minimal model, then the procedure works as described further shortly. If, however, satisfying θ at any state in the obtained minimal model will be sufficient for the purposes of the intended minimization, then a slightly different approach may be preferable: consider all states $t \in \|\theta\|_{\mathcal{M}}$, call the minimisation procedure to $(\mathcal{M}, t)$ for each of them, and finally select one of the obtained minimal models. Alternatively, to avoid some of that work, select amongst all states $t \in \|\theta\|_{\mathcal{M}}$ only those, for which the generated at t submodel of $\mathcal{M}$ is minimal by inclusion with respect to the others, and only apply the minimisation procedure to them.

We emphasize that either of these approaches may be preferable, depending on the concrete case. So, we are only listing them here as reasonable options, but the actual choice of concrete approach is left to the agent (or tool) performing the minimisation.

We assume hereafter that the possible selection of states indicated above has already been performed and the task now is to minimise a given pointed model $(\mathcal{M}, s)$ so that the formula θ is eventually satisfied at (the image of) the same state s in the minimised model.

As noted in the introduction, the minimisation procedure that we develop aims at detecting and eliminating two kinds of redundancies in $\mathcal{M}$, described below. These may have to be applied repeatedly, in an order discussed further, in Section 4.1.

1. *Model-based redundancies*, that arise when the model contains different states that are bisimilar with respect to the language of the input formula. These redundancies are eliminated by applying a well-known bisimulation minimisation procedure, after ignoring the atomic propositions not occurring in the formula. This procedure is deterministic and produces a unique (up to state renaming) reduced model – the bisimulation quotient. 5
2. *Formula-based redundancies*, that arise when the model contains "unnecessary" states, that can be removed without affecting the truth of the formula. Typically, such redundancies arise when:
   - **(i)** the model satisfies both disjunctive components of a disjunctive (sub)formula at some state, instead of only one of them, or
   - **(ii)** a state has more successors than what is needed to satisfy the (sub)formulas that have to be true there, or
   - **(iii)** a state is not reached in the process of the evaluation of the formula. These include all states that are not reachable by finite transition paths from the root state. In the case of a BML formula of modal depth $\leq n$ these are also all states not reachable in n transition steps from the root state.

These redundancies are eliminated by applying a tableaux-like procedure on the given input model, systematically selecting a single branch in the search / decision tree whenever a disjunction is to be satisfied, and selecting only a *minimal subset of necessary successors*

of each state added to the selection; this notion is precisely defined in Section 3.4. This procedure is non-deterministic and produces at least one, but possibly many reduced models, some of which may contain others. After its completion we also remove all obtained reduced models that are not minimal by inclusion.

Note that the preliminary global model checking is also useful in the tableaux-like minimisation procedure to select only minimal subsets of necessary successors of the current state, as well as to select in advance the shortest possible paths in the model realizing required eventualities. This will be illustrated on the running examples of minimising redundant models presented further.

3.2 Running examples

▶ **Example 6.** Consider the rooted model $(\mathcal{M}_1, s)$ shown in Figure 1 and the following formulas :

$\phi_1 = \mathsf{EX}\, p \wedge \mathsf{AF}\,(q \vee \mathsf{EF}\, p)$, $\phi_2 = \mathsf{EX}\, \neg p \wedge \mathsf{EX}\, q \wedge \mathsf{AG}\,(q \rightarrow p)$,
$\phi_3 = \mathsf{EX}\, \neg p \wedge \mathsf{EX}\, \mathsf{E}((p \wedge q)\, \mathsf{U}\, \neg q)$, $\phi_4 = \mathsf{EX}\, q \wedge \mathsf{EG}\,(\neg q \wedge p)$,
$\theta_1 = \phi_1 \vee \phi_2$, $\theta_2 = \phi_1 \vee \phi_3$, $\theta_3 = \phi_1 \vee \phi_4$.

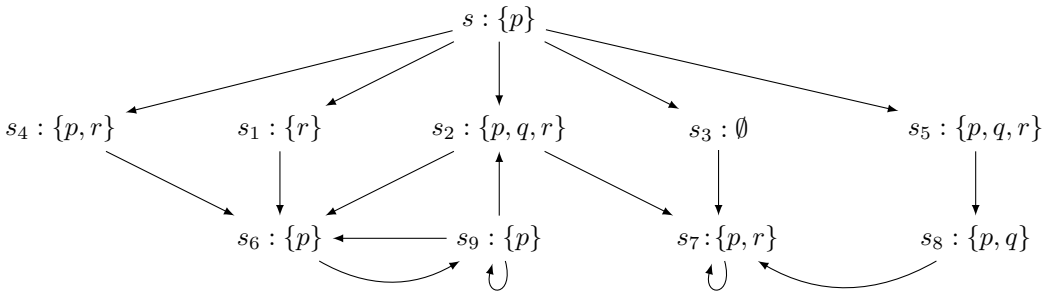


Figure 1 The model $\mathcal{M}_1$.

$\mathcal{M}_1$ satisfies at s all ϕ_i, for $i = 1..4$. Hence, it satisfies each of θ_1, θ_2 and θ_3 but, as we will show, it has unnecessarily many states.

▶ **Example 7.** Model $\mathcal{M}_2$ in Figure 4 satisfies $(\mathcal{M}_2, s) \models \mathsf{EX}\,(\neg p \wedge \mathsf{EX}\,(p \wedge \mathsf{EX}\,(p \wedge q)))$. Again, we will show that it contains states that are unnecessary for that purpose.

3.3 Bisimulation reduction (BR)

As explained in Section 2.3, in our procedure of bisimulation minimization of a (pointed) model $(\mathcal{M}, s)$ satisfying a given CTL formula θ, in order to obtain a smallest possible bisimulation collapse of $\mathcal{M}$ that still satisfies θ we only need to compute the bisimulation collapse of *the reduction* $\mathcal{M}_\theta = \mathcal{M}|_{\mathrm{prop}(\theta)}$ *of* $\mathcal{M}$ *to the language of* θ. The resulting pointed ITS $(\widetilde{\mathcal{M}|_\theta}, \tilde{s})$ still satisfies θ and has the minimal number of states amongst all ITS that satisfy θ and are θ-bisimilar to $\mathcal{M}$. We call this formula-oriented procedure **θ-bisimulation minimisation of $\mathcal{M}$**.

Some essential remarks are in order.

(i) In order to preserve the satisfaction of θ it suffices to compute a *local* bisimulation collapse, of the submodel of $\mathcal{M}_\theta$ that is generated by s.

(ii) If θ is a BML-formula of modal depth n, then it suffices to compute the *n-bisimulation collapse of* $(\mathcal{M}_\theta, s)$, that identifies any two states satisfying the same formulas of depth up to n in the language of θ. That will, in general, produce an even smaller model.

(iii) The issue arises of what happens to the atomic propositions not occurring in θ. The procedure above ignores and forgets them completely. But that may be neither necessary nor desirable, even though we are currently only concerned with $\mathcal{M}$ as a model satisfying θ. This is because there may be other properties of $\mathcal{M}$, involving atoms not occurring in θ, the truth of which may be affected by the minimisation procedure and may be of importance later. So, we propose the following refinement: to keep a best possible record of the truth of each atom r not occurring in θ in the resulting reduced model $(\widetilde{\mathcal{M}|_\theta}, \tilde{s})$ by introducing, besides *true* and *false*, a third truth-value *both*, that will be assigned to r at each state in the collapsed model where original states with different truth values of r have been identified. Thus, the resulting refined model allows for 3-valued valuation of the truth of formulas involving such atomic propositions, that can be used for evaluating the truth of some formulas that contain them. We will not pursue systematically this idea here, but leave it to future work.

There are well-known efficient procedures for bisimulation minimisation based on partition refinement such as the Kanellakis-Smolka algorithm [13], optimized to the Paige-Tarjan algorithm in [16]. (For other, more involved and efficient algorithms see [14]; see also [1].) It is quite easy to refine most of these θ-bisimulation minimisation procedures to account for the refinements above, but for lack of space we will not spell out the details.

▶ **Example 8** (Example 6 continued)**.** Let us apply BR to the model $\mathcal{M}_1$ with respect to the language of the formulas θ_i of Example 6, i.e. over the set of atomic propositions $P = \{p, q\}$. The coarsest partition of the set of states corresponding to the maximal bisimulation relation in $\mathcal{M}_1|_P$ contains six clusters: $C_\square = \{s\}$, $C_\bullet = \{s_1, s_3\}$, $C_\circ = \{s_2, s_8\}$, $C_\times = \{s_4\}$, $C_\otimes = \{s_5\}$, $C_+ = \{s_6, s_7, s_9\}$. Note that, for instance, s_6 and s_9 are in the same cluster even if they do not agree on the valuation of the propositional letter r, as it does not belong to the language of our interest. These clusters of bisimilar states are visualized in Figure 2. The corresponding quotient model $\mathcal{M}'_1$, collapsing all states belonging to the same cluster into a unique state, is given in Figure 3.

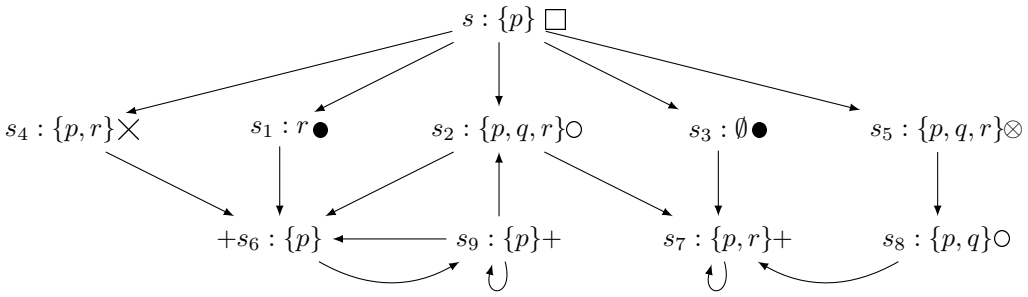


Figure 2 $\{p, q\}$-bisimilar states in the model $\mathcal{M}_1$.

3.4 Tableaux-based reduction (TR)

As explained earlier, the purpose of this reduction is to remove parts of the model that are unnecessary for satisfying the target input formula, typically when satisfying disjunctive choices and selecting successors. The input of the procedure TR is a pointed ITS $(\mathcal{M}, s)$

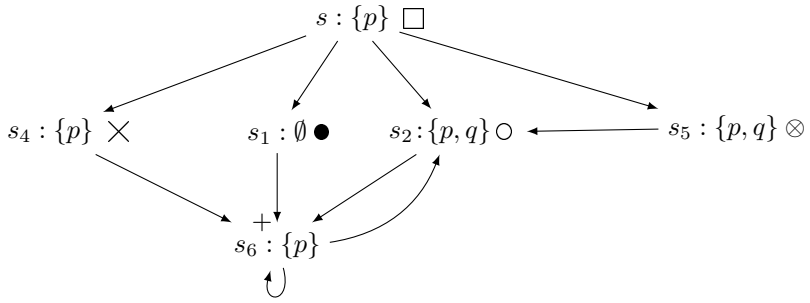


Figure 3 The bisimulation quotient model $\mathcal{M}'_1 = \widetilde{\mathcal{M}_1|_{\{p,q\}}}$

and a formula θ such that $M, s \models \theta$ is given/known to be true (our initial assumption). The output is a family of reduced pointed ITS $(\mathcal{M}_1, s), \ldots (\mathcal{M}_q, s)$ satisfying θ. Here is an informal outline of the overall procedure:

1. TR starts with a global model checking in $\mathcal{M}$ of the formulas in the extended closure $\mathsf{ecl}(\theta)$ of the input formula θ.
2. Then TR runs a tableau-like procedure that iteratively labels states of $\mathcal{M}$ with sets of formulas. At start, the root state s of M is labeled with $\{\theta\}$, while all other states have an empty label. Then labels are possibly modified repeatedly until stabilisation, according to a sub-procedure LAB that we outline later. A non-deterministic run of LAB produces a submodel $\mathcal{M}'$ of $\mathcal{M}$ with state space S' consisting of all states in S with non-empty labels.

   When all the possible runs of LAB are executed, in parallel or consecutively, a list of reduced pointed models $(\mathcal{M}_1, s), \ldots (\mathcal{M}_k, s)$ is produced.
3. Check for *subset inclusion*[1]: if $\mathcal{M}_i$ is included as a substructure in $\mathcal{M}_j$, then remove $\mathcal{M}_j$ from the list. The procedure eventually returns the family of minimal by inclusion reduced pointed models that remain in the list.

We are now going to describe more formally and precisely the procedure outlined above.

▶ **Definition 9.** *Let $(\mathcal{M}, s)$ be a pointed ITS and let Γ be a set of formulas that hold at s. A **(non-deterministic) optimal saturation** of Γ is a procedure OS that, when applied non-deterministically to Γ produces a set of formulas Δ such that $\Gamma \subseteq \Delta$ by repeatedly applying the following operations until saturation:*

1. *Initially, $\Delta := \Gamma$.*
2. *If a conjunctive formula φ is in Δ then OS adds both its components to Δ;*
3. *If a disjunctive formula φ is in Δ and none of its disjunctive components is in Δ, then OS chooses non-deterministically any of these components which is true at s and adds it to Δ. However, the following exception applies: if φ is an eventuality, i.e. $\mathsf{E}(\chi \,\mathsf{U}\, \psi)$, $\mathsf{EF}\, \psi$, $\mathsf{A}(\chi \,\mathsf{U}\, \psi)$, or $\mathsf{AF}\, \psi$, and none of its components is in Δ but ψ is true at s, then OS adds only ψ to Δ.*

*The sets Δ produced by runs of OS are called **(optimally) saturated extensions** of Γ. Γ is said to be **optimally saturated** if it equals an optimally saturated extension of itself.*

[1] More generally, TR can check for isomorphic embeddings, but that may increase substantially the complexity of the whole procedure.

The adjective "optimal" in the above definition is due to the third item, that minimizes the number of disjunctive components required to be true and aims at fulfilling eventualities as soon as possible. Note that if $\Gamma \subseteq \mathsf{ecl}(\theta)$ for a given formula θ and Δ is an optimally saturated extension of Γ, then $\Delta \subseteq \mathsf{ecl}(\theta)$. Moreover all the elements of Δ are true at s, by construction. In particular, so are all the successor formulas occurring in Δ.

▶ **Definition 10.** *Let $\mathcal{M}$ be an ITS, $s \in \mathcal{M}$, let Γ be an optimally saturated set of formulas true at s, and let $\Gamma_{suc} = \{\neg\mathsf{EX}\psi_1, ..., \neg\mathsf{EX}\psi_k, \mathsf{EX}\varphi_1, ..., \mathsf{EX}\varphi_m\}$ be its subset of successor formulas (where each of k and m can be 0). A **minimal set of successors of s** w.r.t. Γ_{suc} is a set U of states in $\mathcal{M}$ that are (immediate) successors of s and:*

1. *Each existential successor formula $\mathsf{EX}\varphi_j$ in Γ_{suc} has a "witness" in U, viz. some state $w(\varphi_j) \in U$ such that $\mathcal{M}, w(\varphi_j) \models \varphi_j$;*
2. *U is minimal with respect to the above property: if any state is removed from U then the resulting set S' lacks a witness for at least one $\mathsf{EX}\,\varphi_j \in \Gamma_{suc}$.*
3. *In case when $m = 0$, an arbitrary self-looping successor of s is added to U, just for the sake of seriality.*

By hypothesis, all formulas in Γ_{suc} are true at s. Therefore, for all $\neg\mathsf{EX}\,\psi_i \in \Gamma_{suc}$, the formula $\neg\psi_i$ is true at each state $s' \in U$.

The procedure ANALYSE given below takes as input an ITS, a state s in it, and a set of formulas $\mathcal{L}(s)$ currently labelling that state. It updates $\mathcal{L}(s)$ by saturating it and adding formulas to the current labels of some successors of s, to produce the updated labels as an output. The top procedure LAB calls ANALYSE.

The procedure ANALYSE**.**

1. Construct an optimal saturation Δ of $\mathcal{L}(s)$ and reset the value of $\mathcal{L}(s)$ to Δ.
2. If $\mathcal{L}(s)_{suc} = \{\neg\mathsf{EX}\psi_1, ..., \neg\mathsf{EX}\psi_k, \mathsf{EX}\varphi_1, ..., \mathsf{EX}\varphi_m\}$ is the subset of successor formulas of $\mathcal{L}(s)$, then build a minimal set U of successors of s w.r.t. $\mathcal{L}(s)_{suc}$.
3. For each $s' \in U$: if $s' = w(\varphi_j)$, then add φ_j and all $\neg\psi_i$, $1 \leq i \leq k$, to the current value of $\mathcal{L}(s')$ (if they are not already in it).

The procedure LAB**.**

1. Initialization: set s to be the current state, $\mathcal{L}(s) := \{\theta\}$ and $\mathcal{L}(s') := \emptyset$ for each other state of $\mathcal{M}$.
2. Until all labels $\mathcal{L}(s')$ of states s' of $\mathcal{M}$ become stable, do:
   a. Apply ANALYSE to the current state t.
   b. Then for each state t' in the minimal set of successors U of t produced by ANALYSE at t, set t' to be the current state and recursively apply ANALYSE there.

Note that, for the sake of simplicity, here we are giving the pseudo-code for a non-deterministic run of LAB. It can be converted to a deterministic algorithm, producing the entire family of reduced models, by using suitable bookkeeping and backtracking mechanisms.

▶ **Example 11** (Example 1 continued)**.** Let us apply LAB to the model $\mathcal{M}'_1$ of Figure 3 and the formula $\theta_1 = \phi_1 \vee \phi_2$ that holds at s. At the initialisation, $\mathcal{L}(s) = \{\theta_1\}$, while the labels of all other states are the empty set. Since both ϕ_1 and ϕ_2 are true at s, a non-deterministic run of LAB makes a choice of which of them to put in an optimized non-deterministic saturation of $\mathcal{L}(s)$. Consider two cases:

1. Suppose that the choice $\phi_1 = \mathsf{EX}\,p \wedge \mathsf{AF}\,(q \vee \mathsf{EF}\,p)$ is made. Then both conjunctive components of ϕ_1, $\mathsf{EX}\,p$ and $\mathsf{AF}\,(q \vee \mathsf{EF}\,p)$, are added to the saturation. The latter formula is an eventuality, whose disjunctive components are $q \vee \mathsf{EF}\,p$ and $\mathsf{AXAF}\,(q \vee \mathsf{EF}\,p)$. Here both components are true at s, but optimality forces us to choose $q \vee \mathsf{EF}\,p$. Now only $\mathsf{EF}\,p$ is true at s, so it is the chosen disjunctive component. In turn, $\mathsf{EF}\,p$ is an eventuality whose disjunctive components are p and $\mathsf{EXEF}\,p$. Since p is true at s then p is chosen. To summarise, the corresponding non-deterministic saturation of $\{\theta_1\}$ built here is the set $\{\theta_1,\ \phi_1,\ \mathsf{EX}\,p,\ \mathsf{AF}\,(q \vee \mathsf{EF}\,p),\ q \vee \mathsf{EF}\,p,\ \mathsf{EF}\,p,\ p\}$. It becomes the new value of $\mathcal{L}(s)$. Its set of successor formulas is $\{\mathsf{EX}\,p\}$, for which we obtain three minimal sets of successors of s, namely $\{s_2\}$, $\{s_4\}$ and $\{s_5\}$. A non-deterministic run of LAB chooses one of them, and adds the formula p to the corresponding state. In each of the three cases, the analysis of the newly labeled state produces no new label and the run halts, respectively producing: the sub-model $\mathcal{M}_a$ of $\mathcal{M}'_1$ containing just the states $\{s, s_2\}$, the sub-model $\mathcal{M}_b$ containing just the states $\{s, s_4\}$, and the sub-model $\mathcal{M}_c$ containing just the states $\{s, s_5\}$ (with loops, respectively, on s_2, s_4 and s_5).
2. Suppose now that the choice $\phi_2 = \mathsf{EX}\,\neg p \wedge \mathsf{EX}\,q \wedge \mathsf{AG}\,(q \to p)$ is made.
   Reasoning as above, by choosing suitable minimal sets of successors, we get:
   - either a candidate model having s, s_1 and s_2 as states, hence strictly including $\mathcal{M}_a$, and therefore excluded as a true minimal model by the inclusion-check that follows the application of LAB procedure in TR,
   - or, a candidate model that strictly includes $\mathcal{M}_c$ and is also disregarded.

   Hence, after the inclusion-check, the complete run of TR on $\mathcal{M}'_1$ produces the family of reduced models consisting of $\mathcal{M}_a$, $\mathcal{M}_b$ and $\mathcal{M}_c$.

▶ **Example 12** (Example 7 continued)**.** Consider the model $\mathcal{M}_2$ of Figure 4 that satisfies $\psi = \mathsf{EX}\,(\neg p \wedge \mathsf{EX}\,(p \wedge \mathsf{EX}\,(p \wedge q)))$ at s. An application of the procedure BR w.r.t. the set of propositions $\{p, q\}$ identifies states s_1 and s_6 as bisimilar, producing the model $\mathcal{M}'_2$ described in Figure 4. Then, running TR on that model and ψ removes s_5 and produces the model $\mathcal{M}''_2$ described in Figure 4. The states s_3 and s_4 are now bisimilar, so a new application of BR to $\mathcal{M}''_2$ is necessary. It produces the model $\mathcal{M}^*_2$ of Figure 4, where s_3 and s_4 are now collapsed into one state. Such a model of $\mathsf{EX}\,(\neg p \wedge \mathsf{EX}\,(p \wedge \mathsf{EX}\,(p \wedge q)))$ cannot be further reduced. This example shows that *the procedure* BR *may have to be applied again after an application of* TR *in order to minimise further the model.*

4 Analysis and results

4.1 Minimisation procedures running BR and TR together

The examples run so far show that it may be necessary to alternate the procedures BR and TR in order to produce truly minimal models of the target formula. Indeed, none of the two procedures subsumes the other in terms of the outcomes. This can be seen by a simple example. Take, for instance, $\mathcal{M}$ to be the model $\mathcal{M}'_2$ of Figure 4 and $\psi = \mathsf{EX}\,(\neg p \wedge \mathsf{EX}\,(p \wedge \mathsf{EX}\,(p \wedge q)))$, as in Example 12. If we run again BR on this input, we trivially get again $\mathcal{M}'_2$, since $\mathcal{M}'_2$ is already minimal with respect to ψ-bisimulation. However, running TR on $\mathcal{M}'_2$ and ψ produces the model $\mathcal{M}''_2$ shown in Figure 4. Thus, the two results are incomparable. More generally, observe also that both BR and TR are idempotent, i.e. neither of them produces new models if applied consecutively twice. These suggest that a minimising procedure might either start with BR and then alternate TR and BR phases (on the input produced by the previous phase) until stabilisation, or else start with TR and then alternate BR and TR phases until stabilisation. However we can bound the number of such alternations until stabilisation in both cases, due to the following result.

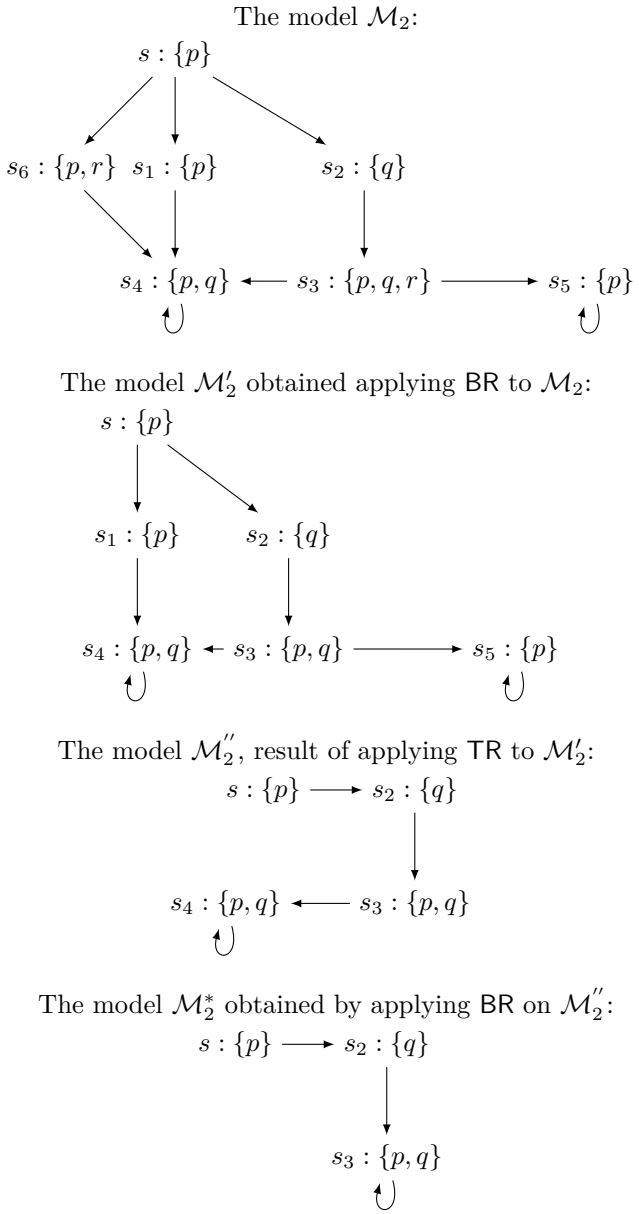


Figure 4 A complete reduction of the model $\mathcal{M}_2$.

▶ **Lemma 13.** *The reduction* TR *has to be applied only at most once, that is: given a pointed model* $(\mathcal{M}_1, t)$ *and a formula* θ*, let* $(\mathcal{M}_2, t)$ *be a reduced model produced by a run of* TR *on* $\mathcal{M}_1$ *and let* $(\mathcal{M}_3, \tilde{t})$ *be the result of running* BR *on* $(\mathcal{M}_2, t)$*. Then any run of* TR *on* $(\mathcal{M}_3, \tilde{t})$*,* θ *produces again* $(\mathcal{M}_3, \tilde{t})$ *as a result.*

Proof. Note that TR only removes states from its input model if they remain with empty labels. So, it suffices to observe that, if any formula $\phi \in \mathsf{ecl}(\theta)$ was added by the first run of TR to the label of a state $s \in \mathcal{M}_2$, then the same formula will be added to the label of the respective collapse state $\tilde{s} \in \mathcal{M}_3$ produced by applying BR to $\mathcal{M}_2$, and therefore that state will be preserved in the application of TR to $\mathcal{M}_3$. The proof can be done by tracing step by step the run of TR on $\mathcal{M}_1$ producing $\mathcal{M}_2$ and the respective run of TR on $\mathcal{M}_2 = \widetilde{\mathcal{M}_2}$. We omit the routine details. ◀

Therefore, there are only two different ways to organize the whole procedure:

MMP1: Start with TR, then apply BR to each obtained model.

MMP2: Start with BR, then apply TR to the obtained model, then again BR to each resulting model.

▶ **Example 14** (Example 12 continued)**.** In Example 12, we have actually run MMP2 on the model $\mathcal{M}_2$ (Figure 4) and the formula $\psi = \mathsf{EX}(\neg p \wedge \mathsf{EX}(p \wedge \mathsf{EX}(p \wedge q)))$. If we rather run MMP1 on the model $\mathcal{M}_2$, TR immediately produces the model $\mathcal{M}_2''$, then an application of BR to such a model makes s_3 and s_4 collapse and produces the minimal model $\mathcal{M}_2^*$.

4.2 Convergence and comparison of MM1 and MM2

▶ **Lemma 15.** *Given any pointed model $(\mathcal{M}, s)$ and a formula θ, every reduced pointed model produced from $(\mathcal{M}, s)$ by applying first* BR *and then* TR *can also be produced by applying first* TR *and then* BR*.*

Proof. Let $(\widetilde{\mathcal{M}}, \widetilde{s})$ be produced from $(\mathcal{M}, s)$ by applying BR and let $(\widetilde{\mathcal{M}}', \widetilde{s})$ be produced from $(\widetilde{\mathcal{M}}, \widetilde{s})$ by applying TR. It suffices to note that every run of procedure TR applied to $(\widetilde{\mathcal{M}}, \widetilde{s})$ to produce $(\widetilde{\mathcal{M}}', \widetilde{s})$ can be simulated, step by step, by a run of TR applied to $(\mathcal{M}, s)$, by selecting at every step a set of successors which are respectively θ-bisimulation equivalent to successors selected at the respective step of the run of TR applied to $(\widetilde{\mathcal{M}}, \widetilde{s})$. That would eventually produce a pointed model, on which BR would produce $(\widetilde{\mathcal{M}}', \widetilde{s})$. ◀

▶ **Theorem 16.** *For every initial pointed model $(\mathcal{M}, s)$ and a given formula φ:*

1. ***MMP1*** *and* ***MMP2*** *produce the same families of reduced models.*
2. *Every reduced pointed model produced by either of* ***MMP1*** *and* ***MMP2*** *is minimal in the following senses:*
   a. Bisimulation-minimal *with respect to the language of φ.*
   b. State-minimal*, in the sense that no state can be removed from $\mathcal{M}$ to still preserve the truth of φ at s.*

Proof. We first prove the second claim. The bisimulation-minimality is immediate, as both procedures end with BR. The state minimality follows from the minimality of every set of successors preserved by TR, and using Lemma 13.

Now, the first claim. First, every reduced pointed model produced by **MMP2** can also be produced by **MMP1**, by Lemma 15 and the idempotency of BR. For the converse inclusion, note that every run ρ of TR applied to a pointed model $(\mathcal{M}, s)$ and input formula θ can be lifted to a run $\widetilde{\rho}$ of TR on the θ-bisimulation quotient $(\widetilde{\mathcal{M}}, \widetilde{s})$ by selecting there the respective clusters of the selected successors in $(\mathcal{M}, s)$. Eventually, applying again BR to the resulting submodel $(\widetilde{\mathcal{M}}', \widetilde{s})$ would produce the same θ-bisimulation quotient as BR applied to the submodel of $(\mathcal{M}, s)$ produced by the run ρ of TR. ◀

We note that neither of the procedures **MMP1** and **MMP2** is intended, nor guaranteed, to produce a *smallest* possible model of the input formula, but only to minimise the input model in the senses described above. Indeed, e.g. the formula ψ in Example 12 has a smaller model than the model $\mathcal{M}_2^*$ in f Figure 4 that was obtained from $\mathcal{M}_2$ by the reduction procedure: a model with just two states, s, having label $\{p\}$, and its looping successor s_2 having label $\{p, q\}$.

We end this section with some complexity analysis. First, note that, despite the equivalence, the procedures **MMP1** and **MMP2** may have quite different performances. For instance, the deterministic version of **MMP1** can take in some cases an exponentially larger number of steps than **MMP2**, as shown by the following example.

▶ **Example 17.** Let θ be a formula of the form $\mathsf{EX}...\mathsf{EX}\,p$, where EX occurs n times, and let $\mathcal{M}$ be a pointed model that is a fully balanced binary tree of height n, satisfying p at each leaf and where all the states at the same level are bisimilar. Note that **MMP2**, starting with BR, will collapse all branches into one, and then TR will not make any change. On the other hand, **MMP1**, starting with TR, will produce 2^n isomorphic reduced models, each of them being a branch in the original model, i.e. a linear chain of length n. After checking for isomorphisms at the end, TR will leave just one of them, which BR will not change.

Now, to analyse the complexity, we can focus on the procedure **MMP1**, taking as inputs a formula θ and a pointed model $(\mathcal{M}, s)$, and returning a set[2] of minimal reduced models. **MMP1** first computes $\mathsf{ecl}(\theta)$ and does global model checking of all formulas in it in $\mathcal{M}$, in time linear in both $|\theta|$ (the size of θ) and $|\mathcal{M}|$ (the size of $\mathcal{M}$). Then, a non-deterministic run of the sub-procedure LAB in the worst case treats all formulas in $\mathsf{ecl}(\theta)$ and visits all the states in $\mathcal{M}$. Thus, it runs in time polynomial in $|\theta|$ and $|\mathcal{M}|$. Eventually, it produces a family of (possibly exponentially many, as evident from Example 17) minimal submodels, but for the sake of comparing and selecting the smallest of them, they can be produced consecutively, thus reusing space. Thus, TR can produce its output consecutively, in PSPACE. Bisimulation reduction of each of the models obtained by TR can be done in $O(m\ log\ n)$[16], where m is the number of transitions and n is the number of states of the model. Thus, finally, it takes polynomial space to produce every reduced pointed model consecutively, as an output of **MMP1**. A similar complexity analysis applies to **MMP2**.

5 Further work and concluding remarks

We have proposed a formula-oriented minimization procedure in two versions, **MMP1** and **MMP2**, that reduces the number of states of a model M satisfying a given CTL formula θ, by taking into account both possible θ-bisimulation redundancies as well as redundancies induced by the structure of θ. Using a tableau-like procedure for handling the second type of redundancies and combining the two kinds of reduction procedures are the main original ideas of our contribution. As already observed in the literature, to reduce the size of components with respect to their corresponding specification formulas can help to tackle the space explosion problems of product transition systems.

Our approach is related to, but different from, [15], as mentioned in the introduction. Not only we do not modify CTL syntax, but our notion of minimality is different and we solve a different algorithmic problem, too. Indeed, a formula $\phi_1 \Xi \phi_2$ in [15] holds at a state s of a model $\mathcal{M}$ when there is a minimal (and conservative, as defined in that work) *sub-structure* of $\mathcal{M}$ verifying ϕ_2 at s that verifies also ϕ_1. Here, minimality is with respect to an ordering of sub-structures of $\mathcal{M}$. In our case, minimisation includes also bisimulation reduction. Thus, for instance, consider again the rooted model $(\mathcal{M}_1, s)$ and the formula θ_1 of Example 6 and let θ_1' be $\theta_1 \wedge \mathsf{EX}\,\mathsf{EX}\,(p \wedge \neg q)$. Then running BR produces the quotient model $\mathcal{M}_1'$ exhibited by Figure 3, then a run of TR gives the model whose states are s, s_5, s_6 (with s connected to s_5, s_5 connected to s_6 and a loop on s_6). The latter is *not* a sub-structure of $\mathcal{M}_1$, and model-checking the formula $\top \Xi \theta_1'$ of the logic in [15] cannot produce it.

Future work includes extending our approach to model minimization to richer logics, in particular to the multi-agent extension ATL of CTL, whose models are minimized only with respect to (alternating) bisimulation in [5]. We also intend to implement **MMP1** and **MMP2** and to test experimentally and compare their performance in practical cases.

[2] Thus, the minimisation problem that this procedure solves is not a decision problem.

References

1 Adnan Aziz, Thomas Shiple, Vigyan Singhal, Robert Brayton, and Alberto Sangiovanni-Vincentelli. Formula-Dependent Equivalence for Compositional CTL Model Checking. *Formal Methods in System Design*, 21(2):193–224, 2002.

2 A. Bouajjani, J.-C. Fernandez, N. Halbwachs, P. Raymond, and C. Ratel. Minimal state graph generation. *Science of Computer Programming*, 18(3):247–269, 1992.

3 Ahmed Bouajjani, Jean-Claude Fernandez, and Nicolas Halbwachs. Minimal Model Generation. In *Proc of CAV '90*, pages 197–203, 1990.

4 Laura Bozzelli and David Pearce. On the Expressiveness of Temporal Equilibrium Logic. In *Proc. of JELIA 2016*, pages 159–173, 2016.

5 Serenella Cerrito and Amélie David. Minimisation of ATL* Models. In *Proc. of TABLEAUX 2017*, pages 193–208, 2017.

6 Massimiliano Chiodo, Thomas R. Shiple, Alberto L. Sangiovanni-Vincentelli, and Robert K. Brayton. Automatic compositional minimization in CTL model checking. In *Proc. of ICCAD'1992*, pages 172–178, 1992.

7 E. Clarke and E.A. Emerson. Design and synthesis of synchronisation skeletons using branching time temporal logic. In *Logics of Programs*, pages 52–71. Springer, 1981.

8 Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors. *Handbook of Model Checking*. Springer, 2018. doi:10.1007/978-3-319-10575-8.

9 E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

10 Stéphane Demri, Valentin Goranko, and Martin Lange. *Temporal Logics in Computer Science*, volume 58 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, October 2016. URL: http://www.cambridge.org/9781107028364.

11 V. Goranko and M. Otto. Model Theory of Modal Logic. In *Handbook of Modal Logic*, pages 249–330. Elsevier, 2007.

12 Susanne Graf and Bernhard Steffen. Compositional minimization of finite state systems. In Edmund M. Clarke and Robert P. Kurshan, editors, *Computer-Aided Verification*, pages 186–196, Berlin, Heidelberg, 1991. Springer.

13 Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.

14 Aceto L., Ingolfsdottir, and Jiri S. The Algorithmics of Bisimilarity. In Sangiorgi D. and Rutten J., editors, *Advanced topics in bisimulation and coinduction*, pages 100–171. Cambridge University Press, 2012.

15 Fabio Mogavero and Aniello Murano. Branching-Time Temporal Logics with Minimal Model Quantifiers. In *Developments in Language Theory, 13th International Conference, DLT 2009, Stuttgart, Germany, June 30 - July 3, 2009. Proceedings*, pages 396–409, 2009. doi:10.1007/978-3-642-02737-6_32.

16 R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

17 Thomas R. Shiple, Massimiliano Chiodo, Alberto L. Sangiovanni-Vincentelli, and Robert K. Brayton. Automatic Reduction in CTL Compositional Model Checking. In *Proc. of CAV'92*, pages 234–247, 1992.

18 Colin Stirling. Bisimulation and Logic. In Sangiorgi D. and Rutten J., editors, *Advanced topics in bisimulation and coinduction*, pages 173–195. Cambridge University Press, 2012.

On the Utility of Neighbourhood Singleton-Style Consistencies for Qualitative Constraint-Based Spatial and Temporal Reasoning

Michael Sioutis[1]
Department of Computer Science, Aalto University, Espoo, Finland
https://msioutis.gitlab.io/
michael.sioutis@aalto.fi

Anastasia Paparrizou
CRIL CNRS UMR 8188, Artois University, Lens, France
http://www.cril.univ-artois.fr/~paparrizou/
paparrizou@cril.fr

Tomi Janhunen
Department of Computer Science, Aalto University, Espoo, Finland
Tampere University, Tampere, Finland
https://users.ics.aalto.fi/ttj/
tomi.janhunen@aalto.fi

Abstract

A singleton-style consistency is a local consistency that verifies if each base relation (atom) of each constraint of a qualitative constraint network (QCN) can serve as a support with respect to the closure of that network under a (naturally) weaker local consistency. This local consistency is essential for tackling fundamental reasoning problems associated with QCNs, such as the satisfiability checking or the minimal labeling problem, but can suffer from redundant constraint checks, especially when those checks occur far from where the pruning usually takes place. In this paper, we propose singleton-style consistencies that are applied just on the neighbourhood of a singleton-checked constraint instead of the whole network. We make a theoretical comparison with existing consistencies and consequently prove some properties of the new ones. In addition, we propose algorithms to enforce our consistencies, as well as parsimonious variants thereof, that are more efficient in practice than the state of the art. We make an experimental evaluation with random and structured QCNs of Interval Algebra in the phase transition region to demonstrate the potential of our approach.

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

Keywords and phrases Qualitative constraints, spatial and temporal reasoning, singleton-style consistencies, neighbourhood, minimal labeling problem

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.14

1 Introduction

Qualitative Spatial and Temporal Reasoning (QSTR) is a Symbolic AI approach that deals with the fundamental cognitive concepts of space and time in a qualitative, human-like, manner [28, 16]. For instance, in natural language one uses expressions such as *inside*, *before*, and *north of* to spatially or temporally relate one object with another object or oneself, without resorting to providing quantitative information about these entities. QSTR provides a concise framework that allows for rather inexpensive reasoning about entities located in

[1] Corresponding author

space and time and, hence, further boosts research and applications to a plethora of areas and domains that include, but are not limited to, dynamic GIS [7], cognitive robotics [18], deep learning [25], and qualitative model generation from video [15]. The interested reader may look into a more comprehensive review of the emerging applications, the trends, and the future directions of QSTR in [6].

The problem of representing and reasoning about qualitative spatial or temporal information can be modeled as a *qualitative constraint network* (QCN), i.e., a network of qualitative constraints corresponding to spatial or temporal relations between spatial or temporal variables respectively. Two fundamental reasoning problems associated with a given QCN $\mathcal{N}$ are the problems of *satisfiability checking* and *minimal labeling* (or *deductive closure*) [37]. In particular, the satisfiability checking problem is about deciding if there exists a valuation of the variables of $\mathcal{N}$ that satisfies its constraints, and the minimal labeling problem concerns finding the strongest implied constraints and consequently obtaining its minimal sub-network. In general, for most well-known spatio-temporal calculi the satisfiability checking problem is NP-hard [17]. Further, the minimal labeling problem is polynomial-time Turing reducible to the satisfiability checking problem [20].

Motivation

In this paper, we focus mostly on the minimal labeling problem, which, since its introduction in 1974 by Montanari [31], has been studied in the domain of both CSPs [21, 46] and QCNs [19, 30]. As noted in [21], *a minimal network is a quite useful knowledge compilation, since it allows one to answer a number of queries in polynomial time that would otherwise be* NP*-hard*; indeed, in the context of QSTR, for instance, one could exploit minimality of a QCN to immediately deduce whether a task A could be scheduled before a task B, or an object X could be placed on top of an object Y. Difficult problems such as the minimal labeling problem and alike are, in general, either approximated by the use of local consistencies [46] or even solved by the aid of such consistencies [2]. Among the local consistencies introduced in the literature, we study *singleton-style consistencies* in the aforementioned context, which are consistencies that entail support for each base relation (atom) of the constraints of a QCN with respect to the closure of that network under a weaker local consistency (typically $\overset{\diamond}{}_G$-consistency [10, 35]). Specifically, we investigate how these consistencies behave when the underlying weaker local consistency that they build upon is restricted to the neighbourhood of a singleton-checked constraint. As noted in [47], neighbourhood-based restrictions can hit the sweet spot between effectiveness and efficiency in singleton-style consistencies for CSPs; therefore, it is imperative that we introduce and study such restrictions in the context of QCNs as well, and consequently provide a foundation for further work in understanding this kind of network structures, which have received much attention over the past years [16].

Contributions

Our contributions are fourfold and are described as follows:

- **(i)** we enrich the family of consistencies for QCNs by proposing singleton-style consistencies that are applied just on the neighbourhood of the singleton-checked constraint instead of the entire network;
- **(ii)** we theoretically obtain a strength-based hierarchy among existing consistencies for QCNs and the novel ones;
- **(iii)** we present algorithms to enforce the proposed consistencies for QCNs, as well as parsimonious variants thereof;

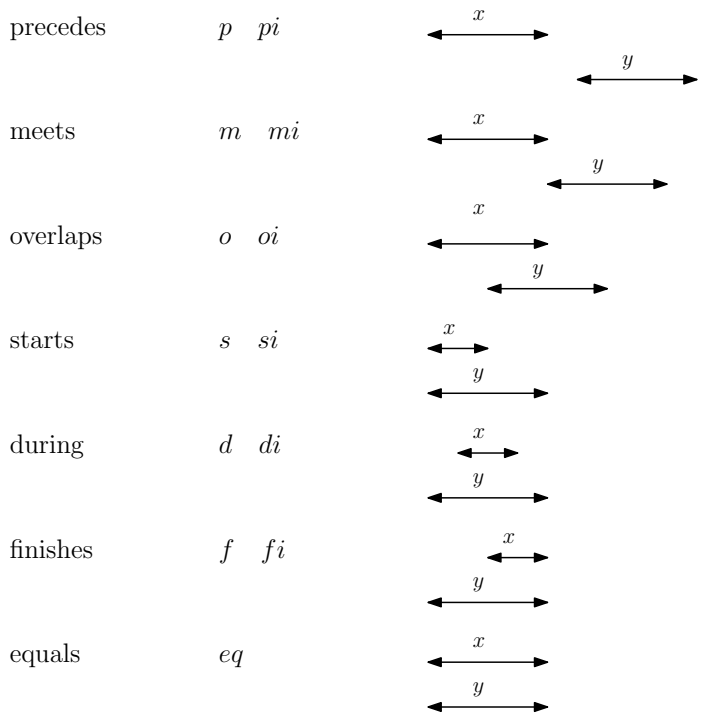


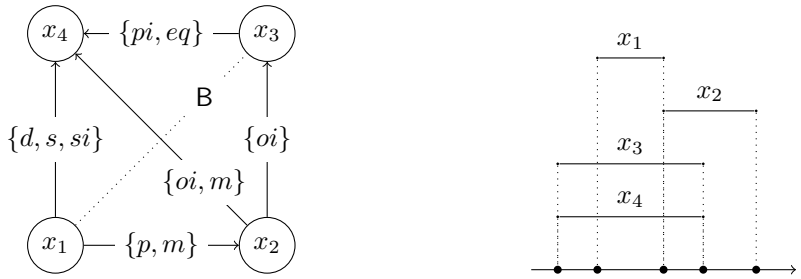
Figure 1 The base relations of IA; $\cdot i$ denotes the converse of $\cdot$.

(iv) we make an experimental evaluation with random and structured QCNs of Interval Algebra to measure and compare the performance of all considered algorithms, especially in terms of how fast and how well they can independently approximate the minimal sub-network of a QCN.

The rest of the paper is organized as follows. In Section 2 we give some preliminaries on qualitative spatial and temporal reasoning. Next, in Section 3 we overview some known state-of-the-art local consistencies for QCNs. Then, in Section 4 we introduce, formally define, and thoroughly study the proposed neighbourhood-based consistencies for QCNs, and present the algorithms for enforcing these consistencies, as well as parsimonious variants thereof. In Section 5 we evaluate our approach with random and structured QCNs of Interval Algebra and comment on the outcome; one finding is that neighbourhood-focused singleton-style algorithms are $\sim 30\%$ faster in the phase transition region than the standard algorithms. Finally, in Section 6 we draw some conclusive remarks and give 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 the set of *base relations* [29], that is defined over an infinite domain D. The base relations of a particular qualitative constraint language can be used to represent the definite knowledge between any two of its entities with respect to the level of granularity provided by the domain D. The set B contains the identity relation Id, and is closed under the *converse* operation $(^{-1})$. Indefinite knowledge can be specified by a union of possible base relations, and is represented by the set containing them. Hence, 2^{B} represents the total set of relations. The set 2^{B} is equipped with the usual set-theoretic



(a) A satisfiable QCN $\mathcal{N}$. **(b)** A solution σ of $\mathcal{N}$.

Figure 2 Figurative examples of QCN terminology using IA.

operations of union and intersection, the converse operation, and the *weak composition* operation denoted by the symbol $\diamond$ [29]. For all $r \in 2^{\mathsf{B}}$, we have that $r^{-1} = \bigcup\{b^{-1} \mid b \in r\}$. The weak composition ($\diamond$) of two base relations $b, b' \in \mathsf{B}$ is defined as the smallest (i.e., strongest) relation $r \in 2^{\mathsf{B}}$ that includes $b \circ b'$, or, formally, $b \diamond b' = \{b'' \in \mathsf{B} \mid b'' \cap (b \circ b') \neq \emptyset\}$, where $b \circ b' = \{(x, y) \in \mathsf{D} \times \mathsf{D} \mid \exists z \in \mathsf{D}$ such that $(x, z) \in b \wedge (z, y) \in b'\}$ is the (true) composition of b and b'. For all $r, r' \in 2^{\mathsf{B}}$, we have that $r \diamond r' = \bigcup\{b \diamond b' \mid b \in r, b' \in r'\}$.

As an illustration, consider the well-known qualitative temporal constraint language of Interval Algebra (IA), introduced by Allen [1]. IA considers time intervals (as temporal entities) and the set of base relations $\mathsf{B} = \{eq, p, pi, m, mi, o, oi, s, si, d, di, f, fi\}$ to encode knowledge about the temporal relations between intervals on the timeline, as depicted in Figure 1. Specifically, each base relation represents a particular ordering of the four endpoints of two intervals on the timeline, and eq is the identity relation Id.

Notably, most of the well-known and well-studied qualitative constraint languages, such as Interval Algebra [1] and RCC8 [34], are in fact *relation algebras* [17]. In what follows, we restrict ourselves to such calculi in order to facilitate discussion of the consistencies and of the algorithms for enforcing them.

The problem of representing and reasoning about qualitative spatial or temporal information can be modeled as a *qualitative constraint network*, defined in the following manner:

▶ **Definition 1.** A *qualitative constraint network* (QCN) is a tuple (V, C) where:
- $V = \{v_1, \ldots, v_n\}$ is a non-empty finite set of variables, each representing an entity of an infinite domain D;
- and C is a mapping $C : V \times V \to 2^{\mathsf{B}}$ such that $C(v, v) = \{\mathsf{Id}\}$ for all $v \in V$ and $C(v, v') = (C(v', v))^{-1}$ for all $v, v' \in V$, where $\bigcup \mathsf{B} = \mathsf{D} \times \mathsf{D}$.

An example of a QCN of IA is shown in Figure 2a; for clarity, converse relations as well as Id loops are not mentioned or shown in the figure.

▶ **Definition 2.** Let $\mathcal{N} = (V, C)$ be a QCN, then:
- a *solution* of $\mathcal{N}$ is a mapping $\sigma : V \to \mathsf{D}$ such that $\forall (u, v) \in V \times V$, $\exists b \in C(u, v)$ such that $(\sigma(u), \sigma(v)) \in b$ (see Figure 2b);
- $\mathcal{N}$ is *satisfiable* iff it admits a solution;
- a *sub*-QCN $\mathcal{N}'$ of $\mathcal{N}$, denoted by $\mathcal{N}' \subseteq \mathcal{N}$, is a QCN (V, C') such that $C'(u, v) \subseteq C(u, v)$ $\forall u, v \in V$; if in addition $\exists u, v \in V$ such that $C'(u, v) \subset C(u, v)$, then $\mathcal{N}' \subset \mathcal{N}$;
- a base relation $b \in C(v, v')$ with $v, v' \in V$ is *feasible* (resp. *unfeasible*) in $\mathcal{N}$ iff there exists (resp. there does not exist) a solution $\sigma : V \to \mathsf{D}$ of $\mathcal{N}$ such that $(\sigma(v), \sigma(v')) \in b$;
- $\mathcal{N}$ is *minimal* iff $\forall v, v' \in V$ and $\forall b \in C(v, v')$, b is a feasible base relation in $\mathcal{N}$;

- the *constraint graph* of $\mathcal{N}$, denoted by $\mathsf{G}(\mathcal{N})$, is the graph (V, E) where $\{u, v\} \in E$ iff $C(u, v) \neq \mathsf{B}$ and $u \neq v$;
- $\mathcal{N}$ is the *empty* QCN on V, denoted by $\bot^V$, iff $C(u, v) = \emptyset$ for all $u, v \in V$.

Let us further introduce the following operation that substitutes $C(v, v')$ with $r \in 2^{\mathsf{B}}$ in a given QCN:

- given a QCN $\mathcal{N} = (V, C)$ and $v, v' \in V$, we define that $\mathcal{N}_{[v,v']/r}$ with $r \in 2^{\mathsf{B}}$ yields the QCN $\mathcal{N}' = (V, C')$ defined by $C'(v, v') = r$, $C'(v', v) = r^{-1}$ and $C'(u, u') = C(u, u')$ $\forall (u, u') \in (V \times V) \setminus \{(v, v'), (v', v)\}$.

3 State-of-the-art Consistencies

We view a consistency ${}^{\phi}_{G}$, where ϕ is some operation (such as the *weak composition* operation) and G a graph, as a predicate on QCNs, i.e., a function that receives an input QCN and returns true or false depending on whether ${}^{\phi}_{G}$ holds on that QCN or not respectively. In what follows, given some operation ϕ (such as the weak composition operation) and a graph G, the unique $\subseteq$-maximal ${}^{\phi}_{G}$-consistent sub-QCN of $\mathcal{N}$ is called the *closure* of $\mathcal{N}$ under the consistency ${}^{\phi}_{G}$ and is denoted by ${}^{\phi}_{G}(\mathcal{N})$.

We recall the definition of ${}^{\diamond}_{G}$-consistency, which is a basic and widely used local consistency for reasoning with QCNs.

▶ **Definition 3.** Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V', E)$, where $V' \subseteq V$, $\mathcal{N}$ is said to be ${}^{\diamond}_{G}$*-consistent* iff $\forall \{v_i, v_j\}, \{v_i, v_k\}, \{v_k, v_j\} \in E$ we have that $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$.

Intuitively, ${}^{\diamond}_{G}$-consistency entails consistency for *all* triples of variables of a QCN that correspond to triangles of a given graph G. If G is the complete graph on the variables of a given QCN, then ${}^{\diamond}_{G}$-consistency becomes identical to $\diamond$-consistency [35], and, hence, $\diamond$-consistency can be seen as a special case of ${}^{\diamond}_{G}$-consistency.

In [39] the authors build upon ${}^{\diamond}_{G}$-consistency and propose a local consistency in the context of qualitative constraint-based reasoning that serves as the counterpart of *directional path consistency* in traditional constraint programming [14] or quantitative temporal reasoning [13], and is mainly distinguished by the fact that the involved consistency notions are tailored to handle infinite domains and qualitative relations. This local consistency is called ${}^{\overleftarrow{\diamond}}_{G}$*-consistency* and, in particular, it entails consistency for all *ordered* triples of variables of a QCN that correspond to triangles of a given graph G; this ordering can be specified by a bijection between the set of the variables of a QCN and a set of integers, and can be chosen randomly or via an algorithm or heuristic. We recall the formal definition of that consistency as follows:

▶ **Definition 4.** Given a QCN $\mathcal{N} = (V, C)$, an ordering $(\alpha^{-1}(0), \alpha^{-1}(1), \ldots, \alpha^{-1}(n-1))$ of V defined by a bijection $\alpha : V \to \{0, 1, \ldots, n-1\}$, and a graph $G = (V', E)$, where $V' \subseteq V$, $\mathcal{N}$ is said to be ${}^{\overleftarrow{\diamond}}_{G}$*-consistent* iff $\forall v_i, v_j, v_k \in V$ such that $\{v_i, v_j\}, \{v_i, v_k\}, \{v_k, v_j\} \in E$ and $\alpha(v_i), \alpha(v_j) < \alpha(v_k)$ we have that $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$.

Since ${}^{\overleftarrow{\diamond}}_{G}$*-consistency* is basically ${}^{\diamond}_{G}$-consistency restricted to some ordering of the triples of variables of a given QCN, it is expected that it will perform worse than ${}^{\diamond}_{G}$-consistency in terms of tackling the satisfiability checking or the minimal labeling problem of that QCN, in the general case. However, that behaviour of ${}^{\overleftarrow{\diamond}}_{G}$*-consistency* in the context of the aforementioned reasoning problems for *arbitrary* QCNs has yet to be investigated (cf. [40]), and we shall use this work as an opportunity to do so (see Section 5).

We continue with the presentation of some state-of-the-art singleton-style consistencies. Given a graph $G = (V', E)$, where $V' \subseteq V$, a QCN $\mathcal{N} = (V, C)$ is $\overset{\blacklozenge}{}_G$-consistent iff for every pair of variables $\{v, v'\} \in E$ and every base relation $b \in C(v, v')$, after instantiating $C(v, v')$ with $\{b\}$ as the singleton and applying $\overset{\diamond}{}_G$-consistency on $\mathcal{N}$, the revised constraint $C(v, v')$ is always supported by $\{b\}$. Formally, $\overset{\blacklozenge}{}_G$-consistency of a QCN is defined as follows:

▶ **Definition 5.** Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V', E)$, where $V' \subseteq V$, $\mathcal{N}$ is said to be $\overset{\blacklozenge}{}_G$-*consistent* iff $\mathcal{N}$ is $\overset{\diamond}{}_G$-consistent and $\forall\{v, v'\} \in E$ and $\forall b \in C(v, v')$ we have that $C'(v, v') = \{b\}$, where $(V, C') = \overset{\diamond}{}_G(\mathcal{N}_{[v,v']/\{b\}})$.

If G is the complete graph on the variables of a given QCN, we can easily verify that $\overset{\blacklozenge}{}_G$-consistency corresponds to $\overset{\diamond}{}_{\mathsf{B}}$-consistency of the family of $\overset{\diamond}{}_f$-consistencies studied in [11]. Interestingly, $\overset{\blacklozenge}{}_G$-consistency for QCNs can also be seen as a counterpart of singleton arc consistency (SAC) [12] for CSPs.

Finally, in [42] the authors define a local consistency that is more restrictive than any of the *practical*[2] local consistencies known to date for QCNs, called *collective* $\overset{\blacklozenge}{}_G$*-consistency*, or $\overset{\blacklozenge^{\cup}}{}_G$-consistency for short. This singleton-style consistency is inspired by *k-partitioning consistency* for CSPs [5]. We recall the formal definition of that consistency as follows:

▶ **Definition 6.** Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V', E)$, where $V' \subseteq V$, $\mathcal{N}$ is said to be $\overset{\blacklozenge^{\cup}}{}_G$-*consistent* iff $\mathcal{N}$ is $\overset{\blacklozenge}{}_G$-consistent and $\forall\{v, v'\} \in E$, $\forall b \in C(v, v')$, and $\forall\{u, u'\} \in E$ we have that $\exists b' \in C(u, u')$ such that $b \in C'(v, v')$, where $(V, C') = \overset{\diamond}{}_G(\mathcal{N}_{[u,u']/\{b'\}})$.

This underlying filtering condition of $\overset{\blacklozenge^{\cup}}{}_G$-consistency is based on relation partitioning combined with $\overset{\diamond}{}_G$-consistency, and allows for improved pruning capability over $\overset{\blacklozenge}{}_G$-consistency [42].

4 Neighbourhood Singleton-style Consistencies

In this section we propose and study a variety of singleton-style consistencies that are applied just on the neighbourhood of the singleton-checked constraint instead of the whole network.

Before doing so, let us first formally introduce a preorder in order to compare the pruning (or inference) capability of different consistencies. Let $\overset{\phi}{}_G$ and $\overset{\psi}{}_G$ be two consistencies defined by some operations ϕ and ψ respectively and a graph G. Then, $\overset{\phi}{}_G$ is *stronger* than $\overset{\psi}{}_G$ iff whenever $\overset{\phi}{}_G$ holds on a QCN $\mathcal{N}$ with respect to a graph G, $\overset{\psi}{}_G$ also holds on $\mathcal{N}$ with respect to G, and $\overset{\phi}{}_G$ is *strictly stronger* than $\overset{\psi}{}_G$ iff $\overset{\phi}{}_G$ is stronger than $\overset{\psi}{}_G$ and there exists at least one QCN $\mathcal{N}$ and a graph G such that $\overset{\psi}{}_G$ holds on $\mathcal{N}$ with respect to G, but $\overset{\phi}{}_G$ does not hold on $\mathcal{N}$ with respect to G. (The terms *weaker* and *strictly weaker* can be defined likewise.) Finally, $\overset{\phi}{}_G$ and $\overset{\psi}{}_G$ are *incomparable* iff there exist QCNs $\mathcal{N}$ and $\mathcal{N}'$ such that $\overset{\phi}{}_G$ is strictly stronger than $\overset{\psi}{}_G$ with respect to $\mathcal{N}$ and some graph G, and $\overset{\phi}{}_G$ is strictly weaker than $\overset{\psi}{}_G$ with respect to $\mathcal{N}'$ and some graph G (we note that the graph G can be different in the two cases).

In general, standard singleton-style consistencies can make a lot of redundant checks, which consequently can slow down their efficacy. It has been observed in the domain of CSPs that the majority of constraint revisions occur close to the relation that is being singleton checked, and rarely too far from it [47]. For that purpose, constraint programming researchers have proposed weaker singleton-style consistencies that localize propagation to the neighbourhood of the variable at hand [47, 33]. Neighbourhood singleton-style consistencies for CSPs, despite being strictly weaker than SAC [12] in general, can produce almost as much

[2] Clearly, in special cases notions like *k-consistency* can be defined and exploited theoretically [9], but these can be hardly implemented efficiently and are therefore not suitable for applications.

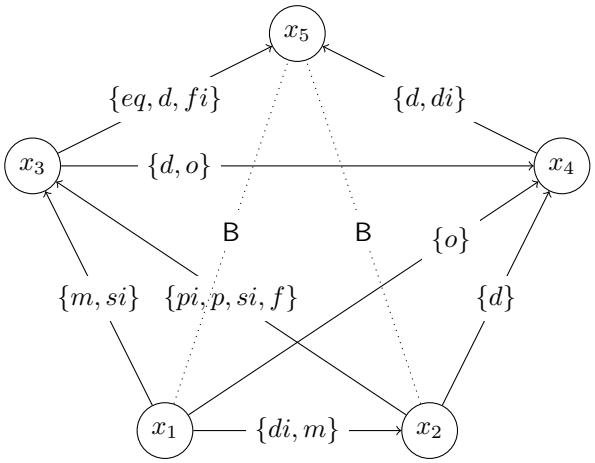


Figure 3 Given the QCN $\mathcal{N} = (V, C)$ above and the graph G that results by removing the edge $\{x_1, x_5\}$ from the complete graph on V, we have that $\mathcal{N}$ is neighbourhood $\overset{\blacklozenge}{}^{\cup}_{G}$-consistent (and neighbourhood $\overset{\blacklozenge}{}_{G}$-consistent), but not $\overset{\blacklozenge}{}_{G}$-consistent (or $\overset{\blacklozenge}{}^{\cup}_{G}$-consistent).

filtering as SAC with substantially less cost on many problems [33]. In what follows, we define two neighbourhood singleton-style consistencies for QCN*s*, and then we proceed to present algorithms and parsimonious variants thereof for applying these consistencies efficiently.

In order to define the new consistencies, we first need to define what exactly is meant by "neighbourhood of a relation" in the context of QCNs. Informally, given a QCN $\mathcal{N}$ and a graph G, the neighbourhood of a relation in $\mathcal{N}$ comprises all the triangles that involve the corresponding edge in G, and all the edges among the vertices of those triangles as well. Noting that in a given graph $G = (V, E)$, for each $u \in V$ the set of adjacent vertices of u, denoted by $\mathsf{adj}(u)$, is the set $\{w \mid \{u, w\} \in E\}$, we can formally define the neighbourhood of a relation of a QCN as follows:

▶ **Definition 7.** Given a QCN $\mathcal{N} = (V, C)$, a graph $G = (V', E)$, where $V' \subseteq V$, and two variables $v, v' \in V$ such that $\{v, v'\} \in E$, the *neighbourhood of* $C(v, v')$, denoted by $G_{\mathsf{N}(vv')}$, is the induced subgraph $G[S]$, where $S = (\mathsf{adj}(v) \cap \mathsf{adj}(v')) \cup \{v, v'\}$.

As an example, consider the QCN and its accompanying graph shown in Figure 3. The neighbourhood of $C(x_1, x_3)$ is the induced subgraph of the set of vertices $\{x_1, x_2, x_3, x_4\}$.

With the aforementioned definition in mind, we can define the notion of *neighbourhood* $\overset{\blacklozenge}{}_{G}$*-consistency* as follows:

▶ **Definition 8.** Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V', E)$, where $V' \subseteq V$, $\mathcal{N}$ is said to be *neighbourhood* $\overset{\blacklozenge}{}_{G}$*-consistent*, or $\mathsf{N}\overset{\blacklozenge}{}_{G}$-consistent for short, iff $\mathcal{N}$ is $\overset{\diamond}{}_{G}$-consistent and $\forall \{v, v'\} \in E$ and $\forall b \in C(v, v')$ we have that $C'(v, v') = \{b\}$, where $(V, C') = \overset{\diamond}{}_{G_{\mathsf{N}(vv')}}(\mathcal{N}_{[v,v']/\{b\}})$.

Similarly, we can define the notion of *neighbourhood* $\overset{\blacklozenge}{}^{\cup}_{G}$*-consistency* as follows:

▶ **Definition 9.** Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V', E)$, where $V' \subseteq V$, $\mathcal{N}$ is said to be *neighbourhood* $\overset{\blacklozenge}{}^{\cup}_{G}$*-consistent*, or $\mathsf{N}\overset{\blacklozenge}{}^{\cup}_{G}$-consistent for short, iff $\mathcal{N}$ is $\mathsf{N}\overset{\blacklozenge}{}_{G}$-consistent and $\forall \{v, v'\} \in E$, $\forall b \in C(v, v')$, and $\forall \{u, u'\} \in E$ we have that $\exists b' \in C(u, u')$ such that $b \in C'(v, v')$, where $(V, C') = \overset{\diamond}{}_{G_{\mathsf{N}(vv')}}(\mathcal{N}_{[u,u']/\{b'\}})$.

The reader can note that Definitions 8 and 9 mirror Definitions 5 and 6 respectively, the difference being that the closure under $\overset{\diamond}{}_G$-consistency is restricted to the neighbourhood of the constraint at hand.

We recall the following result from [42] in our effort here to build a strength-based hierarchy among all discussed consistencies:

▶ **Proposition 1** ([42]). *We have that $\overset{\blacklozenge^{\cup}}{}_G$-consistency is strictly stronger than $\overset{\blacklozenge}{}_G$-consistency.*

In the sequel, Figure 3 will be crucial in proving the results that follow.

▶ **Proposition 2.** *We have that $\overset{\blacklozenge^{\cup}}{}_G$-consistency is strictly stronger than $\mathsf{N}\overset{\blacklozenge^{\cup}}{}_G$-consistency.*

Proof. Consider the QCN along with its accompanying graph depicted in Figure 3. As noted in its caption the QCN is $\mathsf{N}\overset{\blacklozenge^{\cup}}{}_G$-consistent and $\mathsf{N}\overset{\blacklozenge}{}_G$-consistent, but not $\overset{\blacklozenge}{}_G$-consistent or $\overset{\blacklozenge^{\cup}}{}_G$-consistent. Specifically, in order for the QCN to become $\overset{\blacklozenge}{}_G$-consistent and $\overset{\blacklozenge^{\cup}}{}_G$-consistent, the base relation mi needs to be removed from $C(x_2, x_5)$. In addition, by the definitions of $\overset{\blacklozenge^{\cup}}{}_G$-consistency and $\mathsf{N}\overset{\blacklozenge^{\cup}}{}_G$-consistency, we have that every $\overset{\blacklozenge^{\cup}}{}_G$-consistent QCN is $\mathsf{N}\overset{\blacklozenge^{\cup}}{}_G$-consistent. Specifically, given a QCN $\mathcal{N}$ and two graphs G and G' such that $G \subseteq G'$, it holds that if $\mathcal{N}$ is $\overset{\diamond}{}_G$-consistent then $\mathcal{N}$ is $\overset{\diamond}{}_{G'}$-consistent. ◀

Following the same line of reasoning as that of the proof of Proposition 2, we assert the next result:

▶ **Proposition 3.** *We have that $\overset{\blacklozenge}{}_G$-consistency is strictly stronger than $\mathsf{N}\overset{\blacklozenge}{}_G$-consistency.*

We proceed with presenting the next result:

▶ **Proposition 4.** *We have that $\mathsf{N}\overset{\blacklozenge^{\cup}}{}_G$-consistency is strictly stronger than $\mathsf{N}\overset{\blacklozenge}{}_G$-consistency.*

Proof. Consider the QCN along with its accompanying graph depicted in Figure 3 in [42]. It is the case that the QCN is $\mathsf{N}\overset{\blacklozenge}{}_G$-consistent, but not $\mathsf{N}\overset{\blacklozenge^{\cup}}{}_G$-consistent. Additionally, by definition of $\mathsf{N}\overset{\blacklozenge^{\cup}}{}_G$-consistency, we have that every $\mathsf{N}\overset{\blacklozenge^{\cup}}{}_G$-consistent QCN is $\mathsf{N}\overset{\blacklozenge}{}_G$-consistent. ◀

We continue with another result as follows:

▶ **Proposition 5.** *We have that $\mathsf{N}\overset{\blacklozenge^{\cup}}{}_G$-consistency is incomparable to $\overset{\blacklozenge}{}_G$-consistency.*

Proof. Consider again the QCN along with its accompanying graph depicted in Figure 3 in [42]. It is the case that the QCN is $\overset{\blacklozenge}{}_G$-consistent, but not $\mathsf{N}\overset{\blacklozenge^{\cup}}{}_G$-consistent. On the other hand, as noted also in the proof of Proposition 2, the QCN of Figure 3 here is $\mathsf{N}\overset{\blacklozenge^{\cup}}{}_G$-consistent, but not $\overset{\blacklozenge}{}_G$-consistent, with respect to its accompanying graph. ◀

From Propositions 2 and 4 (or 1 and 3) we obtain the following result:

▶ **Corollary 1.** *We have that $\overset{\blacklozenge^{\cup}}{}_G$-consistency is strictly stronger than $\mathsf{N}\overset{\blacklozenge}{}_G$-consistency.*

Finally, to complete our strentgh-based hierarchy we close off with some results that involve the non-singleton-style consistencies $\overset{\diamond}{}_G$-consistency and $\overset{\overleftarrow{\diamond}}{}_G$-consistency.

▶ **Proposition 6.** *We have that $\mathsf{N}\overset{\blacklozenge}{}_G$-consistency is strictly stronger than $\overset{\diamond}{}_G$-consistency.*

Proof. Consider the QCN depicted in Figure 14 in [36], which was used to prove that $\diamond$-consistency cannot decide the minimality of a QCN in general. It is the case that the QCN is $\overset{\diamond}{}_G$-consistent, but not $\mathsf{N}\overset{\blacklozenge}{}_G$-consistent, with respect to the complete graph on the set of variables of that QCN. Notably, applying $\mathsf{N}\overset{\blacklozenge}{}_G$-consistency on that QCN makes it minimal. Additionally, by definition of $\mathsf{N}\overset{\blacklozenge}{}_G$-consistency, we have that every $\mathsf{N}\overset{\blacklozenge}{}_G$-consistent QCN is $\overset{\diamond}{}_G$-consistent. ◀

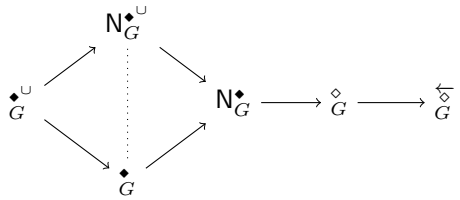


Figure 4 A strength-based hierarchy of consistencies for QCNs; an arrow denotes the (transitive) *strictly stronger* relationship and a dotted line the (symmetric) *incomparable* relationship.

Algorithm 1 $\mathsf{PSWC}^{\cup}_{\underline{\mathsf{N}}}(\mathcal{N}, G)$.

in : A QCN $\mathcal{N} = (V, C)$, and a graph $G = (V' \subseteq V, E)$.
out : A sub-QCN of $\mathcal{N}$.

begin
  $\mathcal{N} \leftarrow {}^{\diamond}_{G}(\mathcal{N})$;
  $Q \leftarrow list(E)$;
  while $Q \neq \emptyset$ **do**
    $\{v, v'\} \leftarrow Q.pop()$;
    $(V, C') \leftarrow \perp^{V}$;
    foreach $b \in C(v, v')$ **do**
      $(V, C') \leftarrow (V, C') \cup {}^{\diamond}_{G_{\underline{\mathsf{N}}(vv')}}(\mathcal{N}_{[v,v']/\{b\}})$;
    if $(V, C') \subset \mathcal{N}$ **then**
      $flag \leftarrow$ False;
      foreach $\{u, u'\} \in E$ **do**
        if $C'(u, u') \subset C(u, u')$ **then**
          $C(u, u') \leftarrow C'(u, u')$;
          $C(u', u) \leftarrow C'(u', u)$;
          $flag \leftarrow$ True;
      if $flag$ **then** $Q \leftarrow list(E)$;
  return $\mathcal{N}$;

From Propositions 1, 2, 3, 4, and 6 we obtain the following result:

▶ **Corollary 2.** *We have that each of the consistencies of* ${}^{\blacklozenge\cup}_{G}$*-consistency,* $\mathsf{N}^{\blacklozenge\cup}_{G}$*-consistency,* ${}^{\blacklozenge}_{G}$*-consistency, and* $\mathsf{N}^{\blacklozenge}_{G}$*-consistency is strictly stronger than* ${}^{\diamond}_{G}$*-consistency.*

From [40] we have the following result:

▶ **Proposition 7** ([40]). *We have that* ${}^{\diamond}_{G}$*-consistency is strictly stronger than* ${}^{\overleftarrow{\diamond}}_{G}$*-consistency.*

From Corollary 2 and Proposition 7 we obtain the following last result:

▶ **Corollary 3.** *We have that each of the consistencies of* ${}^{\blacklozenge\cup}_{G}$*-consistency,* $\mathsf{N}^{\blacklozenge\cup}_{G}$*-consistency,* ${}^{\blacklozenge}_{G}$*-consistency,* $\mathsf{N}^{\blacklozenge}_{G}$*-consistency, and* ${}^{\diamond}_{G}$*-consistency is strictly stronger than* ${}^{\overleftarrow{\diamond}}_{G}$*-consistency.*

A visual representation of the established strength-based hierarchy of consistencies is shown in Figure 4.

Algorithm 2 $\mathsf{PSWC}_{\underline{\mathsf{N}}}(\mathcal{N}, G)$.

in : A QCN $\mathcal{N} = (V, C)$, and a graph $G = (V' \subseteq V, E)$.
out : A sub-QCN of $\mathcal{N}$.
begin
  $\mathcal{N} \leftarrow \overset{\diamond}{G}(\mathcal{N})$;
  $Q \leftarrow list(E)$;
  while $Q \neq \emptyset$ **do**
    $\{v, v'\} \leftarrow Q.pop()$;
    $(V, C') \leftarrow \bot^V$;
    foreach $b \in C(v, v')$ **do**
      $(V, C') \leftarrow (V, C') \cup \overset{\diamond}{G}_{\underline{\mathsf{N}(vv')}}(\mathcal{N}_{[v,v']/\{b\}})$;
    if $C'(v, v') \subset C(v, v')$ **then**
      $C(v, v') \leftarrow C'(v, v')$;
      $C(v', v) \leftarrow C'(v', v)$;
      $Q \leftarrow list(E)$;
  return $\mathcal{N}$;

Algorithm 3 $\ell\mathsf{PSWC}^{\cup}_{\underline{\mathsf{N}}}(\mathcal{N}, G)$.

in : A QCN $\mathcal{N} = (V, C)$, and a graph $G = (V' \subseteq V, E)$.
out : A sub-QCN of $\mathcal{N}$.
begin
  $\mathcal{N} \leftarrow \overset{\diamond}{G}(\mathcal{N})$;
  $Q \leftarrow list(S \subseteq E)$;
  while $Q \neq \emptyset$ **do**
    $\{v, v'\} \leftarrow Q.pop()$;
    $(V, C') \leftarrow \bot^V$;
    foreach $b \in C(v, v')$ **do**
      $(V, C') \leftarrow (V, C') \cup \overset{\diamond}{G}_{\underline{\mathsf{N}(vv')}}(\mathcal{N}_{[v,v']/\{b\}})$;
    $C(v, v') \leftarrow C'(v, v')$;
    if $(V, C') \subset \mathcal{N}$ **then**
      foreach $\{u, u'\} \in E \setminus \{v, v'\}$ **do**
        if $C'(u, u') \subset C(u, u')$ **then**
          $C(u, u') \leftarrow C'(u, u')$;
          $C(u', u) \leftarrow C'(u', u)$;
          $Q.push(\{u, u'\})$;
  return $\mathcal{N}$;

Algorithms and Complexities

For the sake of completeness, we present here algorithms $\mathsf{PSWC}^{\cup}_{\mathsf{N}}$ and $\mathsf{PSWC}_{\mathsf{N}}$, shown in Algorithms 1 and 2 respectively, which given a QCN $\mathcal{N}$ and a graph G as input apply $\mathsf{N}^{\blacklozenge\cup}_{G}$-consistency and $\mathsf{N}^{\blacklozenge}_{G}$-consistency on $\mathcal{N}$ respectively. By dropping the red underlined parts in the aforementioned algorithms, the reader can verify that they fall back to algorithms $\mathsf{PSWC}^{\cup}$ and PSWC respectively, which were introduced in [42].

Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V', E)$, where $V' \subseteq V$, the worst-case time complexity for both $\mathsf{PSWC}_{\mathsf{N}}^{\cup}$ and $\mathsf{PSWC}_{\mathsf{N}}$ is $O(\alpha|\mathsf{B}|^2|E|^2)$, where α is the worst-case time complexity for computing $^{\diamond}_{G'}(\mathcal{N})$ with respect to the largest graph $G' \subseteq G$ that is used in Line 8 of the algorithms (as each constraint defines its own neighbourhood G'). For any given QCN $\mathcal{N} = (V, C)$ and a graph $G = (V', E)$, where $V' \subseteq V$, we note that α is $O(\Delta|\mathsf{B}||E|)$, where Δ is the maximum vertex degree of G [10].

Finally, given a QCN $\mathcal{N}$ and a graph G, a parsimonious variant for approximating $\mathsf{N}_G^{\blacklozenge\cup}$-consistency in $\mathcal{N}$ is algorithm $\ell\mathsf{PSWC}_{\mathsf{N}}^{\cup}$, shown in Algorithm 3. Again, by dropping the red underlined parts in the aforementioned algorithm, the reader can verify that it falls back to a slight generalization of algorithm $\ell\mathsf{PSWC}^{\cup}$, which was introduced in [41]. Specifically, contrary to the algorithm as it appears in [41], in Line 3 of Algorithm 3 we allow any subset S of the set of edges of the input graph to be used; this subset serves as the seed of constraints from which the singleton checks will start propagating themselves. Algorithm $\ell\mathsf{PSWC}_{\mathsf{N}}^{\cup}$ is lazy in the sense that it relies upon previously revised constraints to allow itself to continue propagation. Therefore, depending on the subset S to be used, and the order in which the constraints are processed, the algorithm may produce different outputs for the same input (see [41]).

The worst-case time complexity of $\ell\mathsf{PSWC}_{\mathsf{N}}^{\cup}$ is the same as that of $\mathsf{PSWC}_{\mathsf{N}}^{\cup}$ (and $\mathsf{PSWC}_{\mathsf{N}}$), although we will see later on in Section 5 that it is much faster in practice.

5 Experimental Evaluation

In this section we investigate the utility of the proposed neighbourhood singleton-style consistency algorithms, as well as the discussed state-of-the-art local consistency algorithms that appear in the literature, with respect to the fundamental reasoning problems of *satisfiability checking* and *minimal labeling* that are associated with QCNs. Specifically, we explore their efficiency in determining the satisfiability of a given network instance and in discovering the unfeasible base relations of that network instance (in regard to both CPU time and correctness of decision).

Technical specifications

The evaluation was carried out on a computer with an Intel Core i5-4570 processor, 16 GB of RAM, and the Xenial Xerus x86_64 OS (Ubuntu Linux). All algorithms were coded in Python and run using the PyPy intepreter under version 5.1.2, which implements Python 2.7.10. Only one CPU core was used.

Dataset

We used the dataset employed in [43]. That dataset comprises 1 000 random and structured QCNs of IA that were created using models $\mathsf{A}(\mathsf{n}, \mathsf{l}, \mathsf{d})$ [32] and $\mathsf{BA}(\mathsf{n}, \mathsf{m})$ [38] respectively. Pertaining to $\mathsf{A}(\mathsf{n}, \mathsf{l}, \mathsf{d})$, there are 100 QCNs of IA of $\mathsf{n} = 70$ variables and with $\mathsf{l} = 6.5$ base relations per non-universal constraint on average, for all values of the average constraint graph degree d from 7 to 12 with a step of 1. Pertaining to $\mathsf{BA}(\mathsf{n}, \mathsf{m})$, there are 100 QCNs of IA of $\mathsf{n} = 150$ variables for all values of the constraint graph *preferential attachment* m [4] from 2 to 5 with a step of 1. Finally, regarding the graphs that were given as input to our algorithms, the *maximum cardinality search* algorithm [45] was used to obtain triangulations of the constraint graphs of our QCNs. The choice of such chordal graphs was not only reasonable but also crucial given their important theoretical and practical implications in

qualitative constraint-based spatial and temporal reasoning, as reviewed in [44]; the use of those graphs itself was inspired by [23, 22, 27] among other works, where preliminary results pertainining to tree decompositions were established.

Tools

In addition to our implementations of the algorithms that were presented in Section 4, we utilized the following four software tools:[3]

- Solver, the state-of-the-art reasoner for checking the satisfiability of QCNs of Interval Algebra and RCC8 that was introduced in [38] (and in particular the reasoner called Phalanx$_{\triangledown}$ in that work);
- Minimizer, our own implementation of the approach for solving the minimal labeling problem of QCNs of Interval Algebra and RCC8 that was first presented in [2];[4]
- PWC, the state-of-the-art algorithm for applying $\overset{\diamond}{}_G$-consistency on QCNs of Interval Algebra and RCC8 that was used in [38] (which is a module of the Phalanx$_{\triangledown}$ reasoner mentioned earlier);
- DPWC, the state-of-the-art algorithm for applying $\overset{\leftarrow}{\overset{\diamond}{}}_G$-consistency on QCNs of Interval Algebra and RCC8 that was introduced in [40] (and in particular the reasoner called Pyrrhus in that work).

Results

The results of our experimental evaluation are detailed in Tables 1 and 2, where a fraction $\frac{x}{y}$ for Solver denotes that it required x seconds of CPU time on average per dataset of network instances during its operation and detected y such instances as being unsatisfiable in total, a fraction $\frac{x}{z}$ for Minimizer denotes that it determined $z\%$ of base relations to be unfeasible in total, and a fraction $\frac{x}{y|z}$ for the rest of the algorithms denotes all the previous information combined together (from the viewpoint of the respective algorithm).

The first thing to note is that Solver has no competition whatsoever in terms of deciding the satisfiability of a network instance. This was expected, as this type of reasoner is tailored to avoid "bad" branches in the search tree and to reach a leaf (i.e., a solution) in the most efficient way possible. On the other hand, when the entire search tree needs to be taken into account, as is the case with Minimizer, the computational task is much more time-consuming; therefore, Minimizer has by far the worst performance among its competition.

Regarding the singleton-style consistency algorithms, we can note that they of course have an overhead compared to Solver, but they are much faster in general than Minimizer and they can, in many cases, simulate its pruning capability in an almost exact manner. It is worth mentioning that the neighbourhood-focused singleton-style algorithms PSWC$^{\cup}_{\mathsf{N}}$ and PSWC$_{\mathsf{N}}$ are around 30% faster in the phase transition region than the standard algorithms PSWC$^{\cup}$ and PSWC respectively, whilst retaining much of the good performance characteristics (viz., unfeasible base relations elimination and satisfiability decision) of the latter. The

[3] These software tools are available at `https://msioutis.gitlab.io/software`.

[4] In particular, we ported the code to Python and included all recent advances that are associated with the components that comprise that approach, such as improvements in its underlying satisfiability checking module. It must also be noted that the strongest of the local consistencies discussed here, viz., $\overset{\blacklozenge^{\cup}}{}_G$-consistency, was used as a preprocessing step to enhance the performance of Minimizer.

Table 1 Evaluation with random IA networks that were generated using model $\mathsf{A}(\mathsf{n} = 70, \mathsf{l} = 6.5, \mathsf{d})$ [32].

d	Solver	Minimizer	$\mathsf{PSWC}^{\cup}$	$\mathsf{PSWC}^{\cup}_{\mathsf{N}}$	$\ell\mathsf{PSWC}^{\cup}$	$\ell\mathsf{PSWC}^{\cup}_{\mathsf{N}}$	PSWC	$\mathsf{PSWC}_{\mathsf{N}}$	PWC	DPWC
7	$\frac{0.16s}{2}$	$\frac{12.29s}{3.84\%}$	$\frac{2.54s}{2\|3.84\%}$	$\frac{2.27s}{2\|3.84\%}$	$\frac{0.44s}{2\|3.84\%}$	$\frac{0.40s}{2\|3.84\%}$	$\frac{3.00s}{2\|3.84\%}$	$\frac{2.72s}{2\|3.84\%}$	$\frac{0.00s}{2\|3.77\%}$	$\frac{0.00s}{2\|2.48\%}$
8	$\frac{0.17s}{5}$	$\frac{27.40s}{8.75\%}$	$\frac{9.92s}{5\|8.72\%}$	$\frac{7.80s}{5\|8.68\%}$	$\frac{1.96s}{5\|8.69\%}$	$\frac{1.58s}{5\|8.66\%}$	$\frac{11.22s}{5\|8.72\%}$	$\frac{9.64s}{5\|8.64\%}$	$\frac{0.00s}{4\|7.23\%}$	$\frac{0.00s}{3\|3.70\%}$
9	$\frac{0.29s}{6}$	$\frac{281.59s}{13.67\%}$	$\frac{24.80s}{6\|12.31\%}$	$\frac{17.47s}{6\|11.69\%}$	$\frac{4.69s}{6\|12.03\%}$	$\frac{3.41s}{6\|11.44\%}$	$\frac{28.23s}{6\|12.31\%}$	$\frac{20.96s}{6\|11.36\%}$	$\frac{0.01s}{1\|4.82\%}$	$\frac{0.00s}{0\|0.63\%}$
10	$\frac{1.77s}{55}$	$\frac{1\,541.88s}{70.57\%}$	$\frac{41.13s}{54\|64.13\%}$	$\frac{31.15s}{51\|57.06\%}$	$\frac{7.71s}{50\|57.82\%}$	$\frac{5.46s}{44\|49.93\%}$	$\frac{48.04s}{54\|64.11\%}$	$\frac{36.58s}{51\|56.45\%}$	$\frac{0.01s}{5\|10.02\%}$	$\frac{0.00s}{1\|1.92\%}$
11	$\frac{4.52s}{100}$	$\frac{5.99s}{100\%}$	$\frac{6.98s}{99\|98.97\%}$	$\frac{8.23s}{97\|97.06\%}$	$\frac{2.52s}{96\|96.13\%}$	$\frac{2.57s}{91\|91.40\%}$	$\frac{8.44s}{99\|98.97\%}$	$\frac{10.83s}{97\|97.06\%}$	$\frac{0.01s}{31\|34.93\%}$	$\frac{0.00s}{5\|6.07\%}$
12	$\frac{0.23s}{100}$	$\frac{0.96s}{100\%}$	$\frac{0.59s}{100\|100\%}$	$\frac{0.96s}{100\|100\%}$	$\frac{0.58s}{100\|100\%}$	$\frac{0.79s}{100\|100\%}$	$\frac{0.70s}{100\|100\%}$	$\frac{1.51s}{100\|100\%}$	$\frac{0.01s}{44\|47.91\%}$	$\frac{0.00s}{4\|5.09\%}$

Table 2 Evaluation with structured IA networks that were generated using model BA(n = 150, m) [38].

m	Solver	Minimizer	$PSWC^{\cup}$	$PSWC^{\cup}_{N}$	$\ell PSWC^{\cup}$	$\ell PSWC^{\cup}_{N}$	PSWC	$PSWC_{N}$	PWC	DPWC
2	$\frac{0.15s}{2}$	$\frac{6.57s}{3.14\%}$	$\frac{0.53s}{2\|3.14\%}$	$\frac{0.45s}{2\|3.14\%}$	$\frac{0.12s}{2\|3.14\%}$	$\frac{0.10s}{2\|3.14\%}$	$\frac{0.67s}{2\|3.14\%}$	$\frac{0.56s}{2\|3.14\%}$	$\frac{0.00s}{2\|3.12\%}$	$\frac{0.00s}{2\|2.26\%}$
3	$\frac{0.17s}{7}$	$\frac{34.95s}{9.42\%}$	$\frac{9.55s}{7\|9.42\%}$	$\frac{7.85s}{7\|9.40\%}$	$\frac{2.52s}{7\|9.41\%}$	$\frac{1.91s}{7\|9.40\%}$	$\frac{12.40s}{7\|9.42\%}$	$\frac{10.14s}{7\|9.38\%}$	$\frac{0.01s}{7\|8.82\%}$	$\frac{0.00s}{4\|3.92\%}$
4	$\frac{0.23s}{60}$	$\frac{101.33s}{66.89\%}$	$\frac{95.64s}{60\|66.83\%}$	$\frac{69.81s}{60\|66.39\%}$	$\frac{26.06s}{60\|66.64\%}$	$\frac{19.39s}{60\|66.24\%}$	$\frac{126.69s}{60\|66.83\%}$	$\frac{94.51s}{60\|65.77\%}$	$\frac{0.01s}{42\|44.61\%}$	$\frac{0.00s}{12\|12.06\%}$
5	$\frac{0.16s}{100}$	$\frac{0.71s}{100\%}$	$\frac{0.04s}{100\|100\%}$	$\frac{0.07s}{100\|100\%}$	$\frac{0.06s}{100\|100\%}$	$\frac{0.06s}{100\|100\%}$	$\frac{0.05s}{100\|100\%}$	$\frac{0.07s}{100\|100\%}$	$\frac{0.01s}{92\|92.32\%}$	$\frac{0.00s}{29\|28.91\%}$

parsimonious variants ℓPSWC$^{\cup}$ and ℓPSWC$^{\cup}_{\mathsf{N}}$ are up to 6 times faster in the phase transition region than PSWC$^{\cup}$ and PSWC$^{\cup}_{\mathsf{N}}$ respectively, but detect in general slightly fewer unsatisfiable network instances and eliminate slightly fewer unfeasible base relations as well. We should note that for a given QCN $\mathcal{N} = (V, C)$ and a graph $G = (V', E)$, where $V' \subseteq V$, the subset S that was used in Line 3 of the parsimonious variants (see Algorithm 3) corresponds to the set of edges $E(\mathsf{G}(^{\diamond}_{G}(N)))$, i.e., the set of edges of the constraint graph of $^{\diamond}_{G}(N)$.

Synopsis

In conclusion, and with respect to the involved datasets here, we observe that the considered singleton-style consistency algorithms are not good options for just checking the satisfiability of a network instance, as they present an overhead when compared to a state-of-the-art reasoner that is tailored to this specific task. However, we also point out that they are ideal candidates for efficiently approximating and even determining in many cases the minimal labeling of a network instance; this becomes even more prominent if one considers the comparatively bad pruning capability of PWC, and the even worse one of DPWC for that matter. It should be noted that even if the state-of-the-art reasoner Minimizer is provided with a minimal network instance (as it was usually the case in our evaluation due to the preprocessing with $^{\blacklozenge}_{G}{}^{\cup}$-consistency, see again Footnote 4 about this), it is an NP-hard problem to decide the satisfiability of that instance, and an NP-hard problem to verify its minimality as a consequence [30]. We emphasize again the fact that the neighbourhood-focused singleton-style algorithms PSWC$^{\cup}_{\mathsf{N}}$ and PSWC$_{\mathsf{N}}$ were found to be around 30% faster in the phase transition region than the standard algorithms PSWC$^{\cup}$ and PSWC respectively, for both random and structured QCNs, whilst they were able to retain much of the good performance characteristics in terms of unfeasible base relations elimination and satisfiability decision of the latter. Regarding the parsimonious variants in particular, viz., ℓPSWC$^{\cup}$ and ℓPSWC$^{\cup}_{\mathsf{N}}$, even though they exhibited arguably impressive performance characteristics, a major disadvantage is that they do not yield unique closures for a same QCN (see again the discussion in the previous section), which inhibits their theoretical study.

6 Conclusion and Future Work

We proposed singleton-style consistencies for QCNs that are applied just on the neighbourhood of a singleton-checked constraint instead of the whole network, and attained a strength-based hierarchy among all discussed consistencies here. Further, we proposed algorithms to enforce our consistencies, as well as parsimonious variants thereof, that were shown to be much more efficient in practice than the state-of-the-art algorithms for a dataset comprising random and structured QCNs of Interval Algebra. It should be noted that are approach is generic and applies to other calculi as well, such as the spatial calculus RCC8.

Future work consists in obtaining structure-based tractability results focused on the neighbourhood of constraints, developing faster inference mechanisms that will only partially singleton-check a constraint (i.e., only some of the base relations of a constraint will be used for singleton checks), much like *quick shaving* [26], establishing adaptive constraint propagators for QCNs (see [3] for instance in the context of CSPs), and looking into prioritizing or even solely focusing on singleton checks for base relations that play a crucial role in the computational properties of a given qualitative constraint language [24, 8]. Therefore, we argue that our approach can drive both theoretical and practical future research and provide a foundation for further work in the study of QCNs, which are pertinent in Symbolic AI [16].

References

1 James F. Allen. Maintaining Knowledge about Temporal Intervals. *Commun. ACM*, 26:832–843, 1983.
2 Nouhad Amaneddine, Jean-François Condotta, and Michael Sioutis. Efficient Approach to Solve the Minimal Labeling Problem of Temporal and Spatial Qualitative Constraints. In *IJCAI*, 2013.
3 Amine Balafrej, Christian Bessiere, El-Houssine Bouyakhf, and Gilles Trombettoni. Adaptive Singleton-Based Consistencies. In *AAAI*, 2014.
4 Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286:509–512, 1999.
5 Hachemi Bennaceur and Mohamed-Salah Affane. Partition-k-AC: An Efficient Filtering Technique Combining Domain Partition and Arc Consistency. In *CP*, 2001.
6 Mehul Bhatt, Hans Guesgen, Stefan Wölfl, and Shyamanta Hazarika. Qualitative Spatial and Temporal Reasoning: Emerging Applications, Trends, and Directions. *Spatial Cognition & Computation*, 11:1–14, 2011.
7 Mehul Bhatt and Jan Oliver Wallgrün. Geospatial Narratives and Their Spatio-Temporal Dynamics: Commonsense Reasoning for High-Level Analyses in Geographic Information Systems. *ISPRS Int. J. Geo-Information*, 3:166–205, 2014.
8 Manuel Bodirsky, Peter Jonsson, Barnaby Martin, and Antoine Mottet. Classification Transfer for Qualitative Reasoning Problems. In *IJCAI*, 2018.
9 Manuel Bodirsky and Stefan Wölfl. RCC8 is polynomial on networks of bounded treewidth. In *IJCAI*, 2011.
10 Assef Chmeiss and Jean-François Condotta. Consistency of Triangulated Temporal Qualitative Constraint Networks. In *ICTAI*, 2011.
11 Jean-François Condotta and Christophe Lecoutre. A Class of $\overset{\diamond}{f}$-Consistencies for Qualitative Constraint Networks. In *KR*, 2010.
12 Romuald Debruyne and Christian Bessière. Some Practicable Filtering Techniques for the Constraint Satisfaction Problem. In *IJCAI*, 1997.
13 Rina Dechter, Itay Meiri, and Judea Pearl. Temporal Constraint Networks. *Artif. Intell.*, 49:61–95, 1991.
14 Rina Dechter and Judea Pearl. Network-Based Heuristics for Constraint-Satisfaction Problems. *Artif. Intell.*, 34:1–38, 1987.
15 Krishna Sandeep Reddy Dubba, Anthony G. Cohn, David C. Hogg, Mehul Bhatt, and Frank Dylla. Learning Relational Event Models from Video. *J. Artif. Intell. Res.*, 53:41–90, 2015.
16 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.
17 Frank Dylla, Till Mossakowski, Thomas Schneider, and Diedrich Wolter. Algebraic Properties of Qualitative Spatio-Temporal Calculi. In *COSIT*, 2013.
18 Frank Dylla and Jan Oliver Wallgrün. Qualitative Spatial Reasoning with Conceptual Neighborhoods for Agent Control. *J. Intell. Robotic Syst.*, 48:55–78, 2007.
19 Alfonso Gerevini and Alessandro Saetti. Computing the minimal relations in point-based qualitative temporal reasoning through metagraph closure. *Artif. Intell.*, 175:556–585, 2011.
20 Martin Charles Golumbic and Ron Shamir. Complexity and Algorithms for Reasoning about Time: A Graph-Theoretic Approach. *J. ACM*, 40:1108–1133, 1993.
21 Georg Gottlob. On minimal constraint networks. *Artif. Intell.*, 191-192:42–60, 2012.
22 Jinbo Huang. Compactness and Its Implications for Qualitative Spatial and Temporal Reasoning. In *KR*, 2012.
23 Jinbo Huang, Jason Jingshi Li, and Jochen Renz. Decomposition and tractability in qualitative spatial and temporal reasoning. *Artif. Intell.*, 195:140–164, 2013.
24 Peter Jonsson and Victor Lagerkvist. Why are CSPs Based on Partition Schemes Computationally Hard? In *MFCS*, 2018.

25 Nikhil Krishnaswamy, Scott Friedman, and James Pustejovsky. Combining Deep Learning and Qualitative Spatial Reasoning to Learn Complex Structures from Sparse Examples with Noise. In *AAAI*, 2019.

26 Olivier Lhomme. Quick Shaving. In *AAAI*, 2005.

27 Jason Jingshi Li, Jinbo Huang, and Jochen Renz. A divide-and-conquer approach for solving interval algebra networks. In *IJCAI*, 2009.

28 Gérard Ligozat. *Qualitative Spatial and Temporal Reasoning*. Wiley, 2013.

29 Gérard Ligozat and Jochen Renz. What Is a Qualitative Calculus? A General Framework. In *PRICAI*, 2004.

30 Weiming Liu and Sanjiang Li. Solving Minimal Constraint Networks in Qualitative Spatial and Temporal Reasoning. In *CP*, 2012.

31 Ugo Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inf. Sci.*, 7:95–132, 1974.

32 Bernhard Nebel. Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class. *Constraints*, 1:175–190, 1997.

33 Anastasia Paparrizou and Kostas Stergiou. On Neighborhood Singleton Consistencies. In *IJCAI*, 2017.

34 David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions & Connection. In *KR*, 1992.

35 Jochen Renz and Gérard Ligozat. Weak Composition for Qualitative Spatial and Temporal Reasoning. In *CP*, 2005.

36 Jochen Renz and Bernhard Nebel. On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus. *Artif. Intell.*, 108:69–123, 1999.

37 Jochen Renz and Bernhard Nebel. Qualitative Spatial Reasoning Using Constraint Calculi. In *Handbook of Spatial Logics*, pages 161–215. Springer, 2007.

38 Michael Sioutis, Jean-François Condotta, and Manolis Koubarakis. An Efficient Approach for Tackling Large Real World Qualitative Spatial Networks. *Int. J. Artif. Intell. Tools*, 25:1–33, 2016.

39 Michael Sioutis, Zhiguo Long, and Sanjiang Li. Efficiently Reasoning about Qualitative Constraints through Variable Elimination. In *SETN*, 2016.

40 Michael Sioutis, Zhiguo Long, and Sanjiang Li. Leveraging Variable Elimination for Efficiently Reasoning about Qualitative Constraints. *Int. J. Artif. Intell. Tools*, 27:1860001, 2018.

41 Michael Sioutis, Anastasia Paparrizou, and Jean-François Condotta. A Lazy Algorithm to Efficiently Approximate Singleton Path Consistency for Qualitative Constraint Networks. In *ICTAI*, 2017.

42 Michael Sioutis, Anastasia Paparrizou, and Jean-François Condotta. Collective Singleton-Based Consistency for Qualitative Constraint Networks. In *TIME*, 2017.

43 Michael Sioutis, Anastasia Paparrizou, and Jean-François Condotta. Collective Singleton-Based Consistency for Qualitative Constraint Networks: Theory and Practice. *Theor. Comput. Sci*, 2019. In press.

44 Michael Sioutis, Yakoub Salhi, and Jean-François Condotta. Studying the use and effect of graph decomposition in qualitative spatial and temporal reasoning. *Knowl. Eng. Rev.*, 32:e4, 2016.

45 Robert E. Tarjan and Mihalis Yannakakis. Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. *SIAM J. Comput.*, 13:566–579, 1984.

46 Peter van Beek and Rina Dechter. On the Minimality and Decomposability of Row-Convex Constraint Networks. *J. ACM*, 42:543–561, 1995.

47 Richard J. Wallace. Neighbourhood SAC: Extensions and new algorithms. *AI Commun.*, 29:249–268, 2016.

A Bounded Domain Property for an Expressive Fragment of First-Order Linear Temporal Logic

Quentin Peyras
ONERA DTIS, Toulouse, France
Université fédérale de Toulouse, France
quentin.peyras@onera.fr

Julien Brunel
ONERA DTIS, Toulouse, France
Université fédérale de Toulouse, France
https://www.onera.fr/fr/staff/julien-brunel
julien.brunel@onera.fr

David Chemouil
ONERA DTIS, Toulouse, France
Université fédérale de Toulouse, France
https://www.onera.fr/fr/staff/david-chemouil
david.chemouil@onera.fr

Abstract

First-Order Linear Temporal Logic (FOLTL) is well-suited to specify infinite-state systems. However, FOLTL satisfiability is not even semi-decidable, thus preventing automated verification. To address this, a possible track is to constrain specifications to a decidable fragment of FOLTL, but known fragments are too restricted to be usable in practice. In this paper, we exhibit various fragments of increasing scope that provide a pertinent basis for abstract specification of infinite-state systems. We show that these fragments enjoy the Bounded Domain Property (any satisfiable FOLTL formula has a model with a finite, *bounded* FO domain), which provides a basis for complete, automated verification by reduction to LTL satisfiability. Finally, we present a simple case study illustrating the applicability and limitations of our results.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases First-Order Linear Temporal Logic, Bounded Domain Property, Finite Domain Property, Decidability

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.15

Funding Work partly financed by the European Regional Development Fund (ERDF) through the Operational Programme for Competitiveness and Internationalisation (COMPETE2020) and by National Funds through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT) within project POCI-01-0145-FEDER-016826.

1 Introduction

First-Order Logic (FO) has proven to be useful to reason about the structure of a system, *i.e.*, the objects of the domain (which may be infinite), their relations and the properties they satisfy. Temporal logics, on the other hand, provide a natural way to specify the evolution of a system. First-Order Temporal Logics combine both dimensions and offer a flexible way of specifying systems with a rich structure, dynamic aspects and a possibly infinite number of states. First-Order Linear Temporal Logic (FOLTL) [7, 4] is the most studied among those.

However, formally verifying these properties is hard since FOLTL is not even semi-decidable. As we aim at verifying abstract specifications of infinite-state systems, a source of inspiration for the syntactic shape of fragments can be found in formal specification

approaches such as Lamport's TLA$^+$ [9] or the present authors' Electrum [2, 10]. In the latter for instance, a specification typically has the following form (ignoring relational and data structuring features):

$$\mathit{spec} = \mathit{init} \wedge \mathbf{G}\,\mathit{trans} \wedge \mathit{fair} \rightarrow \mathit{prop}$$

where:

- *init* is an FO formula that expresses *initial conditions* of the system;
- *trans* is an FOLTL formula that describes the *system transitions* and that only includes the LTL connective **X** (next) and first-order quantifiers;
- *fair* is an FOLTL formula, which expresses *fairness conditions* and thus includes nested LTL connectives **G** (always) and **F** (eventually);
- *prop* is an FOLTL formula that expresses a *property* expected of the system under specification. It is in principle arbitrarily complex but, in practice, for a large class of systems, it often remains in a relatively simple fragment of FOLTL.

Checking the validity of *spec* ($\models \mathit{spec}$) can be reduced to verifying that $\neg\mathit{spec}$ is *unsatisfiable* ($\mathrm{UNSAT}_{\mathrm{FOLTL}}(\neg\mathit{spec})$), with $\neg\mathit{spec} = \mathit{init} \wedge \mathbf{G}\,\mathit{trans} \wedge \mathit{fair} \wedge \neg\mathit{prop}$. Typically, however, $\neg\mathit{spec}$ does not belong to any formerly known decidable fragment of FOLTL.

Our main contribution is precisely to devise some novel *decidable* fragments of FOLTL encompassing formulas of the shape $\neg\mathit{spec}$.

We introduce in particular the Geneva[1] fragment, which consists of the set of NNF formulas of shape $\psi \wedge \mathbf{G}(\phi)$, where ψ is an FOLTL formula featuring existential quantifiers only at the root and where universal sub-formulas do not contain temporal connectives; and where ϕ is an FOLTL formula featuring only **X** and **F** as temporal connectives and where universal sub-formulas do not contain temporal connectives.

In practice, we prove that this fragment and some variants enjoy the *Bounded Domain Property* (BDP): any satisfiable formula (in any of these fragments) admits a model with finite, *bounded FO domain* (where the bound depends on the shape of the formula)[2]. Remark that the bound does not apply to the temporal dimension of models, only to the FO domain.

Written in a contrapositive way, provided $\neg\mathit{spec}$ belongs to one of these fragments, there is a bound k such that $\mathrm{UNSAT}_{\mathrm{FOLTL}}(\neg\mathit{spec})$ can be reduced to $\mathrm{UNSAT}^{\leqslant k}_{\mathrm{FOLTL}}(\neg\mathit{spec})$, where $\mathrm{UNSAT}^{\leqslant k}_{\mathrm{FOLTL}}$ means unsatisfiability in interpretation structures with FO domain of size $\leqslant k$.

Using this bound k, the FOLTL formula *spec* can be expanded into a plain LTL formula spec' (by unfolding quantifiers over the bounded domain). This way, the $\mathrm{UNSAT}^{\leqslant k}_{\mathrm{FOLTL}}(\neg\mathit{spec})$ problem is itself reduced to an $\mathrm{UNSAT}_{\mathrm{LTL}}(\neg\mathit{spec}')$. As LTL satisfiability is decidable, this ultimately yields a complete, automated decision procedure for the original problem.

Additionally, we make the following two remarks:

- for several of our fragments, the bound is linear in the size of formulas and exponential in certain formula-related criteria that are usually small in practice;
- for several fragments, the characterized bound is effectively reached, in the sense that $\mathrm{UNSAT}_{\mathrm{FOLTL}}(\neg\mathit{spec})$ can even be reduced to $\mathrm{UNSAT}^{=k}_{\mathrm{FOLTL}}(\neg\mathit{spec})$, which can in practice be leveraged to produce a smaller LTL formula to check for unsatisfiability.

The remainder of the article is organized as follows. In Sect. 2, we provide preliminary definitions about FOLTL. In Sect. 3, we exhibit axioms of infinity, i.e. formulas that *do not* enjoy the FDP, in order to guide the search for logical fragments enjoying the BDP. Then

[1] Geneva is a mnemonic for "**G**, Exists, Next/Eventually, (for)All".

[2] This work extends [8] where various simple fragments of FOLTL were studied. In particular, only one fragment including all LTL connectives, an extension of the classic Ramsey FO fragment (cf. Ex. 9), had been shown to enjoy the FDP.

we state some lemmas useful for subsequent proofs. In Sect. 4, we provide a step-by-step definition of a fragment of FOLTL that is relevant in the context of system specification. We establish that it enjoys the FDP and exhibit a bound on the FO domain. Then we illustrate our method on a toy example in Sect 5. Finally, we draw a comparison with other work in Sect. 6.

2 Syntax and Semantics of FOLTL

2.1 FOLTL

The basic vocabulary of FOLTL is defined out of a signature $\Sigma = (\mathcal{F}, \mathcal{R})$ where $\mathcal{F} = (\mathcal{F}_i)_{i\in\mathbb{N}}$ (resp. $\mathcal{R} = (\mathcal{R}_i)_{i\in\mathbb{N}}$) is a family of sets of *function* (resp. *predicate*) *symbols*), with $\mathcal{F}_i$ (resp. $\mathcal{R}_i$) the set of function (resp. predicate) symbols of *arity* i. We write *Const* for the set $\mathcal{F}_0$ of constant symbols. Given a set $\mathcal{V}$ of *variables*, the set $\mathcal{T}_{\Sigma,\mathcal{V}}$ of terms over Σ and $\mathcal{V}$ is defined in the usual way. Terms in $\mathcal{T}_{\Sigma,\varnothing}$ are called *closed terms*.

▶ **Definition 1** (Formulas). *Given a signature $\Sigma = (\mathcal{F}, \mathcal{R})$ and a set of variables $\mathcal{V}$, FOLTL* formulas *over Σ and $\mathcal{V}$ are defined inductively by the following grammar (with $x \in \mathcal{V}$, $r \in \mathcal{R}_n$ and every t_i in $\mathcal{T}_{\Sigma,\mathcal{V}}$):*

$$\psi ::= r(t_1, \ldots, t_n) \mid \neg\psi \mid \psi \vee \psi \mid \mathbf{X}\,\psi \mid \psi\,\mathbf{U}\,\psi \mid \forall x \cdot \psi \mid \exists x \cdot \psi$$

$\mathbf{X}$ and $\mathbf{U}$ stand for the "next" and "until" connectives. We also extend the set of temporal connectives by defining "eventually" as $\mathbf{F}\,\psi = \top\,\mathbf{U}\,\psi$, "always" as $\mathbf{G}\,\psi = \neg\,\mathbf{F}(\neg\psi)$ and "releases" as $\psi_1\,\mathbf{R}\,\psi_2 = \neg((\neg\psi_1)\,\mathbf{U}\,(\neg\psi_2))$. Similarly, classical propositional connectives $\wedge$, $\Rightarrow$ and $\Leftrightarrow$ are defined in the natural way.

Additionally,

- we write $\psi[x]$ for a formula ψ having x as a free variable.
- We write $\mathrm{FV}(\phi)$ for the set of *free variables* of a formula, defined in the obvious way. Also, a formula ϕ is said to be *closed* if $\mathrm{FV}(\phi) = \varnothing$.
- Given a formula ψ, we write $\mathcal{T}_\psi$ for the set of terms, including sub-terms, appearing in ψ.
- Classically, a formula is in *negation normal form* (NNF) if negations only appear in front of predicate symbols.
- We denote by $\mathrm{LTL}_{\Sigma,\mathcal{V}}$ the set of FOLTL formulas, built over Σ with variables in $\mathcal{V}$, that do not contain any first-order quantifier. We write $\mathrm{LTL}_{\Sigma,\mathcal{V}}(\mathbf{X})$ (resp. $\mathrm{LTL}_{\Sigma,\mathcal{V}}(\mathbf{X},\mathbf{F})$) for the set of formulas from $\mathrm{LTL}_{\Sigma,\mathcal{V}}$ that are in NNF and that contain no other temporal connective than $\mathbf{X}$ (resp. $\mathbf{X}$ and $\mathbf{F}$).
- A formula l is called literal if $l = r(t_1, \ldots, t_n)$ or $l = \neg r(t_1, \ldots, t_n)$ where $x \in \mathcal{V}$, $r \in \mathcal{R}_n$ and every t_i in $\mathcal{T}_{\Sigma,\mathcal{V}}$).

We now introduce the semantics of FOLTL. In the interpretation structures defined below, the interpretation of predicates *varies* over time while that of function symbols *does not*. Notice we rely on the Kleene star in the definition.

▶ **Definition 2** ((Interpretation) Structure). *Given a signature $\Sigma = (\mathcal{F}, \mathcal{R})$, an* (interpretation) structure $\mathcal{M}$ (over Σ) *is a triple (D, σ, ρ) where:*

- *D, called the domain, is a non-empty set.*
- *σ is a map s.t. for any $c \in \mathcal{F}_0$, $\sigma(c) \in D$, and for any $f \in \mathcal{F}_n$, $\sigma(f) : D^n \to D$.*
- *$\rho : \mathbb{N} \times D^\star \to \mathcal{P}(\mathcal{R})$ is a map s.t. for any instant $i \in \mathbb{N}$ and any $\vec{a} = (a_1 \ldots, a_n) \in D^\star$, $\rho_i(\vec{a}) \subseteq \mathcal{R}_n$.*

$\mathcal{M}$ is said to be domain-finite *if D is finite. We also define the* domain size *(simply called* size *in the remainder of this paper) of $\mathcal{M}$ as $|D|$.*

▶ Remark 3 (Type of ρ). Usually, FOLTL structures would be defined with ρ a function $\mathbb{N} \times \mathcal{R} \to \mathcal{P}(D^\star)$ mapping at any instant a predicate to the set of tuples (of the domain) satisfying it. We turn this definition upside down, which is trivially equivalent to the classical one, to simplify the presentation of forthcoming definitions (in particular, partial structures introduced in Def. 10) and proofs.

▶ **Definition 4** (Assignment). *An* assignment $\mathcal{C}$ *in a domain* D *for variables in* $\mathcal{V}$ *is a map* $\mathcal{V} \to D$. *We write* $\mathcal{C}[x \mapsto d]$ *the assignment defined as* $\mathcal{C}[x \mapsto d](x) = d$ *and* $\mathcal{C}[x \mapsto d](y) = \mathcal{C}(y)$ *if* $y \neq x$. *The* extension *of* $\mathcal{C}$ *to* terms*, also written* $\mathcal{C}$*, is defined in the obvious way.*

▶ **Definition 5** (Satisfaction). *Given a structure* $\mathcal{M} = (D, \sigma, \rho)$ *and an assignment* $\mathcal{C}$*, the* satisfaction *relation* $\vDash$ *is defined by induction on formulas, for any* $i \in \mathbb{N}$*, as follows:*

- $\mathcal{M}, i, \mathcal{C} \vDash r(t_1, \ldots, t_n)$ *iff* $r \in \rho_i(\mathcal{C}(t_1), \ldots, \mathcal{C}(t_n))$;
- $\mathcal{M}, i, \mathcal{C} \vDash \neg\phi$ *iff* $\mathcal{M}, i, \mathcal{C} \nvDash \phi$;
- $\mathcal{M}, i, \mathcal{C} \vDash \phi_1 \vee \phi_2$ *iff* $\mathcal{M}, i, \mathcal{C} \vDash \phi_1$ *or* $\mathcal{M}, i, \mathcal{C} \vDash \phi_2$;
- $\mathcal{M}, i, \mathcal{C} \vDash \mathbf{X}\,\phi$ *iff* $\mathcal{M}, i+1, \mathcal{C} \vDash \phi$;
- $\mathcal{M}, i, \mathcal{C} \vDash \phi_1\,\mathbf{U}\,\phi_2$ *iff there exists* $k \in \mathbb{N}$ *s.t.* $\mathcal{M}, i+k, \mathcal{C} \vDash \phi_2$ *and for every* $0 \leq j < k$*, we have* $\mathcal{M}, i+j, \mathcal{C} \vDash \phi_1$;
- $\mathcal{M}, i, \mathcal{C} \vDash \exists y \cdot \phi$ *iff there exists* $d \in D$ *s.t.* $\mathcal{M}, i, \mathcal{C}[y \mapsto d] \vDash \phi$;
- $\mathcal{M}, i, \mathcal{C} \vDash \forall x \cdot \phi$ *iff for every* $d \in D$*, we have* $\mathcal{M}, i, \mathcal{C}[x \mapsto d] \vDash \phi$.

Given a closed formula ϕ*, we write* $\mathcal{M}, k \vDash \phi$ *if* $\mathcal{M}, k, [] \vDash \phi$*, where* $[]$ *is the empty assignment.*

Let ϕ, ϕ' be two FOLTL formulas. If for any structure $\mathcal{M}$ and an assignment $\mathcal{C}$, we have $\mathcal{M}, 0, \mathcal{C} \vDash \phi$ iff $\mathcal{M}, 0, \mathcal{C} \vDash \phi'$ then we say that ϕ and ϕ' are logically equivalent, written $\phi \equiv \phi'$.

▶ **Definition 6** (Finite Domain Property, Bounded Domain Property). *A closed formula* ϕ *of FOLTL enjoys the* finite domain property *(FDP) if* ϕ *is not satisfiable, or there is a domain-finite structure* $\mathcal{M}$ *s.t.* $\mathcal{M}, 0 \vDash \phi$. *Additionally, if the bound is computable, the formula is said to enjoy the Bounded Domain Property (BDP). A fragment of a logic enjoys the FDP (resp. BDP) if every formula in this fragment does.*

▶ Remark 7 (BDP and decidability). For pure FO, if a fragment enjoys the FDP (usually called the *Finite Model Property*), then it is decidable. As FOLTL is not recursively enumerable (contrary to FO), the FDP does not suffice to show decidability of a given fragment, while the BDP does.

▶ Remark 8 (BDP and complexity). If a fragment enjoys the BDP, then from the expression of the bound on the domain, we can easily deduce an upper bound of the complexity of satisfiability for this fragment, using the results from [8]. Indeed, in [8], the complexity of the satisfiability problem on bounded models is studied for full FOLTL.

▶ **Example 9.** The following fragments of FO enjoy the FDP (following the book and notations of Börger et al. [3]):

- $[\exists^\star\forall^\star, all]_=$(Ramsey 1930) the class of formulas with quantifier prefix $\exists^\star\forall^\star$, without function symbols, with arbitrary predicate symbols, with equality.
- $[\exists^\star, all, all]_=$(Gurevich 1976) the class of formulas with quantifier prefix $\exists^\star$, with arbitrary predicate and function symbols, with equality.

2.2 Partial Structures

We defined the notion of structures for FOLTL, however proving the BDP requires to define a model in several steps. Indeed, we will need to define predicate interpretations for a finite numbers of instants, and to define the truth values of predicates for the remaining time later

on. This is clumsy to do with structures since we will need to redefine the entire structure at each step. For this reason, we introduce a notion of partial structures, which is easier to handle.

▶ **Definition 10** (Partial (interpretation) structures). *A* partial (interpretation) structure $\mathcal{M}$ *(over* Σ*) is a triple* (D, σ, ρ) *satisfying the same conditions as in Def.* 2 *except that* ρ *is a partial function. We denote by* $\rho_i(\vec{x}) = \bot$ *the fact that* ρ *is not defined on the pair* $(i, \vec{x})$*.*

Structures are then the maximal elements of the set of partial structures for the following partial order.

▶ **Definition 11** (Extension ordering of partial structures). *Given two partial structures* $\mathcal{M} = (D, \sigma, \rho)$ *and* $\mathcal{M}' = (D', \sigma', \rho')$*, we define the partial order* $\preccurlyeq$ *over partial structures as follows:* $\mathcal{M}'$ extends $\mathcal{M}$*, written* $\mathcal{M} \preccurlyeq \mathcal{M}'$*, iff* $D = D'$*,* $\sigma = \sigma'$*, and* $\rho_i(\vec{a}) \neq \bot$ *implies* $\rho'_i(\vec{a}) = \rho_i(\vec{a})$*.*

This allows a natural generalization of satisfaction to partial structures by saying that a partial structure satisfies a formula if *all* its extensions that are structures satisfy it.

▶ **Definition 12** (Semantics over partial structures I). *Given a partial structure* $\mathcal{M}$*, we say that* $\mathcal{M}, i, \mathcal{C} \vDash \phi$ *iff for all structure* $\mathcal{M}'$ *s.t.* $\mathcal{M} \preccurlyeq \mathcal{M}'$*, we have* $\mathcal{M}', i, \mathcal{C} \vDash \phi$*.*

There is another, natural way to define the semantics over partial structures. We introduce it as it will be required in forthcoming proofs. This semantics can be defined by induction on formulas in NNF. Such a restriction is necessary because we cannot evaluate the truth value of $\neg\phi$ out of that of ϕ, therefore we cannot define a general semantics for the "not" connective. This is because if a partial structure can be extended to either satisfy ϕ or satisfy $\neg\phi$, then this partial structure satisfies neither of these formulas.

▶ **Definition 13** (Semantics over partial structures II). *Given a partial structure* $\mathcal{M} = (D, \sigma, \rho)$ *and an assignment* $\mathcal{C}$*, the* satisfaction *relation* $\Vdash$ *is defined by induction on formulas in negation normal form (NNF), for all non-negative integers* i *as follows:*

- $\mathcal{M}, i, \mathcal{C} \Vdash r(t_1, \ldots, t_n)$ *iff* $r \in \rho_i(\mathcal{C}(t_1), \ldots, \mathcal{C}(t_n))$*.*
- $\mathcal{M}, i, \mathcal{C} \Vdash \neg r(t_1, \ldots, t_n)$ *iff* $\rho_i(\mathcal{C}(t_1), \ldots, \mathcal{C}(t_n)) \neq \bot$ *and* $r \notin \rho_i(\mathcal{C}(t_1), \ldots, \mathcal{C}(t_n))$*.*
- $\mathcal{M}, i, \mathcal{C} \Vdash \phi_1 \wedge \phi_2$ *if and only if* $\mathcal{M}, i, \mathcal{C} \Vdash \phi_1$ *and* $\mathcal{M}, i, \mathcal{C} \Vdash \phi_2$*.*
- $\mathcal{M}, i, \mathcal{C} \Vdash \phi_1 \vee \phi_2$ *if and only if* $\mathcal{M}, i, \mathcal{C} \Vdash \phi_1$ *or* $\mathcal{M}, i, \mathcal{C} \Vdash \phi_2$*.*
- $\mathcal{M}, i, \mathcal{C} \Vdash \mathbf{X}\,\phi$ *iff* $\mathcal{M}, i+1, \mathcal{C} \Vdash \phi$*.*
- $\mathcal{M}, i, \mathcal{C} \Vdash \phi_1\,\mathbf{U}\,\phi_2$ *iff there exists* $k \in \mathbb{N}$ *s.t.* $\mathcal{M}, i+k, \mathcal{C} \vDash \phi_2$ *and for every integer* $0 \leq j < k$*, we have* $\mathcal{M}, i+j, \mathcal{C} \Vdash \phi_1$*.*
- $\mathcal{M}, i, \mathcal{C} \Vdash \phi_1\,\mathbf{R}\,\phi_2$ *iff for each* $k \in \mathbb{N}$ $\mathcal{M}, i+k, \mathcal{C} \Vdash \phi_2$ *or there exists an integer* j *s.t.* $\mathcal{M}, i+j, \mathcal{C} \Vdash \phi_1$ *and for every integer* $0 \leq k \leq j$*,* $\mathcal{M}, i+k, \mathcal{C} \Vdash \phi_2$*.*
- $\mathcal{M}, i, \mathcal{C} \Vdash \exists y \cdot \phi(y)$ *if and only if there exists* $d \in D$ *s.t.* $\mathcal{M}, i, \mathcal{C}[y \mapsto d] \Vdash \phi(y)$*.*
- $\mathcal{M}, i, \mathcal{C} \Vdash \forall x \cdot \phi(x)$ *if and only if for every* $d \in D$*, we have* $\mathcal{M}, i, \mathcal{C}[x \mapsto d] \Vdash \phi(x)$*.*

▶ **Lemma 14** (Equivalence of semantics). *Given a partial structure* $\mathcal{M}$*, a formula* ϕ *in NNF,* $k \in \mathbb{N}$ *and an assignment* $\mathcal{C}$*, we have* $\mathcal{M}, k, \mathcal{C} \vDash \phi$ *iff* $\mathcal{M}, k, \mathcal{C} \Vdash \phi$*.*

▶ **Definition 15** (Enrichment of a structure). *Given a partial structure* $\mathcal{M} = (D, \sigma, \rho)$ *s.t.* $\rho_i(\vec{d}) = \bot$*, we define the* enrichment *of* $\mathcal{M}$ *at instant* i *on tuple* $\vec{d}$ *for* $A \in \mathcal{P}(\mathcal{R})$*, written* $\mathcal{M}[(i, \vec{d}) \mapsto A]$*, as the triple* (D, σ, ρ') *where:* $\rho'_i(\vec{d}) = A$ *and for any* $j \in \mathbb{N}$ *and any tuple* $\vec{d'}$*,* $\rho'_j(\vec{d'}) = \rho_j(\vec{d'})$ *if* $(j, \vec{d'}) \neq (i, \vec{d})$*. Notice that* $\mathcal{M}[(i, \vec{d}) \mapsto A]$ *is an extension of* $\mathcal{M}$*.*

Some sort of induction is required to extend a partial structure for tuples over all instants of time. It is possible to proceed by extending a partial structure step-by-step using the previous definition. The result is then an increasing sequence of partial structures. Intuitively, it can be seen that such a sequence somehow converges to a partial structure where all steps of extension have been performed on it. The following definition formalizes this notion.

▶ **Definition 16** (Limit structure). *Let $(\mathcal{M}^k)_{k\in\mathbb{N}}$ be a $\preccurlyeq$-increasing sequence of partial structures, with $\mathcal{M}^k = (D, \sigma, \rho^k)$. Then we define the (partial)* limit structure *$\mathcal{M}^\infty = (D, \sigma, \rho^\infty)$ s.t., for any $i \in \mathbb{N}$ and vector $\vec{d} \in D^\star$: (1) if there exists k s.t. $\rho_i^k(\vec{d}) \neq \bot$, then $\rho_i^\infty(\vec{d}) = \rho_i^k(\vec{d})$; (2) if for every $k \in \mathbb{N}$ we have $\rho_i^k(\vec{d}) = \bot$, then $\rho_i^\infty(\vec{d}) = \bot$.*

As we focus on the BDP, we aim at building a domain-finite model of a formula out of any structure satisfying it. However, to ensure that we have a general method working for a fragment as expressive as possible, we need to make this domain-finite model as similar as possible to the original one. For this reason, we define the following notion of partial embedding. Informally, this embedding between two partial structures expresses that any element of the domain of the former has, for each instant, an equivalent element in the domain of the latter, meaning they satisfy the same predicates. In the case of n-ary predicates, two tuples with one-to-one equivalent elements are considered equivalent, so they satisfy the same predicates. A partial embedding is used to deduce information about the satisfaction of non-temporal universally quantified properties at some particular instant.

▶ **Definition 17** (Partial embedding). *Let $\mathcal{M}_0 = (D_0, \sigma_0, \rho^0)$, $\mathcal{M}_1 = (D_1, \sigma_1, \rho^1)$ be two partial structures and $f : \mathbb{N} \times D_0 \nrightarrow D_1$ be a partial function. We say that f is a* (partial) embedding *from $\mathcal{M}_0$ to $\mathcal{M}_1$, denoted $\mathcal{M}_0 \overset{f}{\hookrightarrow} \mathcal{M}_1$, if there exists $m \in \mathbb{N}$ s.t.:*

- *for each $c \in Const$, each $i \in \mathbb{N}$, we have $f_i(\sigma_0(c)) = \sigma_1(c)$;*
- *for each $g \in \mathcal{F}_n$ $(n > 0)$, $\vec{d} \in D_0^n$, and each $i \in \mathbb{N}$ s.t. $f_i(\vec{d}) \neq \bot$[3], $f_i(\sigma_0(g)(\vec{d})) = \sigma_1(g)(f_i(\vec{d}))$; and*
- *for each $\vec{d} \in D_0^\star$ and each $i \in \mathbb{N}$, if $f_{i+m}(\vec{d}) \neq \bot$ then $\rho_i^0(\vec{d}) = \rho_i^1(f_{i+m}(\vec{d}))$, otherwise $\rho_i^0(\vec{d}) = \bot$.*

3 Preliminary Results

Here we exhibit various formulas of FOLTL that *do not* enjoy the FDP, which hints on possible directions to find a fragment that *does* enjoy it. Then, we introduce some technical lemmas about elementary fragments of FOLTL, which will be useful to establish the BDP of the fragments studied in Sect. 4.

3.1 Axioms of Infinity

In this section, we report on various ways not to enjoy the FDP. Our study of FOLTL fragments was partly guided by the need to avoid syntactic fragments. We call *axiom of infinity* an FOLTL formula that does not satisfy the FDP. Finding such axioms is easy, even with strong constraints on first-order quantifiers.

Due to some results from [8], we start our study with formulas featuring existential quantifiers. For instance, the following axiom of infinity involves only one existential quantifier: $\mathbf{G}(\exists y \cdot P(y) \wedge \mathbf{X}\,\mathbf{G}\,\neg P(y))$. Indeed, to satisfy this formula, we need to find some element in the domain satisfying P at each instant of time; however this element will never satisfy P again so an infinite domain is needed to pick a different element at every instant.

[3] $f_i(\vec{d}) \neq \bot$ denotes the fact that $f_i(\vec{d})$ is defined

It then appears that existential quantification under a $\mathbf{G}$ connective can be problematic. However, this problem occurs only when several *nested* $\mathbf{G}$ connectives appear in the formula. Notice that nesting $\mathbf{G}$ connectives is often unnecessary in practical system specifications.

Therefore let us now focus on cases where we have a formula of the form $\mathbf{G}(\exists y \cdot \psi[y])$ where ψ does not contain any $\mathbf{G}$ or first-order quantifier.

Now, what about universal quantification? Unfortunately, even with a prefix within the Ramsey fragment, axioms of infinity can be found, such as the following: $\mathbf{G}(\exists y \forall x \cdot \neg P(y) \wedge \mathbf{X}\, P(y) \wedge (P(x) \Rightarrow \mathbf{X}\, P(x)))$. Here, the universal quantifier allows us to specify by induction that any element in the domain used for the existential quantifier satisfies $\mathbf{G}\, P(y)$. This is actually similar to the first axiom. In order to avoid this behaviour, a new restriction is needed. A possibility is to forbid the use of temporal connectives under the scope of a universal quantifier.

Another issue lies in the use of constant predicates (predicates whose value does not change along time). Assume we are given a constant order $<$ (axiomatized by universally quantified formula without temporal connectives). Then the following formula defines an axiom of infinity: $\mathbf{G}(\exists y \cdot P(y) \wedge \mathbf{X}(\forall x \cdot P(x) \Rightarrow y < x))$. Indeed it forces at each instant the existence of an element in the domain which is greater than all elements that have already been used. Satisfying the formula then requires to have an infinite domain.

Thus, to obtain a fragment enjoying the BDP, one should at least:

- forbid nested $\mathbf{G}$ connectives;
- forbid temporal connectives under the scope of a universal quantifier;
- and forbid constant predicates if universal quantifiers are allowed.

3.2 Preliminary Lemmas

We now introduce lemmas that are basic elements of the BDP proof for the FOLTL fragments studied in this article.

The following lemma allows us to consider a formula with a well-suited syntactic form for the upcoming proofs (*without* impact on computed bounds). Indeed, we will need to define interpretations of predicates such that a formula is true at every instant. However, in case of a disjunction, there may be various ways to satisfy a formula. For example, consider $\phi = (a \Rightarrow \mathbf{X}\, b) \wedge (a \Rightarrow \mathbf{F}\, c)$; in this case, at every instant, ϕ may be satisfied by having $\neg a$ or $\mathbf{X}\, b \wedge \mathbf{F}\, c$. So we transform ϕ into a disjunctive normal form allowing us to differentiate and pick in which way it can be satisfied. In the case of ϕ, we obtain $(\neg a) \vee (\mathbf{X}\, b \wedge \mathbf{F}\, c)$. Within each disjunct, we distinguish between the sub-formulas under an $\mathbf{F}$ connective (which need to be satisfied at an unspecified instant) and the other ones (which need to be satisfied at a specified number of instants, depending on the number of nested $\mathbf{X}$ connectives).

▶ **Lemma 18** (Disjunctive normal form (DNF)). *If ϕ is a formula in $LTL_{\Sigma,\mathcal{V}}(\mathbf{X},\mathbf{F})$ then there exists $\psi \equiv \phi$ s.t.: (1) ψ is a disjunction of the form $\psi_1 \vee \ldots \vee \psi_n$ (notice that each ψ_i is in NNF); (2) Each ψ_i is a conjunction of the form $\alpha_i \wedge \mathbf{F}\, \beta_{i,1} \wedge \ldots \wedge \mathbf{F}\, \beta_{i,j}$, with $\alpha_i = \mathbf{X}^{n_{i,1}}\, \ell_{i,1} \wedge \ldots \wedge \mathbf{X}^{n_{i,k_i}}\, \ell_{i,k_i}$ (writing $\mathbf{X}^n$ for a sequence of n $\mathbf{X}$ connectives) and where each $\ell_{i,k}$ is a literal and each $\beta_{i,k}$ is in $LTL_{\Sigma,\mathcal{V}}(\mathbf{X},\mathbf{F})$.*

▶ Remark 19 (Inocuity of the DNF transformation). The transformation of a formula into DNF induces an exponential blow-up. In this paper, it is only used to prove the BDP of the considered fragments: since the considered bounds are not affected by the transformation in DNF, the blow-up does not influence the complexity of the decision procedure.

The size of the finite model resulting from the construction presented in this paper depends on the depth of nested $\mathbf{X}$ connectives. For example, there is a structure of size 1 satisfying $\mathbf{G}(\exists y \cdot P(y))$. However any structure satisfying $\mathbf{G}(\exists y \cdot P(y) \wedge \mathbf{X}(\neg P(y)) \wedge \mathbf{X}\,\mathbf{X}(\neg P(y))$ is at least of size 3. This depends on the number of instants it refers to using $\mathbf{X}$ connectives:

▶ **Definition 20** (Stride of a formula). *Given a formula ϕ in DNF, we define its* stride K_ϕ *as the maximal depth of nested $\mathbf{X}$ connectives not under an $\mathbf{F}$, that is $K_\phi = \max_{i=1..n} \max_{j=1..k_i} n_{i,j}$ (with $n_{i,j}$ following the notations of Lemma 18).*

The following lemma applies to formulas containing only $\mathbf{X}$ and $\mathbf{F}$ connectives, as well as featuring only existential quantification over a single variable. Given such a formula, a model of this formula and a partial structure where constant symbols are interpreted as in the model of the formula, the lemma states that we can extend this partial structure into a partial model of the formula by providing an interpretation for the predicates (1) for a finite set of instants only and (2) over a single element in the domain.

▶ **Lemma 21.** *Consider a formula ψ in $LTL_{\Sigma,\{y\}}(\mathbf{X},\mathbf{F})$ and a structure $\mathcal{M} = (\mathcal{D}, \sigma, \rho)$ s.t. $\mathcal{M}, k \vDash \exists y \cdot \psi[y]$ for some $k \in \mathbb{N}$. Consider also a partial structure $\mathcal{M}_0 = (\mathcal{D}, \sigma_0, \rho^0)$ s.t. $\mathcal{M}_0 \overset{f^0}{\hookrightarrow} \mathcal{M}$ and s.t. there exists some a in $\mathcal{D}$ s.t. for each integer $j \geq k$ we have $f^0_j(a) = \bot$. Then, there exists an integer $k' > k$ (where $k' = k + K_\psi + 1$ if $\psi \in LTL_{\Sigma,\{y\}}(\mathbf{X})$) and a structure $\mathcal{M}_1 = (\mathcal{D}, \sigma_1, \rho^1)$ satisfying:*

- *$\mathcal{M}_1 \overset{f^1}{\hookrightarrow} \mathcal{M}$ for some f^1,*
- *for any $x \in \mathcal{D}$ and any $i \geq k'$, $f^1_i(x) \neq \bot$ iff $x \in \sigma_1(\mathcal{T}_{\Sigma,\varnothing})$,*
- *for any $i \in \mathbb{N}$ s.t. $k \leq i < k'$, and any $x \in D$, $x \neq a$ implies $f^0_i(x) = f^1_i(x)$,*
- *$\mathcal{M}_0 \preccurlyeq \mathcal{M}_1$,*
- *$\mathcal{M}_1, k, [y \mapsto a] \vDash \psi[y]$.*

Proof. First notice that the truth value of a formula in $\mathrm{LTL}_{\Sigma,\{y\}}(\mathbf{X},\mathbf{F})$ can be determined by only "looking at" a finite set of instants I, in the sense that changing the interpretation of predicates outside I does not change the truth value of the formula. This can be shown for instance by induction on the number of nested $\mathbf{F}$.

Let ψ be a formula in $\mathrm{LTL}_{\Sigma,\{y\}}(\mathbf{X},\mathbf{F})$ s.t. $\mathcal{M}, k \models \exists y \cdot \psi[y]$. Let k' be the greatest instant in the set I as introduced above. Let d be an element in the domain such that $\mathcal{M}, k, [y \mapsto d] \models \psi[y]$. Then, we can extend $\mathcal{M}_0$ into $\mathcal{M}_1$ in a way s.t. $f^1_i(a) = d$ and $\rho^1_i(a) = \rho_i(d)$ for $i \in [k, k']$. ◀

The next lemma focuses on formulas containing $\mathbf{X}$ connectives only. It establishes that formulas of the form $\mathbf{G}(\exists y \cdot \psi)$, where the only temporal connective in ψ is $\mathbf{X}$, enjoy the BDP. However, this lemma is formulated in a more suitable way for the proof of Theorem 26. In particular, we limit the result to a finite temporal window $[k_1, k_2]$.

▶ **Lemma 22.** *Assume that there exists $k_1, k_2 \in \mathbb{N}$ s.t. for any integer $i \in [k_1, k_2]$ we have $\mathcal{M}, i \vDash \exists y \cdot \alpha[y]$, where $\alpha \in LTL_{\Sigma,\{y\}}(\mathbf{X})$. Let $\mathcal{M}^0$ be a partial structure s.t. $\mathcal{M}^0 \overset{f^0}{\hookrightarrow} \mathcal{M}$ for some f^0, and there exists $A = \{a_0, \ldots, a_{K_\alpha}\}$ s.t. for each integer $j \in [k_1, k_2 + K_\alpha]$ and all $a \in A$, we have $f^0_j(a) = \bot$. Then there exists $\mathcal{M}^1$ s.t.:*

- *$\mathcal{M}^1 \overset{f_1}{\hookrightarrow} \mathcal{M}$ for some f^1;*
- *$\mathcal{M}^0 \preccurlyeq \mathcal{M}^1$;*
- *$f^1_j(x) \neq f^0_j(x)$ implies that $j \in [k_1, k_2 + K_\alpha]$ and $x \in A$;*
- *For any $i \in [k_1, k_2]$, there exists $m \leq K_\alpha$ s.t. $\mathcal{M}^1, i, [y \mapsto a_m] \vDash \alpha[y]$.*

Figure 1 First step of the partial structure construction.

	0	1	2	3	...
a_0	P	$\neg P$	?	?	...
a_1	?	P	$\neg P$	?	...
a_2	?	?	P	$\neg P$	...
...	...	...	...	...	

$\longrightarrow$

	0	1	2	3	...
$a_0 = a_2$	P	$\neg P$	P	$\neg P$	...
a_1	?	P	$\neg P$	?	...
...	...	...	...	...	

Figure 2 Trace of $\mathcal{M}^\infty$.

	0	1	2	3	...
d_0	P	$\neg P$	P	$\neg P$	...
d_1	?	P	$\neg P$	P	...
	$P(d_0) \wedge \mathbf{X}(\neg P(d_0))$	$P(d_1) \wedge \mathbf{X}(\neg P(d_1))$	$P(d_0) \wedge \mathbf{X}(\neg P(d_0))$	...	

Proof. Let α be a formula in $\mathrm{LTL}_{\Sigma,\{y\}}(\mathbf{X})$. We prove the theorem by induction over k_2.

If $k_1 = k_2$ then the result is reduced to lemma 21.

Induction step: we assume that the statement of the lemma holds for $[k_1, k_2]$. Now suppose that the premises of the lemma hold for $[k_1, k_2 + 1]$. From induction hypothesis, we know that there exists $\mathcal{M}_1$ satisfying the conclusion of the lemma for $i \in [k_1, k_2]$. We can then extend $\mathcal{M}^1$ by applying lemma 21 using instant $k_2 + 1$ and one element in A. Then the resulting structure satisfies the conclusion of the lemma for the set $[k_1, k_2 + 1]$. ◀

▶ **Example 23.** The main ideas of the proof of Lemma 22 are illustrated through an example. Let us consider the formula $\psi[y] = P(y) \wedge \mathbf{X}\neg P(y)$. For the sake of simplicity, instead of considering a finite temporal window $[k_1, k_2]$ among which $\exists y \cdot \psi[y]$ is satisfied, we consider a structure $\mathcal{M}$ s.t. for any $k \in \mathbb{N}$, $\mathcal{M}, k \models \exists y \cdot \psi[y]$, which is equivalent to $\mathcal{M}, 0 \models \mathbf{G}(\exists y \cdot \psi[y])$.

Let us build a finite partial model of this formula. Following the semantics of FOLTL, for any $k \in \mathbb{N}$, there is some a_k in the domain of $\mathcal{M}$ s.t. $\mathcal{M}, k, [y \mapsto a_k] \models P(y) \wedge \mathbf{X}\neg P(y)$. So, for any $k \in \mathbb{N}$, $P \in \rho_k(a_k)$ and $P \notin \rho_{k+1}(a_k)$. Consider the constraints a_0, a_1 and a_2 must satisfy: a_0 has constraints only at instants 0 (to satisfy P) and 1 (not to satisfy P); a_1 only has some constraints at instants 1 and 2; and a_2 only has some constraints at instants 2 and 3. Thus, we can reuse a_0 to play the role of a_2 at instants 2 and 3, as shown in Fig. 1.

Then ψ can be satisfied for the first three instants with only two elements in the domain. By the same argument, we can reuse a_1 instead of using a_3. This can be generalized to reuse a_0 (resp. a_1) instead of every a_k, where k is an even (resp. odd) number. We then see that we can satisfy our formula with a structure of size 2. Let us call d_0 and d_1 the corresponding elements of the domain. Let us define a first structure $\mathcal{M}^0 = (D, \sigma, \rho^0)$, where $D = \{d_0, d_1\}$, σ is an empty map (since there is no function symbols in ψ) and ρ^0 is defined as the partial function that is undefined over all entries. Now let us define $\mathcal{M}^{i+1}$ from $\mathcal{M}^i$. If i is even then $m = 0$, else $m = 1$ then Lemma 21 gives us $\mathcal{M}^{k+1} = \mathcal{M}^k[(k, d_m) \mapsto \{P\}][(k + 1, d_m) \mapsto \varnothing]$. Then we have $\mathcal{M}^{k+1}, k, [y \mapsto d_m] \models \psi[y]$.

We get a $\preccurlyeq$-increasing sequence $(\mathcal{M}^i)_{i\in\mathbb{N}}$, the limit structure of which $(\mathcal{M}^\infty)$ is illustrated in Fig. 2. Since for any integer k, $\mathcal{M}^{k+1}, k, [y \mapsto d_0] \models \psi[y]$ or $\mathcal{M}^{k+1}, k, [y \mapsto d_1] \models \psi[y]$, we have that $\mathcal{M}^\infty, k \models \mathbf{G}(\exists y \cdot \psi[y])$. ⌟

The reasoning that we had for this particular example can be easily generalized for any formula of the form $\mathbf{G}(\exists y \cdot \psi[y])$ where $\psi \in LTL_{\Sigma,\{y\}}(\mathbf{X})$. We then get a partial model of the formula with a domain of size $K_\psi + 1$.

Now, we want to extend the fragment to allow for the temporal connective $\mathbf{F}$ in ψ. Suppose that there is a model $\mathcal{M}$ of $\phi = \mathbf{G}(\exists y \cdot \psi[y])$ and that $\psi = \psi_1 \vee \ldots \vee \psi_n$ is in DNF, as in Lemma 18. Also suppose that several ψ_i have the form $\mathbf{F}\,\psi_i'$. Then, some of these $\mathbf{F}\,\psi_i'$ can be true at a finite number of instants in $\mathcal{M}$, which makes it complicated to build a finite partial model of ϕ. The following lemma states that we can get rid of such $\mathbf{F}\,\psi_i'$.

▶ **Lemma 24.** *Let $\mathcal{M}$ be a partial structure satisfying $\mathcal{M}, 0 \vDash \mathbf{G}(\psi_1 \vee \psi_2) \wedge \neg\,\mathbf{G}\,\mathbf{F}(\psi_2)$. Then there exists $\mathcal{M}'$ s.t. $\mathcal{M}', 0 \vDash \mathbf{G}(\psi_1)$ and $\mathcal{M}' \overset{Id}{\hookrightarrow} \mathcal{M}$, with Id defined as $Id(i,d) = d$ for all instant i and domain element d.*

Sketch. To get $\mathcal{M}'$ from $\mathcal{M}$, we simply make a translation in time, starting from the first instant k s.t. for any $k' \geqslant k$, $\mathcal{M}, k' \vDash \neg\psi_2$. ◀

4 Finite Domain Property

We now present our main results. We start in Sect. 4.1 with the BDP of our core fragment, limited to a single existential quantifier and without functions. In Sect. 4.2, we establish the BDP for extended fragments including functions and first-order quantifiers used in a restricted way. In Sect. 4.3, we study how these fragments can be extended with equality.

4.1 Core Theorem

Theorem 26 says that given a formula ϕ (1) in NNF, (2) with only one existential quantifier, (3) containing no other temporal connectives than $\mathbf{X}$ and $\mathbf{F}$, (4) without function symbols other than constants, (5) with only unary predicates, then $\mathbf{G}\,\phi$ enjoys the BDP. Most of these restrictions are unnecessary for the BDP but, while keeping the main ideas of the proof, they make it simpler to understand. Releasing them will lead to Theorem 28.

▶ **Definition 25.** *We say that $\phi \in Gur^{-}(\mathbf{X}, \mathbf{F})$ (for "Gurevich") if there exists a signature $\Sigma = (\mathcal{F}, \mathcal{R})$ s.t. (1) for any $n > 0$, $\mathcal{F}_n = \varnothing$, (2) for any $n > 1$, $\mathcal{R}_n = \varnothing$ and (3) there exists $\psi \in LTL_{\Sigma,\{y\}}(\mathbf{X}, \mathbf{F})$ s.t. $\phi = \exists y \cdot \psi$.*

▶ **Theorem 26.** *If ϕ is a formula in $Gur^{-}(\mathbf{X}, \mathbf{F})$, then $\mathbf{G}\,\phi$ enjoys the FDP. Moreover, if $\mathbf{G}\,\phi$ is satisfiable, it has a model of size $|Const| + 2 \times (K_\psi + 1)$.*

Proof. Let us consider that $\psi' \in \mathrm{LTL}_{\Sigma,\{y\}}(\mathbf{X}, \mathbf{F})$. By using Lemma 18, w.l.o.g., we consider ψ' in DNF: $\psi' = \psi_1 \vee \ldots \vee \psi_m$. Considering a model $\mathcal{M}$ of $\mathbf{G}(\exists y \cdot \psi'[y])$, some ψ_i are satisfied at an infinite number of instants. Let us consider that $\psi_1, \ldots, \psi_n$ are satisfied at an infinite number of instants and $\psi_{n+1}, \ldots, \psi_m$ are satisfied at a finite number of instants. Then by application of Lemma 24, there is a structure $\mathcal{M}$ satisfying: $\mathcal{M}, 0 \vDash \mathbf{G}(\exists y \cdot \psi_1[y] \vee \ldots \vee \psi_n[y])$.

Let us write $\psi = \psi_1 \vee \ldots \vee \psi_n$ and remind that each ψ_i is in NNF and each ψ_i is of the form $\alpha_i \wedge \mathbf{F}\,\beta_{i,1} \wedge \ldots \wedge \mathbf{F}\,\beta_{i,j_i}$ where $\alpha_i = \mathbf{X}^{n_{i,1}}\,\ell_{i,1} \wedge \ldots \wedge \mathbf{X}^{n_{i,k_i}}\,\ell_{i,k_i}$.

We introduce $\alpha = \bigvee_{\ell=1}^{n} \alpha_\ell$ and $\beta = \bigwedge_{\ell=1}^{n} \bigwedge_{p=1}^{j_\ell} \mathbf{F}\,\beta_{\ell,p}$.

The main step of the proof consists in defining a sequence $(\mathcal{M}^i, f^i, k_i)_{i \in \mathbb{N}}$ where, for each $i \in \mathbb{N}$:

- $\mathcal{M}^i$ is a partial structure, $\mathcal{M}^i \overset{f^i}{\hookrightarrow} \mathcal{M}$ and $\mathcal{M}^i \preccurlyeq \mathcal{M}^{i+1}$,
- up to instant $k_i - 1$, $\mathcal{M}^{i+1}$ coincides with $\mathcal{M}^i$,
- $\mathcal{M}^i$ is built as an extension of $\mathcal{M}^{i-1}$ s.t. for each $k < k_{i-1}$ $\mathcal{M}^i, k \models \exists y \cdot \psi[y]$,
- the limit $\mathcal{M}^\infty$ satisfies $\mathbf{G}(\exists y \cdot \psi[y])$ at instant 0.

The domain $\mathcal{D}$ of the different structures consists of the union of the two disjoints sets $\mathcal{D}_{\mathbf{X}} = \{d_0, \ldots, d_{K_\psi}\}$ and $\mathcal{D}_{\mathbf{F}} = \{e_0, \ldots e_{K_\psi}\}$, and of the set *Const* of constants. That is, $\mathcal{D} = \mathcal{D}_{\mathbf{X}} \cup \mathcal{D}_{\mathbf{F}} \cup Const$.

For $i = 0$, $\mathcal{M}^0$ and the partial function f^0 are defined by: (1) for any $k \in \mathbb{N}$ and $a \in \mathcal{D}_{\mathbf{X}} \cup \mathcal{D}_{\mathbf{F}}$, $f^0_k(a) = \bot$; and (2) $\mathcal{M}^0 \overset{f^0}{\hookrightarrow} \mathcal{M}$.

For any $i > 0$, let us now define $\mathcal{M}^i, k_i$ and f^i. $\mathcal{M}^i = (\mathcal{D}, \sigma^i, \rho^i)$ is defined as an extension of $\mathcal{M}^{i-1}$ in the following way. By application of Lemma 21, it is possible to extend $\mathcal{M}^{i-1}$ up to an instant k_i and satisfy β for one value of the domain. Within the time interval $[k_{i-1}, k_i[$, if i is an odd (resp. even) number, then $\mathcal{M}^i$ is s.t. β is satisfied at instant k_{i-1} for any $a \in \mathcal{D}_{\mathbf{F}}$ (resp. any $a \in \mathcal{D}_{\mathbf{X}}$): $\mathcal{M}^i, k_{i-1}, [y \mapsto a] \models \beta[y]$. If i is an odd (resp. even) number, this defines how $\mathcal{M}^i$ extends $\mathcal{M}^{i-1}$ for elements in $\mathcal{D}_{\mathbf{F}}$ (resp. $\mathcal{D}_{\mathbf{X}}$).

Now, by Lemma 22, if i is an odd (resp. even) number, we can extend $\mathcal{M}^{i-1}$ s.t. for any k in $[k_{i-1}, k_i[$, there is some $a \in \mathcal{D}_{\mathbf{X}}$ (resp. $a \in \mathcal{D}_{\mathbf{F}}$) s.t. $\mathcal{M}^i, k, [y \mapsto a] \models \alpha[y]$. If i is an odd (resp. even) number, this defines how $\mathcal{M}^i$ extends $\mathcal{M}^{i-1}$ for elements in $\mathcal{D}_{\mathbf{X}}$ (resp. $\mathcal{D}_{\mathbf{F}}$). Following this definition, for any $i \in \mathbb{N}$ and any $k < k_i$, $\mathcal{M}^i, k \models \exists y \cdot \psi[y]$.

The limit structure $\mathcal{M}^\infty$ of $(\mathcal{M}^i)_{i \in \mathbb{N}}$ is then a partial model of $\mathbf{G}(\exists y \cdot \psi[y])$, and its domain $\mathcal{D}$ is finite, of size $|Const| + 2 \times (K_\psi + 1)$. ◀

4.2 Relaxing the Use of Quantifiers

The next theorem generalizes the previous result to formulas:

- over n-ary predicates,
- with function symbols,
- and containing any number of existential quantifiers.

▶ **Definition 27.** *We say that $\phi \in Gur(\mathbf{X}, \mathbf{F})$ if there exists a signature Σ and a formula $\psi \in LTL_{\Sigma,\{y_1,\ldots y_n\}}(\mathbf{X}, \mathbf{F})$ such that $\phi = \exists y_1 \ldots y_n \cdot \psi$.*

▶ **Theorem 28.** *Given a formula ϕ in $Gur(\mathbf{X}, \mathbf{F})$, $\mathbf{G}\,\phi$ enjoys the BDP. Denoting $\mathcal{T}_\phi$ the set of terms appearing in ϕ, then, if $\mathbf{G}(\phi)$ is satisfiable, it has a model of size $|\mathcal{T}_\phi \cap \mathcal{T}_{\Sigma,\varnothing}| + 2 \times (K_\phi + 1) \times |\mathcal{T}_\phi \cap \mathcal{T}_{\Sigma,\mathcal{V}}|$.*

The fragment used in Theorem 28 forbids formulas outside the scope of $\mathbf{G}$. This prevents the specification of initial conditions. Proving the BDP for a fragment allowing such conditions requires to handle clauses in the DNF that are satisfied only a *finite* number of times, contrary to what we dealt with until then, using in particular Lemma 24. Theorem 31 states that we can actually extend the fragment used in Theorem 28 by adding a conjunct ψ to $\mathbf{G}(\phi)$ which refers to the initial state (and more generally to a finite set of states). However, the bound of the domain is significantly larger.

▶ **Definition 29.** *Let us assume that ϕ is in the form given in Lemma 18. Then we write $\beta_\phi = |\{\beta \mid \exists i \cdot \psi_i = \alpha_i \wedge \ldots \wedge \mathbf{F}\,\beta \wedge \ldots\}|$.*

▶ **Definition 30** (Genev fragment)**.** *We call* Genev fragment *the set of FOLTL formulas of shape $\psi \wedge \mathbf{G}(\phi)$ s.t. ϕ is a formula of class $Gur(\mathbf{X}, \mathbf{F})$ and $\psi = \exists y_1 \ldots y_2 \cdot \theta[y_1, \ldots, y_n]$ with $\theta \in LTL_{\Sigma,\{y_1,\ldots,y_n\}}$.*

▶ **Theorem 31.** *The Genev fragment enjoys the FDP. If $\psi \wedge \mathbf{G}(\phi)$ is a satisfiable formula in this fragment, it has a model of size $|(\mathcal{T}_\psi \cup \mathcal{T}_\phi) \cap \mathcal{T}_{\Sigma,\varnothing}| + (1 + 2^{\beta_\phi}) \times (K_\phi + 1) \times |\mathcal{T}_\phi \cap \mathcal{T}_{\Sigma,\mathcal{V}}|$.*

Let FO(∀) denote the fragment of purely universal FO formulas containing no other function symbols than constants. The next theorem extends Theorem 31 by allowing formulas of FO(∀) as leaves of the formula instead of basic predicates. However non-constant function symbols can not be used under the scope of a universal quantifier, since even the FO fragment of universally quantified formulas with non-restricted function symbols does not enjoy the FDP.

▶ **Definition 32.** *An FOLTL formula ψ is in FOLTL$(\exists\uparrow, \forall\downarrow)$ if $\psi = \exists y_1 \dots y_2 \cdot \theta[y_1, \dots, y_n]$, where θ has the following syntax: $\theta ::= \ell \mid \alpha \mid \theta \vee \theta \mid \theta \wedge \theta \mid \mathbf{X}\,\theta \mid \theta\,\mathbf{U}\,\theta \mid \theta\,\mathbf{R}\,\theta$, where $\alpha \in$ FO$(\forall)$ and ℓ is a literal.*

▶ Remark 33. Notice in particular that a formula in FOLTL$(\exists\uparrow, \forall\downarrow)$ satisfies the following two conditions: (1) no existential quantifier is in the scope of a temporal operator, (2) no temporal operator is in the scope of a universal quantifier. This is the case, for example, of the following formula: $\exists x, y \cdot (\forall z \cdot \neg P_1(z))\,\mathbf{U}\,(P_1(y)) \wedge (\forall z \cdot \neg P_2(x, z) \Rightarrow P_1(z))$.

▶ **Definition 34.** *FOLTL$(\mathbf{X}, \mathbf{F}, \forall\downarrow)$ is defined by the following grammar: $\phi ::= \ell \mid \alpha \mid \phi \vee \phi \mid \phi \wedge \phi \mid \mathbf{X}\,\phi \mid \mathbf{F}\,\phi \mid \exists y \cdot \phi$, with $\alpha \in$ FO$(\forall)$, ℓ a literal and $y \in \mathcal{V}$.*

▶ **Definition 35** (Geneva fragment)**.** *We call* Geneva fragment *the set of FOLTL formulas of shape $\psi \wedge \mathbf{G}(\phi)$ s.t. ϕ is a closed formula of FOLTL$(\mathbf{X}, \mathbf{F}, \forall\downarrow)$ and ψ is a closed formula of FOLTL$(\exists\uparrow, \forall\downarrow)$.*

▶ **Theorem 36.** *The Geneva fragment enjoys the FDP. If $\psi \wedge \mathbf{G}(\phi)$ is a satisfiable formula in this fragment, it has a model of size $|(\mathcal{T}_\psi \cup \mathcal{T}_\phi) \cap \mathcal{T}_{\Sigma,\varnothing}| + (1 + 2^{\beta_\phi}) \times (K_\phi + 1) \times |\mathcal{T}_\phi \cap \mathcal{T}_{\Sigma,\mathcal{V}}|$.*

4.3 Extension with Equality

We now address the problem of adding the equality predicate to the previous fragments. The interpretation of equality is constant over time. As mentioned in Sect. 3.1, this could be a source of infinity axioms if universal quantification is allowed. We show that we can add equality to the ∀-free fragments of our previous theorems 26, 28, and 31 and still enjoy the BDP. However, the bound on the domain becomes much larger and not exact anymore.

▶ **Definition 37.** *Given an FOLTL formula ϕ, we write $Eq(\phi)$ the set of equality tests of ϕ, i.e. the set of predicates of the form $t_1 = t_2$ in ϕ.*

In the following, Gur$^{=}(\mathbf{X}, \mathbf{F})$ (resp. LTL$^{=}_{\Sigma,\mathcal{V}}$) denotes Gur$(\mathbf{X}, \mathbf{F})$ (resp. LTL$_{\Sigma,\mathcal{V}}$) augmented with equality. Theorem 38 (resp. 39) generalizes Theorem 28 (resp. 31).

▶ **Theorem 38.** *If ϕ is a formula of class Gur$^{=}(\mathbf{X}, \mathbf{F})$ then $\mathbf{G}(\phi)$ enjoys the FDP. Writing $\mathcal{T}_\phi$ for the set of terms appearing in ϕ, then if $\mathbf{G}(\phi)$ is satisfiable, it has a model of size at most $|\mathcal{T}_\phi \cap \mathcal{T}_{\Sigma,\varnothing}| + 2 \times (K_\phi + 1) \times |\mathcal{T}_\phi \cap \mathcal{T}_{\Sigma,\mathcal{V}}| \times 2^{|Eq(\phi)|}$.*

Proof. Consider $\mathcal{M}$ a model of $\mathbf{G}(\phi)$ where $\phi = \exists \vec{y} \cdot \psi$. Theorem 28 can be applied to this formula after replacing equality tests by $\top$. This operation yields a partial structure that we call $\mathcal{M}_0$. Now we want to use $\mathcal{M}_0$ to build a model of $\mathbf{G}(\phi)$. Building such a model requires that, at each instant i, it is possible to find a tuple of elements in the domain that:

- satisfies the same relations as the tuple used to satisfy existential quantifiers at instant i in $\mathcal{M}_0$;
- satisfies the same equality relations as the tuple used to satisfy existential quantifiers at instant i in $\mathcal{M}$.

This can be done by making $2^{|Eq(\phi)|}$ copies of the domain of $\mathcal{M}_0$. Let us remember that this domain is an union of the tuples used to satisfy the existential quantifiers at different instants. Then, for each copy of each tuple, it is possible to define an equivalence relation between terms formed from this tuple. It requires to define these equivalence relations in order to cover all possibilities of interpretation for the equality relations appearing in ϕ, the number of possibilities being $2^{|Eq(\phi)|}$.

Once this is done, quotienting each part of the domain by this relation gives a structure where there are tuples:

- satisfying the same relations as any of the tuple of the first built finite model;
- satisfying any possible subset of $Eq(\phi)$.

So at any instant it is only needed to look in the original model what equality tests of the formula were satisfied and to take the tuple in the appropriate copy of the domain. ◀

▶ **Theorem 39.** *If ϕ is a formula of class $Gur^{=}(\mathbf{X},\mathbf{F})$ and $\psi = \exists y_1,\dots,y_n \cdot \theta[y_1,\dots,y_n]$, where $\theta \in LTL^{=}_{\Sigma,\{y_1,\dots,y_n\}}$, then $\psi \wedge \mathbf{G}(\phi)$ enjoys the FDP. If $\psi \wedge \mathbf{G}(\phi)$ is satisfiable, it has a model of size at most $|(\mathcal{T}_\psi \cup \mathcal{T}_\phi) \cap \mathcal{T}_{\Sigma,\varnothing}| + (1 + 2^{\beta_\phi}) \times 2^{|Eq(\phi)|} \times (K_\phi + 1) \times |\mathcal{T}_\phi \cap \mathcal{T}_{\Sigma,\mathcal{V}}|$.*

Proof. The proof of Theorem 38 can easily be adapted to Theorem 39. ◀

Now, if we extend the fragment of Theorem 36 with equality, it becomes possible to use equality predicates in the scope of a universal quantifier. In that case, our approach does not stand anymore. Therefore, the question of generalizing Theorem 36 by adding equality remains open.

5 Toy Example: a Notification System

Here, a simple example of a notification system in a ring is presented to illustrate the expressiveness of the fragment[4]. The structure of the network is defined by a predicate `succ` relating a node to its successor, and a formula **Ring** specifying that `succ` forms a ring topology. The formula **Ring** is specified as proposed in [15], with the use of a ternary predicate, in pure FO (without transitive closure). Each node x of the ring may be aware ($\texttt{notified}(x)$) of a certain piece of information, or not. Any node that has been notified may notify its successor (by *sending* a message), and other nodes do not change during this operation.

$$\begin{aligned}
\mathbf{Same}(z) &:= \texttt{notified}(z) \Leftrightarrow \mathbf{X}\,\texttt{notified}(z)\\
\mathbf{Send}(x) &:= \exists y \cdot \Big[\texttt{succ}(x,y) \wedge (\forall z \cdot z \neq y \Rightarrow \mathbf{Same}(z))\\
&\qquad \wedge (\texttt{notified}(x) \Rightarrow \mathbf{X}\,\texttt{notified}(y)) \wedge (\neg\texttt{notified}(x) \Rightarrow \mathbf{Same}(y))\Big]\\
\mathbf{Trans} &:= \mathbf{G}(\exists p \cdot \mathbf{Send}(p))
\end{aligned}$$

5.1 Safety Property

Now consider the safety property "if a node is notified, it remains notified", described by the following formula: $\mathbf{Safety} := \mathbf{G}(\forall x \cdot \texttt{notified}(x) \Rightarrow \mathbf{X}\,\texttt{notified}(x))$. Proving that our protocol ensures this property ($\mathbf{Ring} \wedge \mathbf{Trans} \models \mathbf{Safety}$) amounts to proving that $\mathbf{Ring} \wedge \mathbf{Trans} \wedge \neg\mathbf{Safety}$ is unsatisfiable.

[4] The complete Electrum specification is available at [16]. Electrum is available at `http://huit.re/electrum/`.

Notice that $\neg\mathbf{Safety} \equiv \exists x \cdot \mathbf{F}(\mathtt{notified}(x) \wedge \mathbf{X}\,\neg\mathtt{notified}(x))$, therefore an equisatisfiable formula can be obtained by Skolemization: $\mathbf{SkNegSafety} := \mathbf{F}(\mathtt{notified}(c) \wedge \mathbf{X}\,\neg\mathtt{notified}(c))$. However $\varphi := \mathbf{Ring} \wedge \mathbf{Trans} \wedge \mathbf{SkNegSafety}$ is not in any of our fragments because of the universal quantification over $\mathbf{Same}(z)$, which is a temporal formula, and because of the use of equality.

We now devise a more abstract specification of the protocol which is a semantic consequence of φ that fits into the fragment of Theorem 36. First, we get rid of equality: we use an equivalence predicate $\approx$ instead, which can be axiomatized (using a formula $\overline{\mathbf{Eq}}$) in our fragment (notice that the semantics of $\approx$ may vary over time). Second, we get rid of the universal quantifier over z. To do that, a solution is to instantiate the variable z for the values x and c, which yields: $\overline{\mathbf{Send}}(x) := (\exists y \cdot \mathtt{succ}(x,y) \wedge (\mathtt{notified}(x) \Rightarrow \mathbf{X}\,\mathtt{notified}(y)) \wedge (\neg\mathtt{notified}(x) \Rightarrow \mathbf{Same}(y)) \wedge (x \approx y \vee \mathbf{Same}(x)) \wedge c \approx y \vee \mathbf{Same}(c)) \wedge (c \approx y \Leftrightarrow \mathbf{X}\,c \approx y))$. Notice it is necessary to add $c \approx y \Leftrightarrow \mathbf{X}\,c \approx y$ in the previous formula. Indeed, $\approx$ is not necessarily constant so it would be possible to have that $c \approx y$ and $\neg\,\mathbf{X}\,c \approx y$ and, in this case, no constraint would apply to the truth value of $\mathbf{X}\,\mathtt{notified}(c)$. Then, it is possible to define an abstraction by the following formulas: $\overline{\mathbf{Trans}} := \mathbf{G}(\exists p \cdot \overline{\mathbf{Send}}(p))$ and $\mathbf{AbsSatS} := \overline{\mathbf{Eq}} \wedge \overline{\mathbf{Trans}} \wedge \mathbf{SkNegSafety}$.

It is easy to show that $\mathbf{Ring} \wedge \mathbf{Trans} \wedge \neg\mathbf{Safety} \models \mathbf{AbsSatS}$ and that $\mathbf{AbsSatS}$ belongs to the Geneva fragment. Applying Theorem 36, we compute a size 5 for the domain. Using the Electrum tool, $\mathbf{AbsSatS}$ can be shown to be unsatisfiable for a domain of size 5, which ultimately proves the original property.

5.2 Liveness Property

An interesting liveness property to prove on the considered system is "all nodes eventually become notified", formalized as: $\mathbf{Liveness} := \forall x \cdot \mathbf{F}(\mathtt{notified}(x))$. This property can be shown under the assumption that all notified nodes eventually perform the send transition: $\mathbf{Progress} := \mathbf{G}(\forall x \cdot \mathtt{notified}(x) \Rightarrow \mathbf{F}\,\mathbf{Send}(x))$.

The complete abstraction that allows us to prove this liveness property is available with the full example specification. We basically need to Skolemize the negation of the liveness property and to instantiate the universally quantifiers that are out of our fragment with the Skolem constant. An axiom abstracting the ring topology needs to be added for proving the liveness property. In the end, the obtained formula fits into the fragment of Theorem 36, which provides the domain size 6. The formula can be shown in Electrum to be unsatisfiable for a domain of size 6, which proves the property.

6 Related Work

In [8], Kuperberg and the last two authors of the present article show that the FDP for some FO fragments can be lifted to some FOLTL fragments. However, they only allow to add $\mathbf{X}$ and $\mathbf{F}$ connectives, which is not enough for real specifications. An extension of the Ramsey fragment is also proposed, allowing the use of all temporal connectives, but preventing existential quantifiers under a $\mathbf{G}$.

The decidable monodic fragment studied by Hodkinson et al. [5,6] does not enjoy the FDP. Indeed, $\mathbf{G}(\exists y \cdot P(y) \wedge \mathbf{G}(\neg P(y)))$ belongs to the monodic extension of the Gurevich fragment (first-order formulas containing existential quantifiers only) but it is an axiom of infinity: the monodic fragment helps preserve decidability but says nothing about the FDP. Additionally, on the practical side, the monodic fragment limits the use of free variables in temporal formulas to only one, which does not really fit with real specifications of systems. Indeed,

any transition system implying relations between different components (list of messages, topology of a network, etc) requires to be specified by using at least binary relations in the temporal transitions, thus breaking the monodicity condition.

Padon et al. [15] propose yet another approach: they reduce specific temporal problems to FO and even, in many cases, to a *decidable* fragment of it. This method was improved in [13, 14] to address the verification of liveness properties. It was implemented in the Ivy tool and gives good results in practice. However, it is not complete and it requires the user to understand rather deeply both the specified system and the verification technique itself. Additionally, the user must devise an inductive invariant manually.

7 Discussion

In the introduction, we drew as an inspiration for our work the following classical shape for specifications of systems and of their properties: $\mathit{spec} = \mathit{init} \wedge \mathbf{G}\,\mathit{trans} \wedge \mathit{fair} \rightarrow \mathit{prop}$ (with *trans* using only the $\mathbf{X}$ connective). Checking the validity of *spec* amounts to assessing the satisfiability of $\neg\mathit{spec} = \mathit{init} \wedge \mathbf{G}\,\mathit{trans} \wedge \mathit{fair} \wedge \neg\mathit{prop}$. Our results then say this satisfiability can be decided provided $\neg\mathit{spec}$ can written as $\psi \wedge \mathbf{G}(\phi)$ and respect the conditions of Theorems 31, 36 or 39.

Beyond the obvious *init* and *trans*, one can see that, depending on their shape, *fair* and *prop* will have to be, possibly, split into sub-formulas, and then "dispatched" into either ψ or ϕ, or both. As an example, for *prop*, any combination of an $\mathbf{F}$ or $\mathbf{G}$ connective and of some quantification on a variable is acceptable, except for the shape $\exists x \cdot \mathbf{G}(P(x))$ (as we would have $\neg\mathit{prop} = \forall x \cdot \mathbf{F}(\neg P(x))$, in which case $\forall$ would not appear as a leaf). Similarly, for a strong fairness property of the shape $\mathbf{G}\,\mathbf{F}\,\mathit{enabled} \rightarrow \mathbf{G}\,\mathbf{F}\,\mathit{effect} = \mathbf{F}\,\mathbf{G}\,\neg\mathit{enabled} \vee \mathbf{G}\,\mathbf{F}\,\mathit{effect}$, the disjunction distributes over the rest of the formula, and then *enabled* may only contain existential quantifiers at the leaves and universal ones at the root, and *effect* may only have universal quantifiers at the leaves (without any constraint on existential ones). This may sometimes be restrictive, in which case alternative expressions should be sought. Another limitation lies in the possible uses of (constant) equality: in our first experiments, we were often able to abstract it into a dynamic equivalence relation, as we did in Sect. 5.

Now, if the specification falls into one of our fragments, then the bound on the domain is known (and even exact, without equality). To be sure, this bound grows exponentially but only in the number of $\mathbf{F}$ connectives under a $\mathbf{G}$. This ultimately yields a decision procedure for the validity of *spec*. Notice that existing tools, such as our own Electrum [10], can readily be used to support it, as was shown in Sect. 5.

In the future, we will study ways to augment the expressiveness of our fragments to address some of the current limitations (e.g. fairness, equality). Apart from trying to extend the fragments themselves, we will also devise a many-sorted version thereof, in the spirit of [11, 12, 1, 15], to extend their expressiveness and applicability and to fit the data structuring features of Electrum [10] more closely. We will also assess our approach on more realistic case studies. Finally, we will build on our results in the setting of complete, automated verification for system specification.

References

1 Aharon Abadi, Alexander Rabinovich, and Mooly Sagiv. Decidable fragments of many-sorted logic. *Journal of Symbolic Computation*, 45(2):153–172, 2010. doi:10.1016/j.jsc.2009.03.003.

2 Julien Brunel, David Chemouil, Alcino Cunha, and Nuno Macedo. The Electrum Analyzer: Model Checking Relational First-Order Temporal Specifications. In *33rd ACM/IEEE International Conference on Automated Software Engineering (ASE '18)*, Montpellier, France, September 2018. ACM Press. doi:10.1145/3238147.3240475.

3 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997. doi:10.1007/978-3-642-59207-2.

4 Dov M. Gabbay, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*, chapter Fragments of first-order temporal logics. Elsevier, 2003.

5 Ian Hodkinson, Frank Wolter, and Michael Zakharyaschev. Decidable fragments of first-order temporal logics. *Annals of Pure and Applied logic*, 106(1-3):85–134, 2000. doi:10.1016/S0168-0072(00)00018-X.

6 Ian Hodkinson, Frank Wolter, and Michael Zakharyaschev. Monodic Fragments of First-Order Temporal Logics: 2000–2001 A.D. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 1–23. Springer, 2001. doi:10.1007/3-540-45653-8_1.

7 Fred Kröger and Stephan Merz. *Temporal Logic and State Systems (Texts in Theoretical Computer Science. An EATCS Series)*. Springer, 2008.

8 Denis Kuperberg, Julien Brunel, and David Chemouil. On Finite Domains in First-Order Linear Temporal Logic. In *Automated Technology for Verification and Analysis*, pages 211–226. Springer, 2016. doi:10.1007/978-3-319-46520-3_14.

9 Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Professional, 2002.

10 Nuno Macedo, Julien Brunel, David Chemouil, Alcino Cunha, and Denis Kuperberg. Lightweight Specification and Analysis of Dynamic Systems with Rich Configurations. In *Foundations of Software Engineering*, Seattle, United States, November 2016. doi:10.1145/2950290.2950318.

11 Timothy Nelson, Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. On the finite model property in order-sorted logic. Technical report, Worcester Polytechnic Institute, 2010.

12 Timothy Nelson, Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. Toward a More Complete Alloy. In *Abstract State Machines, Alloy, B, VDM, and Z*, pages 136–149. Springer, 2012. doi:10.1007/978-3-642-30885-7_10.

13 Oded Padon, Jochen Hoenicke, Giuliano Losa, Andreas Podelski, Mooly Sagiv, and Sharon Shoham. Reducing liveness to safety in first-order logic. *Proceedings of the ACM Conference on Principles of Programming Languages (POPL)*, 2:26, 2017. doi:10.1145/3158114.

14 Oded Padon, Jochen Hoenicke, Kenneth L McMillan, Andreas Podelski, Mooly Sagiv, and Sharon Shoham. Temporal Prophecy for Proving Temporal Properties of Infinite-State Systems. In *Formal Methods in Computer Aided Design (FMCAD)*, 2018. doi:10.23919/FMCAD.2018.8603008.

15 Oded Padon, Kenneth L. McMillan, Aurojit Panda, Mooly Sagiv, and Sharon Shoham. Ivy: safety verification by interactive generalization. *ACM SIGPLAN Notices*, 51(6):614–630, 2016. doi:10.1145/2980983.2908118.

16 Quentin Peyras, Julien Brunel, and David Chemouil. Electrum specification of a toy notification system, August 2019. doi:10.5281/zenodo.3369542.

Hybrid SAT-Based Consistency Checking Algorithms for Simple Temporal Networks with Decisions

Matteo Zavatteri
Department of Computer Science, University of Verona, Italy
matteo.zavatteri@univr.it

Carlo Combi
Department of Computer Science, University of Verona, Italy
carlo.combi@univr.it

Romeo Rizzi
Department of Computer Science, University of Verona, Italy
romeo.rizzi@univr.it

Luca Viganò
Department of Informatics, King's College London, UK
luca.vigano@kcl.ac.uk

Abstract

A Simple Temporal Network (STN) consists of time points modeling temporal events and constraints modeling the minimal and maximal temporal distance between them. A Simple Temporal Network with Decisions (STND) extends an STN by adding *decision time points* to model temporal plans with decisions. A decision time point is a special kind of time point that once executed allows for deciding a truth value for an associated Boolean proposition. Furthermore, STNDs label time points and constraints by conjunctions of literals saying for which *scenarios* (i.e., complete truth value assignments to the propositions) they are relevant. Thus, an STND models a family of STNs each obtained as a *projection* of the initial STND onto a *scenario*. An STND is consistent if there exists a consistent scenario (i.e., a scenario such that the corresponding STN projection is consistent). Recently, a hybrid SAT-based consistency checking algorithm (HSCC) was proposed to check the consistency of an STND. Unfortunately, that approach lacks experimental evaluation and does not allow for the synthesis of all consistent scenarios. In this paper, we propose an incremental HSCC algorithm for STNDs that (i) is faster than the previous one and (ii) allows for the synthesis of all consistent scenarios and related early execution schedules (offline temporal planning). Then, we carry out an experimental evaluation with KAPPA, a tool that we developed for STNDs. Finally, we prove that STNDs and disjunctive temporal networks (DTNs) are equivalent.

2012 ACM Subject Classification Theory of computation → Timed and hybrid models; Computing methodologies → Temporal reasoning; Computing methodologies → Planning and scheduling; Mathematics of computing → Graph algorithms; Hardware → Theorem proving and SAT solving

Keywords and phrases Simple temporal network with decisions, HSCC algorithms, incremental SAT-solving, disjunctive temporal network, KAPPA

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.16

Supplement Material `https://github.com/matteozavatteri/kappa`

1 Introduction

Context and motivations. Temporal networks are a possible formalism to model temporal plans and check the coherence of temporal constraints that impose lower and upper bounds on the temporal distance of the modeled events. A *Simple Temporal Network* (*STN*, [11]) is a formalism able to model an unconditional temporal plan in which all components (events

and their time distances) are under control. Temporal events are modeled as *time points* and their occurrence is modeled by executing the corresponding time points (i.e., by assigning them real values).

A temporal plan is *consistent* if we can schedule all the events, each at a specific time instant, such that all constraints are satisfied. If this is possible, a schedule can be synthesized for both offline planning (the plan is available before starting) or online planning (the plan is generated while executing). The difference between these two planning approaches is that in the former the consistency checking algorithm returns a solution, whereas in the latter the algorithm returns a *minimal network* to generate any solution.

However, STNs fail to model temporal plans where the occurrence of some events or the satisfaction of some constraints must happen only if some decision has been made. The decisions we are interested in have Boolean domain[1]. Thus, a temporal plan subject to decisions would ask us (not) to execute some time points or to satisfy some constraints depending on what decisions we have made.

A few proposals to handle decisions within the temporal network formalisms built on top of STNs have been put forth. For instance, *Drake* [9] provides an executive for temporal plans with choices based on *Labeled STNs* that do not specify decision points (in a node-sense), whereas *Temporal Plan Networks (TPNs, [21])* extend STNs with decision nodes and model decisions as outgoing edges from such nodes. *Simple Temporal Networks with Decisions* (*STNDs*, [4, 29]) extend STNs by adding decision points that can influence both the execution of time points and the satisfaction of constraints. Several (possibly different) STNs may arise when projecting an STND onto a scenario that models the complete interpretation of the decisions. For any scenario, we are only interested in executing the time points and satisfy the constraints *entailed* by it.

So far, only one hybrid SAT-based consistency checking algorithm (*HSCC*) has been devised to check the consistency of an STND [4]. If the algorithm finds a consistent scenario, i.e., a scenario for which the STN projection is consistent, then a solution (*scenario* plus *schedule*) exists. In this case, any schedule in the solution set involves the STN-projection corresponding to the related scenario.

This algorithm allows for an offline planning where all decisions are made before starting and the corresponding schedule is already known. However, this algorithm has never been implemented nor evaluated, and it does not allow for the synthesis of all consistent scenarios.

Contributions. Our contribution is three-fold. First, we provide `STND-HSCC2`, a novel HSCC algorithm for STNDs that (i) is faster than the existing algorithm as it rules out inconsistent scenarios as early as possible, and (ii) allows for the synthesis of all consistent scenarios and related early execution schedules (offline temporal planning). The algorithm is hybrid because a SAT-solver and a shortest path algorithm mutually influence each other (the output of the former becomes the input of the latter and vice versa continuously). Second, we discuss Kappa, a tool that we developed for STNDs for the experimental evaluation and in which, as a minor contribution, we adapted the previous algorithm to support the synthesis of all consistent scenarios. Third, we prove that STNDs are equivalent to disjunctive temporal networks (DTNs).

[1] This is not a restriction as every finite set $\{0, 1, 2, \ldots, n-1\} \subset \mathbb{N}$ of different discrete choices (i.e., decisions) can be represented in an equivalent binary notation by using $\lceil \log n \rceil$ bits where if the i^{th} bit is 1 (respectively, 0), then it means that the i^{th} decision holds (respectively, does not hold). Of course this binary representation may express more than n possibilities (e.g., to express $\{0, 1, 2\}$ we need $\lceil \log 3 \rceil = 2$ bits with which we can also express "3" when all bits are set). Again, this is not a problem if we add unsatisfiable constraints for all combinations representing numbers $\geq n$ (if any).

Organization. Section 2 provides background on STNs and STNDs, and the current consistency checking algorithm for STNDs. Section 3 discusses `STND-HSCC2`, a faster HSCC algorithm for STNDs. Section 4 discusses KAPPA and the related experimental evalutation. Section 5 proves that STNDs are equivalent to DTNs. Section 6 discusses related work. Section 7 sums up and discusses future work.

2 Background

A *Simple Temporal Network* (*STN*, [11]) is a pair $\langle \mathcal{T}, \mathcal{C} \rangle$, where $\mathcal{T} = \{X, \dots\}$ is a set of time points (continuous variables) and $\mathcal{C} = \{(Y - X \leq k), \dots\}$ is a set of constraints, for $X, Y \in \mathcal{T}$, $k \in \mathbb{R} \cup \pm\{\infty\}$.

An STN is *consistent* if there exists an assignment of real values to the time points such that all constraints are satisfied.

Given a consistent STN, a *schedule* is a function $S \colon \mathcal{T} \to \mathbb{R}$ assigning real values to time points such that if X is executed before Y, then $S(X) \leq S(Y)$. An *early execution* of an STN consists of finding a schedule executing the time points as soon as possible (e.g., by using the Floyd-Warshall algorithm [11]).

Given a set $\mathcal{P} = \{d, \dots\}$ of Boolean propositions, a *label* $\ell = \lambda_1 \dots \lambda_n$ is any finite conjunction of literals λ_i, where a literal is either d (positive literal) or $\neg d$ (negative literal), and we omit the $\wedge$ connective to ease reading. The *empty label* is denoted by $\boxdot$. The *label universe of* $\mathcal{P}$, denoted by $\mathcal{P}^*$, is the set of all possible (consistent) labels drawn from $\mathcal{P}$; e.g., if $\mathcal{P} = \{d, e\}$, then $\mathcal{P}^* = \{\boxdot, d, e, \neg d, \neg e, de, d\neg e, \neg de, \neg d\neg e\}$. Two labels ℓ_1, ℓ_2 are *consistent* if their conjunction $\ell_1\ell_2$ is satisfiable. A label ℓ_1 *entails* a label ℓ_2 (written $\ell_1 \Rightarrow \ell_2$) if all literals in ℓ_2 appear in ℓ_1 too (i.e., if ℓ_1 is more *specific* than ℓ_2). For instance, if $\ell_1 = d\neg e$ and $\ell_2 = d$, then ℓ_1 and ℓ_2 are consistent since $d\neg ed$ is satisfiable, and ℓ_1 entails ℓ_2 since $d\neg e \Rightarrow d$.

A *scenario* is a mapping $s \colon \mathcal{P} \to \{\top, \bot\}$ assigning a truth value to each $d \in \mathcal{P}$. A scenario *satisfies* a label ℓ (in symbols $s \models \ell$) if ℓ evaluates to true under the interpretation given by s (e.g., if $s(d) = \top$ and $s(e) = \bot$, then $s \models d\neg e$).

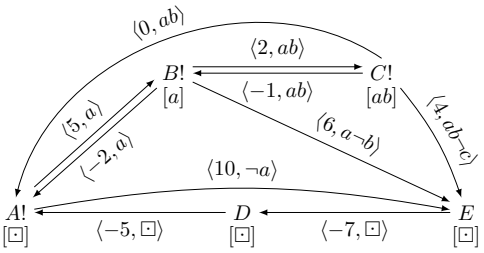
▶ **Definition 1.** *A* Simple Temporal Network with Decisions *(*STND*, [4, 29, 33]) is a tuple* $\mathcal{S} = \langle \mathcal{T}, \mathcal{DT}, \mathcal{P}, O, L, \mathcal{C} \rangle$*, where:*

- $\mathcal{T} = \{X, \dots, Z\}$ *is a finite set of time points.*
- $\mathcal{DT} \subseteq \mathcal{T} = \{D!, \dots, H!\}$ *is a finite set of decision time points.*
- $\mathcal{P} = \{d, \dots, g\}$ *is a finite set of Boolean propositions.*
- $O \colon \mathcal{P} \to \mathcal{DT}$ *is a bijection assigning a unique proposition to each decision time point* $D!$ *that controls the truth value assignment to* d *(*$O^{-1} \colon \mathcal{DT} \to \mathcal{P}$ *models the inverse).*
- $L \colon \mathcal{T} \to \mathcal{P}^*$ *is a function assigning labels to time points.*
- $\mathcal{C}$ *is a finite set of* labeled *constraints* $(Y - X \leq k, \ell)$ *where* $X, Y \in \mathcal{T}$, $k \in \mathbb{R} \cup \pm\{\infty\}$ *and* $\ell \in \mathcal{P}^*$.

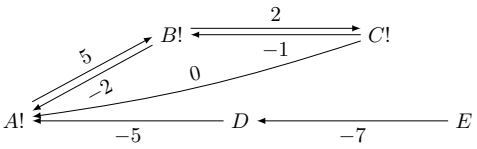
The STN-projection of an STND $\mathcal{S}$ *with respect to a scenario* s *(written* $\pi_s(\mathcal{S})$*) is an STN* $\langle \mathcal{T}_s, \mathcal{C}_s \rangle$ *built as follows:*

- $\mathcal{T}_s = \{X \mid X \in \mathcal{T} \wedge s \models L(X)\}$
- $\mathcal{C}_s = \{(Y - X \leq k) \mid (Y - X \leq k, \ell) \in \mathcal{C} \wedge s \models \ell\}$

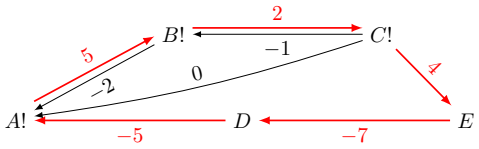
$\mathcal{S}$ *is consistent if there exists a scenario* s *such that* $\pi_s(\mathcal{S})$ *is consistent. A solution is a pair* $\langle s, S \rangle$*, where* s *is a scenario,* S *is a schedule with domain* $\mathcal{T}_s$*, and* $S(Y) - S(X) \leq k$ *holds for each* $(Y - X \leq k) \in \mathcal{C}_s$*. Checking consistency of STNDs is NP-complete [4].*



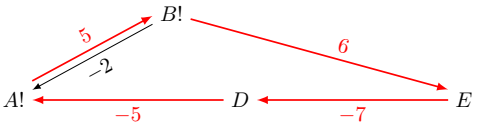
(a) Simple Temporal Network with Decisions.



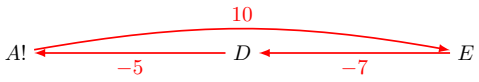
(b) STN-Projection onto $s(a) = s(b) = s(c) = \top$.



(c) STN-Projection onto $s(a) = s(b) = \top, s(c) = \bot$.



(d) STN-Projection onto $s(a) = \top, s(b) = \bot$, $s(c) \in \{\top, \bot\}$.



(e) STN-Projection onto $s(a) = \bot, s(b)$, $s(c) \in \{\top, \bot\}$.

Figure 1 An example of STND and related STN-projections (where thick red edges highlight negative cycles).

We represent an STND graphically by extending the distance graph of an STN into a labeled distance (multi)graph. The set of nodes still coincides with the set of time points, whereas each edge $X \to Y$ labeled by $\langle k, \ell \rangle$ represents $(Y - X \leq k, \ell) \in \mathcal{C}$. Time points' labels are shown below the nodes. Many $\langle k, \ell \rangle$ can be specified for the same $X \to Y$ provided their ℓ are different (if two labels are equal, we keep the smallest k). Figure 1a shows an example of STNDs, whereas Figures 1b–1e show its STN-projections.

A label ℓ labeling a time point or a constraint is *honest* if for each literal d or $\neg d$ in ℓ we have that $\ell \Rightarrow L(D!)$, where $D! = O(d)$ is the decision time point associated to d; ℓ is dishonest otherwise. For example, consider $L(C!) = ab$ in Figure 1a. $C!$ appears in a solution only if $s(a) = s(b) = \top$. However, deciding $\top$ for b implies that $B!$ appears in the solution too, which in turn requires that $A!$ appears (before $B!$) in the same solution with $\top$ decided for a. Thus, an honest ℓ containing b or $\neg b$ should also contain a. A label on a constraint is *coherent* if it is at least as expressive as the labels of the time points appearing in the constraint (i.e., ℓ contains all literals in the labels of the two connected time points).

▶ **Definition 2** (Well-definedness). *An STND is well-defined [17, 29, 33] if*

- $L(X) \Rightarrow L(O(d))$ *and* $(O(d) - X \leq 0) \in \mathcal{C}$ *for each* $X \in \mathcal{T}$ *and* $\{d, \neg d\} \in L(X)$*, and*
- $\ell \Rightarrow L(Y) \wedge L(X)$ *for each* $(Y - X \leq k, \ell) \in \mathcal{C}$*, and* $\ell \Rightarrow L(O(d))$ *for each literal* $\{d, \neg d\} \in \ell$*.*

Figure 1a is an example of well-defined STND modeling a temporal plan with 3 decisions ($A!$, $B!$ and $C!$) and two (instantaneous) activities (D and E). $A!$ is the first time point to execute. We execute $B!$ if and only if we decided $\top$ for a ($L(B!) = a$) and $C!$ if we further decided $\top$ for b too ($L(C!) = ab$). Instead, $A!$, D and E are always executed ($L(A!) = L(D) = L(C) = \boxdot$). We can execute $B!$ (if we decide so) after at least 2 ($B! \to A!$

Algorithm 1 `STND-HSCC1`($\mathcal{S}$).

Input: An STND $\mathcal{S} = \langle \mathcal{T}, \mathcal{DT}, \mathcal{P}, O, L, \mathcal{C} \rangle$.
Output: A solution $\langle s, S \rangle$ for $\mathcal{S}$ if $\mathcal{S}$ is consistent; "*inconsistent*" if $\mathcal{S}$ is inconsistent.
`STN-CC` is a consistency checking algorithm for STNs.

```
1 φ ← ⋀_{p∈P}(p ∨ ¬p)                                  ▷ Make every assignment possible
2 while true do
3     s ← SAT-SOLVE(φ)                                  ▷ Try to find a satisfying assignment s
4     if φ is unsatisfiable then
5         return inconsistent
6     ⟨T_s, C_s⟩ ← π_s(S)                               ▷ Project S onto s
7     if STN-CC(⟨T_s, C_s⟩) then
8         return ⟨s, S⟩                                 ▷ where S is an early schedule for ⟨T_s, C_s⟩
9     φ ← φ ∧ CUT-SCENARIO(S, ⟨T_s, C_s⟩)               ▷ Exclude this scenario
```

labeled by $\langle -2, a \rangle$) and within 5 time units since $A!$ was executed ($A! \to B!$ labeled by $\langle 5, a \rangle$). The same happens for $C!$ with respect to $B!$ (after 1 and within 2 time units since $B!$). Instead, we always execute D after minimum 5 time units since $A!$ ($D \to A!$ labeled by $\langle -5, \boxdot \rangle$) and E after 7 time units since D ($E \to D$ labeled by $\langle -7, \boxdot \rangle$). Furthermore, if we decide

- $\bot$ for a, then we must execute E within 10 time units since $A!$ ($A! \to E$ labeled by $\langle 10, \neg a \rangle$),
- $\top$ for a and $\bot$ for b, then we must execute E within 6 time units since $B!$ ($B! \to E$ labeled by $\langle 6, a\neg b \rangle$),
- $\top$ for both a and b and $\bot$ for c, then we must execute E within 4 time units since $C!$ ($C! \to E$ labeled by $\langle 4, ab\neg c \rangle$).

Therefore negative values on edges model delays, positive ones model deadlines.

To check the consistency of an STND, we can use the *hybrid SAT-based consistency checking (HSCC)* algorithm proposed in [4]. `STND-HSCC1`, specified in Algorithm 1, (`STND-CC` in [4]) maintains a formula φ specifying CNF clauses over propositions in $\mathcal{P}$. Initially, φ allows for all truth value assignments. In each round of the algorithm we ask the SAT solver for a truth value assignment making φ true. Such an assignment (if any) corresponds to a scenario s over which we can project the STND and check if the resulting STN is consistent ("SAT-solver influences directed weighted graph algorithm"). If so, we return this scenario and a valid schedule for the projected STN (i.e., a solution). Otherwise, we apply De Morgan's rules to the negation of the *relevant part* of the scenario containing the negative cycle (`CUT-SCENARIO` in Algorithm 2) and add the resulting clause to φ and go ahead with the next round ("directed weighted graph algorithm influences the SAT-solver"). This makes the approach hybrid. If φ has become unsatisfiable it means that all STN-projections are inconsistent and therefore the STND is inconsistent. Similar approaches are described in [23, 28].

An example of round for the network in Figure 1a is as follows. Suppose that `SAT-SOLVE`(φ) $= ab\neg c$ (i.e., $s(a) = s(b) = \top$ and $s(c) = \bot$). Since the STN $\pi_{ab\neg c}(\mathcal{S})$ is inconsistent (Figure 1c admits a negative cycle), we add to φ the clause $\neg(a \wedge b \wedge \neg c)$, which simplifies to $(\neg a \vee \neg b \vee c)$, to ask the SAT solver for a different truth value assignment excluding this projection (if any). Figure 1a is consistent if and only if $s(a) = s(b) = s(c) = \top$. A possible schedule for Figure 1b is $S(A!) = 0, S(B!) = 2, S(C!) = 3, S(D) = 5, S(E) = 12$. Any other combination leads to a projection containing a negative cycle (Figures 1c-1e).

Algorithm 2 `CUT-SCENARIO`$(\mathcal{S}, \langle \mathcal{T}_s, \mathcal{C}_s \rangle)$.

Input: An STND $\mathcal{S} = \langle \mathcal{T}, \mathcal{DT}, \mathcal{P}, O, L, \mathcal{C} \rangle$ and one of its STN-projections $\langle \mathcal{T}_s, \mathcal{C}_s \rangle$
Output: A clause ψ expressing the cut of the relevant part of the scenario.

1 $\psi \leftarrow \top$ ▷ ψ will contain the relevant part of s
2 **foreach** *constraint* $c \in \mathcal{C}_s$ **do**
3 $\quad \psi \leftarrow \psi \wedge \ell_c$ ▷ where ℓ_c is the corresponding label of c in $\mathcal{S}$
4 **return** `DeMorgan`$(\neg(\psi))$ ▷ the clause expressing the cut of ψ

3 A Faster HSCC Algorithm for STNDs

`STND-HSCC1` is correct [4], but it suffers from the limitation that projections are tested only when the SAT solver returns a complete truth value assignment. Consider Figure 1a and assume that the SAT solver starts on the formula $\varphi = (a \vee \neg a) \wedge (b \vee \neg b) \wedge (c \vee \neg c)$, which makes every truth value assignment possible. Suppose that in the search tree the SAT solver decides $\bot$ for a proposition d going down to the left and $\top$ going down to the right in the search tree, and assume that the order of visit is left then right. The first truth value assignment returned is $a = \bot$, $b = \bot$ and $c = \bot$ (corresponding to the scenario $s(a) = s(b) = s(c) = \bot$). Now `STND-HSCC1` would project the STND in Figure 1a onto s to obtain the STN shown in Figure 1e and eventually detect the negative cycle. However, the negative cycle could have been detected much earlier, say, when a was assigned $\bot$. Indeed, all projections of any scenario containing $s(a) = \bot$ boil down to Figure 1e (no matter which Boolean values are assigned to b and c). Therefore, a clever implementation of this algorithm calls for an early detection of negative cycles. Before proceeding with it, we must refine the concept of scenario and projection so that they support "unknown" propositions (i.e., propositions that have not been assigned a value yet).

▶ **Definition 3.** *A* scenario *is (now) a mapping* $s \colon \mathcal{P} \to \{\top, \bot, -\}$ *assigning either* true, false *or* unknown *to each proposition* $d \in \mathcal{P}$*. A scenario* s satisfies *a label* ℓ *if* ℓ *evaluates to true under the following interpretation given by* s*:*

1. $s \models \lambda$ *iff* $(\lambda = d \wedge s(d) = \top)$ *or* $(\lambda = \neg d \wedge s(d) = \bot)$,
2. $s \models \ell$ *iff* $s \models \lambda_1$ *and* $\ldots$ *and* $s \models \lambda_n$ *for* $\ell = \lambda_1 \ldots \lambda_n$.

Note that s *never satisfies a label containing a literal for which the corresponding proposition is unknown in* s*.*

The definition of STN projection remains the same as that given in Definition 1 but extended to the new definition of scenario. As a result, Figure 1e now becomes a representative also for any scenario s such that $s(a) = \bot$ and $s(b), s(c) \in \{\top, \bot, -\}$. Another example is Figure 1d, extending $s(c) \in \{\top, \bot\}$ to $s(c) \in \{\top, \bot -\}$. Now we have everything we need to hunt down inconsistent scenarios as early as possible.

`STND-HSCC2` (Algorithm 3) is a brand new algorithm to check the consistency of STNDs. It allows for the synthesis of a single or all scenarios admitting a consistent schedule for the corresponding STN-projection. `STND-HSCC2` still initializes a CNF formula φ making all truth value assignments possible. Then, it starts the SAT-solver and hooks a listener to the corresponding run. Such a listener is able to operate on φ by adding CNF clauses on the fly if needed and is triggered by two main events: *assume* and *solution found*.

An *assume* ($d = \top$ or $d = \bot$) event (Algorithm 3, lines 8-12) triggers an action of the listener to "look ahead" if the STN-projection obtained by projecting the STND onto the scenario built from the current truth value assignment and *extended* with this assumption

Algorithm 3 `STND-HSCC2`$(\mathcal{S}, all)$.

Input: An STND $\mathcal{S} = \langle \mathcal{T}, \mathcal{DT}, \mathcal{P}, O, L, \mathcal{C} \rangle$ and a Boolean value *all* meaning *all consistent scenarios* iff $all = \top$.
Output: A single or all scenarios s along with schedule(s) S for the projection $\pi_s(\mathcal{S})$ if $\mathcal{S}$ is consistent, "*inconsistent*" otherwise.

1 $\varphi \leftarrow \bigwedge_{p \in \mathcal{P}} (p \vee \neg p)$ ▷ Make every assignment possible
2 $Sols = \emptyset$ ▷ The set of (all) consistent scenarios (global variable)
3 Hook a *listener* to the run of the SAT solver and make it detect negative cycles as early as possible (**on** blocks (below) define the event-driven behavior).
4 `SAT-SOLVE`(φ)
5 **if** $Sols = \emptyset$ **then**
6 **return** *inconsistent*
7 **return** *Sols*
8 **on** *assume* $d = \top$ *or assume* $d = \bot$*:* ▷ Partial model
9 Build a scenario s from the current truth value assignment extended with $s(d) = \top$ or $s(d) = \bot$ (depending on the case)
10 $\langle \mathcal{T}_s, \mathcal{C}_s \rangle \leftarrow \pi_s(\mathcal{S})$ ▷ Get the STN projection
11 **if** *BellmanFord*$(\langle \mathcal{T}_s, \mathcal{C}_s \rangle)$ *detects a negative cycle* **then**
12 $\varphi \leftarrow \varphi \wedge$ `CUT-SCENARIO`$(\mathcal{S}, \langle \mathcal{T}_s, \mathcal{C}_s \rangle)$ ▷ Add clause
13 **on** *solution found:* ▷ Complete model
14 Build a scenario s from the current truth value assignment
15 $\langle \mathcal{T}_s, \mathcal{C}_s \rangle \leftarrow \pi_s(\mathcal{S})$ ▷ Get the STN projection
16 **if** *BellmanFord*$(\langle \mathcal{T}_s, \mathcal{C}_s \rangle)$ *does not detect any negative cycle* **then**
17 $Sols \leftarrow Sols \cup \{\langle s, S \rangle\}$ ▷ S is an early schedule for $\langle \mathcal{T}_s, \mathcal{C}_s \rangle$
18 **if** *BellmanFord*$(\langle \mathcal{T}_s, \mathcal{C}_s \rangle)$ *detects negative cycle or all is true* **then**
19 $\varphi \leftarrow \varphi \wedge$ `CUT-SCENARIO`$(\mathcal{S}, \langle \mathcal{T}_s, \mathcal{C}_s \rangle)$ ▷ Add clause

contains a negative cycle. If so, we extend φ by adding (on the fly) a CNF clause modeling the negation of the part of s containing a negative cycle in order to avoid getting the same scenario again. If the projection is consistent, `STND-HSCC2` does nothing and lets the run go.

A *solution found* event (Algorithm 3, lines 13–19) extends the behavior of the listener described for *assume* as follows. When triggered, the listener builds a scenario from the current truth value assignment (which does not need to be extended with anything else this time). Then, it checks if the corresponding STN-projection contains a negative cycle. If it does not, then it computes an (early) schedule S for the STN projected onto s and adds the pair $\langle s, S \rangle$ to the set of solutions. If it detects a negative cycle (or all consistent scenarios are sought), then it acts as for *assume* events.

Eventually, when the run of the SAT solver ends, either $Sols = \emptyset$ (and thus the starting STND is inconsistent), or *Sols* contains at least 1 solution (scenario-schedule).

Like `STND-HSCC1`, `STND-HSCC2` is sound and complete because it is based on a SAT-solver that allows us to iterate on all the models. Whenever we add a clause, we exclude a relevant part of a scenario that we do not want to get anymore. The sooner, the better.

Besides the SAT solver, all other internal sub-procedures (mostly, algorithms for directed weighted graphs) are well known to be sound and complete, and run in polynomial time.

4 Experimental Evaluation

We developed Kappa, a tool for STNDs that takes in input a specification of an STND and acts both as a solver and as a solution verifier. Kappa relies on SAT4J [3], a Java library compliant with the IPASIR interface that specifies how to interact with a SAT solver [1].

Kappa implements both `STND-HSCC1` and `STND-HSCC2`. We extended `STND-HSCC1` so that it allows for the synthesis of all consistent scenarios as well. In this way, we could carry out a more accurate experimental evaluation comparing the two algorithms when seeking single or all consistent scenarios.

We randomly generated 2200 STNDs partitioned in benchmark 11 sets, each one containing 100 consistent STNDs and 100 inconsistent STNDs. Regardless of the set, each STND has exactly 100 time points. The first set (`100TimePoints/10Decisions`) specifies 10 decision time points, the second set (`100TimePoints/11Decisions`) specifies 11 decision time points and so on, up to the eleventh one (`100TimePoints/20Decisions`) that specifies 20 decision time points. Each STND has a maximum number of constraints of $|\mathcal{T}| \times |\mathcal{DT}|$. Time points and constraints are randomly labeled so that the resulting STND is well defined. The weights on labeled edges range from -100 to 100. See the link "Supplement Material" before Section 1 to get Kappa and these benchmark sets.

We ran Kappa on these benchmark sets without imposing any limit to collect data (time and space) for both `STND-HSCC1` and `STND-HSCC2` when seeking a single or all consistent scenarios.

We graphically show the results in Figure 2, where x-axes always represent the number (#) of decision time points (i.e., the set under analysis) and y-axes represent either the average time elapsed or space consumed when analyzing the instances in that set.

Figure 2a shows the results of the analysis run on the sets containing consistent STNDs when seeking a single consistent scenario. The graph shows that `STND-HSCC2` is significantly faster than `STND-HSCC1` for STNDs specifying more than 16 decision time points. Figure 2b shows the results of the same analysis when seeking all consistent scenarios: despite a normal general worsening of performances (all consistent scenarios are sought and not just one) `STND-HSCC2` is faster than `STND-HSCC1` for STNDs specifying 20 decisions. Figure 2c shows the results of the analysis on the sets containing inconsistent STNDs. `STND-HSCC2` has no competitors here, whereas `STND-HSCC1` starts having a serious exponential blow up for STNDs specifying more than 14 decisions. Figure 2d shows the average space consumed when synthesizing all consistent scenarios. The curve grows exponentially according to the number of decision time points (recall that `STND-HSCC1` and `STND-HSCC2` return the same set of consistent scenarios in such an analysis, therefore we only show the data for `STND-HSCC2`).

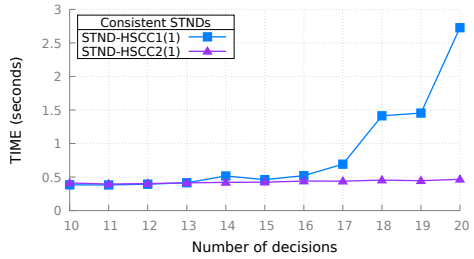
We verified all synthesized solutions. No constraint was violated.

5 Equivalence with Disjunctive Temporal Networks

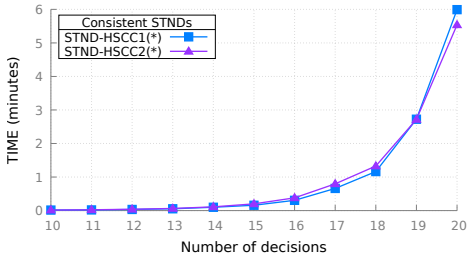
Disjunctive temporal networks (*DTNs*, [25]) allow for disjunctions of temporal constraints (i.e., alternatives) in a temporal problem and are a possible formalism to model the *disjunctive temporal problem* (*DTP*). For example, we might want that once an event modeled by a time point X happened, another event modeled by a time point Y happens *either* after 10 (seconds, minutes, hours, ...) *or* within 5. Such a constraint would look like

$$(X - Y \leq -10) \vee (Y - X \leq 5)$$

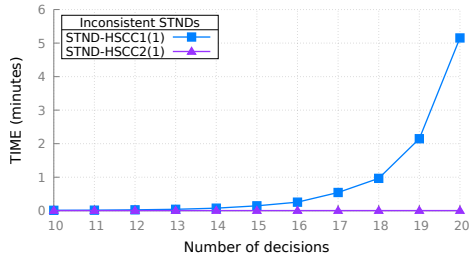
Any assignment of real values to X and Y satisfies the constraint if it satisfies (at least) one disjunct.



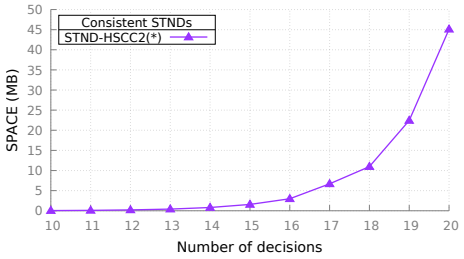
(a) Consistency checking time on consistent STNDs when seeking a single consistent scenario.



(b) Consistency checking time on consistent STNDs when seeking all consistent scenarios.



(c) Consistency checking time on inconsistent STNDs.



(d) Space consumed when synthesizing all consistent scenarios.

Figure 2 Experimental evaluation with KAPPA.

Differently from the initial proposal in [11], where disjunctions of intervals were allowed on the *same pairs of time points* only, the work we consider here is the one in [25], not having such a restriction.

▶ **Definition 4.** *A* disjunctive temporal network *(*DTN*) is a pair* $\langle \mathcal{T}, \mathcal{C} \rangle$*, where*

- $\mathcal{T}$ *is the usual finite set of time points, and*
- $\mathcal{C}$ *is a finite set of temporal constraints each one having the form*

$$\underbrace{(Y_1 - X_1 \leq k_1) \vee \cdots \vee (Y_n - X_n \leq k_n)}_{n \text{ disjuncts (atoms)}}$$

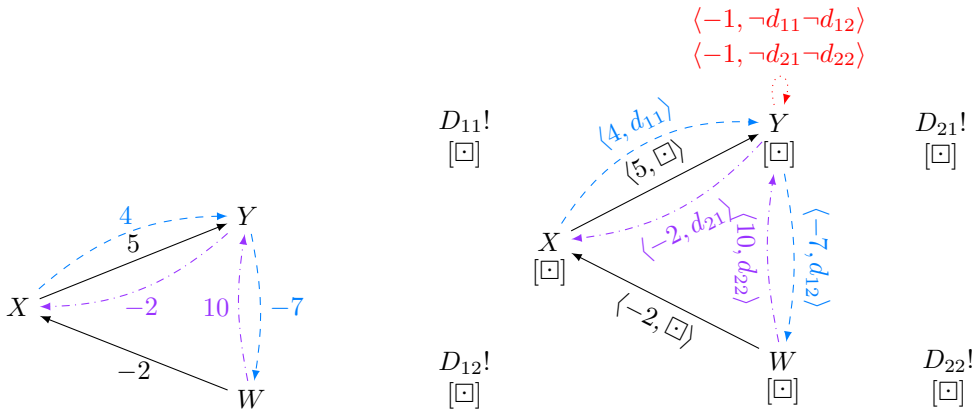
where $X_i, Y_i \in \mathcal{T}$ *and* $k_i \in \mathbb{R}$*. A temporal constraint is* non-disjunctive *if and only if it contains one disjunct,* disjunctive *otherwise.*

A DTN is consistent *if there exists an assignment of real values to all time points (i) always satisfying all non-disjunctive constraints and (ii) satisfying at least one disjunct for each disjunctive constraint.*

We write $D(i)$ to shorten the i^{th} disjunctive constraint and more specifically $D(i, j)$ to refer to the j^{th} disjunct of the i^{th} disjunctive constraint [25].

We represent a DTN graphically through a *colored multi graph*, where *black edges* model non-disjunctive constraints (i.e., those constraints that must always hold), whereas *colored edges* (different from black) model disjunctive constraints (i.e., those $D(i)$s for which at least one disjunct must hold). Each disjunctive constraint is assigned to a different color (we also use a unique line pattern for each color).

To give an example, consider the following DTN, whose corresponding colored multi graph is shown in Figure 3a.



(a) Example of DTN. **(b)** Corresponding STND.

Figure 3 Representing and encoding DTNs into STNDs.

- $\mathcal{T} = \{X, Y, W\}$
- $\mathcal{C} = \{\overbrace{(Y - X \leq 5), (X - W \leq -2)}^{\text{must always hold}}, \overbrace{\overbrace{(Y - X \leq 4)}^{D(1,1)} \vee \overbrace{(W - Y \leq -7)}^{D(1,2)}}^{D(1)},$
  $\underbrace{\underbrace{(X - Y \leq -2)}_{D(2,1)} \vee \underbrace{(Y - W \leq 10)}_{D(2,2)}}_{D(2)}\}$

The DTN in Figure 3a is consistent, and, for example, the assignment $X = 0$, $Y = 3$ and $W = 5$ satisfies:

- all non-disjunctive constraints (solid black edges),
- $D(1,1)$ but not $D(1,2)$ for the disjunctive constraints $D(1)$ (dashed blue edges),
- $D(2,1)$ and also $D(2,2)$ for the disjunctive constraint $D(2)$ (dashdotted purple edges).

We now proceed by proving that STNDs and DTNs are equivalent. We first give a strongly polynomial time encoding from DTNs to STNDs and then the vice versa (and we provide examples throughout this discussion).

5.1 Encoding DTNs into STNDs

We encode the DTN in Figure 3a into the corresponding STND in Figure 3b as follows.

We generate a "core" STND containing all time points and all non-disjunctive constraints of the starting DTN and labeling them by $\boxdot$, since all time points must always be assigned a value and all non-disjunctive constraints must always be satisfied.

For each disjunctive constraint $D(i)$ in the DTN, we add to the STND as many decision time points $D_{ij}!$ as the number of disjuncts $D(i,j)$. These decision time point are not constrained to any other time point in the STND (i.e., free to take any value). Any disjunct $D(i,j)$ in the DTN appears as a constraint in the STND labeled by d_{ij} (the proposition associated to $D_{ij}!$) so that when $d_{ij} = \top$, the disjunct of the DTN (labeled constraint in the STND) must hold and when $d_{ij} = \bot$ we are not obliged to satisfy it. Moreover, we impose that at least one disjunct $D(i,j)$ for any disjunctive constraint $D(i)$ must hold (otherwise, it would be possible to disable them all by setting all d_{ij} to $\bot$). We enforce this condition by adding a negative self loop labeled by $\neg d_{ij_1} \dots \neg d_{ij_n}$ on any time point of the STND.

In Figure 3b we added four decision time points $D_{11}!$, $D_{12}!$, $D_{21}!$ and $D_{22}!$ and the labeled constraints $X \to Y$ labeled by $\langle 4, d_{11}\rangle$, $Y \to W$ labeled by $\langle -7, d_{12}\rangle$, $Y \to X$ labeled by $\langle -2, d_{21}\rangle$ and $W \to Y$ labeled by $\langle 10, d_{22}\rangle$ to switch on and off the disjuncts $D(1,1)$, $D(1,2)$, $D(2,1)$ and $D(2,2)$ through the truth value assignments to d_{11}, d_{12}, d_{21} and d_{22}. Finally, we added two negative self loops $Y \to Y$ labeled by $\langle -1, \neg d_{11}\neg d_{12}\rangle$ and $\langle -1, \neg d_{21}\neg d_{22}\rangle$ to prevent a disjunctive constraint $D(i)$ from being excluded (red self loop at Y). Note that the "-1" is meaningless: *any negative number* (e.g., -3, -159 or $-\epsilon$) is fine for this purpose. Likewise, the choice of time point Y is meaningless too. Any time point would be fine for this purpose (e.g., $X \to X$ labeled by the same constraints). Negative self loops are the more intuitive way to enforce these conditions. However, nothing would have prevented us from creating cycles of negative sum with respect to these labels involving many time points.

To ease reading and understand "what goes where", we colored the STND in Figure 3b with the same colors of the DTN in Figure 3a and showed the added negative cycles in red.

This encoding is strongly polynomial. The number of time points in the STND is equal to the number of time points in the DTN plus as many decision time points as the number of disjuncts $D(i,j)$ contained in all disjunctive constraints $D(i)$ in the DTN. The number of constraints in the STND is equal to the the number of non-disjunctive constraints plus as many constraints as the number of disjuncts $D(i,j)$ contained in all disjunctive constraints $D(i)$ in the DTN plus as many constraints as the number of disjunctive constraints $D(i)$ to model negative loops.

Any consistent scenario in the STND says which disjuncts (at least one for each disjunctive constraint) are satisfied for the solution. If the STND is inconsistent, then so is the DTN.

5.2 Encoding STNDs into DTNs

We encode the STND in Figure 4a into the corresponding STND in Figure 4b as follows.

First of all, if the STND has labels on nodes we convert it to its streamlined version having only label on edges. The process of streamlining a temporal network was first discussed in [5] for CSTNs. However, that process works for STNDs as well (as consistency is basically entailed by controllability). Then, we generate a "core" DTN having the same set of time points of the STND (we drop all "!" from the names) and all constraints labeled by $\boxdot$ in the STND as non-disjunctive constraints in the DTN.

For each proposition d associated to a decision time point $D!$ in the STND, we add to the DTN a time point d and the disjunctive constraint

$$\underbrace{(d - D \leq 0)}_{\text{means } d = \bot} \vee \underbrace{(D - d \leq -1)}_{\text{means } d = \top}$$

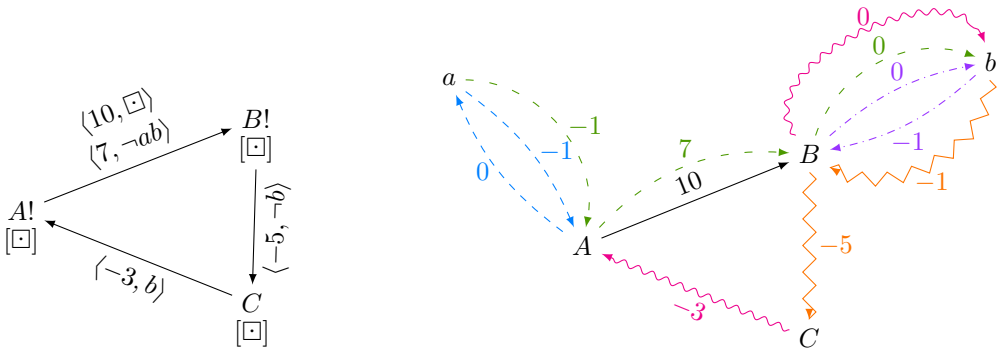
where the former says that d "occurs within" D, whereas the latter says that d occurs at least 1 after D (a way to to simulate a Boolean condition).

Now, every constraint $X \to Y$ labeled by $\langle k, d\neg e f \dots\rangle$ (in the STND) implies the following "meta constraint" in the DTN:

$$(\underbrace{D - d \leq -1}_{d} \wedge \underbrace{e - E \leq 0}_{\neg e} \wedge \underbrace{F - f \leq -1}_{f} \dots) \Rightarrow Y - X \leq k$$

which can be rewritten as

$$\neg(D - d \leq -1 \wedge e - E \leq 0 \wedge F - f \leq -1 \dots) \vee Y - X \leq k$$



(a) Example of (streamlined) STND. **(b)** Corresponding DTN.

Figure 4 Encoding STNDs into DTNs.

and finally simplified to

$$\underbrace{(d - D \leq 0)}_{\neg d} \vee \underbrace{(E - e \leq -1)}_{e} \vee \underbrace{(f - F \leq 0)}_{\neg f} \cdots \vee (Y - X \leq k)$$

Note that for any D and d (in the DTN) we define $\neg(D - d \leq -1)$ as $d - D \leq 0$ and $\neg(d - D \leq 0)$ as $D - d \leq -1$ since they are abstracting Boolean conditions only and we are not therefore interested in a specific numeric value. Therefore, for any labeled constraint in the STND we add such a disjunctive constraint to the DTN.

In Figure 4b we add two time points a and b and the following constraints:

- $B - A \leq 10$ (solid black edges),
- $D(1)\colon (a - A \leq 0) \vee (A - a \leq -1)$ (dashed blue edges),
- $D(2)\colon (b - B \leq 0) \vee (B - b \leq -1)$ (dashdotted purple edges),
- $D(3)\colon (A - a \leq -1) \vee (b - B \leq 0) \vee (B - A \leq 7)$ (loosely dashed green edges),
- $D(4)\colon (B - b \leq -1) \vee (C - B \leq -5)$ (zigzag orange edges),
- $D(5)\colon (b - B \leq 0) \vee (A - C \leq -3)$ (snake magenta edges).

We show the "colored" DTN graph in Figure 4b. Now, the DTN is consistent if and only if the STND is so. A solution of the DTN corresponds to a consistent scenario in the STND. The truth value assignment to the propositions in the STND depends on the real value assignments to the time points modeling those propositions in the DTN. For any proposition d in the STND, d is false iff in the DTN the time point d has a value not greater than $D!$ and d is true in the STND iff in the DTN the value of time point d is greater than $D!$ (the assignment to the other time points defines a schedule consistent for the scenario).

This encoding is strongly polynomial. The number of time points in the DTN is the same of that in the STND plus as many time points as the number of propositions in the STND. The number of constraints in the DTN is given by the number of unlabeled constraints in the STND (non-disjunctive constraints in the DTN), plus as many disjunctive constraints as the number of labeled constraints in the STND (whose labels are different from ⊡). Also, for any disjunctive constraint $D(i)$ in the DTN, the number of disjuncts of $D(i)$ is $n+1$ where n is the number of literals contained in the label of the corresponding constraint in the STND and the "+1" refers to the inequality.

6 Related Work

STNs [11] model fully-controllable plans but do not employ decisions. *Drake* [9] is an executive for temporal plans with choices modeled as Labeled STNs that extend STNs by labeling constraints with environments (set of instantiated discrete variables). There are no decision points and time points are unlabeled. During execution, choices are discriminated by generating conflicts according to the timing Drake decides to schedule some event. STNDs differ from Drake in their specification and in how decisions are made. STNDs rely on a more structured approach by using labels instead of environments, labeling time points to prevent them from being executed when some literal in the label is still unknown, enforcing well-defined properties and making decisions only upon the execution of the related decision time points. A *Disjunctive Temporal Network (DTN)* [25] extends an STN with disjunctive constraints. Any solution to a DTN must satisfy all non-disjunctive constraints (i.e., STN-constraints) and at least one disjunct for each disjunctive constraint. Labeled STNs and DTNs are equivalent [9]. DTNs and STNDs are equivalent too, therefore, Labeled STNs are equivalent to STNDs as well.

Temporal Plan Networks (TPNs) [21] extend STNs by adding decision nodes and symbolic constraints to model temporal plans with controllable choices modeled as outgoing edges from a decision node. Taking one of these outgoing edges means making a particular decision. Time points are not labeled and activities are modeled as pair of non-decision nodes (start,end). A symbolic constraint is either Ask(c) (is c true?) and Tell(c) (c is true!) where c a literal. Symbolic constraints may exclude activities from being executed. A plan is consistent if it satisfies both temporal and symbolic constraints. TPNs do not specify more than one temporal constraint on the same edge. Consistency is checked by visiting the nodes of the graph from start to end taking one edge (modeling a decision) at a time. If the resulting STN-projection is inconsistent, the algorithm backtracks to the last decision node that still has unexplored outgoing edges. STNDs label nodes and consistency is checked in a hybrid way.

A *Controllable Conditional Temporal Problem (CCTP)* [28] is an optimization problem for temporal plans with choices and thus incomparable with STNDs.

Pike [22] is an executive for temporal plans with both controllable and uncontrollable choices modeled as *Temporal Plan Networks with Uncertainty (TPNUs)*, which extend TPNs with uncontrollable choices. Pike adapts its controllable choices to the uncontrollable ones made by a human. STNDs do not have uncontrollable choices. *CCTPs with Uncertainty (CCTPUs)* [27] address temporal plans with controllable choices and uncontrollable durations, whereas in [18], TPNUs are extended to support uncontrollable durations (strong controllability only). In both works, relaxation techniques are used to restore controllability of an uncontrollable plan. STNDs do not have uncontrollable parts.

Several extensions of *STNs* address uncertain domains. For example, *Simple Temporal Networks with Uncertainty* (*STNUs*, [24]) add uncontrollable (but bounded) durations by means of a finite set of contingent links, whereas *Conditional Simple Temporal Networks* (*CSTNs*, [17]), and the Conditional Temporal Problem (*CTP*, [26]) considered formerly, extend STNs by turning the constraints conditional with uncontrollable truth value assignments observable upon the execution of some special kind of time points called observations. Finally, *Conditional Simple Temporal Networks with Uncertainty* (*CSTNUs*, [16, 15]) merge STNUs and CSTNs, whereas *Conditional Simple Temporal Networks with Uncertainty and Decisions* (*CSTNUDs*, [29, 33]) add conditional constraints with controllable truth value assignments decidable upon the execution of some special kind of time points called decision. CSTNUDs

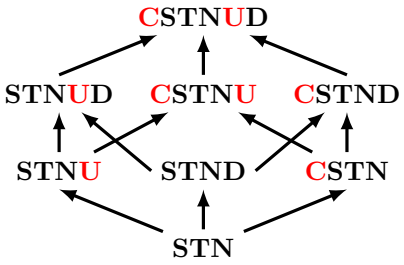


Figure 5 A hierarchy of simple temporal networks. Acronyms containing **D** (resp., **C**) mean that formalisms deal with controllable (resp., uncontrollable) conditionals, whereas those containing **U**, mean that formalisms deal with uncontrollable durations. We highlight uncontrollable parts in red.

encompass all previous formalisms. All these networks extend STNs by adding at least an uncontrollable part. In this work, we do not address any uncontrollable part. STNDs derive from [4] by removing uncontrollable conditionals and from [29, 33] by removing uncontrollable conditionals and uncontrollable durations. This work extends [4] by providing `STND-HSCC2` both to speed up the consistency checking phase and to allow for the synthesis of all consistent scenarios. This work is, however, incomparable with [29, 33] as that work employs timed game automata. Figure 5 provides a hierarchy of simple temporal networks.

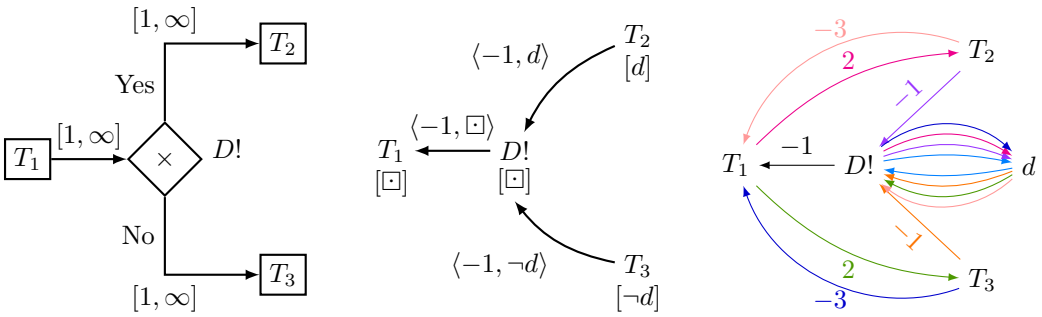
There also have been attempts to consider time and resources together, e.g., *Access Controlled Temporal Networks* (*ACTNs*, [6]) and *Conditional Simple Temporal Networks with Uncertainty and Resources* (*CSTNURs*, [7]), which were preceded by an initial proposal in [8]. However, neither ACTNs nor CSTNURs employ decision time points. Research on temporal networks has inspired a recent line of work in which controllability analysis focused on resource allocation under uncertainty employing a qualitative temporal approach instead of a quantitative one. This is the case of access controlled workflows investigated in [31, 36] and of extensions of constraint networks proposed in [34, 35], where *Constraint Networks Under Conditional Uncertainty* (*CNCUs*) are introduced (see also [32]). As we said, temporal relations are only qualitative (specifically, "before/after") and these proposals do not employ decision time points. A short summary of temporal and resource controllability based on constraint networks and considered either in isolation or simultaneously can be found in [30].

Planning as satisfiability was formally introduced in [19, 20] and relies on a set of axioms where any model corresponds to a valid plan. Before that, planning was based on deduction. Recently, more performant SAT encodings have been provided (e.g., [14]). However, none of these approaches is incremental and thus they are incomparable with ours.

Gocht and Balyo [13] provide an incremental SAT solving approach for SAT-based planning and prove that incremental SAT solving outperforms the non-incremental one but they do not address temporal constraints. Our work does not model "transitions" but applies shortest path algorithms, incrementally, on STN-projections.

Temporal induction is an incremental technique to check safety properties on finite state machines and it is strongly related to bounded model checking [12]. It is similar to SAT-based planning and allows for the detection of the unreachability of a goal. Our analysis is not bounded with respect to the "depth".

Satisfiability modulo theory (*SMT*, [2]) can describe STNDs by using a fragment of Linear Real Arithmetic called Difference Logic. However, SMT-solvers do not guarantee to find early schedules. A run of an HSCC algorithm and a run of an SMT-solver are not guaranteed to return the same consistent scenarios (Boolean part). A fairer comparison is when both HSCC-algorithms and SMT-solvers seek all consistent scenarios, but then we should make sure that the SMT-solver does not return more than one schedule for each consistent scenario.



(a) Temporal Workflow with controllable conditional paths.

(b) Simple Temporal Network with Decisions.

(c) DTN. Edges $D! \to d$ are labeled by 0, $d \to D!$ by -1.

Figure 6 Possible encodings of a temporal workflow (a) into STNDs (b) and into DTNs (c). Note that (c) is the DTN equivalent to (b) streamlined. That is, (b) without labels on time points and plus the constraints $(T_2 - T_1 \leq 2, d)$, $(T_1 - T_2 \leq -3, \neg d)$, $(T_3 - T_1 \leq 2, \neg d)$ and $(T_1 - T_3 \leq -3, d)$. There, we chose 2 as an horizon (see [5] for more details on how to streamline a temporal network).

Incremental task planning adopts incremental features of SMT-solvers to extend a constraint-based task planning to motion domains [10]. Our approach is not probabilistic and does not consider a motion domain.

7 Conclusions and Future Work

We provided `STND-HSCC2`, a novel hybrid SAT-based consistency checking algorithm for STNDs. This new version of the algorithm still relies on a SAT solver but differently from the previous one, it exploits partial truth value assignments to hunt down negative cycles in STN-projections as early as possible. The previous algorithm tested STN-projections for negative cycles by iterating on *complete* models returned by the SAT solver. When the SAT solver makes an assumption, we project the STND over the current truth value assignment of the propositions (i.e., partial model) extended with this new assumption. If the projected STN is inconsistent, we add a clause to the SAT solver to exclude that scenario, else we let the solver go. We implemented our approach and provided Kappa, a tool for STNDs that implements `STND-HSCC1` and `STND-HSCC2` both supporting the synthesis of single or all consistent scenarios[2] and we compared the results. The more inconsistent STN-projections an STND admits, the better `STND-HSCC2` performs. The solutions saved to file allow for an offline planning in which all decisions are made before starting and all time points have already been scheduled to execute as soon as possible. Finally, we proved that STNDs and DTNs are equivalent. Considering this equivalence result, one could fairly wonder why use STNDs instead of DTNs. Here is a possible reason: STNDs offer a more "structured" language, which, exploiting labels, allows for an easier modeling of temporal workflows with controllable conditional paths (see Figure 6 for a comparison of workflow modeling methods).

As future work, we plan to investigate optimizations to reduce the size of the CNF clauses added on the fly. We also plan to give a metric suggesting the best algorithm to use depending on the form of the STND in input.

[2] Another minor contribution is the extension of `STND-HSCC1` in our tool to support the synthesis of all consistent scenarios.

References

1 Tomas Balyo, Armin Biere, Markus Iser, and Carsten Sinz. SAT race 2015. *Artificial Intelligence*, 241:45–65, 2016.

2 Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, pages 825–885. IOS Press, 2009.

3 Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *JSAT*, 7(2-3):59–6, 2010. URL: `http://jsat.ewi.tudelft.nl/content/volume7/JSAT7_4_LeBerre.pdf`.

4 Massimo Cairo, Carlo Combi, Carlo Comin, Luke Hunsberger, Roberto Posenato, Romeo Rizzi, and Matteo Zavatteri. Incorporating Decision Nodes into Conditional Simple Temporal Networks. In *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*, volume 90 of *LIPIcs*, pages 9:1–9:17. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. `doi:10.4230/LIPIcs.TIME.2017.9`.

5 Massimo Cairo, Luke Hunsberger, Roberto Posenato, and Romeo Rizzi. A Streamlined Model of Conditional Simple Temporal Networks - Semantics and Equivalence Results. In *24th International Symposium on Temporal Representation and Reasoning, TIME 2017*, volume 90 of *LIPIcs*, pages 10:1–10:19. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. `doi:10.4230/LIPIcs.TIME.2017.10`.

6 Carlo Combi, Roberto Posenato, Luca Viganò, and Matteo Zavatteri. Access Controlled Temporal Networks. In *9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 118–131. INSTICC, ScitePress, 2017. `doi:10.5220/0006185701180131`.

7 Carlo Combi, Roberto Posenato, Luca Viganò, and Matteo Zavatteri. Conditional Simple Temporal Networks with Uncertainty and Resources. *Journal of Artificial Intelligence Research*, 64:931–985, 2019. `doi:10.1613/jair.1.11453`.

8 Carlo Combi, Luca Viganò, and Matteo Zavatteri. Security Constraints in Temporal Role-Based Access-Controlled Workflows. In *6th ACM Conference on Data and Application Security and Privacy*, CODASPY '16, pages 207–218. ACM, 2016. `doi:10.1145/2857705.2857716`.

9 Patrick R. Conrad and Brian C. Williams. Drake: An Efficient Executive for Temporal Plans with Choice. *Journal of Artificial Intelligence Research*, 42(1):607–659, 2011.

10 Neil T. Dantam, Zachary K. Kingston, Swarat Chaudhuri, and Lydia E. Kavraki. Incremental Task and Motion Planning: A Constraint-Based Approach. In *Robotics: Science and Systems XII*, 2016. URL: `http://www.roboticsproceedings.org/rss12/`.

11 Rina Dechter, Itay Meiri, and Judea Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.

12 Niklas Eén and Niklas Sörensson. Temporal induction by incremental SAT solving. *Electronic Notes in Theoretical Computer Science*, 89(4):543–560, 2003.

13 Stephan Gocht and Tomas Balyo. Accelerating SAT Based Planning with Incremental SAT Solving. In *ICAPS 2017*, pages 135–139. AAAI Press, 2017.

14 Ruoyun Huang, Yixin Chen, and Weixiong Zhang. A Novel Transition Based Encoding Scheme for Planning as Satisfiability. In *AAAI 2010*. AAAI Press, 2010.

15 Luke Hunsberger and Roberto Posenato. Sound-and-Complete Algorithms for Checking the Dynamic Controllability of Conditional Simple Temporal Networks with Uncertainty. In *25th International Symposium on Temporal Representation and Reasoning (TIME 2018)*, volume 120 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. `doi:10.4230/LIPIcs.TIME.2018.14`.

16 Luke Hunsberger, Roberto Posenato, and Carlo Combi. The Dynamic Controllability of Conditional STNs with Uncertainty. In *Workshop on Planning and Plan Execution for Real-World Systems (PlanEx) at ICAPS-2012*, pages 1–8, 2012. `arXiv:1212.2005`.

17 Luke Hunsberger, Roberto Posenato, and Carlo Combi. A Sound-and-Complete Propagation-based Algorithm for Checking the Dynamic Consistency of Conditional Simple Temporal Networks. In *22nd International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 4–18. IEEE, 2015. `doi:10.1109/TIME.2015.26`.

18 Erez Karpas, Steven James Levine, Peng Yu, and Brian Charles Williams. Robust Execution of Plans for Human-Robot Teams. In *ICAPS 2015*, pages 342–346. AAAI Press, 2015.

19 Henry A. Kautz, David A. McAllester, and Bart Selman. Encoding Plans in Propositional Logic. In *KR '96*, pages 374–384. Morgan Kaufmann, 1996.

20 Henry A. Kautz and Bart Selman. Planning As Satisfiability. In *ECAI '92*, pages 359–363. John Wiley & Sons, Inc., 1992.

21 Phil Kim, Brian C. Williams, and Mark Abramson. Executing Reactive, Model-based Programs through Graph-based Temporal Planning. In *IJCAI 2001*, pages 487–493. Morgan Kaufmann, 2001.

22 Steven James Levine and Brian Charles Williams. Concurrent Plan Recognition and Execution for Human-Robot Teams. In *ICAPS 2014*. AAAI Press, 2014.

23 Hui Li and Brian Williams. Generalized Conflict Learning for Hybrid Discrete/Linear Optimization. In *CP 2005*, pages 415–429. Springer, 2005.

24 Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic Control Of Plans With Temporal Uncertainty. In *17th International Joint Conference on Artificial Intelligence, IJCAI 2001*, pages 494–502. Morgan Kaufmann, 2001.

25 Kostas Stergiou and Manolis Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120(1):81–117, 2000.

26 Ioannis Tsamardinos, Thierry Vidal, and Martha E. Pollack. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints*, 8(4):365–388, 2003.

27 Peng Yu, Cheng Fang, and Brian Charles Williams. Resolving Uncontrollable Conditional Temporal Problems Using Continuous Relaxations. In *ICAPS 2014*. AAAI Press, 2014.

28 Peng Yu and Brian Charles Williams. Continuously Relaxing Over-Constrained Conditional Temporal Problems through Generalized Conflict Learning and Resolution. In *IJCAI 2013*, pages 2429–2436. IJCAI/AAAI, 2013.

29 Matteo Zavatteri. Conditional Simple Temporal Networks with Uncertainty and Decisions. In *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*, volume 90 of *LIPIcs*, pages 23:1–23:17. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.TIME.2017.23.

30 Matteo Zavatteri. Temporal and Resource Controllability of Workflows Under Uncertainty. In *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2019*, volume 2420, pages 9–14. CEUR-WS.org, 2019. URL: http://ceur-ws.org/Vol-2420/paperDA3.pdf.

31 Matteo Zavatteri, Carlo Combi, Roberto Posenato, and Luca Viganò. Weak, Strong and Dynamic Controllability of Access-Controlled Workflows Under Conditional Uncertainty. In *Business Process Management - 15th International Conference, BPM 2017*, pages 235–251. Springer, 2017. doi:10.1007/978-3-319-65000-5_14.

32 Matteo Zavatteri, Carlo Combi, and Luca Viganò. Resource Controllability of Workflows Under Conditional Uncertainty. In *Business Process Management Workshops (AI4BPM 2019) (to appear)*. Springer International Publishing, 2019.

33 Matteo Zavatteri and Luca Viganò. Conditional simple temporal networks with uncertainty and decisions. *Theoretical Computer Science*, (In press):, 2018. doi:10.1016/j.tcs.2018.09.023.

34 Matteo Zavatteri and Luca Viganò. Constraint Networks Under Conditional Uncertainty. In *10th International Conference on Agents and Artificial Intelligence – Volume 2 (ICAART 2018)*, pages 41–52. SciTePress, 2018. doi:10.5220/0006553400410052.

35 Matteo Zavatteri and Luca Viganò. Conditional Uncertainty in Constraint Networks. In *Agents and Artificial Intelligence*, pages 130–160. Springer, 2019. doi:10.1007/978-3-030-05453-3_7.

36 Matteo Zavatteri and Luca Viganò. Last Man Standing: Static, Decremental and Dynamic Resiliency via Controller Synthesis. *Journal of Computer Security*, 27(3):343–373, 2019. doi:10.3233/JCS-181244.

Synthesis of LTL Formulas from Natural Language Texts: State of the Art and Research Directions

Andrea Brunello[1]
University of Udine, Italy
andrea.brunello@uniud.it

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

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

Abstract

Linear temporal logic (LTL) is commonly used in model checking tasks; moreover, it is well-suited for the formalization of technical requirements. However, the correct specification and interpretation of temporal logic formulas require a strong mathematical background and can hardly be done by domain experts, who, instead, tend to rely on a natural language description of the intended system behaviour. In such situations, a system that is able to automatically translate English sentences into LTL formulas, and vice versa, would be of great help. While the task of rendering an LTL formula into a more readable English sentence may be carried out in a relatively easy way by properly parsing the formula, the converse is still an open problem, due to the inherent difficulty of interpreting free, natural language texts. Although several partial solutions have been proposed in the past, the literature still lacks a critical assessment of the work done. We address such a shortcoming, presenting the current state of the art for what concerns the English-to-LTL translation problem, and outlining some possible research directions.

2012 ACM Subject Classification General and reference → Surveys and overviews; Computing methodologies → Natural language processing; Computing methodologies → Machine learning; Computing methodologies → Temporal reasoning; Theory of computation → Evolutionary algorithms

Keywords and phrases Evolutionary algorithms, Machine learning, Natural language processing, Semantic parsing, Temporal logic

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.17

1 Introduction

Linear temporal logic (LTL) is a formalism which is widely used in model checking (see, for instance, [40, 45]); moreover, LTL formulas make it possible to unambiguously describe the relationships among occurrences of events over time, a capability which turns out to be quite important in many automated reasoning fields. For these reasons, LTL can be considered as a preferred means to formalize and then reason upon natural language texts, which is most useful in tasks such as requirement specification and, more generally, the analysis of temporally-declined semantic content in texts [90].

Despite of their usefulness, the correct specification and interpretation of temporal logic formulas typically requires a strong mathematical background, and this severely limits their applicability by untrained domain experts. In such contexts, a system capable of automatically translating between English and temporal logic would be of great help.

[1] Corresponding author

Concerning the LTL-to-English translation, it may be achieved in a relatively easy way, by simply parsing the logical formula by means of an attribute grammar and applying some heuristics to make the English translation sound as natural as possible [81]. As for the opposite direction, several issues have to be dealt with, such as the inherent ambiguity of natural language, the possible lack of an explicit context in the sentences, or reference to background knowledge which may not be included in the utterance. For all these reasons, the task of translating a free, unconstrained English text into a free, unbounded LTL formula is still an open problem, although several partial solutions have been proposed over the years (see Section 3).

In this paper, we give an overall assessment of the state of the art for what concerns English-to-LTL translation. In addition, we make a critical evaluation of some of the currently available tools which may be adapted and combined for such a task, and we outline some possible future research directions.

The paper is organized as follows. In Section 2, we introduce the problem of English-to-LTL translation. Then, in Section 3, we provide a short state of the art. Next, in Section 4, we outline possible future research directions. Finally, in Section 5, the outcomes of an experimental evaluation of some of the most significant tools proposed in the literature that may be used to develop and English-to-LTL translation architecture are reported. Conclusions summarize the main contributions of the work done.

2 Problem definition

The problem of extracting temporal logic formulas from a natural language utterance may be considered as an instance of the *Semantic Parsing* task, that is, the process of mapping a natural-language sentence into a formal, typically machine-understandable representation of its meaning [95].

It must be observed that such a problem differs from the one of inferring logical formulas from examples. In the latter, a dataset of logs, or traces, resulting from the execution of a given system is considered, with the goal of determining which formulas characterize and distinguish between examples that describe a good behaviour of the system from those that do not (see, for instance, [70]).

In the present work, we focus on *Linear-Time Temporal Logic* (LTL for short) [79]. An LTL formula is built from a finite set of proposition letters by making use of Boolean connectives and the temporal modalities **X** (next) and **U** (until). Formally, LTL formulas can be defined as follows:

- if $p \in \mathcal{AP}$, then p is an LTL formula;
- if ψ and ϕ are LTL formulas, then $\neg\psi$, $\phi \vee \psi$, $\mathbf{X}\psi$, and $\phi\mathbf{U}\psi$ are LTL formulas.

On the basis of these temporal operators, additional modalities can be defined, most commonly the Boolean connectives $\wedge$ (and), $\rightarrow$ (imply), **true**, and **false**, and the temporal modalities **G** (globally) and **F** (eventually).

As for the semantics, let $w = a_0, a_1, a_2, \ldots$ $(a_i \in 2^{\mathcal{AP}})$ be an infinite sequence of truth evaluations for proposition letters in $\mathcal{AP}$, and let $w(i)$ denote the truth evaluation at position i. The satisfaction relation $\models$ between w and an LTL formula can be defined as follows:

- $w \models p$ if $p \in w(0)$;
- $w \models \neg\psi$ if $w \not\models \psi$;
- $w \models \phi \vee \psi$ if $w \models \phi$ or $w \models \psi$;
- $w \models \mathbf{X}\psi$ if $w(1) \models \psi$;
- $w \models \phi\mathbf{U}\psi$ if $\exists\, i \geq 0$ s.t. $w(i) \models \psi$ and $\forall\, 0 \leq k < i$ $w(k) \models \phi$.

The semantics of the derived modalities is then defined as follows:

- $\mathbf{true} \equiv p \vee \neg p$;
- $\mathbf{false} \equiv \neg\mathbf{true}$;
- $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$;
- $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$;
- $\mathbf{F}\psi \equiv \mathbf{true}\mathbf{U}\psi$;
- $\mathbf{G}\psi \equiv \neg\mathbf{F}\neg\psi$.

As an example, consider the following statement:

> *"After the button is pressed, the light will turn red until the elevator arrives at the floor and the doors open."*

Such a situation can be captured by the following LTL formula:

$$p \rightarrow \mathbf{X}\ (\ q\ \mathbf{U}\ (\ s \wedge v\ )\ )$$

where p, q, s, and v are proposition letters corresponding to the button being pressed, the light turning red, the elevator arriving, and the doors opening, respectively.

3 A short state of the art

Over the years, several authors have devised methodologies to translate English sentences into LTL formulas. However, they typically do not deal with the most general case of translating natural, unbounded, English sentences into general, unbounded LTL formulas: assumptions are made that either reduce the generality of the input text or the output formula.

In [29], the authors present a pattern-based approach to the encoding of property specifications for finite-state system verification. Among other contributions, they provide a set of 55 LTL-based formulas that should capture typical patterns that come into play in the design of concurrent and reactive systems, such as the existence of a specific condition or the conditional response to a stimulus. A detailed description of such patterns, together with a repository of examples, are available on the SAnToS laboratory website [5]. Such a set of patterns has been later extended to deal with more advanced scenarios, such as, for instance, timed property specifications [42] and real-time systems [49].

In [72], the authors start from the LTL repository available in [5], identifying the 8 patterns that, according to their analysis, are the most frequently used in temporal requirement specification, covering over 80% of the cases they encountered, that all lie in the aerospace domain. Then, they develop a set of shallow classifiers (such as Random Forests [22] and Support Vector Machines [25]) that are capable of detecting the presence of such patterns in textual technical specifications. As for the predictor attributes, they make use of a bag-of-words approach, enriched with information derived from Part of Speech (PoS) tagging. It should be observed that no solution is given to the problem of instantiating a detected pattern on the specific textual data. Thus, such an approach is not able of deriving a complete translation from English to the restricted set of LTL formulas they consider.

In [100], an algorithm is presented to translate a property, which is specified by making use of predefined subset of English (referred to as *controlled English)*, to LTL. The approach relies on syntactical properties only, being based on text processing techniques like grammatical dependency parsing.

In [59], LTL is used as a formalization technique within an integrated system for generating, managing, and executing controllers for autonomous robots. The idea is that a user should be capable of instructing a robot by means of natural language, so no assumption is made on the

form of English text that is given as input to the system. A textual instruction is processed making use of tools for speech tagging, dependency parsing, and null element restoration (see, for instance, [36]). Then, all verbs are identified, together with their arguments. Finally, each verb is mapped into a set of *senses* in VerbNet [84]. Although in principle no restriction is made on the kind of input phrases that a user may submit to the system, in practice the translation is based on a mapping between *senses* and LTL formulas, that is in turn based on different, manually specified, combinations of so-called *macros*.

In [96], a framework for requirement consistency management is presented. In order to formalize requirements, the system automatically translates natural language descriptions of functionalities into a logic representation. However, a restricted English grammar is considered, to avoid problems such as semantic ambiguity and to make the overall translation process easier.

In [41], the ARSENAL framework for translating natural language requirements into analyzable formal representations is discussed. Given an input sentence, the first step exploits both domain-specific and domain-independent knowledge to identify entity n-grams, such as, for instance, "Lower Desired Temperature", which are then converted into single terms like, e.g., `Lower_Desired_Temperature`. In addition, common expressions, such as "is greater than or equal to", are converted into simple terms, like `dominates`, in an attempt to regularize the input text and reduce its complexity. Next, a dependency parsing step is performed to extract grammatical relationships between phrase elements. These pieces of information, together with PoS tags, are then fed to a so-called *semantic processor*, that guides the translation from English to an internal, intermediate representation. It should be observed that such a module relies on a set of hand-made rules, that inevitably are domain-specific and can only work for restricted scenarios. Finally, the intermediate representation can be converted into several formalisms, including LTL.

Last but not least, a *controlled natural language* is defined in [83], which can be used to specify restrictions on how a system model interacts with its environment. Sentences formulated in such a constrained language are then automatically translated to LTL by means of hard-coded rules.

Broadening the attention scope to other kinds of (temporal) logical formalisms, the following contributions are worth mentioning.

In [32], the authors present a tool for the automatic translation of natural language sentences into formulas of the action-based temporal logic ACTL, in the context of the formalization of reactive systems requirements. A corpus of sentences is analyzed and, starting from it, a context-free grammar is manually built that allows one to parse the considered instances. Such a grammar is augmented with attributes, that allow one to generate the target ACTL formula during the parsing process.

In [71], a translation method is discussed that converts constrained English utterances into a representation level based on Kamp's *Discourse Representation Theory* [46]. Such an intermediate representation is then translated into an ACTL temporal logic formula.

In [30], the authors present a methodology to translate natural language instructions into a formal logical description of goals, that can then be issued to and followed by robots. The target formalisms are CTL* and first-order dynamic logic (FDL). The translation process is carried out by means of a hand-made combinatory categorial grammar [87].

In [43], hand-made attribute grammars are employed to generate Computation Tree Logic (CTL) formulas from Hardware Description Language (HDL) code comments written in English.

Finally, in [89], a two-phase approach to parsing natural language into formal logic is presented. The first phase aims at extracting the generic structure of the logical expression associated with a given natural language utterance. The general idea is that of building a dependency parse of the input utterance, and then attaching to its nodes λ-expressions chosen from a pre-specified finite set, by means of a specifically trained model. The node assignment to λ-expressions is then evaluated to a generic logical form structure. The second phase instantiates the discovered pattern on the specific utterance data. Some considerations are made about the possibility of adopting LTL as a target formalism in future work.

Overall, the typical approach followed by these studies can be summarized as follows: given an input English utterance, preprocess it to extract syntactical information, which may include part of speech tagging, dependency parsing, semantic role labelling, and so on. Then, enrich the input with these pieces of information. Finally, run an attribute grammar-based parser, or rely on some hand-made rules, to derive a translation into a target logical format. A notable exception is the work of [89], where a fully-supervised learning setting is considered.

4 Possible future research directions

In the following, on the basis of the analysis of related work done in the previous section, we outline some possible approaches to the problem of translating open English utterances into general LTL formulas. For each of them, we identify existing tools and solutions from the literature that can possibly be exploited.

4.1 LTL synthesis via classical semantic parsing

Classical approaches to semantic parsing involve the definition of a suitable grammar, which it is possible to rely on to parse a given phrase, and to contextually generate a desired output. Typically, attribute grammars or combinatory categorial grammars have been considered in the literature for this task (see Section 3), although specifically developed grammars have also been proposed [69].

As already pointed out, the problem of generating LTL formulas from English texts can be viewed as an instance of semantic parsing. Some frameworks that allow one to develop semantic parsers are available in the literature, most notably KRISP [47], SEMPRE [19], Cornell Semantic Parsing Framework [18], SippyCup [11], and WASP [15].

However, the main difficulty remains that of defining a suitable grammar, a task that for real-world applications cannot be performed by hand, especially when, instead of restricting to a specific domain, we refer to a general translation scenario. In an attempt to facilitate this task, several approaches to grammar induction have been proposed over the years. Some of them are specifically oriented to natural language processing, like, for instance, those presented in [28, 44, 86, 101]; others are more general, e.g., [53, 77]. Nevertheless, they all require a quite large amount of training data, a problem that is shared also by the (more promising) strategy presented in Section 4.2.

4.2 LTL synthesis as a translation problem

Machine translation can be viewed as the use of software to translate text or speech from one language to another. Several approaches to translation have been proposed over the years.

The first proposed one was the so-called *rule-based paradigm*, in which explicit rules are given that guide the translation. The translation can be done in two different ways: either directly mapping the source language into the target one, or relying on some sort

of interlingual representation. Of course, the richness and complexity of natural language imply that such handcrafted approaches are only suitable for very constrained domains. Nevertheless, some platforms for rule-based translation are available, such as, for instance, Apertium [35].

Subsequently, *statistical machine methods* have gained in popularity. In such a case, translations are not generated by ad-hoc developed rules anymore, but are, instead, generated on the basis of statistical models, whose parameters are trained on bilingual text corpora.

At the moment, the most popular approach is the one based on *neural machine translation*, where a large artificial neural network is used to predict, given an utterance in the source language, the likelihood of a sequence of words in the target language. Several frameworks have been proposed in the literature to develop general-purpose neural networks, such as, for instance, PyTorch [9], Tensorflow [13], and Keras [6]. Recently, OpenNMT [48] has been presented as an open source toolkit specifically oriented to neural machine translation. The system prioritizes efficiency, modularity, and extensibility, and it allows one to develop state-of-the-art solutions, based on Convolutional Neural Networks, LSTM, attention mechanisms, and word- as well as character-based embeddings, through a simple general-purpose interface, that in principle requires only source and target files to be provided. Besides language translation, the framework also supports other sequence generation tasks, such as, for example, summarization, image-to-text and speech recognition[2].

It is worth pointing out that most of the translation models from the literature are based on a sequence-to-sequence architecture [91]. Nevertheless, since a logic formula can be represented by a tree-like structure, it might be better to opt for a sequence-to-tree approach [64], or for other *constrained decoding* techniques (see, for instance, [10]).

Since the English-to-LTL mapping can be viewed as a language translation problem, it makes sense to think of training a neural network model to perform such a task. The best candidate framework for doing that seems to be, to date, the previously-discussed OpenNMT. Unfortunately, a major problem has to be solved in order to actually pursue this approach, that is, the lack of a large training dataset in which English utterances are paired with the corresponding LTL formulas. The University of Kansas provides an online repository of temporal logic formulas that represent the content of technical specifications [5]. However, just about a hundred of English-to-LTL pairings are available. LTLStore[3] is an online repository of around one thousand LTL formulas, collected from previous studies. Although they lack an English counterpart, they may still be useful as *monolingual* data [85].

Various approaches can be followed to address the problem of the scarcity of training data. We would like to outline three of them.

The first, most trivial one consists of randomly generating a large set of LTL formulas by means of a suitable grammar. Then, rules may be derived to translate each formula into an English utterance. Also, some post-processing techniques can be applied to make the resulting text more realistic, such as replacing proposition letters by a set of words that represent them (e.g., `button_pressed` might be mapped to *"the button is pressed"*). Of course, the generated sentences would still make use of a very constrained kind of English. Nevertheless, such an artificial training dataset may be used as a starting point to determine which are the most promising translation techniques.

The second approach consists of finding a kind of "bridging dataset", i.e., a tranining dataset that maps each English utterance in a formal representation that, in turn, can be more or less easily converted into LTL by means of hardcoded rules. As an example, it may

[2] Additional details can be found on OpenNMT website: `http://opennmt.net/`.
[3] LTLStore project website: `https://gitlab.lrz.de/i7/ltlstore`.

be worth investigating the data used in semantic parsing competitions, such as, for instance, SemEval[4], that over the years included some tracks concerned with the evaluation of temporal content in natural language phrases (this is the case with TempEval). Other possibly useful resources are the TimeML [80] annotated corpora TimeBank [82] and AQUAINT [2], and the ACE (Automatic Content Extraction) [1] and WikiWars [66] corpora, that all provide texts manually labeled with events, temporal expressions, and relationships.

A third recent research line has managed to train both neural machine and statistical machine translation systems using monolingual corpora only [16, 17, 51, 52]. Although accuracy results are still far from those guaranteed by state of the art, bilingual data-based models, such approaches are worth some further investigation.

4.3 LTL synthesis through evolutionary computation

Both LTL synthesis via semantic parsing and LTL synthesis via machine translation follow consolidated paths in the field. Now, we work out an alternative path that makes use of evolutionary computation.

Evolutionary Algorithms (EAs) are adaptive meta-heuristic search algorithms, inspired by the process of natural selection, biology, and genetics. Unlike blind random search, they are capable of exploiting historical information to direct the search into the most promising regions of the search space and, in order to achieve that, their basic characteristics are designed to mimic the processes that, in natural systems, lead to adaptive evolution [31].

In nature, a population of individuals tends to evolve, in order to adapt to the environment; in EAs, each individual represents a possible solution for the optimization problem, and its degree of "adaptation" to the problem is evaluated by means of a *fitness* function, which can be single- or multi-objective. In the first case, at the end of the optimization process, the single, best solution is returned. In the second case, the algorithm searches for multiple optimal solutions in parallel, that are returned in the form of a so-called *Pareto-front*. Multi-objective approaches are particularly suitable for multi-objective optimization. In both cases, the elements of the population tend to iteratively evolve toward better solutions, going through a series of generations in which the *crossover* (hybridization of two solutions to generate a solution offspring) and *mutation* (random changes performed to a solution) operators are applied to selected individuals. At each generation, the individuals that are considered best by the fitness function are given a higher probability of being transmitted to the following one.

The English-to-LTL mapping problem may be thought of as an optimization one (that in turn can be solved via evolutionary computation): given an input utterance, the task is that of finding the temporal formula that best matches the content of the text.

In the following, the main issues concerning the exploitation of an evolutionary algorithm for the English-to-LTL mapping problem are discussed. They can be summarized as follows: (*i*) how the single solutions are represented; (*ii*) how the population is initialized; (*iii*) which evolutionary operators (crossover, mutation) are employed; (*iv*) which fitness function is used.

4.3.1 Solution representation and population

Each instance in the population represents an LTL formula, which may be conveniently coded by means of a tree-based data structure. As for the initialization of the population, one may proceed as follows. In the first step, all proposition letters are extracted from the

[4] SemEval 2019 website: `http://alt.qcri.org/semeval2019/`.

given utterance. This can be done by means of a suitable predicate extraction process, that can rely on available tools, such as, for instance, PredPatt [7, 97], or on custom-tailored solutions based on *Semantic Role Labeling* or *Open Information Extraction*, making use of natural language processing frameworks such as Stanford CoreNLP [65], the SpaCy-based [12] AllenNLP [39], and TextPro [78]. Once all proposition letters have been extracted, candidate LTL formulas may be randomly generated, ensuring both to use all of the propositional letters (since the goal is that of fully encoding what is happening in the utterance), and to respect LTL syntactical correctness constraints.

4.3.2 Evolutionary operators

The representation of LTL formulas by means of tree-like structures makes it quite straightforward to implement the crossover and mutation operators. Both of them can, indeed, modify or generate a solution by acting on subtrees, e.g., by adding, removing, raising, or swapping them, always making sure that each proposition letter is appearing at least one time in the resulting formula.

4.3.3 Fitness function

In order to establish how good a given formula is, it is necessary to determine how well it captures the pieces of information contained in the utterance. This is the most difficult step, since a properly designed fitness function should be capable of capturing a semantic parallelism between the logical formula and the natural language text, with an accuracy well beyond the one that can be provided, for instance, by commonly used techniques such as Latent Dirichlet Allocation (LDA) [21].

A possible solution can be that of extracting a suitable model from the text, capable of capturing all pieces of information that are relevant for the evaluation of the LTL formula. In order to do that, an initial step, as already discussed in the case of population initialization, can be the extraction of all proposition letters from the utterance. Then, these letters can be arranged in a Kripke structure [50], that is, a model that keeps track of how and when they hold. To generate such a model, the best approach seems to be the one pursued in [94], where a deep learning solution is developed that is capable of assigning time intervals to the verbs in a given text[5] (a similar work is discussed in [58]). Once such time intervals have been determined, the structure can be derived in a fairly straightforward way and, based on it, the (still non trivial) question now becomes "how much and how well the given LTL formula is describing what is happening in the Kripke model?". Finally, a second objective must be considered, that is, generating easy to read, and thus understandable, formulas. Such a condition can be enforced by using fairly natural metrics such as the nesting degree of the formula, the number of Boolean and/or temporal operators employed, and so on.

4.4 LTL synthesis through event identification and ordering

Finally, the English-to-LTL translation problem can be addressed with an approach which turns out to be somehow more hardwired than the ones discussed before, but, nevertheless, makes use of some advanced machine learning methodologies.

The idea is that of first determining the syntactic structure of the utterance and the events it refers to, then identifying the main agent and action of each of them, and finally establishing an ordering over them. Based on such pieces of information, it should be easier

[5] A working implementation can be found at: `https://hub.docker.com/r/sidvash/temporal`.

to derive a set of rules for the generation of the corresponding LTL formula, at least up to a fixed, maximum degree of nesting. Of course, the translated utterances could be fed in as training data to any other solution developed according to the approaches described in Section 4.1 and Section 4.2. In the following, we give a more detailed account of these phases.

As for the syntactic analysis, several frameworks can be found in the literature, most notably SpaCy [12], AllenNLP [39], which is based on SpaCy, Stanford CoreNLP [65], NLTK [63], and TextPro [78]. All of them allow one to perform the following syntactic analysis tasks, that are necessary to carry out the translation:

- *part of speech tagging*, by which components such nouns, verbs, and adjectives are identified;
- *construction of the dependency tree*, that establishes the relationships between "head" words and words which modify such heads;
- *coreference resolution*, that is, the identification of all the expressions that refer to the same entity in the text;
- *recognition of noun/prepositional/verb phrases*, that may help in the later identification of proposition letters (for instance, *"the big blue ocean"* should be considered as a single entity).

Then, proposition letters are to be extracted, relying on tools like, e.g., PredPatt [7, 97], or on custom solutions, based on the previously-introduced frameworks, making use of either:

- *semantic role labeling*, that is, the process that assigns labels to words or phrases in a sentence that indicate their semantic role, such as, for instance, agent or action, or
- *open information extraction*, that generates a structured, machine-readable representation of the relevant data in the text, typically in the form of n-ary propositions.

The next step is that of correlating proposition letters by making use of Boolean connectives and temporal modalities. To this end, other than relying on the dependency tree and on specific keyword extraction, it is possible to consider:

- *temporal expression recognition and normalization*, that is, the task of identifying sequences of tokens in a given text that denote a point in time, a duration, or a frequency, and of interpreting them. A lot of work has been done in this respect (see, e.g., [27, 54, 61, 98]), including a number tools that can be used for the task: PTime [8], SUTIME [23], UWTIME [56], ClearTK's TimeML module [4], HeidelTime [88], cogCompTime [76], SynTime [99], and TextPro's TimePro module [78]. In addition, AllenNLP and Stanford coreNLP contain some modules that can be used as well. The typical performance is higher than 80% precision and recall on benchmark data such as the Platinum dataset from the TempEval3 workshop [93].
- *temporal relation identification*, i.e., the task of determining a partial or total ordering of events (e.g., identified by verbs) happening in an utterance. While this can still be considered an open problem [26], a lot of work has already been done in the literature [20, 37, 60, 67, 73, 74, 75], also driven by important domain requirements, such as those in the medical field [34, 55, 92]. Some tools are available that either (*i*) encode temporal relationships by means of graphs, as it happens with CATENA [68], ClearTK's TimeML module [4], TextPro's TempRelPro module [78], cogCompTime [76], CAEVO [3], and TIPsem [14, 62], or, perhaps more naturally, (*ii*) assign a time interval to each event, like in [94] (we already refer to it in Section 4.3) and [58].

Other resources that can be successfully exploited are WordNet [33], a well-known lexical database for the English language that groups words into sets of synonyms and records a number of relations among them, VerbNet [84], which is the largest on-line verb lexicon currently available for English, PPDB [38], a database containing hundreds of million

paraphrase pairs, which capture many meaning-preserving syntactic transformations, and VerbOcean, a repository that encompasses the semantic relationships that in English language typically hold between verbs, including a *happens-before* relation [24].

5 An experimental evaluation of existing tools

This section is devoted to a critical experimental evaluation of some of the tools that have been previously discussed. For such a purpose, two datasets have been considered.

The first one is the *US Government's Accident Injuries* collection[6], that contains information on all accidents, injuries, and illnesses reported by mine operators and contractors beginning with 1/1/2000. Each instance corresponds to an accident and is characterized, among all other pieces of information, by a free-text narrative describing the event.

The second dataset has been provided by *Main Roads Western Australia*[7], and it includes information on fatal, serious, and medical car crashes recorded between 2013 and 2017 in Western Australia. As it happens with the previous collection, each instance is characterized by a narrative describing how the incident occurred.

The two datasets are representative of two different kinds of narratives: in the mining case, the textual description is typically short, and involves a single participant. On the contrary, the car crash narratives are longer and more articulated, may involve several participants, and tend to thoroughly describe the event.

For reference, the following representative utterance from the mining accidents dataset is taken into account:

> *Employee was cleaning up at the Primary Crusher with the Dingo skid steer. The employee slipped and fell while operating the skid steer and the machine pinned him against the cement retaining wall.*

As for the car crashes dataset, the following representative instance is considered:

> *B drove east on Gibbins Road, Coolup. It appears B has stopped at the intersection of Maryfield Road and allowed a vehicle to pass before proceeding into the intersection. B2 was driving north on Maryfield Road, has observed B's vehicle at the intersection noticing that B appeared to be looking straight ahead. As B has proceeded into the intersection, B2 has applied the brakes on his vehicle, however he was unable to avoid B's vehicle and has 't-boned' it, colliding with the drivers side of B's vehicle. B has died at the scene from injuries sustained. A preliminary test conducted on B2 returned a negative reading.*

The remainder of this section considers two main tasks that, according to Section 4.4, have to be carried out in order to synthesize an LTL formula that correctly formalizes an utterance, that is, the extraction of the proposition letter candidates, and the identification of the temporal relationships among them.

Concerning the extraction of proposition letter, we may identify two distinct steps: first, *coreference resolution*, by which all references to the same entity are identified and normalized[8]; second, the *extraction of predicates*, i.e., tuples composed of an agent, the action that is being carried out, and possible other arguments.

---

[6] `https://catalog.data.gov/dataset/accident-injuries`

[7] `https://www.mainroads.wa.gov.au/Pages/default.aspx`

[8] Observe that coreference resolution implies entity identification which, nevertheless, is a much easier task, that in general can be reliably solved by investigating the dependency tree and the PoS tags of the given utterance.

Table 1 Predicates extracted by PredPatt on the mining accidents dataset instance. Results have been reworked for the ease of reading.

Agent	Action
Employee	was cleaning up
Employee	slipped
Employee	fell
the Dingo skid	steer
the Dingo skid	pinned Employee

As for coreference resolution, AllenNLP [39] relies on the state-of-the-art algorithm presented in [57]. Tested on the mining accidents dataset instance, it identifies two entities with coreferences, highlighted in red and blue colours:

> ***Employee*** *was cleaning up at the Primary Crusher with* ***the Dingo skid steer****.* ***The employee*** *slipped and fell while operating* ***the skid steer*** *and the machine pinned* ***him*** *against the cement retaining wall.*

Although the result is fairly accurate, the algorithm fails to detect the *the machine* reference to the blue entity. Turning to a more challenging example, this is the result of its execution on the second text[9]:

> ***B*** *drove east on Gibbins Road,* ***Coolup****. It appears* ***B*** *has stopped at the intersection of Maryfield Road and allowed a vehicle to pass before proceeding into the intersection.* ***B2*** *was driving north on Maryfield Road, has observed* ***B's*** *vehicle at the intersection noticing that* ***B*** *appeared to be looking straight ahead. As* ***B*** *has proceeded into the intersection,* ***B2*** *has applied the brakes on* ***his*** *vehicle, however* ***he*** *was unable to avoid* ***B's*** *vehicle and has 't-boned' it, colliding with the drivers side of* ***B's*** *vehicle.* ***B*** *has died at the scene from injuries sustained. A preliminary test conducted on* ***B2*** *returned a negative reading.*

Focusing on just two entities, it is immediately clear that the algorithm has erroneously associated *Coolup* with *B2*. Even worse, judging from the result it seems that *B2* has been applying the brakes on *B*'s vehicle, and that *B* was unable to avoid himself. These two errors are enough to totally misinterpret the semantic content of the utterance.

Once the entities have been found and normalized, it is possible to look for the candidate proposition letter. In this respect, we investigated the use of PredPatt [7, 97], since it is still a major component in many recent, state-of-the-art NLP tools (see, for instance, [94]). To do that, we relied on two versions of the chosen instances in which coreference resolution has been already correctly solved by hand.

The predicates extracted by PredPatt on the mining data utterance are reported in Table 1. As it can be seen, the algorithm outputs 5 candidates. Again, with the exception of the fourth candidate, the result is quite natural and intuitive, except perhaps for the fact that the fourth and fifth candidates refer to the *the Dingo skid* entity instead of *the Dingo skid steer*, considering *steer* as a verb.

[9] Note that the coreference resolution process extracted more than two entities. However, for the sake of clarity, we concentrate here on the two main ones, that are, *B* and *B2*.

Table 2 Predicates extracted by PredPatt on the car crashes dataset instance. Results have been reworked for the ease of reading.

Agent	Action
B	drove east
It	appears
A vehicle	allowed to pass
[UNK]	was driving north
B	appeared to be looking straight ahead
B	has proceeded into the intersection
B2	has applied the brakes on B2's vehicle
[UNK]	avoid B's vehicle
[UNK]	't-boned' it
[UNK]	has died at the scene from injuries
A preliminary test	returned negative reading

Table 2 reports the predicates extracted by PredPatt algorithm on the car crash dataset instance. Here, there are some spurious candidates as, for example, the one with *It* as the main agent. Moreover, for several results, the agent is not clear (*UNK*), or simply wrong (*A vehicle allowed to pass*). Finally, some predicates are entirely missed, like in the case of *B has stopped at the intersection*.

Finally, let us consider the task of extracting the temporal relationships that hold between each pair of predicates. Our original intention was that of evaluating cogCompTime [76] as the tool for generating an ordering graph over the set of events, and the approach presented in [94] as the tool for assigning time intervals to events, since they are two recently proposed solutions. Unfortunately, since various weeks cogCompTime is experiencing some server problems (`http://groupspaceuiuc.com/temporal/#`), that do not allow us to experiment with it. Thus, in the following, just the time interval approach presented in [94] is evaluated with respect to the two selected utterances.

Figure 1 shows its results when applied on the mining dataset instance. As it can be seen, the two occurrences of *steer* are still erroneously recognized as actions. Moreover, while the interval assigned to *operating* seems to be an appropriate choice, the tool fails in identifying its overlap with the interval related to *cleaning*. Furthermore, the events *fell* and *slipped* appear to be swapped.

Considering the car crashes instance, as depicted in Figure 2, the outcomes are unfortunately quite poor, and do not seem to follow any reasonable pattern.

At this point, by looking at the underwhelming results obtained by the tools we tested, we decided not to proceed with the LTL formula synthesis task any further.

6 Conclusions

Linear-time temporal logic (LTL) is a logical formalism which is commonly adopted in formal verification, in particular in consistency and model checking, and it can be a very useful tool for capturing the temporal content of natural language utterances. Such capabilities can be exploited in many important applications, including disambiguation of technical requirements, log analysis, and automated reasoning over temporal data.

Unfortunately, as it emerges from our comprehensive analysis of the literature and of existing tools, a general enough solution, that is capable of translating free, natural English texts into unbounded, general LTL formulas is still missing.

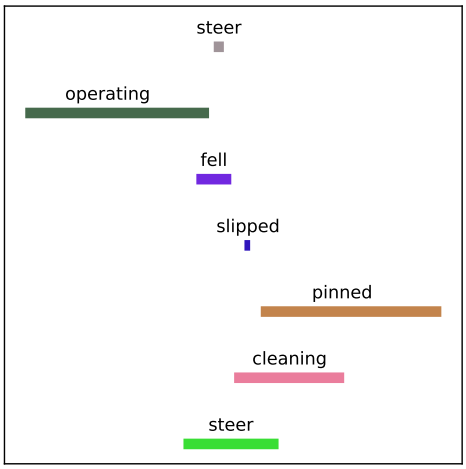


Figure 1 Events timeline in the mining dataset instance.

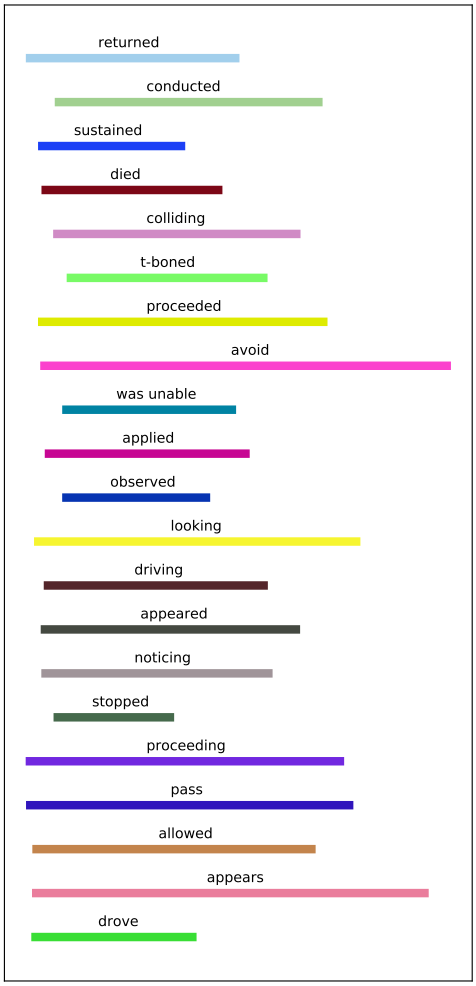


Figure 2 Events timeline in the car crashes dataset instance.

In this paper, we provided a comprehensive review of the state of the art, as defined by the relevant literature; moreover, an empirical evaluation has been conducted on some of the currently available NLP tools, based on two real world instances. Results show that further research work is needed on methods and tools for various crucial tasks, such as coreference resolution, predicate extraction, and temporal relation identification, in order to achieve a performance which is good enough to allow for the synthesis of LTL formulas from unconstrained, natural language texts. We highlighted some possible research directions that can be followed to tackle such a complex problem.

References

1 ACE project website. Accessed: April 2019. URL: https://www.ldc.upenn.edu/collaborations/past-projects/ace.

2 AQUAINT corpus website. Accessed: April 2019. URL: https://catalog.ldc.upenn.edu/LDC2002T31.

3 CAEVO corpus website. Accessed: April 2019. URL: https://www.usna.edu/Users/cs/nchamber/caevo/.

4 ClearTK GitHub page. Accessed: April 2019. URL: https://cleartk.github.io/cleartk/.

5 Kansas State University CIS Department, Laboratory for Specification, Analysis, and Transformation of Software (SAnToS Laboratory), Property Pattern Mappings for LTL. Accessed: April 2019. URL: http://patterns.projects.cs.ksu.edu/.

6 Keras framework website. Accessed: April 2019. URL: https://keras.io.

7 PredPatt GitHub page. Accessed: April 2019. URL: https://github.com/hltcoe/PredPatt.

8 PTime website. Accessed: April 2019. URL: http://ws.nju.edu.cn/ptime/.

9 PyTorch GitHub page. Accessed: April 2019. URL: https://github.com/pytorch.

10 Semantic parsing with AllenNLP. Accessed: April 2019. URL: https://github.com/allenai/allennlp/blob/master/tutorials/getting_started/semantic_parsing.md.

11 Sippycup GitHub page. Accessed: April 2019. URL: https://github.com/wcmac/sippycup.

12 SpaCy website. Accessed: April 2019. URL: https://spacy.io/.

13 TensorFlow framework website. Accessed: April 2019. URL: https://www.tensorflow.org/.

14 TIPSem GitHub page. Accessed: April 2019. URL: https://github.com/hllorens/otip.

15 WASP semantic parser website. Accessed: April 2019. URL: http://www.cs.utexas.edu/users/ml/wasp/.

16 Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Unsupervised Statistical Machine Translation. *CoRR*, abs/1809.01272, 2018. arXiv:1809.01272.

17 Mikel Artetxe, Gorka Labaka, and Eneko Agirre. An Effective Approach to Unsupervised Machine Translation. *CoRR*, abs/1902.01313, 2019. arXiv:1902.01313.

18 Yoav Artzi. Cornell SPF: Cornell semantic parsing framework. *arXiv preprint*, 2013. arXiv:1311.3011.

19 Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1533–1544, 2013. URL: http://aclweb.org/anthology/D/D13/D13-1160.pdf.

20 Steven Bethard, Timothy A. Miller, Dmitriy Dligach, Chen Lin, and Guergana Savova. Neural Temporal Relation Extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pages 746–751, 2017. URL: https://aclanthology.info/papers/E17-2118/e17-2118.

21 David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. URL: http://jmlr.org/papers/v3/blei03a.html.

22 Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. doi:10.1023/A:1010933404324.

23 Angel X. Chang and Christopher D. Manning. SUTime: A library for recognizing and normalizing time expressions. In *Proceedings of the 8th International Conference on Language Resources and Evaluation, LREC*, pages 3735–3740, 2012. URL: `http://www.lrec-conf.org/proceedings/lrec2012/summaries/284.html`.

24 Timothy Chklovski and Patrick Pantel. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP*, pages 33–40, 2004. URL: `http://www.aclweb.org/anthology/W04-3205`.

25 Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995. `doi:10.1007/BF00994018`.

26 Leon Derczynski. *Automatically Ordering Events and Times in Text*, volume 677 of *Studies in Computational Intelligence*. Springer, 2017. `doi:10.1007/978-3-319-47241-6`.

27 Wentao Ding, Guanji Gao, Linfeng Shi, and Yuzhong Qu. A Pattern-based Approach to Recognizing Time Expressions. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019.

28 Arianna D'Ulizia, Fernando Ferri, and Patrizia Grifoni. A survey of grammatical inference methods for natural language learning. *Artificial Intelligence Review*, 36(1):1–27, 2011. `doi:10.1007/s10462-010-9199-1`.

29 Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in Property Specifications for Finite-State Verification. In *Proceedings of the 1999 International Conference on Software Engineering, ICSE*, pages 411–420, 1999. `doi:10.1145/302405.302672`.

30 Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul W. Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *2009 IEEE International Conference on Robotics and Automation, ICRA*, pages 4163–4168, 2009. `doi:10.1109/ROBOT.2009.5152776`.

31 A. E. Eiben and James E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, 2015. `doi:10.1007/978-3-662-44874-8`.

32 Alessandro Fantechi, Stefania Gnesi, Gioia Ristori, Michele Carenini, Massimo Vanocchi, and Paolo Moreschini. Assisting Requirement Formalization by Means of Natural Language Translation. *Formal Methods in System Design*, 4(3):243–263, 1994. `doi:10.1007/BF01384048`.

33 Christiane Fellbaum. WordNet. *The Encyclopedia of Applied Linguistics*, 2012.

34 Olivier Ferret, Xavier Tannier, Aurélie Névéol, and Julien Tourille. Temporal information extraction from clinical text. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pages 739–745, 2017. URL: `https://aclanthology.info/papers/E17-2117/e17-2117`.

35 Mikel L. Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M. Tyers. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144, 2011. `doi:10.1007/s10590-011-9090-0`.

36 Ryan Gabbard, Seth Kulick, and Mitchell P. Marcus. Fully Parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, 2006. URL: `http://aclweb.org/anthology/N/N06/N06-1024.pdf`.

37 Diana Galvan, Naoaki Okazaki, Koji Matsuda, and Kentaro Inui. Investigating the Challenges of Temporal Relation Extraction from Clinical Text. In *Proceedings of the 9th International Workshop on Health Text Mining and Information Analysis 31, 2018*, pages 55–64, 2018. URL: `https://aclanthology.info/papers/W18-5607/w18-5607`.

38 Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: the paraphrase database. In *Proceedings of the Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 758–764, 2013. URL: `http://aclweb.org/anthology/N/N13/N13-1092.pdf`.

39 Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640, 2018. `arXiv:1803.07640`.

40 Rob Gerth, Doron A. Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proceedings of the 15th International Symposium on Protocol Specification*, pages 3–18, 1995.

41 Shalini Ghosh, Daniel Elenius, Wenchao Li, Patrick Lincoln, Natarajan Shankar, and Wilfried Steiner. ARSENAL: automatic requirements specification extraction from natural language. In *Proceedings of the 8th NASA Formal Methods International Symposium*, pages 41–46, 2016. doi:10.1007/978-3-319-40648-0_4.

42 Volker Gruhn and Ralf Laue. Patterns for Timed Property Specifications. *Electronic Notes in Theoretical Computer Science*, 153(2):117–133, 2006. doi:10.1016/j.entcs.2005.10.035.

43 Christopher B. Harris and Ian G. Harris. Generating formal hardware verification properties from Natural Language documentation. In *Proceedings of the 9th IEEE International Conference on Semantic Computing, ICSC*, pages 49–56, 2015. doi:10.1109/ICOSC.2015.7050777.

44 Christopher B. Harris and Ian G. Harris. GLAsT: Learning formal grammars to translate natural language specifications into hardware assertions. In *Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition, DATE*, pages 966–971, 2016. URL: http://ieeexplore.ieee.org/document/7459447/.

45 Claude Jard and Thierry Jéron. On-Line Model Checking for Finite Linear Temporal Logic Specifications. In *Proceedings of the International Conference on Computer Aided Verification*, pages 189–196, 1989. doi:10.1007/3-540-52148-8_16.

46 Hans Kamp. A theory of truth and semantic representation. *Formal semantics-the essential readings*, pages 189–222, 1981.

47 Rohit J. Kate and Raymond J. Mooney. Using String-Kernels for Learning Semantic Parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics ACL*, 2006. URL: http://aclweb.org/anthology/P06-1115.

48 Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 67–72, 2017. doi:10.18653/v1/P17-4012.

49 Sascha Konrad and Betty H. C. Cheng. Real-time specification patterns. In *Proceedings of the 27th International Conference on Software Engineering, ICSE*, pages 372–381, 2005. doi:10.1145/1062455.1062526.

50 Saul A. Kripke. Semantical Considerations on Modal Logic. *Acta Philosophica Fennica*, 16(1963):83–94, 1963.

51 Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised Machine Translation Using Monolingual Corpora Only. In *Proceedings of the 6th International Conference on Learning Representations, ICLR*, 2018. URL: https://openreview.net/forum?id=rkYTTf-AZ.

52 Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-Based & Neural Unsupervised Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, 2018. URL: https://aclanthology.info/papers/D18-1549/d18-1549.

53 Pat Langley and Sean Stromsten. Learning Context-Free Grammars with a Simplicity Bias. In *Proceedings of the 11th European Conference on Machine Learning, ECML*, pages 220–228, 2000. doi:10.1007/3-540-45164-1_23.

54 Egoitz Laparra, Dongfang Xu, and Steven Bethard. From Characters to Time Intervals: New Paradigms for Evaluation and Neural Parsing of Time Normalizations. *Transactions of the Association of Computational Linguistics*, 6:343–356, 2018. URL: https://transacl.org/ojs/index.php/tacl/article/view/1318.

55 Hee-Jin Lee, Yaoyun Zhang, Min Jiang, Jun Xu, Cui Tao, and Hua Xu. Identifying direct temporal relations between time and events from clinical notes. *BMC Medical Informatics and Decision Making*, 18(S-2):23–34, 2018. doi:10.1186/s12911-018-0627-5.

56 Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. Context-dependent Semantic Parsing for Time Expressions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*, pages 1437–1447, 2014. URL: `http://aclweb.org/anthology/P/P14/P14-1135.pdf`.

57 Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end Neural Coreference Resolution. *CoRR*, abs/1707.07045, 2017. `arXiv:1707.07045`.

58 Artuur Leeuwenberg and Marie-Francine Moens. Temporal Information Extraction by Predicting Relative Time-lines. *CoRR*, abs/1808.09401, 2018. `arXiv:1808.09401`.

59 Constantine Lignos, Vasumathi Raman, Cameron Finucane, Mitchell P. Marcus, and Hadas Kress-Gazit. Provably correct reactive control from natural language. *Autonomous Robots*, 38(1):89–105, 2015. `doi:10.1007/s10514-014-9418-8`.

60 Zhengzhong Liu, Teruko Mitamura, and Eduard H. Hovy. Graph-Based Decoding for Event Sequencing and Coreference Resolution. *CoRR*, abs/1806.05099, 2018. `arXiv:1806.05099`.

61 Hector Llorens, Leon Derczynski, Robert J. Gaizauskas, and Estela Saquete. TIMEN: an open temporal expression normalisation resource. In *Proceedings of the 8th International Conference on Language Resources and Evaluation, LREC*, pages 3044–3051, 2012. URL: `http://www.lrec-conf.org/proceedings/lrec2012/summaries/128.html`.

62 Hector Llorens, Estela Saquete, and Borja Navarro. TIPSem (English and Spanish): Evaluating CRFs and semantic roles in TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 284–291, 2010. URL: `http://aclweb.org/anthology/S/S10/S10-1063.pdf`.

63 Edward Loper and Steven Bird. NLTK: the natural language toolkit. *CoRR*, cs.CL/0205028, 2002. `arXiv:cs.CL/0205028`.

64 Weicheng Ma, Zhaoheng Ni, Kai Cao, Xiang Li, and Sang Chin. Seq2tree: A tree-structured extension of LSTM network, 2017. URL: `https://openreview.net/forum?id=HJ0WtefAW`.

65 Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*, pages 55–60, 2014. URL: `http://aclweb.org/anthology/P/P14/P14-5010.pdf`.

66 Pawel P. Mazur and Robert Dale. WikiWars: A new corpus for research on temporal expressions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 913–922, 2010. URL: `http://www.aclweb.org/anthology/D10-1089`.

67 Paramita Mirza. Extracting Temporal and Causal Relations between Events. *CoRR*, abs/1604.08120, 2016. `arXiv:1604.08120`.

68 Paramita Mirza and Sara Tonelli. CATENA: CAusal and TEmporal relation extraction from NAtural language texts. In *Proceedings of the 26th International Conference on Computational Linguistics, COLING*, pages 64–75, 2016. URL: `http://aclweb.org/anthology/C/C16/C16-1007.pdf`.

69 Smaranda Muresan, Tudor Muresan, and Judith L Klavans. Lexicalized Well-Founded Grammars: Learnability and Merging. *Technical Report, Columbia University*, 2005.

70 Daniel Neider and Ivan Gavran. Learning Linear Temporal Properties. In *Proceedings of the 2018 Formal Methods in Computer Aided Design Conference, FMCAD*, pages 1–10, 2018.

71 Rani Nelken and Nissim Francez. Automatic Translation of Natural Language System Specifications into Temporal Logic. In *Proceedings of the 8th International Conference on Computer Aided Verification, CAV*, pages 360–371, 1996. `doi:10.1007/3-540-61474-5_83`.

72 Allen P. Nikora and Galen Balcom. Automated Identification of LTL Patterns in Natural Language Requirements. In *Proceedings of the 20th International Symposium on Software Reliability Engineering, ISSRE*, pages 185–194, 2009. `doi:10.1109/ISSRE.2009.15`.

73 Qiang Ning, Zhili Feng, and Dan Roth. A Structured Learning Approach to Temporal Relation Extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1027–1037, 2017. URL: `https://aclanthology.info/papers/D17-1108/d17-1108`.

74 Qiang Ning, Hao Wu, Haoruo Peng, and Dan Roth. Improving Temporal Relation Extraction with a Globally Acquired Statistical Resource. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 841–851, 2018. URL: `https://aclanthology.info/papers/N18-1077/n18-1077`.

75 Qiang Ning, Zhongzhi Yu, Chuchu Fan, and Dan Roth. Exploiting Partially Annotated Data for Temporal Relation Extraction. *CoRR*, abs/1804.08420, 2018. `arXiv:1804.08420`.

76 Qiang Ning, Ben Zhou, Zhili Feng, Haoruo Peng, and Dan Roth. CogCompTime: A tool for understanding time in natural language. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 72–77, 2018. URL: `https://aclanthology.info/papers/D18-2013/d18-2013`.

77 Hari Mohan Pandey, Ankit Chaudhary, and Deepti Mehrotra. Grammar induction using bit masking oriented genetic algorithm and comparative analysis. *Applied Soft Computing*, 38:453–468, 2016. `doi:10.1016/j.asoc.2015.09.044`.

78 Emanuele Pianta, Christian Girardi, and Roberto Zanoli. The TextPro Tool Suite. In *Proceedings of the 6th International Conference on Language Resources and Evaluation, LREC*, 2008. URL: `http://www.lrec-conf.org/proceedings/lrec2008/summaries/645.html`.

79 Amir Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977. `doi:10.1109/SFCS.1977.32`.

80 James Pustejovsky, José M. Castaño, Robert Ingria, Roser Saurí, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. TimeML: Robust specification of event and temporal expressions in text. *New Directions in Question Answering*, pages 28–34, 2003.

81 Aarne Ranta. Translating Between Language and Logic: What is Easy and What is Difficult. In *Proceedings of the International Conference on Automated Deduction*, pages 5–25. Springer, 2011.

82 Nils Reimers, Nazanin Dehghani, and Iryna Gurevych. Temporal Anchoring of Events for the TimeBank Corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*, 2016. URL: `http://aclweb.org/anthology/P/P16/P16-1207.pdf`.

83 Tainã Santos, Gustavo Carvalho, and Augusto Sampaio. Formal Modelling of Environment Restrictions from Natural-Language Requirements. In *Proceedings of the 21st Brazilian Symposium on Formal Methods: Foundations and Applications, SBMF*, pages 252–270, 2018. `doi:10.1007/978-3-030-03044-5_16`.

84 Karin Kipper Schuler, Anna Korhonen, and Susan Windisch Brown. VerbNet overview, extensions, mappings and applications. In *Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics on Human Language Technologies*, pages 13–14, 2009. URL: `http://www.aclweb.org/anthology/N09-4007`.

85 Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving Neural Machine Translation Models with Monolingual Data. *CoRR*, abs/1511.06709, 2015. `arXiv:1511.06709`.

86 Bradford Starkie. Inferring Attribute Grammars with Structured Data for Natural Language Processing. In *Proceedings of the 6th International Colloquium on Grammatical Inference: Algorithms and Applications, ICGI*, pages 237–248, 2002. `doi:10.1007/3-540-45790-9_19`.

87 Mark Steedman. *The syntactic process*, volume 24. MIT press Cambridge, MA, 2000.

88 Jannik Strötgen and Michael Gertz. HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324, 2010. URL: `http://aclweb.org/anthology/S/S10/S10-1071.pdf`.

89 Giancarlo Sturla. *A two-phased approach for natural language parsing into formal logic*. PhD thesis, Massachusetts Institute of Technology, 2017.

90 Weiyi Sun, Anna Rumshisky, and Özlem Uzuner. Temporal reasoning over clinical text: the state of the art. *Journal of the American Medical Informatics Association*, 20(5):814–819, 2013. `doi:10.1136/amiajnl-2013-001760`.

91 Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *CoRR*, abs/1409.3215, 2014. `arXiv:1409.3215`.

92 Julien Tourille. *Extracting Clinical Event Timelines: Temporal Information Extraction and Coreference Resolution in Electronic Health Records. (Création de Chronologies d'Événements Médicaux: Extraction d'Informations Temporelles et Résolution de la Coréférence dans les Dossiers Patients Électroniques)*. PhD thesis, University of Paris-Saclay, France, 2018. URL: `https://tel.archives-ouvertes.fr/tel-01997223`.

93 Naushad UzZaman, Hector Llorens, James F. Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. TempEval-3: Evaluating events, time expressions, and temporal relations. *CoRR*, abs/1206.5333, 2012. `arXiv:1206.5333`.

94 SIddharth Vashishtha, Benjamin Van Durme, and Aaron Steven White. Fine-Grained Temporal Relation Extraction. *CoRR*, abs/1902.01390, 2019. `arXiv:1902.01390`.

95 Yorick Wilks and Dann Fass. The preference semantics family. *Computers & Mathematics with Applications*, 23(2-5):205–221, 1992.

96 Rongjie Yan, Chih-Hong Cheng, and Yesheng Chai. Formal consistency checking over specifications in natural languages. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE*, pages 1677–1682, 2015. URL: `http://dl.acm.org/citation.cfm?id=2757200`.

97 Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. An Evaluation of PredPatt and Open IE via Stage 1 Semantic Role Labeling. In *Proceedings of the 12th International Conference on Computational Semantics, IWCS*, 2017. URL: `https://aclanthology.info/papers/W17-6944/w17-6944`.

98 Xiaoshi Zhong and Erik Cambria. Time Expression Recognition Using a Constituent-based Tagging Scheme. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW*, pages 983–992, 2018. `doi:10.1145/3178876.3185997`.

99 Xiaoshi Zhong, Aixin Sun, and Erik Cambria. Time Expression Analysis and Recognition Using Syntactic Token Types and General Heuristic Rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 420–429, 2017. `doi:10.18653/v1/P17-1039`.

100 Lukáš Žilka. Temporal logic for man. Master's thesis, Brno University of Technology, 2010.

101 Szilvia Zvada and Tibor Gyimóthy. Using Decision Trees to Infer Semantic Functions of Attribute Grammars. *Acta Cybernetica*, 15(2):279–304, 2001. URL: `http://www.inf.u-szeged.hu/actacybernetica/edb/vol15n2/Zvada_2001_ActaCybernetica.xml`.

Complexity Analysis of a Unifying Algorithm for Model Checking Interval Temporal Logic

Laura Bozzelli
University of Napoli "Federico II", Napoli, Italy

Angelo Montanari
University of Udine, Udine, Italy

Adriano Peron
University of Napoli "Federico II", Napoli, Italy

Abstract

The model-checking (MC) problem of Halpern and Shoham Interval Temporal Logic (HS) has been recently investigated in some papers and is known to be decidable. An intriguing open question concerns the exact complexity of the problem for full HS: it is at least **EXPSPACE**-hard, while the only known upper bound is non-elementary and is obtained by exploiting an abstract representation of Kripke structure paths called *descriptors*. In this paper we generalize the approach by providing a uniform framework for model-checking full HS and meaningful (almost maximal) fragments, where a specialized type of descriptor is defined for each fragment. We then devise a general MC alternating algorithm parameterized by the type of descriptor which has a polynomially bounded number of alternations and whose running time is bounded by the length of minimal representatives of descriptors (certificates). We analyze the time complexity of the algorithm and give, by non-trivial arguments, tight bounds on the length of certificates. For two types of descriptors, we obtain exponential upper and lower bounds which lead to an elementary MC algorithm for the related HS fragments. For the other types of descriptors, we provide non-elementary lower bounds. This last result addresses a question left open in some papers regarding the possibility of fixing an elementary upper bound on the size of the descriptors for full HS.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Interval temporal logic, Model checking, Complexity and succinctness issues

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.18

1 Introduction

Model checking (MC) is a well-established formal-method technique to automatically check for global correctness of finite-state reactive systems. Finite systems are usually modelled as labelled state-transition graphs (finite Kripke structures), while the properties of interest are specified in standard *Point-based* temporal logics (PTLs), such as, for instance, the linear-time temporal logic LTL [22] and the branching-time temporal logics CTL and CTL* [9]. *Interval temporal logics* (ITLs) provide an alternative setting for reasoning about time [11, 21, 25]. ITLs assume intervals, instead of points, as their primitive temporal entities allowing to specify relevant temporal properties that involve, e.g., actions with duration, accomplishments, and temporal aggregations, which are inherently "interval-based", and thus cannot be naturally expressed by PTLs. ITLs find applications in a variety of computer science fields, including artificial intelligence (reasoning about action and change, qualitative reasoning, planning, and natural language processing), theoretical computer science (specification and verification of programs), and temporal and spatio-temporal databases (e.g. see [13, 21, 23]). Among ITLs, the landmark is *Halpern and Shoham's modal logic of time intervals* (HS) [11] which features one modality for each of the 13 possible ordering relations between pairs of intervals (the so-called Allen's relations [1]), apart from equality. The satisfiability problem for HS is undecidable over all relevant classes of linear orders, and most of its fragments (with some meaningful exceptions [7, 8, 20]) are undecidable as well [6, 12, 15].

Model checking of (finite) Kripke structures against HS has been investigated only very recently [13, 14, 16, 2, 5, 4, 3, 18, 19]. The idea is to interpret each finite path of a Kripke structure as an interval, whose labelling is defined on the basis of the labelling of the component states: a proposition letter holds over an interval if and only if it holds over each component state (*homogeneity assumption* [24]). In this paper, we focus on the MC problem of HS under the *state-based semantics* (time branches both in the future and in the past) proved decidable in [16]. In this setting, the temporal modalities for the Allen's relations *started-by* (B), *finished-by* (E), and *contains* (D), have a "linear-time" character: they allows to select either proper prefixes (B), or proper suffixes (E), or internal subpaths (D) of the current path. The modalities associated with the other Allen's relations are instead "branching-time": they allow *either* to non-deterministically extend a prefix (resp., suffix, resp., subpath) of the current path in the future or in the past, *or* to non-deterministically select an independent path whose start point (resp., ending point) is reachable from (resp., can reach) the ending point (resp., start point) of the current path. The expressiveness of the state-based semantics of HS has been studied in [5] together with two other decidable variants: the *computation-tree-based semantics*, that allows time to branch only in the future, and the *trace-based semantics*, that disallows time branching. The computation-tree-based variant of HS is expressively equivalent to finitary CTL* (the variant of CTL* with quantification over finite paths), while the trace-based variant is equivalent to LTL (but at least exponentially more succinct). The state-based variant is more expressive than the computation-tree-based variant and expressively incomparable with both LTL and CTL*.

As far as concerns the complexity of the state-based MC problem, for the full logic HS, the problem is at least **EXPSPACE**-hard [2], while the only known upper bound is non-elementary [16]. The approach for full HS [16] consists in defining a finite abstraction over the (possibly infinite) set of finite paths of a Kripke structure. This abstraction is parameterized by a natural number h and is based on the *h-level BE-descriptor* of a path: a tree-like structure of depth h which conveys information about the states occurring in prefixes and suffixes of the path. Paths having the same h-level BE-descriptor (i) are indistinguishable with respect to the fulfillment of HS formulas having nesting depth of modalities for prefixes (B) and suffixes (E) at most h, and (ii) admit a bounded minimal representative (*h-level BE-certificate*) whose length is at most a tower of exponentials of height h. The model-checking procedure for full HS based on BE-descriptors is only sketched in [16] and, in particular, the succinctness of BE-descriptors has not been investigated so far. In subsequent papers [3, 19, 4, 17], the focus has been on some syntactical fragments of HS: the fragment featuring only the modalities for the *contains* relation (D), and fragments featuring modalities for a subset of the Allen relations *meets* (A), *started-by* (B), *finished-by* (E) and their transposed relations $\overline{A}$, $\overline{B}$, and $\overline{E}$, respectively (see Table 1 for a graphical intuition of relations). The complete picture of known results is reported in Figure 1.

In this paper, we first provide a uniform framework for the state-based MC problem against the HS syntactical fragments obtained by combining the modalities of a linear-time basis $\mathcal{B}$ (i.e, a non-empty subset of non-interdefinable Allen's relations in $\{B, E, D\}$) with the modalities for the (branching-time) Allen's relations in $\{A, L, O, \overline{A}, \overline{L}, \overline{B}, \overline{E}, \overline{D}, \overline{O}\}$ but not including either the modalities for *overlap* O or the modalities of its transposed relation $\overline{O}$ (the fragment for the complete basis $\{B, E\}$ expresses the full logic HS). The proposed approach generalizes the one provided in [16], where only the full logic HS is considered: for each basis $\mathcal{B}$, it defines a finite abstraction of the set of paths of a Kripke structure based on the notion of *h-level $\mathcal{B}$-descriptor* (coinciding with the BE-descriptor for the complete basis $\mathcal{B} = \{B, E\}$). As for the basis $\{B, E\}$, we show that for all the other bases with

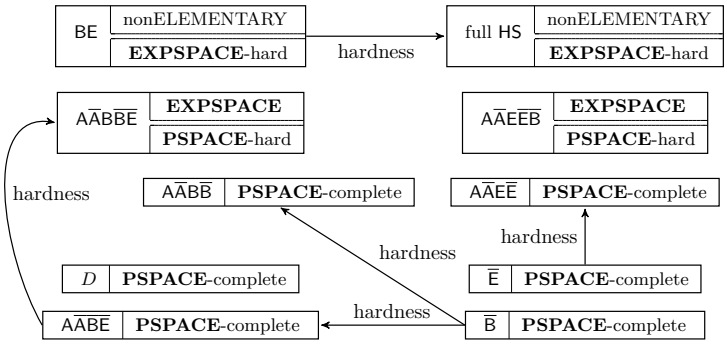


Figure 1 Complexity of the MC problem for HS fragments.

the exception of $\{D\}$, paths having the same h-level $\mathcal{B}$-descriptor (i) are indistinguishable with respect to the fulfillment of HS formulas having nesting depth of modalities for $\mathcal{B}$ at most h, and (ii) admit a bounded minimal representative (*h-level $\mathcal{B}$-certificate*). We exploit these results for devising an *alternating* algorithm, parameterized in the basis $\mathcal{B} \neq \{D\}$, for model-checking the associated fragment, which runs in time bounded by the maximal length of h-level $\mathcal{B}$-certificates of the input Kripke structure, with h being the $\mathcal{B}$-nesting depth of the input formula, and whose number of alternations between existential and universal choices is at most the size of the input formula.

As a second contribution, for each basis $\mathcal{B}$, we provide tight bounds on the length of h-level $\mathcal{B}$-certificates. For the bases $\{B\}$ and $\{E\}$, we prove singly-exponential upper and lower bounds. Hence, by the proposed alternating algorithm, we argue that model-checking for the fragments $\mathsf{A\overline{A}B\overline{B}D\overline{E}L\overline{L}O}$ and $\mathsf{A\overline{A}B\overline{D}E\overline{E}L\overline{L}O}$ is in the complexity class $\mathbf{AEXP_{pol}}$ of problems decided by exponential-time bounded alternating Turing Machines with a polynomially bounded number of alternations (a class included in **EXPSPACE** which captures the precise complexity of some relevant problems, e.g., the first-order theory of real addition with order [10]). On the other hand, for all bases $\mathcal{B}$ distinct from $\{B\}$ and $\{E\}$, we state a non-elementary lower bound. In particular, the result obtained for the basis $\{B, E\}$ negatively answers a question left open in [16] regarding the possibility of fixing an elementary upper bound on the size of BE-descriptors, and at the same time provides new insight on the MC problem for full HS: if elementary procedures are possible, they have certainly to exploit less powerful structures than descriptors.

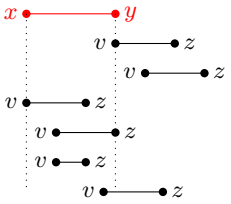
The paper is organised as follows. In Section 2, we recall the state-based model-checking framework for HS. In Section 3, we introduce for each basis $\mathcal{B}$, the notion of $\mathcal{B}$-descriptor, and describe the algorithm to solve the MC problem for the associated fragment. In Section 4, we fix tight bounds on the length of $\mathcal{B}$-certificates giving conclusions in Section 5.

2 Preliminaries

In this section, after introducing some notations we recall in Subsection 2.1 the logic HS [11] and the state-based model-checking framework for verifying HS formulas [16].

Let $\mathbb{N}$ be the set of natural numbers. For all $i, j \in \mathbb{N}$, with $i \leq j$, $[i, j]$ denotes the set of natural numbers h such that $i \leq h \leq j$. For all $n, h \in \mathbb{N}$, $\mathit{Tower}(n, h)$ denotes a tower of exponentials of height h and argument n: $\mathit{Tower}(n, 0) = n$ and $\mathit{Tower}(n, h+1) = 2^{\mathit{Tower}(n,h)}$. Let Σ be a finite alphabet. The set of all the finite words over Σ is denoted by Σ^*, and $\Sigma^+ := \Sigma^* \setminus \{\varepsilon\}$, where ε is the empty word. Let w be a finite word over Σ. We denote by $|w|$ the length of w. For all $i, j \in \mathbb{N}$, with $i \leq j$, $w(i)$ is the i-th letter of w ($w(0)$ is the first

Table 1 Allen's relations and corresponding HS modalities.

Allen relation	HS	Definition w.r.t. interval structures	Example
MEETS	$\langle A \rangle$	$[x,y]\mathcal{R}_A[v,z] \iff y = v$	
BEFORE	$\langle L \rangle$	$[x,y]\mathcal{R}_L[v,z] \iff y < v$	
STARTED-BY	$\langle B \rangle$	$[x,y]\mathcal{R}_B[v,z] \iff x = v \wedge z < y$	
FINISHED-BY	$\langle E \rangle$	$[x,y]\mathcal{R}_E[v,z] \iff y = z \wedge x < v$	
CONTAINS	$\langle D \rangle$	$[x,y]\mathcal{R}_D[v,z] \iff x < v \wedge z < y$	
OVERLAPS	$\langle O \rangle$	$[x,y]\mathcal{R}_O[v,z] \iff x < v < y < z$	

letter) while $w[i,j]$ denotes the infix of w given by $w(i)\cdots w(j)$. If $w \neq \varepsilon$, then we denote by $\mathsf{fst}(w)$ and $\mathsf{lst}(w)$ the first and last symbol of w, and by $\mathsf{internal}(w)$ the set of letters in Σ occurring in $w[1, n-1]$ where $|w| = n+1$. The concatenation of two finite words w and w' is denoted by $w \cdot w'$. Moreover, if $\mathsf{lst}(w) = \mathsf{fst}(w')$, $w \star w'$ represents $w[0, n-1] \cdot w'$, where $|w| = n+1$ ($\star$-concatenation). The set $\mathsf{Pref}(w)$ of *non-empty proper prefixes* of w is the set of non-empty finite words u such that $w = u \cdot v$ for some non-empty word v. The set $\mathsf{Suff}(w)$ of *non-empty proper suffixes* of w is the set of non-empty words u such that $w = v \cdot u$ for some non-empty finite word v. A *subword* (resp., *internal subword*) of w is a word w' such that w is of the form $w = u \cdot w' \cdot v$ for some words (resp., for some non-empty words) u and v.

2.1 The Interval Temporal Logic HS

An interval algebra to reason about intervals and their relative orders was proposed by Allen in [1], while a systematic logical study of interval representation and reasoning was done a few years later by Halpern and Shoham, who introduced the interval temporal logic HS featuring one modality for each Allen relation, but equality [11]. Table 1 depicts 6 of the 13 Allen's relations, together with the corresponding HS (existential) modalities. The other 7 relations are the 6 inverse relations (given a binary relation $\mathcal{R}$, the inverse relation $\overline{\mathcal{R}}$ is such that $b\overline{\mathcal{R}}a$ iff $a\mathcal{R}b$) and equality.

Let $\mathcal{AP}$ be a finite set of atomic propositions. HS formulas ψ over $\mathcal{AP}$ are defined as follows:

$$\psi ::= \top \mid \bot \mid p \mid \neg\psi \mid \psi \wedge \psi \mid \langle \mathrm{X} \rangle \psi$$

where $p \in \mathcal{AP}$ and $\langle \mathrm{X} \rangle$ is the existential temporal modality for the (non-trivial) Allen's relations $X \in \{A, L, B, E, D, O, \overline{A}, \overline{L}, \overline{B}, \overline{E}, \overline{D}, \overline{O}\}$. The size $|\psi|$ of a formula ψ is the number of distinct subformulas of ψ. We also exploit the standard logical connectives $\vee$ (disjunction) and $\rightarrow$ (implication) as abbreviations, and for any temporal modality $\langle \mathrm{X} \rangle$, the dual universal modality $[\mathrm{X}]$ defined as: $[\mathrm{X}]\,\psi := \neg\langle \mathrm{X} \rangle \neg\psi$. An HS formula ψ is in *positive normal form* (*PNF*) if negation is applied only to atomic formulas in $\mathcal{AP}$. By using De Morgan's laws and for any existential modality $\langle \mathrm{X} \rangle$, the dual universal modality $[\mathrm{X}]$, we can convert in linear-time an HS formula ψ into an equivalent formula in *PNF*, called the *PNF* of ψ. For a formula ψ in *PNF*, the *dual* $\widetilde{\psi}$ of ψ is the *PNF* of $\neg\psi$.

Given a set $U \subseteq \{A, L, B, E, D, O, \overline{A}, \overline{L}, \overline{B}, \overline{E}, \overline{D}, \overline{O}\}$ of Allen's relations, the *joint nesting depth of U in a formula* ψ denoted by $\mathrm{depth}_U(\psi)$ is defined as: (i) $\mathrm{depth}_U(p) = 0$, for any $p \in \mathcal{AP}$; (ii) $\mathrm{depth}_U(\neg\psi) = \mathrm{depth}_U(\psi)$; (iii) $\mathrm{depth}_U(\psi \wedge \varphi) = \max\{\mathrm{depth}_U(\psi), \mathrm{depth}_U(\varphi)\}$; (iv) $\mathrm{depth}_U(\langle \mathrm{X} \rangle \psi) = 1 + \mathrm{depth}_U(\psi)$ if $X \in U$, and $\mathrm{depth}_U(\langle \mathrm{X} \rangle \psi) = \mathrm{depth}_U(\psi)$ otherwise; (iv) $\mathrm{depth}_U([\mathrm{X}]\,\psi) = 1 + \mathrm{depth}_U(\psi)$ if $X \in U$, and $\mathrm{depth}_U([\mathrm{X}]\,\psi) = \mathrm{depth}_U(\psi)$ otherwise.

Given any subset of Allen's relations $\{X_1, .., X_n\}$, we denote by $\mathsf{X_1} \cdots \mathsf{X_n}$ the HS fragment featuring existential (and universal) modalities for $X_1, .., X_n$ only.

We assume the *non-strict semantics of HS*, which admits intervals consisting of a single point (all the results proved in the paper hold for the strict semantics as well). Under such an assumption, all HS-temporal modalities can be expressed in terms of $\langle \mathrm{B} \rangle, \langle \mathrm{E} \rangle, \langle \overline{\mathrm{B}} \rangle$, and $\langle \overline{\mathrm{E}} \rangle$ [25]. HS can thus be regarded as a multi-modal logic with $\langle \mathrm{B} \rangle, \langle \mathrm{E} \rangle, \langle \overline{\mathrm{B}} \rangle$, and $\langle \overline{\mathrm{E}} \rangle$ as primitive modalities and its semantics can be defined over a multi-modal Kripke structure, called *abstract interval model* (AIM for short), where intervals are treated as atomic objects and Allen's relations as binary relations over intervals.

▶ **Definition 1** (Abstract interval models [16]). *An* abstract interval model *(AIM) over $\mathcal{AP}$ is a tuple $\mathcal{A} = (\mathcal{AP}, \mathbb{I}, B_{\mathbb{I}}, E_{\mathbb{I}}, Lab_{\mathbb{I}})$, where $\mathbb{I}$ is a possibly infinite set of worlds (abstract intervals), $B_{\mathbb{I}}$ and $E_{\mathbb{I}}$ are two binary relations over $\mathbb{I}$, and $Lab_{\mathbb{I}} : \mathbb{I} \mapsto 2^{\mathcal{AP}}$ is a labeling function, which assigns a set of proposition letters from $\mathcal{AP}$ to each abstract interval.*

In the interval setting, $\mathbb{I}$ is interpreted as a set of intervals and $B_{\mathbb{I}}$ and $E_{\mathbb{I}}$ as Allen's relations B (*started-by*) and E (*finished-by*), respectively; $Lab_{\mathbb{I}}$ assigns to each interval in $\mathbb{I}$ the set of atomic propositions that hold over it. Given an interval $I \in \mathbb{I}$, the truth of an HS formula over I is inductively defined as follows (the Boolean connectives are treated as usual):

- $\mathcal{A}, I \models p$ if $p \in Lab_{\mathbb{I}}(I)$, for any $p \in \mathcal{AP}$;
- $\mathcal{A}, I \models \langle \mathrm{X} \rangle \psi$, for $X \in \{B, E\}$, if $I \, X_{\mathbb{I}} \, J$ and $\mathcal{A}, J \models \psi$ for some $J \in \mathbb{I}$;
- $\mathcal{A}, I \models \langle \overline{\mathrm{X}} \rangle \psi$, for $\overline{X} \in \{\overline{B}, \overline{E}\}$, if $J \, X_{\mathbb{I}} \, I$ and $\mathcal{A}, J \models \psi$ for some $J \in \mathbb{I}$.

As an example, $\langle \mathrm{D} \rangle$ can be expressed in terms of $\langle \mathrm{B} \rangle$ and $\langle \mathrm{E} \rangle$ as $\langle \mathrm{D} \rangle \, \psi := \langle \mathrm{B} \rangle \, \langle \mathrm{E} \rangle \, \psi$, while $\langle \mathrm{A} \rangle$ can be expressed in terms of $\langle \mathrm{E} \rangle$ and $\langle \overline{\mathrm{B}} \rangle$ as $\langle \mathrm{A} \rangle \, \psi := ([\mathrm{E}] \, \bot \wedge (\psi \vee \langle \overline{\mathrm{B}} \rangle \, \psi)) \vee \langle \mathrm{E} \rangle ([\mathrm{E}] \, \bot \wedge (\psi \vee \langle \overline{\mathrm{B}} \rangle \, \psi))$.

State-based model-checking against HS. In the context of MC, finite state systems are usually modelled as finite Kripke structures over a finite set $\mathcal{AP}$ of atomic propositions which represent predicates over the states of the system.

▶ **Definition 2.** *A* Kripke structure *over $\mathcal{AP}$ is a tuple $\mathcal{K} = (\mathcal{AP}, S, E, Lab, s_0)$, where S is a set of states, $E \subseteq S \times S$ is a transition relation, $Lab : S \mapsto 2^{\mathcal{AP}}$ is a labelling function assigning to each state s the set of propositions that hold over it, and $s_0 \in S$ is the initial state. We say that $\mathcal{K}$ is finite if S is finite.*

Let $\mathcal{K} = (\mathcal{AP}, S, E, Lab, s_0)$ be a Kripke structure. A path π of $\mathcal{K}$ is a non-empty finite word over S such that for all $0 \leq i < |\pi|$, $(\pi(i), \pi(i+1)) \in E$. A *sub-path* (resp., *internal sub-path*) of π is a path of $\mathcal{K}$ which is a subword (resp., internal subword) of π. A path is *initial* if it starts from the initial state of $\mathcal{K}$.

We now recall the state-based approach [16] for model checking Kripke structures against HS formulas which consists in defining a mapping from a Kripke structure $\mathcal{K}$ to an AIM $\mathcal{A}_{\mathcal{K}}$, where the abstract intervals correspond to the paths of the Kripke structure, the relations $B_{\mathbb{I}}$ and $E_{\mathbb{I}}$ of $\mathcal{A}_{\mathcal{K}}$ are interpreted as the Allen's relations B and E over the set of $\mathcal{K}$-paths, respectively, and the following assumption is adopted: a proposition holds over an interval if and only if it holds over all its subintervals (*homogeneity principle*).

▶ **Definition 3.** *Let $\mathcal{K} = (\mathcal{AP}, S, E, Lab, s_0)$ be a Kripke structure. The AIM induced by $\mathcal{K}$ is $\mathcal{A}_{\mathcal{K}} = (\mathcal{AP}, \mathbb{I}, B_{\mathbb{I}}, E_{\mathbb{I}}, Lab_{\mathbb{I}})$, where $\mathbb{I}$ is the set of paths of $\mathcal{K}$, and:*

- $B_{\mathbb{I}} = \{(\pi, \pi') \in \mathbb{I} \times \mathbb{I} \mid \pi' \in \mathsf{Pref}(\pi)\}$, $E_{\mathbb{I}} = \{(\pi, \pi') \in \mathbb{I} \times \mathbb{I} \mid \pi' \in \mathsf{Suff}(\pi)\}$, *and*
- *for all $p \in \mathcal{AP}$, $Lab_{\mathbb{I}}^{-1}(p) = \{\pi \in \mathbb{I} \mid p \in \bigcap_{i=0}^{i<|\pi|} Lab(\pi(i))\}$.*

Note that for a finite Kripke structure $\mathcal{K}$, the number of paths in $\mathcal{K}$ may be infinite (this happens when $\mathcal{K}$ has loops), hence the number of intervals in $\mathcal{A}_\mathcal{K}$ may be infinite. A Kripke structure $\mathcal{K}$ over $\mathcal{AP}$ is a *model* of an HS formula ψ over $\mathcal{AP}$, written $\mathcal{K} \models \psi$, if for all *initial* paths π of $\mathcal{K}$, $\mathcal{A}_\mathcal{K}, \pi \models \psi$. In the following, we also write $\mathcal{K}, \pi \models \psi$ to mean $\mathcal{A}_\mathcal{K}, \pi \models \psi$. The (finite) model-checking problem (against HS) consists in checking whether $\mathcal{K} \models \psi$ for a given HS formula ψ and a finite Kripke structure $\mathcal{K}$.

We observe that the temporal modalities for the Allens's relations in $\{B, E, D\}$ have a "linear-time" semantics, i.e., they allow to select only slices (subpaths) of the current timeline (path). The semantics of the temporal modalities associated with the other Allen's relations (i.e., the ones in $\{A, L, O, \overline{A}, \overline{L}, \overline{B}, \overline{E}, \overline{D}, \overline{O}\}$) is instead "branching-time" (i.e., they allow to non-deterministically extend the current timeline in the future or in the past). Accordingly, a non-empty subset of non-interdefinable Allen's relations in $\{B, E, D\}$ is called a *linear-time basis* $\mathcal{B}$ of HS. Hence, the possible bases are $\{B\}$, $\{E\}$, $\{D\}$, $\{B, D\}$, $\{B, E\}$, and $\{E, D\}$.

3 Decision procedures based on descriptors

In this section, we provide a uniform framework for model-checking finite Kripke structures against the HS syntactical fragments, denoted by $\mathsf{HS}_\mathcal{B}(\mathcal{F})$, obtained by combining the modalities of a linear-time basis $\mathcal{B}$ of HS distinct from $\{D\}$ with the branching-time modalities for the Allen's relations in $\mathcal{F} = \{A, \overline{A}, \overline{B}, \overline{E}, L, \overline{L}\}$. Note that for the complete basis $\{B, E\}$, we obtain the full logic HS. Moreover, the Allen relation $\overline{D}$ can be expressed in terms of $\overline{B}$,$\overline{E}$: $\langle \overline{\mathrm{D}} \rangle \psi := \langle \overline{\mathrm{B}} \rangle \langle \overline{\mathrm{E}} \rangle \psi$. For the remaining branching-time modalities, i.e., the ones associated with O and $\overline{O}$, we have that $\langle \mathrm{O} \rangle$ (resp., $\langle \overline{\mathrm{O}} \rangle$) can be expressed in terms of $\langle \overline{\mathrm{B}} \rangle$ and $\langle \mathrm{E} \rangle$ (resp., $\langle \overline{\mathrm{E}} \rangle$ and $\langle \mathrm{B} \rangle$): $\langle \mathrm{O} \rangle \psi := \langle \mathrm{E} \rangle (\langle \mathrm{E} \rangle \top \wedge \langle \overline{\mathrm{B}} \rangle \psi)$ (resp., $\langle \overline{\mathrm{O}} \rangle \psi := \langle \mathrm{B} \rangle (\langle \mathrm{B} \rangle \top \wedge \langle \overline{\mathrm{E}} \rangle \psi)$). As an example, $\mathsf{HS}_{\{B\}}(\mathcal{F})$ corresponds to $\mathsf{A\overline{A}B\overline{B}D\overline{E}L\overline{L}O}$.

The proposed approach is a generalization of the one provided in [16], where only the full logic HS is considered. In particular, given a finite Kripke structure $\mathcal{K}$, for each linear-time basis $\mathcal{B}$ of HS, we define a finite abstraction of the set of $\mathcal{K}$-paths parameterized by a natural number h. This abstraction induces in turn a *finite* abstract interval model, which, in case $\mathcal{B} \neq \{D\}$, is equivalent to $\mathcal{A}_\mathcal{K}$ with respect to the fulfillment of all the formulas in $\mathsf{HS}_\mathcal{B}(\mathcal{F})$ having joint $\mathcal{B}$-nesting depth at most h. This allows us to provide in Subsection 3.1 an *alternating* algorithm, parameterized in the basis $\mathcal{B} \neq \{D\}$, for model-checking the fragment $\mathsf{HS}_\mathcal{B}(\mathcal{F})$, which given a finite Kripke structure $\mathcal{K}$ and a formula ψ, runs in time bounded by the size of the finite abstraction for the basis $\mathcal{B}$, the Kripke structure $\mathcal{K}$ and the parameter $h = \mathrm{depth}_\mathcal{B}(\psi)$, and whose number of alternations between existential and universal choices is at most $O(|\psi|)$. For each basis $\mathcal{B}$, we define in the following the notion of $\mathcal{B}$-*descriptor* which allows to construct the above mentioned finite abstraction. The definition of $\mathcal{B}$-*descriptors* exploits *h-level* Σ-*terms* and *h-level bipartite* Σ-*terms*, where Σ denotes a given finite alphabet and h a natural number. Intuitively, an h-level Σ-term corresponds to an unordered finite tree of height h such that subtrees rooted at distinct children of the same node are *not* isomorphic. An h-level bipartite Σ-term is similar but additionally we require that each edge in the tree has a color from a set of two colors. Formally, the set of h-level Σ-terms t is inductively defined as follows:

- if $h = 0$, then $t = a$ for some $a \in \Sigma$; otherwise, t has the form (a, T) where $a \in \Sigma$ and T is a (possibly empty) subset of $(h-1)$-level Σ-terms.

The set of h-level bipartite Σ-terms t is inductively defined as follows:

- if $h = 0$, then $t = a$ for some $a \in \Sigma$; otherwise t is of the form (a, T_1, T_2) where $a \in \Sigma$ and T_1 and T_2 are (possibly empty) subsets of $(h-1)$-level Σ-terms.

We say that a is the root of t. The size of an h-level (bipartite) Σ-term is the number of nodes in the associated tree representation. The following holds.

▶ Remark 4. The number of distinct h-level Σ-terms (resp., h-level bipartite Σ-terms) over Σ is $Tower(\Theta(|\Sigma|), h)$.

▶ **Definition 5** (Descriptors). *Let Σ be a finite alphabet and $\mathcal{B}$ be a linear-time basis of HS. Given a non-empty finite word w over Σ and $h \geq 0$, the h-level $\mathcal{B}$-descriptor $\mathcal{B}_h(w)$ of w is the h-level $(\Sigma \times 2^{\Sigma} \times \Sigma)$-term (resp., h-level bipartite $(\Sigma \times 2^{\Sigma} \times \Sigma)$-term) if $|\mathcal{B}| = 1$ (resp., $|\mathcal{B}| = 2$) inductively satisfying the following conditions. For the base case, i.e. $h = 0$, then $\mathcal{B}_0(w) = (\mathsf{fst}(w), \mathsf{internal}(w), \mathsf{lst}(w))$. For the induction step, i.e. $h > 0$, we have:*

- *Case $\mathcal{B} = \{B\}$ (resp., $\mathcal{B} = \{E\}$, resp., $\mathcal{B} = \{D\}$): $\mathcal{B}_h(w) = (\mathcal{B}_0(w), T)$ where T is the set of $(h-1)$-level $\mathcal{B}$-descriptors of the non-empty proper prefixes (resp., non-empty proper suffixes, resp., non-empty internal subwords) of w.*
- *Case $\mathcal{B} = \{B, E\}$: $\mathcal{B}_h(w) = (\mathcal{B}_0(w), T_B, T_E)$ with T_B (resp., T_E) the set of $(h-1)$-level $\mathcal{B}$-descriptors of the non-empty proper prefixes (resp., non-empty proper suffixes) of w.*
- *Case $\mathcal{B} = \{B, D\}$ (resp., $\mathcal{B} = \{E, D\}$): as in case $\mathcal{B} = \{B, E\}$ by replacing T_E (resp., T_B) with the set of $(h-1)$-level $\mathcal{B}$-descriptors of the non-empty internal subwords of w.*

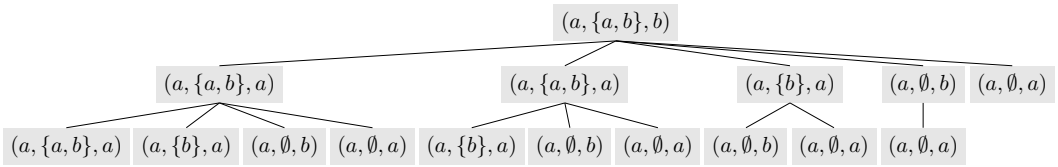


Figure 2 The 2-level $\{B\}$-descriptor for the word $w = abaaaab$ over $\Sigma = \{a, b\}$.

An example of 2-level $\{B\}$-descriptor is depicted in Figure 2. Intuitively, in case $\mathcal{B} \neq \{D\}$, the h-level $\mathcal{B}$-descriptor $\mathcal{B}_h(\pi)$ of a Kripke structure path π has enough information for checking the fulfillment of $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ formulas with joint $\mathcal{B}$-nesting depth at most h:

- for checking the fulfillment of proposition letters, $\mathcal{B}_h(\pi)$ keeps tracks at each node of the set of states visited by the *current subpath* of π;
- to deal with the branching-time modalities for the Allen's relations in $\mathcal{F}$, $\mathcal{B}_h(\pi)$ keeps tracks at each node also of the first and last states of the current subpath;
- finally, for checking the fulfillment of the linear-time modalities for the basis $\mathcal{B}$, $\mathcal{B}_h(\pi)$ keeps information about all the subpaths of the current subpath π' which can be obtained from π' by applying the Allen's relations in the basis $\mathcal{B}$.

For a basis $\mathcal{B} = \{X, Y\}$ (resp., $\mathcal{B} = \{X\}$), an h-level $\mathcal{B}$-descriptor is also called h-level XY-descriptor (resp., h-level X-descriptor) and for a non-empty word, we write $XY_h(w)$ (resp., $X_h(w)$) to mean $\mathcal{B}_h(w)$. For a finite Kripke structure $\mathcal{K}$, a basis $\mathcal{B}$, and $h \geq 0$, we denote by $\mathcal{B}_h(\mathcal{K})$ the finite set of h-level $\mathcal{B}$-descriptors associated with the paths of $\mathcal{K}$.

In the following we show that paths of $\mathcal{K}$ which have the same h-level $\mathcal{B}$-descriptor with $\mathcal{B} \neq \{D\}$ satisfy the same formulas in $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ whose joint $\mathcal{B}$-nesting depth is at most h. As a preliminary step, we show that the property of two paths π and π' to have the same h-level $\mathcal{B}$-descriptor is preserved by right (resp., left) $\star$-concatenation with another path of $\mathcal{K}$. This result is used for handling the branching-time modalities $\langle \overline{\mathrm{B}} \rangle$ and $\langle \overline{\mathrm{E}} \rangle$.

▶ **Proposition 6.** *Let $h \geq 0$, $\mathcal{B} \neq \{D\}$ a basis, and π and π' be two paths of a finite Kripke structure $\mathcal{K}$ having the same h-level $\mathcal{B}$-descriptor. Then, for all paths π_L and π_R of $\mathcal{K}$ such that $\pi_L \star \pi$ and $\pi \star \pi_R$ are defined, the following holds:*
(1) $\mathcal{B}_h(\pi \star \pi_R) = \mathcal{B}_h(\pi' \star \pi_R)$ and (2) $\mathcal{B}_h(\pi_L \star \pi) = \mathcal{B}_h(\pi_L \star \pi')$.

We note that Proposition 6 does not hold in general for the basis $\mathcal{B} = \{D\}$. As an example, let us consider a Kripke structure $\mathcal{K}$ consisting of three states s_1, s_2, and s_3 such that (s_i, s_j) is an edge of $\mathcal{K}$ for all $1 \leq i, j \leq 3$. Let us consider the two paths $\pi = s_1(s_2s_3)^3s_1$ and $\pi' = s_1(s_3s_2)^3s_1$. One can check that π and π' have the same 1-level D-descriptor. On the other hand, $\pi \cdot s_1$ and $\pi' \cdot s_1$ have distinct 1-level D-descriptors: in particular, while $\pi \cdot s_1$ has the internal subword s_3s_1, there is no internal subword ν' of $\pi' \cdot s_1$ such that $\mathsf{fst}(\nu') = s_3$, $\mathsf{lst}(\nu') = s_1$, and $\mathsf{internal}(\nu') = \emptyset$. By Proposition 6, we can obtain the following result.

▶ **Proposition 7.** *Let $h \geq 0$, $\mathcal{B} \neq \{D\}$ a basis, and π and π' be two paths of a finite Kripke structure $\mathcal{K}$ having the same h-level $\mathcal{B}$-descriptor. Then, for each $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ formula ψ with $\mathrm{depth}_{\mathcal{B}}(\psi) \leq h$, it holds that $\mathcal{K}, \pi \models \psi$ iff $\mathcal{K}, \pi' \models \psi$.*

By Proposition 6, for each basis $\mathcal{B} \neq \{D\}$, we can also state *a bounded path property* which intuitively provides a bounded witness for each $\mathcal{B}_h$-descriptor associated with an arbitrary path of a finite Kripke structure. The bounded path property will be crucial in Subsection 3.1 to design the MC algorithm for the logic $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$.

▶ **Proposition 8** (Bounded Path Property)**.** *Let $\mathcal{B} \neq \{D\}$ be a basis, $\mathcal{K}$ a finite Kripke structure, $h \geq 0$ and π a $\mathcal{K}$-path. Then, there exists a path π' having the same h-level $\mathcal{B}$-descriptor of π and whose length is bounded by $|\mathcal{B}_h(\mathcal{K})|$ (i.e., the number of distinct h-level $\mathcal{B}$-descriptors of the $\mathcal{K}$-paths).*

Proof. Let $|\pi| = n$. Since there are n distinct non-empty prefixes of π, if $n > |\mathcal{B}_h(\mathcal{K})|$, then π can be written in the form $\pi = \nu \cdot \nu' \cdot \nu''$ such that $|\nu| > 0$, $|\nu'| > 0$, and ν and $\nu \cdot \nu'$ have the same h-level $\mathcal{B}$-descriptor. By Proposition 6, the strictly smaller path $\nu \cdot \nu''$ has the same h-level $\mathcal{B}$-descriptor as π. We can iterate such a contraction process until there are no more pairs of prefixes with the same h-level $\mathcal{B}$-descriptor proving the statement. ◀

By Propositions 7 and 8 we can define bounded minimal representatives ($\mathcal{B}$-certificates) of paths used in the MC algorithm defined in the next section.

▶ **Definition 9** ($\mathcal{B}$-certificate)**.** *Given a basis $\mathcal{B} \neq \{D\}$, a finite Kripke structure $\mathcal{K}$, and $h \geq 0$, an* h-level $\mathcal{B}$-certificate of $\mathcal{K}$ *is a path π of $\mathcal{K}$ such that there is no path π' so that $|\pi'| < |\pi|$ and π and π' have the same h-level $\mathcal{B}$-descriptor. Given an $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ formula φ, a* $\mathcal{B}$-certificate for $(\mathcal{K}, \varphi)$ *is an h-level $\mathcal{B}$-certificate of $\mathcal{K}$ where $h = \mathrm{depth}_{\mathcal{B}}(\varphi)$.*

By Proposition 8 an upper bound on the length of $\mathcal{B}$-certificates for $(\mathcal{K}, \varphi)$ is $|\mathcal{B}_h(\mathcal{K})|$ with $h = \mathrm{depth}_{\mathcal{B}}(\varphi)$.

3.1 Algorithm for model-checking the logics $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$

In this section, by exploiting Propositions 6–8, for each basis $\mathcal{B} \neq \{D\}$, we provide an alternating MC algorithm for the logic $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ (recall that $\mathcal{F} = \{A, \overline{A}, \overline{B}, \overline{E}, L, \overline{L}\}$). We assume that $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ formulas are in *PNF*. As complexity measures of a formula φ, we consider the size $|\varphi|$ and the standard *alternation depth*, denoted by $\Upsilon(\varphi)$, between the existential $\langle \mathrm{X} \rangle$ and universal modalities $[\mathrm{X}]$ occurring in the *PNF* of φ for $X \in \{\overline{B}, \overline{E}\}$. Formally, we establish the following result, where $\mathsf{MC}_{\mathcal{B}}$ is the set of pairs $(\mathcal{K}, \varphi)$ consisting of a finite Kripke structure $\mathcal{K}$ and a $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ formula φ such that $\mathcal{K} \models \varphi$.

▶ **Theorem 10.** *For each basis $\mathcal{B} \neq \{D\}$, one can construct a time-bounded Alternating Turing Machine (ATM) accepting $\mathsf{MC}_{\mathcal{B}}$ which, given an input $(\mathcal{K}, \varphi)$, has a number of alternations (between existentially and universal choices) at most $\Upsilon(\varphi) + 2$ and runs in time $M_{\mathcal{B}}(\mathcal{K}, \varphi)^{O(|\varphi|^d)}$, where $M_{\mathcal{B}}(\mathcal{K}, \varphi)$ is the maximal length of a $\mathcal{B}$-certificate for the input, and $d = 2$ if $D \in \mathcal{B}$ and $d = 1$ otherwise.*

check$_{\mathcal{B}}(\mathcal{K},\varphi)$ [$\mathcal{K}$ *is a finite Kripke structure and* φ *is an* $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ *formula in PNF*]

---

existentially choose an $\mathsf{A\overline{A}L\overline{L}}$-labeling $\mathcal{L}$ for $(\mathcal{K},\varphi)$;
for each state s and $\psi \in \mathcal{L}(s)$ **do**
  case $\psi = \langle \mathrm{A}\rangle\,\psi'$ (resp., $\psi = [\mathrm{A}]\,\psi'$): **existentially** (resp., **universally**) **choose** a $\mathcal{B}$-certificate π with $\mathsf{fst}(\pi) = s$ and **call** *checkTrue*$_{\mathcal{B}}(\mathcal{K},\varphi,\mathcal{L},\{(\psi',\pi)\})$;
  case $\psi = \langle \overline{\mathrm{A}}\rangle\,\psi'$ (resp., $\psi = [\overline{\mathrm{A}}]\,\psi'$): **existentially** (resp., **universally**) **choose** a $\mathcal{B}$-certificate π with $\mathsf{lst}(\pi) = s$ and **call** *checkTrue*$_{\mathcal{B}}(\mathcal{K},\varphi,\mathcal{L},\{(\psi',\pi)\})$;
  case $\psi = \langle \mathrm{L}\rangle\,\psi'$ (resp., $\psi = [\mathrm{L}]\,\psi'$): **existentially** (resp., **universally**) **choose** state s' s.t. $s \rightarrow_{\mathcal{K}}^{+} s'$ and a $\mathcal{B}$-certificate π with $\mathsf{fst}(\pi) = s'$ and **call** *checkTrue*$_{\mathcal{B}}(\mathcal{K},\varphi,\mathcal{L},\{(\psi',\pi)\})$;
  case $\psi = \langle \overline{\mathrm{L}}\rangle\,\psi'$ (resp., $\psi = [\overline{\mathrm{L}}]\,\psi'$): **existentially** (resp., **universally**) **choose** state s' s.t. $s' \rightarrow_{\mathcal{K}}^{+} s$ and a $\mathcal{B}$-certificate π with $\mathsf{lst}(\pi) = s'$ and **call** *checkTrue*$_{\mathcal{B}}(\mathcal{K},\varphi,\mathcal{L},\{(\psi',\pi)\})$;
universally choose a certificate π for $(\mathcal{K},\varphi)$ with $\mathsf{fst}(\pi) = s_0$ (s_0 is the initial state of $\mathcal{K}$)
  and **call** *checkTrue*$_{\mathcal{B}}(\mathcal{K},\varphi,\mathcal{L},\{(\varphi,\pi)\})$;

Figure 3 Procedure *check*$_{\mathcal{B}}$ for a linear-time basis $\mathcal{B} \neq \{D\}$.

To prove the assertion of Theorem 10 we define a procedure, parametric in the basis $\mathcal{B} \neq \{D\}$, which can be easily translated into an ATM. To this end, we introduce some auxiliary notation. Let us fix a finite Kripke structure $\mathcal{K}$ and an $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ formula φ in *PNF*. For two states s and s', we write $s \rightarrow_{\mathcal{K}}^{+} s'$ to mean that s' is reachable from s by a path of length at least 2. Let π be a $\mathcal{B}$-certificate for $(\mathcal{K},\varphi)$ and $h = \mathrm{depth}_{\mathcal{B}}(\varphi)$. For each $X \in \mathcal{B}$, an X-witness of π is a non-empty proper prefix (resp., non-empty proper suffix, resp., non-empty internal subpath) of π if $X = B$ (resp., $X = E$, resp., $X = D$). A $\overline{B}$-*witness* (resp., $\overline{E}$-*witness*) of π for $(\mathcal{K},\varphi)$, is a $\mathcal{B}$-certificate π' of $(\mathcal{K},\varphi)$ such that π' has the same h-level $\mathcal{B}$ descriptor of a path of the form $\pi \star \pi''$ (resp., $\pi'' \star \pi$) for some $\mathcal{B}$-*certificate* π'' of $(\mathcal{K},\varphi)$ with $|\pi''| > 1$. By $\mathsf{SD}(\varphi)$ we denote the set consisting of the subformulas ψ of φ and the *duals* $\widetilde{\psi}$. By Propositions 6–8, we easily deduce the following property.

▶ **Proposition 11.** *Let* $\mathcal{B} \neq \{D\}$ *be a basis,* φ *an* $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ *formula in PNF,* $\mathcal{K}$ *a finite Kripke structure, and* π *a* $\mathcal{B}$*-certificate for* $(\mathcal{K},\varphi)$*. Then, for each* $\langle \mathrm{X}\rangle\,\psi \in \mathsf{SD}(\varphi)$ *with* $X \in \{\overline{B},\overline{E}\}$, $\mathcal{K},\pi \models \langle \mathrm{X}\rangle\,\psi$ *iff there is an* X*-witness* π' *of* π *for* $(\mathcal{K},\varphi)$ *such that* $\mathcal{K},\pi' \models \psi$.

The set $\mathsf{A\overline{A}L\overline{L}}(\varphi)$ is the set of formulas in $\mathsf{SD}(\varphi)$ of the form $\langle \mathrm{X}\rangle\,\psi'$ or $[\mathrm{X}]\,\psi'$ with $X \in \{A,\overline{A},L,\overline{L}\}$. An $\mathsf{A\overline{A}L\overline{L}}$-labeling $\mathcal{L}$ for $(\mathcal{K},\varphi)$ is a mapping associating with each state s of $\mathcal{K}$ a maximally consistent set of subformulas of $\mathsf{A\overline{A}L\overline{L}}(\varphi)$. More precisely, for all $s \in S$, $\mathcal{L}(s)$ is such that for all $\psi, \widetilde{\psi} \in \mathsf{A\overline{A}L\overline{L}}(\varphi)$, $\mathcal{L}(s) \cap \{\psi,\widetilde{\psi}\}$ is a singleton. $\mathcal{L}$ is *valid* if for all states $s \in S$ and $\psi \in \mathcal{L}(s)$, $\mathcal{K},s \models \psi$ (we consider s as a length-1 path). Finally, a *well-formed set for* $(\mathcal{K},\varphi)$ is a finite set $\mathcal{W}$ consisting of pairs (ψ,π) such that $\psi \in \mathsf{SD}(\varphi)$ and π is a $\mathcal{B}$-certificate of $(\mathcal{K},\varphi)$. $\mathcal{W}$ is said *universal* if each formula occurring in $\mathcal{W}$ is of the form $[\mathrm{X}]\,\psi$ with $X \in \{\overline{B},\overline{E}\}$. The *dual* $\widetilde{\mathcal{W}}$ of $\mathcal{W}$ is the well-formed set obtained by replacing each pair $(\psi,\pi) \in \mathcal{W}$ with $(\widetilde{\psi},\pi)$. A well-formed set $\mathcal{W}$ is *valid* if for each $(\psi,\pi) \in \mathcal{W}$, $\mathcal{K},\pi \models \psi$.

The procedure *check*$_{\mathcal{B}}$ in Figure 3 defines the ATM required to prove the assertion of Theorem 10 for a basis $\mathcal{B} \neq \{D\}$. It takes a pair $(\mathcal{K},\varphi)$ as input, where φ is an $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ formula, and: (1) it guesses an $\mathsf{A\overline{A}L\overline{L}}$-labeling $\mathcal{L}$ for $(\mathcal{K},\varphi)$; (2) it checks that the guessed labeling $\mathcal{L}$ is valid; (3) for every $\mathcal{B}$-certificate π of $(\mathcal{K},\varphi)$ starting from the initial state, it checks that $\mathcal{K},\pi \models \varphi$. To perform steps (2)–(3), it exploits the auxiliary ATM procedure *checkTrue*$_{\mathcal{B}}$ reported in Figure 4. The procedure *checkTrue*$_{\mathcal{B}}$ takes as input a well-formed set $\mathcal{W}$ for $(\mathcal{K},\varphi)$ and, assuming that the current $\mathsf{A\overline{A}L\overline{L}}$-labeling $\mathcal{L}$ is valid, checks whether $\mathcal{W}$ is valid. For each pair $(\psi,\pi) \in \mathcal{W}$ such that ψ is not of the form $[\mathrm{X}]\,\psi'$ with $X \in \{\overline{B},\overline{E}\}$,

checkTrue$_{\mathcal{B}}(\mathcal{K}, \varphi, \mathcal{L}, \mathcal{W})$ [$\mathcal{W}$ *is a well-formed set and* $\mathcal{L}$ *is an* $\mathsf{A\overline{A}L\overline{L}}$*-labeling for* $(\mathcal{K}, \Phi)$]

---

while $\mathcal{W}$ is *not* universal **do**
  deterministically select $(\psi, \pi) \in \mathcal{W}$ such that ψ is not of the form $[\overline{E}]\psi'$ and $[\overline{B}]\psi'$
  update $\mathcal{W} \leftarrow \mathcal{W} \setminus \{(\psi, \pi)\}$;
  case $\psi = p$ (resp., $\psi = \neg p$) with $p \in \mathcal{AP}$: **if** $\mathcal{K}, \pi \not\models p$ (resp., $\mathcal{K}, \pi \not\models \neg p$) **then** *reject*;
  case $\psi = \langle \mathrm{X} \rangle \psi'$ or $\psi = [\mathrm{X}]\psi'$ with $X \in \{A, L\}$: **if** $\psi \notin \mathcal{L}(\mathsf{lst}(\pi))$ **then** *reject*;
  case $\psi = \langle \mathrm{X} \rangle \psi'$ or $\psi = [\mathrm{X}]\psi'$ with $X \in \{\overline{A}, \overline{L}\}$: **if** $\psi \notin \mathcal{L}(\mathsf{fst}(\pi))$ **then** *reject*;
  case $\psi = \psi_1 \vee \psi_2$: **existentially choose** $i = 1, 2$, **update** $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi_i, \pi)\}$;
  case $\psi = \psi_1 \wedge \psi_2$: **update** $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi_1, \pi), (\psi_2, \pi)\}$;
  case $\psi = [\mathrm{X}]\psi'$ with $X \in \mathcal{B}$: **update** $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi', \pi') \mid \pi'$ is an X*-witness* of $\pi\}$;
  case $\psi = \langle \mathrm{X} \rangle \psi'$ with $X \in \mathcal{B} \cup \{\overline{E}, \overline{B}\}$: **existentially choose** an X*-witness* π' of π for $(\mathcal{K}, \varphi)$, **update** $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi', \pi')\}$;
end while
if $\mathcal{W} = \emptyset$ **then** *accept*
else universally choose $(\psi, \pi) \in \widetilde{\mathcal{W}}$ and **call** *checkFalse*$_{\mathcal{B}}(\mathcal{K}, \varphi, \mathcal{L}, \{(\psi, \rho)\})$

Figure 4 Procedure *checkTrue*$_{\mathcal{B}}$ for a linear-time basis $\mathcal{B} \neq \{D\}$.

checkTrue$_{\mathcal{B}}$ directly checks whether $\mathcal{K}, \pi \models \psi$. In order to allow a deterministic choice of the current element of the iteration, we assume that the set $\mathcal{W}$ is implemented as an ordered data structure. At each iteration of the while loop in *checkTrue*$_{\mathcal{B}}$, the current pair $(\psi, \pi) \in \mathcal{W}$ is processed according to the semantics of HS, exploiting the guessed $\mathsf{A\overline{A}L\overline{L}}$-labeling $\mathcal{L}$ and Proposition 11. The processing is either deterministic or based on an existential choice, and the currently processed pair (ψ, π) is either removed from $\mathcal{W}$, or replaced with pairs (ψ', π') such that ψ' is a strict subformula of ψ.

At the end of the while loop, the resulting well formed set $\mathcal{W}$ is either empty or universal. In the former case, the procedure accepts. In the latter case, there is a switch in the current operation mode. For each element (ψ, π) in the dual of $\mathcal{W}$ (note that the root modality of ψ is either $\langle \overline{\mathrm{E}} \rangle$ or $\langle \overline{\mathrm{B}} \rangle$), the auxiliary ATM procedure *checkFalse*$_{\mathcal{B}}$ is invoked, which accepts the input $\{(\psi, \pi)\}$ iff $\mathcal{K}, \pi \not\models \psi$. The procedure *checkFalse*$_{\mathcal{B}}$ is the *dual* of *checkTrue*$_{\mathcal{B}}$: it is simply obtained from *checkTrue*$_{\mathcal{B}}$ by switching *accept* and *reject*, by switching existential choices and universal choices, and by converting the last call to *checkFalse*$_{\mathcal{B}}$ into *checkTrue*$_{\mathcal{B}}$. Thus *checkFalse*$_{\mathcal{B}}$ accepts an input $\mathcal{W}$ iff $\mathcal{W}$ is *not* valid.

Note that the number of alternations of the ATM *check*$_{\mathcal{B}}$ between existential and universal choices is the number of switches between the calls to the procedures *checkTrue*$_{\mathcal{B}}$ and *checkFalse*$_{\mathcal{B}}$ plus two. The correctness of the algorithm follows from Propositions 7, 8 and 11.

4 Tight bounds on the length of certificates

In this section, for each basis $\mathcal{B}$ (except $\{D\}$), we provide tight bounds on the length of h-level $\mathcal{B}$-certificates. For the bases $\{B\}$ and $\{E\}$ (see Subsection 4.1), we prove singly exponential upper bounds and matching lower bounds. By Theorem 10, we deduce that model-checking the logics $\mathsf{HS}_{\{B\}}(\mathcal{F})$ and $\mathsf{HS}_{\{E\}}(\mathcal{F})$ is in the complexity class $\mathbf{AEXP_{pol}}$ of problems decided by exponential-time bounded alternating Turing Machines with a polynomially bounded number of alternations. On the other hand, for all bases $\mathcal{B}$ distinct from $\{B\}$ and $\{E\}$, we state a non-elementary lower bound (see Subsection 4.2). In particular, the result obtained for the basis $\{B, E\}$ negatively answers a question left open in [16] regarding the possibility of fixing an elementary upper bound on the size of BE-descriptors.

4.1 Tight bounds on the length of B-certificates and E-certificates

In this section, we provide exponential upper bounds and exponential lower bounds on the length of h-level B-certificates and E-certificates of a finite Kripke structure.

▶ **Theorem 12.** *The following holds:*

- Upper bound: *let $\mathcal{K}$ be a finite Kripke structure with set of states S and $h \geq 0$. Then, each h-level B-certificate (resp., h-level E-certificate) has length at most $|S|^{2h+2}$.*
- Lower bound: *there is a family $\{\mathcal{K}_n\}_{n\geq 1}$ of finite Kripke structures such that for all $n \geq 1$, $\mathcal{K}_n$ has $O(n)$ states and for all $h \geq 1$, there are h-level B-certificates (resp., h-level E-certificates) of $\mathcal{K}_n$ whose length is at least $\frac{1}{h+1} \cdot (\frac{n}{h+1})^{h+1} \cdot e^h$.*

By Theorem 10 and the upper bound in Theorem 12, and considering that model-checking $\overline{\mathsf{B}}$ and $\overline{\mathsf{E}}$ is already **PSPACE**-hard, we obtain the following result.

▶ **Corollary 13.** *For the basis $\mathcal{B} = \{B\}$ (resp., $\mathcal{B} = \{E\}$), model-checking the logic $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ is in $\mathbf{AEXP_{pol}}$ and at least* **PSPACE***-hard.*

Considering that $\mathbf{AEXP_{pol}} \subseteq \mathbf{EXPSPACE}$, our result improves the **EXPSPACE** upper-bounds for the smaller fragments $\mathsf{A\overline{A}B\overline{B}E}$ and $\mathsf{A\overline{A}E\overline{B}E}$ obtained in [17] by a much more involved technique. In the following, we prove Theorem 12 focusing on B-certificates (the proof for E-certificates is similar and omitted).

Upper bound in Theorem 12 for B-certificates. In order to prove Theorem 12(1), for a given finite alphabet Σ and $h \geq 0$, we first define a variant of the notion of h-level B-descriptor, called *ordered h-prefix descriptor over Σ*, which is not related to a specific word over Σ. The set OPD_h of ordered h-prefix descriptors over Σ is partitioned into $|\Sigma|$ subsets OPD_h^b (for each $b \in \Sigma$), where each of them is equipped with a strict partial order. We show that (i) each strict ascendent chain of elements in OPD_h^b has length at most $O(|\Sigma|^{2h+1})$, (ii) the h-level B-descriptor of a word $w \in \Sigma^+$ is an element in OPD_h, and (iii) for each $w \in \Sigma^+$, the h-level B-descriptors associated to the prefixes of w can be grouped into at most $|\Sigma|$ non-strict ascendent chains. Thus, by Proposition 6 and reasoning as in Proposition 8, we fix the upper bound on the length of h-level B-certificates for a given finite Kripke structure.

Let $h \geq 0$. For a $(\Sigma \times 2^\Sigma \times \Sigma)$-term t with root (a, I, b), we say that a (resp., b) is the *first symbol* (resp., *last symbol*) of t.

▶ **Definition 14** (Ordered prefix descriptors). *Let Σ be a finite alphabet and $h \geq 0$. We define by induction on h a pair $(OPD_h, \prec_h)$ consisting of a set OPD_h of h-level $(\Sigma \times 2^\Sigma \times \Sigma)$-terms, called* ordered h-prefix descriptors over Σ *and a binary relation $\prec_h$ over OPD_h.*

- *$h = 0$: OPD_0 coincides with the set of 0-level $(\Sigma \times 2^\Sigma \times \Sigma)$-terms. Given (a, I, b), $(a', I', b') \in OPD_0$, $(a, I, b) \prec_0 (a', I', b')$ if (i) $a = a'$ (equality between the initial symbols) and (ii) $I \cup \{b\} \subseteq I' \cup \{b'\}$, and either $b \neq b'$ or $I \cup \{b\} \subset I' \cup \{b'\}$.*
- *$h > 0$: OPD_h is the set of h-level $(\Sigma \times 2^\Sigma \times \Sigma)$-terms $t = ((a, I, b), T)$ such that T is a (possibly empty) set of the form $T = \{t_1, \ldots, t_n\}$ where $t_i \in OPD_{h-1}$, t_i has initial symbol a, and $t_i \prec_{h-1} t_j$ for all $i \in [1, n]$ and $j \in [i + 1, n]$. The binary relation $\prec_h$ is defined as follows: $((a, I, b), T) \prec_h ((a', I', b'), T')$ if*
  - *$a = a'$, $I \cup \{b\} \subseteq I' \cup \{b'\}$, $T \subseteq T'$;*
  - either $b \neq b'$ or $I \cup \{b\} \subset I' \cup \{b'\}$ or $T \subset T'$.

By construction for each $b \in \Sigma$, the binary relation $\prec_h$ is a strict partial order over the set OPD_h^b of ordered h-prefix descriptors over Σ having the same last symbol b. Additionally, we show that a strict ascendent chain of elements in OPD_h^b has length at most $|\Sigma|^{2h+1}$.

▶ **Proposition 15.** *Let $h \geq 0$, Σ be a finite alphabet, $b \in \Sigma$, and $t_1, \ldots, t_n$ be ordered h-prefix descriptors having last symbol b such that $t_1 \prec_h t_2 \prec_h \ldots \prec_h t_n$. Then, $n \leq |\Sigma|^{2h+1}$.*

Proof. The proof is by induction on $h \geq 0$. For $h = 0$, there is $a \in \Sigma$ s.t. for all $i \in [1, n]$, $t_i = (a, I_i, b)$ for some $I_i \subseteq \Sigma$, and $I_1 \subset I_2 \subset \ldots \subset I_n$. Hence, $n \leq |\Sigma|$ and the result follows.

Now, let $h > 0$. Hence, there is $a \in \Sigma$ s.t. for all $i \in [1, n]$, t_i is of the form $t_i = ((a, I_i, b), T_i)$. By hypothesis, $I_1 \subseteq I_2 \subseteq \ldots \subseteq I_n$. Moreover, for each $i \in [1, n]$, T_i can be partitioned into at most $|\Sigma|$ strict ascendent chains of ordered $h-1$-prefix descriptors having the same last symbol. Thus, by the induction hypothesis, we have that $|T_i| \leq |\Sigma| \cdot |\Sigma|^{2(h-1)+1} = |\Sigma|^{2h}$ for all $i \in [1, n]$. Fix an arbitrary $i \in [1, n]$. We claim that for each $j \in [i, n]$ such that $I_j = I_i$, it holds that $|j - i| \leq |\Sigma|^{2h}$. Hence, evidently, the result follows. Fix $i, j \in [1, n]$ such that $i < j$ and $I_j = I_i$. Since $t_i \prec_h t_\ell$ for all $\ell \in [i+1, j]$, we have that $|T_i| < |T_{i+1}| < \ldots < |T_j|$. Hence, $j - i \leq |T_j|$ and since $|T_j| \leq |\Sigma|^{2h}$, the result follows. ◀

By exploiting Proposition 15, we deduce the following proposition, from which the upper bound for the h-level B-certificates in Theorem 12 directly follows.

▶ **Proposition 16.** *Let $\mathcal{K}$ be a finite Kripke structure with set of states S, $h \geq 0$, and π a path of $\mathcal{K}$. Then, the following holds:*

1. *for all $i, j \in [0, n]$ where $n = |\pi| - 1$, (i) $B_h(\pi[0, i])$ is an ordered h-prefix descriptor, and (ii) if $j \in [i+1, n]$ and $B_h(\pi[0, i]) \neq B_h(\pi[0, j])$, then $B_h(\pi[0, i]) \prec_h B_h(\pi[0, j])$;*
2. *there is a path π' having the same h-level B-descriptor as π s.t. $|\pi'|$ is at most $|S|^{2h+2}$.*

Proof. Property 1 can be proved by a straightforward induction on $h \geq 0$. Now, let us consider Property 2. By reasoning as in the proof of Proposition 8, there is a path π' of $\mathcal{K}$ having the same h-level B-descriptor as π and such that distinct non-empty prefixes of π' have distinct h-level B-descriptors as well. Let s be a state visited by π', then by Property 1, the set of h-level B-descriptors associated with the non-empty prefixes of π' ending at state s form a strict ascending chain (with respect $\prec_h$) whose length n_s coincides with the set of positions i of π' such that $\pi'(i) = s$. By Proposition 15, $n_s \leq |S|^{2h+1}$. Since $|\pi'| = \sum_{s \in S(\pi')} n_s$ where $S(\pi')$ is the set of states visited by π', we obtain that $|\pi'| \leq |S|^{2h+2}$. ◀

Lower bound in Theorem 12 for B-certificates. For each $n \geq 1$, let $\Sigma_n = \{a_1, \ldots, a_n\}$ be an alphabet consisting of n distinct symbols a_1, ..., a_n. We exhibit a family $(w_n^h)_{h \geq 0}$ of non-empty words over Σ_n such that for each $h \geq 0$, the length of w_n^h is at least $\frac{1}{h+1} \cdot (\frac{n}{h+1})^{h+1} \cdot e^h$ and w_n^h is a minimal representative of the $h+1$-level B-descriptor $B_{h+1}(w_n^h)$.

Fix $n \geq 1$. Formally, for all $i, j \in [1, n]$ and $h \geq 0$, we define by induction on $h \geq 0$, a non-empty word $w_{i,j,h}$ over Σ_n called (i, j, h)-*miniword*:

1. Case $h = 0$: if $i \leq j$, then $w_{i,j,h} = a_i a_{i+1} \ldots a_j$. Otherwise, $w = a_j a_{j-1} \ldots a_i$. The set of *main positions* of $w_{i,j,h}$ is the set of all its positions.
2. Case $h > 0$: if $i \leq j$, then $w_{i,j,h} = a_i \cdot w_i \cdot a_{i+1} \cdot w_{i+1} \cdot \ldots \cdot a_j \cdot w_j$ where for each $\ell \in [i, j]$, w_ℓ is the $(\ell, i, h-1)$-miniword. Otherwise, $w_{i,j,h} = a_i \cdot u_i \cdot a_{i-1} \cdot u_{i-1} \cdot \ldots \cdot a_j \cdot u_j$ where for each $\ell \in [j, i]$, u_ℓ is the $(\ell, i, h-1)$-miniword. The subwords w_ℓ (resp., u_ℓ) with $\ell \in [i, j]$ (resp., $\ell \in [j, i]$) are called *secondary subwords* of $w_{i,j,h}$, while a *main position* of $w_{i,j,h}$ is a position which is not associated to a secondary-subword position.

We say that $w_{i,j,h}$ *has level* h. Note that by construction, for each symbol a occurring in $w_{i,j,h}$, the smallest position ℓ such that $w_{i,j,h}(\ell) = a$ is a main position. We can show that distinct prefixes of h-level miniwords have distinct h-level B-descriptors as well.

▶ **Proposition 17.** *Let $n \geq 1$ and $h \geq 0$. Then, for each miniword w over Σ_n of level h, distinct prefixes of w have distinct h-level B-descriptors.*

For $\Sigma_n = \{a_1, \ldots, a_n\}$, let $K(\Sigma_n)$ be the Kripke structure $(\Sigma_n, \Sigma_n, E, Lab, a_1)$, where Lab is the identity and $(a_i, a_j) \in E$ for all $i, j \in [1, n]$. The set of paths in $K(\Sigma_n)$ is the set of non-empty finite words over Σ_n. Hence, the lower bound in Theorem 12 for B-certificates directly follows from the following result which is obtained by exploiting Proposition 17.

▶ **Proposition 18.** *Let $n \geq 1$, $i, j \in [1, n]$, and $h \geq 0$. For the (i, j, h) miniword $w_{i,j,h}$ over Σ_n, the length of $w_{i,j,h}$ is at least $\frac{1}{h+1} \cdot (\frac{|i-j|+1}{h+1})^{h+1} \cdot e^h$ and there is no smaller word u over Σ_n (i.e., such that $|u| < |w_{i,j,h}|$) having the same $h+1$-level B-descriptor as $w_{i,j,h}$.*

Proof. For the (i, j, k) miniword $w_{i,j,h}$, let $p = |i - j| + 1$. By construction, the length of $w_{i,j,h}$, denoted by $L(p, h)$, depends only on h and p, and satisfies the recurrence: $L(p, h) = p$ if $h = 0$, and $L(p, h) = p + \sum_{\ell=1}^{\ell=p} L(\ell, h-1)$ otherwise. We first show by induction on $h \geq 0$ that $L(p, h) \geq \frac{p^{h+1}}{(h+1)!}$. The base case ($h = 0$) is obvious. Now, let $h > 0$. By the induction hypothesis and the fact that $\sum_{\ell=1}^{\ell=p} \ell^h \geq \frac{p^{h+1}}{h+1}$ (Faulhaber's formula), we have that $L(p, h) = p + \sum_{\ell=1}^{\ell=p} L(\ell, h-1) \geq \sum_{\ell=1}^{\ell=p} \frac{\ell^h}{h!} \geq \frac{p^{h+1}}{(h+1)!}$. Thus, since $(h+1)! \leq \frac{(h+1)^{h+2}}{e^h}$, the claimed lower bound follows. Now, let T be the set of h-level B-descriptors of the non-empty proper prefixes of $w_{i,j,h}$, and u a non-empty word having the same $h+1$-level B-descriptor as $w_{i,j,h}$. Since the number of non-empty proper prefixes of a non-empty word w is $|w| - 1$, by hypothesis, we have that $|u| - 1 \geq |T|$. On the other hand, by Proposition 17, $|w_{i,j,h}| - 1 = |T|$. Hence, $|u| \geq |w_{i,j,h}|$, which concludes the proof of Proposition 18. ◀

4.2 Non-elementary lower bounds on the length of BD-certificates, BE-certificates, and ED-certificates

In this section, for each linear-time basis $\mathcal{B} \in \{\{B, D\}, \{B, E\}, \{E, D\}\}$, we establish a non-elementary lower bound on the length of h-level $\mathcal{B}$-certificates. Hence, in particular, we obtain a non-elementary lower bound on the running time of the algorithm for model-checking the logic $\mathsf{HS}_{\mathcal{B}}(\mathcal{F})$ presented in Section 3.1.

▶ **Theorem 19.** *There is a family $\{\mathcal{K}_n\}_{n \geq 1}$ of finite Kripke structures such that for all $n \geq 1$, $\mathcal{K}_n$ has $O(n)$ states and for all $k \in [0, n-1]$ and basis $\mathcal{B}$ with $\mathcal{B} \in \{\{B, D\}, \{E, D\}\}$ (resp., $\mathcal{B} = \{B, E\}$), there are k-level (resp., $2k$-level) $\mathcal{B}$-certificates of $\mathcal{K}_n$ having length at least $\Omega(Tower(n, k+1))$.*

In the rest of this section we provide a proof of Theorem 19. We first show as an intermediate and crucial step that there is a family $\{\Sigma_n\}_{n \geq 1}$ of finite alphabets such that for all $n \geq 1$, Σ_n has cardinality $O(n)$ and for all $h \in [0, n-1]$, there are $\Omega(Tower(n, h+1))$ words over Σ_n having pairwise distinct h-level D-descriptors (resp., $2h$-level BE-descriptors).

Fix $n \geq 1$ and let Σ_n be the finite alphabet having cardinality $O(n)$ given by

$$\Sigma_n = \bigcup_{i \in [2,n]} \{\$_i\} \cup \bigcup_{bit \in \{0,1\}} \bigcup_{i \in [1,n]} \{(\$_i, bit)\} \cup \bigcup_{bit \in \{0,1\}} \bigcup_{i \in [1,n]} \{(i, bit)\}$$

Moreover, for each $h \in [1, n]$, let Σ_n^h be the subset of Σ_n given by

$$\Sigma_n^h = \Sigma_n \setminus \Big(\bigcup_{i \in [h+1,n]} \{\$_i\} \cup \bigcup_{bit \in \{0,1\}} \bigcup_{i \in [h+1,n]} \{(\$_i, bit)\} \Big)$$

For each $h \in [1, n]$, we define a suitable encoding of the natural numbers in $[0, \mathit{Tower}(n, h)-1]$ by finite words over Σ_n^h, called (n, h)-*blocks*. In particular, for $h > 1$, a (n, h)-block encoding a natural number $m \in [0, \mathit{Tower}(n, h) - 1]$ is a sequence of $\mathit{Tower}(n, h - 1)$ $(n, h - 1)$-blocks, where the i^{th} $(n, h - 1)$-block encodes both the value and (recursively) the position of the i^{th}-bit in the binary representation of m. Formally, the set of (n, h)-blocks is defined by induction on h as follows:

Base Step: $h = 1$. A $(n, 1)$-block is a finite word bl over Σ_n^1 of length $n+2$ having the form $bl = (\$_1, bit)(1, bit_1) \dots (n, bit_n)(\$_1, bit)$ such that $bit, bit_1, \dots, bit_n \in \{0, 1\}$. The *content* of bl is bit, and the *index* of bl is the natural number in $[0, \mathit{Tower}(n, 1) - 1]$ (recall that $\mathit{Tower}(n, 1) = 2^n$) whose binary code is $bit_1 \dots bit_n$.

Induction Step: $1 < h \leq n$. A (n, h)-*block* is a finite word bl over Σ_n^h having the form $(\$_h, bit) \cdot bl_0 \cdot \$_h \cdot \ldots \cdot bl_{\ell-1} \cdot \$_h \cdot bl_\ell \cdot (\$_h, bit)$ such that $\ell = \mathit{Tower}(n, h - 1) - 1$, $bit \in \{0, 1\}$ and for all $i \in [0, \ell]$, bl_i is a $(n, h - 1)$-block having index i. The *content* of bl is bit and the *index* of bl is the natural number in $[0, \mathit{Tower}(n, h) - 1]$ whose binary code is given by $bit_0, \dots, bit_\ell$, where bit_i is the content of the sub-block bl_i for all $0 \leq i \leq \ell$.

By construction, the following holds.

▶ Remark 20. For all $n \geq 1$ and $h \in [1, n]$, there are $2 \cdot \mathit{Tower}(n, h)$ distinct (n, h)-blocks.

▶ **Example 21.** Let $n = 2$ and $h = 2$. In this case $\mathit{Tower}(n, h) = 16$ and $\mathit{Tower}(n, h - 1) = 4$. We can encode by $(2, 2)$-blocks all the integers in $[0, 15]$. Let us consider the number 14 whose binary code (using $\mathit{Tower}(n, h - 1) = 4$ bits) is given by 0111 (the first bit is the least significant). The $(2, 2)$-block with content 0 encoding number 14 is given by $(\$_2, 0) \cdot bl_0 \cdot \$_2 \cdot bl_1 \cdot \$_2 \cdot bl_2 \cdot \$_2 \cdot bl_3 \cdot (\$_2, 0)$, where bl_i is the $(2, 1)$-block encoding the value and the position of the i^{th} bit in 0111. For example, $bl_2 = (\$_1, 1)(1, 0)(2, 1)(\$_1, 1)$ while $bl_3 = (\$_1, 1)(1, 1)(2, 1)(\$_1, 1)$.

We now show that the $(h - 1)$-level D-descriptors (resp., $(2h - 2)$-level BE-descriptors) associated with distinct (n, h)-blocks are distinct as well.

▶ **Lemma 22.** *Let $n \geq 1$. Then, for each $h \in [1, n]$, distinct (n, h)-blocks have distinct $(h - 1)$-level D-descriptors and distinct $(2h - 2)$-level BE-descriptors.*

Proof. For the fixed $n \geq 1$, the proof of Lemma 22 is by induction on $h \in [1, n]$. For the base case, let $h = 1$. Let bl be an $(n, 1)$-block. By construction bl is a word of length $n + 2$ of the form $bl = (\$_1, bit)(1, bit_1) \dots (n, bit_n)(\$_1, bit)$ where $bit, bit_1, \dots, bit_n \in \{0, 1\}$. Hence, the 0-level D-descriptor $D_0(bl)$ (resp., 0-level BE-descriptor $BE_0(bl)$) of bl is the triple $((\$_1, bit), \{(1, bit_1), \dots, (n, bit_n)\}, (\$_1, bit))$, and the result for $h = 1$ easily follows.

Now, for the induction step, assume that $h \in [2, n]$. Let bl and bl' be two (n, h)-blocks such that $bl \neq bl'$. We need to show that the $(h - 1)$-level D-descriptors (resp., $(2h - 2)$-level BE-descriptors) of bl and bl' are distinct. First, assume that bl and bl' have distinct content: let $(\$_h, bit)$ (resp., $(\$_h, bit')$) be the content of bl (resp., bl'). By hypothesis, $bit \neq bit'$. By construction, it follows that $D_0(bl) \neq D_0(bl')$ and $BE_0(bl) \neq BE_0(bl')$. Hence, $D_{h-1}(bl) \neq D_{h-1}(bl')$ and $BE_{2h-2}(bl) \neq BE_{2h-2}(bl')$ and the result follows.

Now, assume that bl and bl' have the same content. Since bl and bl' are distinct (n, h)-blocks, by construction there is $i \in [0, \mathit{Tower}(n, h - 1) - 1]$ such that the $(n, h - 1)$ sub-block sb_i of bl with index i and the $(n, h - 1)$ sub-block sb'_i of bl' with index i have distinct content.

We first consider the D-descriptors. Let $(D_0(bl), T)$ (resp., $(D_0(bl), T')$) be the $(h-1)$-level D-descriptor of bl (resp., bl'). We show that for each non-empty internal subword w of bl, the $(h-2)$-level D-descriptor $D_{h-2}(w)$ of w is distinct from the $(h-2)$-level descriptor $D_{h-2}(sb'_i)$ of sb'_i. Hence, $D_{h-2}(sb'_i) \notin T$. Since $D_{h-2}(sb'_i) \in T'$, we obtain that $T \neq T'$ and the result follows. Fix a non-empty internal subword w of bl. By hypothesis and construction, there is no subword of bl which coincides with sb'_i. We distinguish the following cases:

- w is an $(n, h-1)$-block. Since w is internal subword of bl and no subword of bl coincides with sb'_i, it hold that $w \neq sb'_i$. Hence, by the induction hypothesis, $D_{h-2}(w) \neq D_{h-2}(sb'_i)$.
- w is a proper subword of some $(n, h-1)$-block. By construction $D_0(w)$ is of the form (p, P, p') such that either $p \notin \{(\$_{h-1}, 0), (\$_{h-1}, 1)\}$ or $p' \notin \{(\$_{h-1}, 0), (\$_{h-1}, 1)\}$. Since the 0-level descriptor of an $(n, h-1)$-block is of the form $((\$_{h-1}, bit), P', (\$_{h-1}, bit))$ for some $bit \in \{0, 1\}$, we obtain that $D_0(w) \neq D_0(sb'_i)$. Hence, $D_{h-2}(w) \neq D_{h-2}(sb'_i)$.
- There is some $(n, h-1)$ sub-block w' of bl such that w' is a proper subword of w. By construction, w contains some occurrence of a symbol in $\{\$_h, (\$_h, 0), (\$_h, 1)\}$. Since such symbols do not occur in an $(n, h-1)$-block, the result holds in this case as well.

It remains to consider the BE-descriptors. Let $(BE_0(bl), T_P, T_S)$ (resp., $(BE_0(bl'), T'_P, T'_S)$) be the $(2h-2)$-level BE-descriptor of bl (resp., bl'), and $w_{sb'_i}$ be the unique proper prefix of bl' having sb'_i as a proper suffix. We show that for each non-empty proper prefix w_p of bl, $BE_{2h-3}(w_{sb'_i}) \neq BE_{2h-3}(w_p)$. Hence, $BE_{2h-3}(w_{sb'_i}) \notin T_P$. Since $BE_{2h-3}(w_{sb'_i}) \in T'_P$, we obtain that $T_P \neq T'_P$ and the result follows. Fix a non-empty proper prefix w_p of bl. Note that since $h \geq 2$, $BE_{2h-3}(w_p)$ is of the form $(BE_0(w_p), R_P, R_S)$ and $BE_{2h-3}(w_{sb'_i})$ is of the form $(BE_0(w_{sb'_i}), R'_P, R'_S)$. Thus, it suffices to prove that $R_S \neq R'_S$. Since a proper suffix of a proper prefix of a word u is an internal word of u and $BE_{2h-4}(sb'_i) \in R'_S$, we just need to show that for each non-empty internal subword u of bl, $BE_{2h-4}(sb'_i) \neq BE_{2h-4}(u)$. For this we proceed as for the case of the D-descriptors but this time we apply the induction hypothesis on the BE_{2h-4}-descriptors. This concludes the proof of Lemma 22. ◀

Proof of Theorem 19. Let $n \geq 1$, a_n be a designated letter in the alphabet Σ_n and $\mathcal{K}_n$ the finite Kripke structure over Σ_n given by $\mathcal{K}_n = (\Sigma_n, \Sigma_n, E_n, Lab_n, a_n)$, where $(a, a') \in E_n$ and $Lab_n(a) = \{a\}$ for all $a, a' \in \Sigma_n$. Hence, the paths of $\mathcal{K}_n$ correspond to the non-empty finite words over Σ_n. We show that for all $k \in [0, n-1]$ and basis $\mathcal{B}$ with $\mathcal{B} \in \{\{B, D\}, \{E, D\}\}$ (resp., $\mathcal{B} = \{B, E\}$), there are $\Omega(Tower(n, k+1))$ distinct k-level (resp., $2k$-level) $\mathcal{B}$-certificates of $\mathcal{K}_n$. Hence, Theorem 19 directly follows. By Remark 20, there are $2 \cdot Tower(n, k+1)$ distinct $(n, k+1)$-blocks. Thus, for the basis $\{B, E\}$, the result directly follows from Lemma 22. For the bases $\{B, D\}$ and $\{E, D\}$, the result follows from Lemma 22 and the fact that words having distinct $2k$-level D-descriptors have distinct $2k$-level BD-descriptors (resp., distinct $2k$-level ED-descriptors) as well. ◀

5 Conclusions

We have addressed open complexity issues about the known approach to model-checking the logic HS, based on abstract representations of paths in Kripke structures (BE-descriptors). In particular, we have proposed a unifying framework for model-checking full HS and large HS-fragments obtained by (i) introducing for each basis $\mathcal{B}$, a specialized type of descriptor ($\mathcal{B}$-descriptor) and (ii) designing an alternating-time MC algorithm with a polynomially bounded number of alternations which is parametric w.r.t. the chosen basis $\mathcal{B}$ and runs in time bounded by the length of $\mathcal{B}$-descriptor certificates. As a main result, for each basis $\mathcal{B}$,

we have provided tight bounds on the length of $\mathcal{B}$-certificates: exponential for the bases $\{B\}$ and $\{E\}$ (which lead to $\mathbf{AEXP_{pol}}$ procedures for the related fragments), and non-elementary for the other bases. Future work will be devoted to solve the hard open question about the existence of an elementary procedure for the MC problem for the full logic, and to settle the exact complexity for model-checking the HS-fragments for the bases $\{B\}$ and $\{E\}$.

References

1 J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

2 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Interval Temporal Logic Model Checking: the Border Between Good and Bad HS Fragments. In *Proc. 8th IJCAR*, LNAI 9706, pages 389–405. Springer, 2016.

3 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Satisfiability and Model Checking for the Logic of Sub-Intervals under the Homogeneity Assumption. In *Proc. 44th ICALP*, volume 80 of *LIPIcs*, pages 120:1–120:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

4 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Model checking for fragments of the interval temporal logic HS at the low levels of the polynomial time hierarchy. *Inf. Comput.*, 262(Part):241–264, 2018.

5 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Interval vs. Point Temporal Logic Model Checking: An Expressiveness Comparison. *ACM Trans. Comput. Logic*, 20(1):4:1–4:31, 2019.

6 D. Bresolin, D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco. The dark side of interval temporal logic: marking the undecidability border. *Annals of Mathematics and Artificial Intelligence*, 71(1-3):41–83, 2014.

7 D. Bresolin, V. Goranko, A. Montanari, and P. Sala. Tableau-based decision procedures for the logics of subinterval structures over dense orderings. *Journal of Logic and Computation*, 20(1):133–166, 2010.

8 D. Bresolin, V. Goranko, A. Montanari, and G. Sciavicco. Propositional interval neighborhood logics: Expressiveness, decidability, and undecidable extensions. *Annals of Pure and Applied Logic*, 161(3):289–304, 2009.

9 E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

10 J. Ferrante and C. Rackoff. A Decision Procedure for the First Order Theory of Real Addition with Order. *SIAM Journal of Computation*, 4(1):69–76, 1975.

11 J. Y. Halpern and Y. Shoham. A Propositional Modal Logic of Time Intervals. *Journal of the ACM*, 38(4):935–962, 1991.

12 K. Lodaya. Sharpening the Undecidability of Interval Temporal Logic. In *Proc. 6th ASIAN*, LNCS 1961, pages 290–298. Springer, 2000.

13 A. Lomuscio and J. Michaliszyn. An Epistemic Halpern-Shoham Logic. In *Proc. 23rd IJCAI*, pages 1010–1016, 2013.

14 A. Lomuscio and J. Michaliszyn. Decidability of model checking multi-agent systems against a class of EHS specifications. In *Proc. 21st ECAI*, pages 543–548, 2014.

15 J. Marcinkowski and J. Michaliszyn. The Undecidability of the Logic of Subintervals. *Fundamenta Informaticae*, 131(2):217–240, 2014.

16 A. Molinari, A. Montanari, A. Murano, G. Perelli, and A. Peron. Checking interval properties of computations. *Acta Informatica*, 53(6-8):587–619, 2016.

17 A. Molinari, A. Montanari, and A. Peron. A Model Checking Procedure for Interval Temporal Logics based on Track Representatives. In *Proc. 24th CSL*, pages 193–210, 2015.

18 A. Molinari, A. Montanari, and A. Peron. Complexity of ITL model checking: some well-behaved fragments of the interval logic HS. In *Proc. 22nd TIME*, pages 90–100, 2015.

19 Alberto Molinari, Angelo Montanari, and Adriano Peron. Model checking for fragments of Halpern and Shoham's interval temporal logic based on track representatives. *Inf. Comput.*, 259(3):412–443, 2018.

20 A. Montanari, G. Puppis, and P. Sala. A decidable weakening of Compass Logic based on cone-shaped cardinal directions. *Logical Methods in Computer Science*, 11(4), 2015.

21 B. Moszkowski. *Reasoning About Digital Circuits*. PhD thesis, Dept. of Computer Science, Stanford University, Stanford, CA, 1983.

22 A. Pnueli. The temporal logic of programs. In *Proc. 18th FOCS*, pages 46–57. IEEE, 1977.

23 I. Pratt-Hartmann. Temporal propositions and their logic. *Artificial Intelligence*, 166(1-2):1–36, 2005.

24 P. Roeper. Intervals and Tenses. *Journal of Philosophical Logic*, 9:451–469, 1980.

25 Y. Venema. Expressiveness and Completeness of an Interval Tense Logic. *Notre Dame Journal of Formal Logic*, 31(4):529–547, 1990.

Simplifying Inductive Schemes in Temporal Logic

Pablo Cordero
Dept. Applied Mathematic, University of Málaga, Spain
http://webpersonal.uma.es/~pcordero/My_personal_web/Wellcome.html
pcordero@uma.es

Inmaculada Fortes
Dept. Applied Mathematic, University of Málaga, Spain
ifortes@ctima.uma.es

Inmaculada P. de Guzmán
Dept. Applied Mathematic, University of Málaga, Spain
guzman@ctima.uma.es

Sixto Sánchez
Dept. Applied Mathematic, University of Málaga, Spain
sixto@uma.es

Abstract

In propositional temporal logic, the combination of the connectives "tomorrow" and "always in the future" require the use of induction tools. In this paper, we present a classification of inductive schemes for propositional linear temporal logic that allows the detection of loops in decision procedures. In the design of automatic theorem provers, these schemes are responsible for the searching of efficient solutions for the detection and management of loops. We study which of these schemes have a good behavior in order to give a set of reduction rules that allow us to compute these schemes efficiently and, therefore, be able to eliminate these loops. These reduction laws can be applied previously and during the execution of any automatic theorem prover. All the reductions introduced in this paper can be considered a part of the process for obtaining a normal form of a given formula.

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

Keywords and phrases Linear Temporal Logic, Inductive Schemes, Loop-check

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.19

Funding *Pablo Cordero*: Partially supported by Project TIN2017-89023-P of the Science and Innovation Ministry of Spain, co-funded by the EU Regional Development (ERDF).
Inmaculada Fortes: Partially supported by Project PGC2018-095869-B-I00 of the Science and Innovation Ministry of Spain.
Inmaculada P. de Guzmán: Partially supported by Project TIN2017-89023-P of the Science and Innovation Ministry of Spain, co-funded by the EU Regional Development (ERDF).

1 Introduction

The notion of induction is a relevant topic in temporal logic and related areas. In fact, in [11], for the philosophical question: *What is temporal logic?* five answers are provided by Michael Fisher. One answer is that Propositional Temporal Logic characterizes simple induction. Another is that "Propositional Temporal Logic can be seen as a multi-modal logic, comprising two modalities, $\boxed{1}$ and $\boxed{*}$, which interact closely". The modal operator $\boxed{1}$ corresponds to the "next" relation and $\boxed{*}$ is a modal operator that corresponds to the "always" relation. As usual, possibility modal operators are denoted by $\langle 1\rangle$ and $\langle *\rangle$. Thus, the *interaction* axiom between both modal operators is given by the induction axiom i.e. $\vdash \boxed{*}(\varphi \rightarrow \boxed{1}\varphi) \rightarrow (\varphi \rightarrow \boxed{*}\varphi)$.

It is well-known the mathematical induction principle can be generalized. In the same way, in Propositional Temporal Logic, there are other formulas which match with the idea of induction. Thus, for example, the formula $\boxed{*}(\langle * \rangle \varphi \rightarrow \boxed{1}\boxed{1}\varphi) \rightarrow (\boxed{1}\boxed{1}\boxed{1}\boxed{1}\varphi \rightarrow \boxed{*}\boxed{1}\boxed{1}\varphi)$ is a tautology that follows the induction idea. This kind of scheme is especially relevant in the design of automatic theorem provers for temporal logics where they are responsible for the searching of efficient solutions for the detection and management of loops (see, for example [3, 7, 12]).

The applicability of temporal logics in the field of information sciences does not need to be justified. As a simple example, an important application of temporal logics is the specification of the properties to be formally verified in a model checking [5] where the system requirements are usually expressed by means of temporal logic formulas [13]. What makes temporal logic particularly attractive is the enormous success of model checking, which, under appropriate assumptions on the system specification, can make the verification of temporal logic properties automatic.

In verification, temporal logics are usually enriched with modal connectives such as knowledge, beliefs, intentions, norms, etc. Although, the extension of results in temporal logics to these modal×temporal logics is not straightforward [16], any advance in temporal logic technics is a positive enhancement.

Induction is also a relevant issue in model checking, since it is related to the analysis of invariants and loops and, specifically, with infinite loops detection. Induction itself is relevant because reasoning about the partial correctness of programs often requires proofs by induction, in particular for reasoning about recursive functions. In [10], a proof method for inductive theorem proving is developed in rewriting a system framework. In [9], the author develops a general proof method that combines logic and induction. He provides a solid and uniform mathematical foundation to induction proof methodologies for a wide variety of formal specification frameworks and shows its applicability through several examples of formal verification proof. As it has been mentioned, induction plays a central role in temporal logic. Combining both topics, in [1], a linear temporal logic of rewriting is introduced to be used in model checking.

One of the main problems with the model checking technique is to make it as least expensive as possible. That is, a balance is needed between expressivity of the language and the cost of the logic-based methods. Numerous logics have been used for this objective [16, 2, 17]. In this paper we center on the very influential Linear Temporal Logic, considering the "always" and the "next" operators, introduced by Pnueli [19]. In [21], Sistla and Clarke studied the complexity of the satisfiability and model checking problems in this logic and, in [8], a more general study can be found regarding the complexities of temporal logic model checking.

Specifically, the considered logic in this paper is $FNext$, which is a propositional linear discrete temporal logic with three temporal connectives (using Prior's notation): $\oplus$ (tomorrow), $\mathtt{G}$ (always in the future) and $\mathtt{F}$ (some time in the future). As mentioned, it is well-known that the combination itself of the connectives $\oplus$ and $\mathtt{G}$ requires the study of induction [11].

Among the formulas which involve the induction idea, we are especially interested in those where a unique propositional symbol has incidence. In propositional logic, literals are limited to propositional symbols and their negations (aside from constants) whereas in modal/temporal logics they are extended to include modal/temporal connectives (see Section 2.2). In the medium term, our aim is to extend the notion of literal including induction schemes. There is no doubt about the relevance of literals in issues such as normal forms, automatic reasoning and the SAT problem. As an example, in [14], literals are used to find rough and fuzzy-rough set reducts. In [20] they are used to extend formal concept analysis considering negative information.

In this paper we focus on the study of induction schemes as a previous stage in the generalization of literals towards the improvement of normalization processes. To this aim, we analyze those formulas which involves the induction idea. Thus, in the same way that the interaction axiom can be extended, the pure induction (i.e. $\oplus p \wedge \mathtt{G}(p \rightarrow \oplus p) \equiv \mathtt{G}p$) can be generalized into induction schemes. That is, we consider formulas $A \wedge \mathtt{G}(B \rightarrow C)$ where A semantically implies $\mathtt{F}B$ and C implies $\mathtt{F}B$. As mentioned, we especially center on those cases in which A, B and C are formulas where a unique propositional symbol has incidence. Specifically, in this paper we characterize those formulas that can be simplified providing an equivalent without loops.

All simplifications introduced in this paper have a general purpose and can be used in deduction processes, automatic reasoning, normalization techniques, etc. Some of the given equivalences are able to simplify the formula by reducing the size and are useful in problems where cost depends on the size of the input. Moreover, although we describe our method on $FNext$, our approach can also be applied to totally expressive logic US developed by Hans Kamp [15].

The paper is organized as follows: In Section 2, the preliminary definitions and $FNext$ logic are introduced. In Section 3, the aim of the paper is specified in detail. In Section 4, G-clauses are studied and, in Section 5, inductive schemes are defined and, from among all the studied formulas, we classify them into those that can be transformed into an equivalent expression that does not involve loops and those corresponding with inductive schemes. The final section is devoted to conclusions and future works.

2 $FNext$ Logic

In this section, we introduce a temporal propositional logic, with an infinite, linear and discrete flow of time denoted by $FNext$. We develop the language, semantics, modalities and simplification laws for $FNext$.

The alphabet of the language of the logic $FNext$ consists of the following: A denumerable set, $\mathcal{V}$, of propositional variables, a set of Boolean connectives: $\{\top, \bot, \neg, \wedge, \vee, \rightarrow\}$, and a set of temporal connectives: $\{\oplus, \mathtt{F}, \mathtt{G}\}$. The language is denoted by $\mathcal{L}$ and well-formed formulas, (hereafter, wffs), in $FNext$ are generated by the construction rules of classical propositional logic, plus the following rule: *"If A is a wff, then $\oplus A$, $\mathtt{F}A$ and $\mathtt{G}A$ are wffs"*.

The meaning of an expression like $\oplus A$, is : "*A will occur tomorrow*"; while the meaning of an expression like $\mathtt{F}A$ is: "*in the future, A will occur*" and of an expression like $\mathtt{G}A$ is: "*in the future, A will always occur*". We shall consider $\mathtt{F}$ and $\mathtt{G}$ as connectives of strict future. In what follows, we consider $\oplus^k A = \oplus \oplus^{k-1} A$ if $k \geq 1$ and $\oplus^0 A = A$.

2.1 Semantics for $FNext$

For the semantics of $FNext$ we consider always the temporal flow $(\mathbb{Z}, <)$ (i.e., the existence of a first instant of time is not demanded), where $<$ is the standard ordering on $\mathbb{Z}$, and interpretations, which are mappings from the language to $2^{\mathbb{Z}}$ assigning to each wff the set of instants in $\mathbb{Z}$ where such wff is true.

▶ **Definition 1.** *A* model *is a tuple $(\mathbb{Z}, <, h)$ where $h\colon \mathcal{L} \longrightarrow 2^{\mathbb{Z}}$ is a function, called* temporal interpretation*, satisfying the following conditions:*

1. $h(\top) = \mathbb{Z}$, $h(\bot) = \varnothing$,
2. $h(\neg A) = \mathbb{Z} \setminus h(A)$, $h(A \wedge B) = h(A) \cap h(B)$, $h(A \vee B) = h(A) \cup h(B)$
3. $h(\oplus A) = \{t \in \mathbb{Z} \mid t + 1 \in h(A)\}$

4. $h(\mathtt{F}A) = \{t \in \mathbb{Z} \mid (t,\infty) \cap h(A) \neq \varnothing\}$
5. $h(\mathtt{G}A) = \{t \in \mathbb{Z} \mid (t,\infty) \subseteq h(A)\}$

where $(t,\infty) = \{z \in \mathbb{Z} \mid t < z\}$.

As usual, other classical connectives are introduced as follows:

$$A \to B =_{def} \neg A \vee B \quad \text{and} \quad A \leftrightarrow B =_{def} (A \to B) \wedge (B \to A)$$

Observe that the connective $\mathtt{F}$ can be also introduced as a definite connective like $\mathtt{F}A =_{def} \neg\mathtt{G}\neg A$. Thus, from now on, we focus our study on formulas involving $\mathtt{G}$.

Given two wffs A and B, the notions of validity, satisfiability, logical consequence and equivalence are defined in standard way: A is *valid* if, for every interpretation h, we have that $h(A) = \mathbb{Z}$, and it is denoted by $\models A$. A is *satisfiable* if an interpretation h and $t \in \mathbb{Z}$ exists, such that $t \in h(A)$. In this case, we say that A is true at t, and it is denoted by $(h,t) \models A$ and, when no confusion arises, it is denoted as $t \models A$. B is a *logical consequence* of A, denoted by $A \models B$, if for all interpretation h and all $t \in \mathbb{Z}$, $(h,t) \models A$ implies $(h,t) \models B$. Finally, A and B are *equivalent*, denoted by $A \equiv B$, if $A \models B$ and $B \models A$. That is, for all interpretation h, $h(A) = h(B)$.

In $FNext$, the propositional classical logical laws hold. Moreover, the following proposition establishes the basic laws concerning temporal connectives.

▶ **Proposition 2.** *If $A, B \in \mathcal{L}$, then the following equivalences hold:*
1. $\mathtt{F}\mathtt{F}A \equiv \oplus\mathtt{F}A \equiv \mathtt{F} \oplus A$; $\quad \mathtt{G}\mathtt{G}A \equiv \oplus\mathtt{G}A \equiv \mathtt{G} \oplus A$,
2. *If* $\gamma \in \{\oplus, \mathtt{F}, \mathtt{G}\}$, *then* $\gamma\mathtt{G}\mathtt{F}A \equiv \mathtt{G}\mathtt{F}A$ *and* $\gamma\mathtt{F}\mathtt{G}A \equiv \mathtt{F}\mathtt{G}A$.

The following laws concern the interaction between classical and temporal connectives.

▶ **Proposition 3.** *If $A, B \in \mathcal{L}$, then the following equivalences hold:*
1. $\oplus\bot \equiv \mathtt{F}\bot \equiv \mathtt{G}\bot \equiv \bot$ *and* $\oplus\top \equiv \mathtt{F}\top \equiv \mathtt{G}\top \equiv \top$
2. $\neg \oplus A \equiv \oplus\neg A$; $\quad \neg\mathtt{F}A \equiv \mathtt{G}\neg A$; $\quad \neg\mathtt{G}A \equiv \mathtt{F}\neg A$,
3. $\oplus(A \vee B) \equiv \oplus A \vee \oplus B$; $\quad \oplus(A \wedge B) \equiv \oplus A \wedge \oplus B$,
4. $\mathtt{F}(A \vee B) \equiv \mathtt{F}A \vee \mathtt{F}B$ *and* $\mathtt{G}(A \wedge B) \equiv \mathtt{G}A \wedge \mathtt{G}B$.

2.2 Modalities and literals in $FNext$

Proposition 2 leads to the definition of modalities (sequences of temporal connectives) and their canonical representation.

▶ **Definition 4.** *The set of modalities is defined as follows*

$$\mathcal{M}d = \{\Gamma = \gamma_1 \dots \gamma_n \mid n \in \mathbb{N}, \gamma_i \in \{\oplus, \mathtt{F}, \mathtt{G}\} \text{ for all } 1 \leq i \leq n\}$$

and the set of canonical modalities is as follows

$$\mathcal{M}d_c = \{\mathtt{F}\mathtt{G}, \mathtt{G}\mathtt{F}\} \cup \{\oplus^{k+1}, \mathtt{F}\oplus^k, \mathtt{G}\oplus^k \mid k \in \mathbb{N}\}$$

Thus, as a direct consequence of Proposition 2, we have the following theorem that gives meaning to the name of canonical modalities.

▶ **Theorem 5.** *Given a wff A, for any modality $\Gamma \in \mathcal{M}d$, there exists a canonical modality $\Gamma_c \in \mathcal{M}d_c$ such that $\Gamma A \equiv \Gamma_c A$.*

▶ **Example 6.** $\mathtt{F} \oplus \oplus\mathtt{F} \oplus \oplus \oplus \mathtt{F} \oplus A \equiv \mathtt{F} \oplus^8 A$ and $\mathtt{F} \oplus \mathtt{G}\mathtt{F} \oplus \oplus\mathtt{G} \oplus A \equiv \mathtt{F}\mathtt{G}A$, for any wff A.

▶ **Definition 7** (Literals). *Let $\mathcal{V}$ be the propositional variable set and $p \in \mathcal{V}$. Then:*

1. *A formula p or $\neg p$ will be named classical literal on p and denoted by ℓ_p. Then, $\mathcal{V}^{\pm}$ denotes the set of classical literals $\{p, \neg p \mid p \in \mathcal{V}\}$.*
2. *Given $\ell_p \in \mathcal{V}^{\pm}$, the set of* temporal literals on ℓ_p *is*

$$Lit(\ell_p) = \{\top, \bot\} \cup \{\mathtt{FG}\ell_p, \mathtt{GF}\ell_p\} \cup \{\oplus^k \ell_p, \mathtt{F}\oplus^k \ell_p, \mathtt{G}\oplus^k \ell_p \mid k \in \mathbb{N}\}$$

3. *The* temporal literal set *is:* $Lit = \bigcup_{\ell_p \in \mathcal{V}^{\pm}} Lit(\ell_p)$.

From now on, when no confusion arises, the adjective "temporal" will be omitted.

Theorem 5 ensures that, for any classical literal $\ell_p \in \mathcal{V}^{\pm}$ and any modality $\Gamma \in \mathcal{M}d$ there exists one (and only one) temporal literal $\ell \in Lit$ such that $\Gamma\ell_p \equiv \ell$. The unicity of this literal is because there does not exist two different literals that are equivalent.

On the other hand, in Subsection 2.1, we have introduced the notion of logical consequence, denoted by $\models$, which, in the set of literals, can be seen as an order relation. Moreover, the poset $(Lit, \models)$ is a lattice as the following proposition states.

▶ **Proposition 8** ([6]). *$(Lit, \models)$ is a lattice satisfying the following conditions:*

1. *$(Lit(\ell_p), \models)$ is a sublattice, for all $\ell_p \in \mathcal{V}^{\pm}$ (see Figure 1).*
2. *For all $\ell_1 \in Lit(\ell_p)$ and $\ell_2 \in Lit(\ell_q)$ with $\ell_p \neq \ell_q$.*

$$\ell_1 \models \ell_2 \quad \text{if and only if} \quad \ell_1 = \bot \text{ or } \ell_2 = \top$$

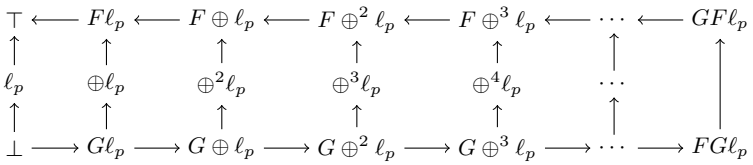


Figure 1 The sublattice $(Lit(\ell_p), \models)$.

2.3 Specific distribution laws

The item 2 in Proposition 2 reveals the special behavior of the modalities `FG` and `GF`: both *"absorb to any other finite sequence of temporal connectives"*. This is due to the fact that, for any interpretation, if there exists an instant in which the formula is true then it is true in every instant. This characteristic will be named using the adjective *atemporal*. Similarly, there exist formulas that can be *projected* toward the future or the past. The following definition formalize these ideas.

▶ **Definition 9.** *Let $[t, \infty) = \{t' \in \mathbb{Z} \mid t \leq t'\}$ and $(-\infty, t] = \{t' \in \mathbb{Z} \mid t' \leq t\}$. A wff A is said to be*

- projectable to the future *if $t \in h(A)$ implies $[t, \infty) \subseteq h(A)$, for all temporal interpretation h, i.e. either $h(A) = \varnothing$, or $h(A) = \mathbb{Z}$, or $h(A) = [t_0, \infty)$ for some $t_0 \in \mathbb{Z}$.*
- projectable to the past *if $t \in h(A)$ implies $(-\infty, t] \subseteq h(A)$, for all temporal interpretation h, i.e. either $h(A) = \varnothing$, or $h(A) = \mathbb{Z}$, or $h(A) = (-\infty, t_0]$ for some $t_0 \in \mathbb{Z}$.*
- atemporal *if $h(A) = \varnothing$ or $h(A) = \mathbb{Z}$, for all temporal interpretation h. So, A is atemporal if and only if A is projectable to the future and to the past.*

The following immediate results allow to characterize syntactically the formulas that are projectable: for all wffs A and B,

1. $\top$ and $\bot$ are atemporal.
2. $\mathtt{G}A$ is projectable to the future.
3. $\mathtt{F}A$ is projectable to the past.
4. If A is projectable to the future, $\oplus A$ and $\mathtt{G}A$ are also projectable to the future, $\neg A$ is projectable to the past and $\mathtt{F}A$ is atemporal.
5. If A is projectable to the past, $\oplus A$ and $\mathtt{F}A$ are also projectable to the past, $\neg A$ is projectable to the future and $\mathtt{G}A$ is atemporal.
6. If A and B are projectable to the future/past, so are $A \vee B$ and $A \wedge B$.

Proposition 3 ensures the distributivity of $\oplus$ and $\mathtt{F}$ over $\vee$, and the distributivity of $\oplus$ and $\mathtt{G}$ over $\wedge$. However, we cannot ensure neither the distributivity of $\mathtt{F}$ over $\wedge$ nor the distributivity of $\mathtt{G}$ over $\vee$, as the following example illustrates.

▶ **Example 10.** The formula $\mathtt{G}(p \vee q)$ is not equivalent to $\mathtt{G}p \vee \mathtt{G}q$ because, for example, given an interpretation such that $h(p) = \{t \in \mathbb{Z} \mid t \text{ is odd}\}$ and $h(q) = \{t \in \mathbb{Z} \mid t \text{ is even}\}$, we have that $0 \models \mathtt{G}(p \vee q)$ but $0 \not\models \mathtt{G}p \vee \mathtt{G}q$.

Counterexamples for the distributivity of $\mathtt{F}$ respect to $\wedge$ can be obtained by duality.

In the following proposition, we give the conditions in which $\mathtt{G}$ distributes respect to $\vee$. The result of the distributivity $\mathtt{F}$ respect to $\wedge$ is obtained by duality.

▶ **Proposition 11.** *Let $\{A_i \mid i \in I\}$ be a finite set of wffs in FNext. Then, the following equivalences hold:*

1. *If A_i is projectable to the future, for all $i \in I$, then $\mathtt{G}\big(\bigvee_{i\in I} A_i\big) \equiv \bigvee_{i\in I} \mathtt{G}A_i$.*
2. *If A_i is projectable to the past, for all $i \in I$, then $\mathtt{G}\big(\bigvee_{i\in I} A_i\big) \equiv \bigvee_{i\in I} \mathtt{G}A_i$.*
3. *If $J = \{i \in I \mid A_i \text{ is atemporal}\}$ then $\mathtt{G}\big(\bigvee_{i\in I} A_i\big) \equiv \bigvee_{i\in J} A_i \vee \mathtt{G}\big(\bigvee_{i\in I\setminus J} A_i\big)$.*

Proof. We shall prove the item 1. The proof of item 2 can be obtained in a similar way.

Let $t \in h\big(\mathtt{G}\big(\bigvee_{i\in I} A_i\big)\big)$. Then $(t,\infty) \subseteq h\big(\bigvee_{i\in I} A_i\big) = \bigcup_{i\in I} h(A_i)$. Thus, there exists $i \in I$ such that $t+1 \in h(A_i)$. Since A_i is projectable to the future, then $(t,\infty) \subseteq h(A_i)$ and therefore $t \in h(\mathtt{G}A_i) \subseteq \bigcup_{i\in I} h(\mathtt{G}A_i)$. Thus, $t \in h\big(\bigvee_{i\in I} \mathtt{G}A_i\big)$.

Conversely, let $t \in h\big(\bigvee_{i\in I} \mathtt{G}A_i\big)$. Then $t \in \bigcup_{i\in I} h(\mathtt{G}A_i)$. Thus, there exists $i \in I$ such that $t \in h(\mathtt{G}A_i)$ and therefore $(t,\infty) \subseteq h(A_i) \subseteq \bigcup_{i\in I} h(A_i)$. Then, $t \in h\big(\mathtt{G}\big(\bigvee_{i\in I} A_i\big)\big)$.

For item 3, let $t \in h\big(\mathtt{G}\big(\bigvee_{i\in I} A_i\big)\big)$. Then, $(t,\infty) \subseteq h\big(\bigvee_{i\in I} A_i\big) = \bigcup_{i\in I} h(A_i)$. Therefore, there exists $i \in I$ such that $t+1 \in h(A_i)$. There are two cases: if $i \in I$ also belongs to J, then A_i with $i \in J$ is atemporal, and $h(A_i) = \mathbb{Z}$. Therefore, $t \in h(A_i) \subseteq \bigcup_{i\in J} h(A_i)$. Thus, $t \in h\big(\bigvee_{i\in J} A_i\big)$. The other case is if $i \in I \setminus J$ then $t \in h\big(\mathtt{G}\big(\bigvee_{i\in I\setminus J} A_i\big)\big)$. So, $t \in h\big(\bigvee_{i\in J} A_i \vee \mathtt{G}\big(\bigvee_{i\in I\setminus J} A_i\big)\big)$.

Conversely, let $t \in h\big(\bigvee_{i\in J} A_i \vee \mathtt{G}\big(\bigvee_{i\in I\setminus J} A_i\big)\big)$. Then, $t \in h\big(\bigvee_{i\in J} A_i\big)$ or $t \in h\big(\mathtt{G}\big(\bigvee_{i\in I\setminus J} A_i\big)\big)$. Then, there exists $i \in J$ such that $t \in h(A_i)$. Since A_i is atemporal, $(t,\infty) \subseteq h(A_i) \subseteq \bigcup_{i\in J} h(A_i)$. Thus, $t \in h\big(\mathtt{G}\bigvee_{i\in J} A_i\big)$. Therefore, $t \in h\big(\mathtt{G}\big(\bigvee_{i\in I} A_i\big)\big)$. ◀

3 Stating the problem

Classical induction in temporal logic can be expressed in FNext by the following equivalence $\oplus A \wedge \mathtt{G}(A \to \oplus A) \equiv \mathtt{G}A$. However, induction can be easily extended by considering formulas as $\oplus^n A \wedge \mathtt{G}(A \to \oplus^m A)$. Each model of these formulas has an infinite sequence of instants in which A is true.

Following with this idea to generalize the induction, we consider formulas $A \wedge \mathtt{G}(B \to C)$ where $A \models \mathtt{F}B$ and $C \models \mathtt{F}B$. These formulas satisfy that, for every model $h \models A \wedge \mathtt{G}(B \to C)$, the set $h(C)$ is infinite (an infinite sequence of instants in which C is true exists).

Looking for a balance between the expressiveness and the complexity of management, we are interested in formulas in which A, B and C are literals. That is, we consider induction schemes like $\ell_1 \wedge \mathtt{G}(\ell_2 \rightarrow \ell_3) \equiv \ell_1 \wedge \mathtt{G}(\neg\ell_2 \vee \ell_3)$. Due to the structure of the set of literals, to ensure $\ell_1 \models \mathtt{F}\ell_2$ and $\ell_3 \models \mathtt{F}\ell_2$, it is necessary that $\ell_1, \ell_2, \ell_3 \in Lit(\ell_p)$ for a classical literal ℓ_p.

For some of these formulas, it is possible to find equivalent formulas in which temporal connectives appear only in the literals. For example,

- $\oplus^2 p \wedge \mathtt{G}(\neg p \vee \oplus p) \equiv \mathtt{G} \oplus p$.
- $\mathtt{G} \oplus^4 p \wedge \mathtt{G}(\neg \oplus^2 p \vee \oplus^3 p) \equiv \mathtt{G} \oplus^4 p \wedge (\neg \oplus^3 p \vee \oplus^4 p)$

However, there exist formulas in which it is not possible, such as $\mathtt{F} \oplus^5 p \wedge \mathtt{G}(\neg \oplus^2 p \vee \oplus^4 p)$. The aim of this paper is to distinguish those formulas that can be simplified from those that are not.

To study the induction schemes, $\ell_1 \wedge \mathtt{G}(\ell_2 \rightarrow \ell_3) \equiv \ell_1 \wedge \mathtt{G}(\neg\ell_2 \vee \ell_3)$ such that $\ell_1 \models \mathtt{F}\ell_2$ and $\ell_3 \models \mathtt{F}\ell_2$, we characterize those formulas as $\mathtt{G}(\neg\ell_2 \vee \ell_3)$ for which no simplest equivalent formulas exist. These formulas are going to be named irreducible 2-$\mathtt{G}$-clauses. Observe that, in particular, we are interested in those irreducible 2-$\mathtt{G}$-clauses satisfying $\ell_3 \models \mathtt{F}\ell_2$.

4 Irreducible 2-G-Clauses on p

In this section we center on the first step. That is, formulas $\mathtt{G}(\ell_1 \vee \ell_2)$ that cannot be simplified are going to be characterized. We begin by studying clauses that can be simplified to a literal (i.e. there exists a literal that is equivalent to the clause). Obviously, there are clauses where it is not possible.

It is not difficult to see that, if a clause $\ell_1 \vee \ell_2$ is equivalent to a literal ℓ, then $\sup\{\ell_1, \ell_2\} = \ell$ in the lattice $(Lit, \models)$ by definition of the order relation. For example, $\sup\{\oplus\ell_p, \mathtt{F}\oplus\ell_p\} = \mathtt{F}\ell_p$ and $\mathtt{F}\ell_p \equiv \oplus\ell_p \vee \mathtt{F} \oplus \ell_p$. However, the converse is not true, e.g. $\sup\{\oplus\ell_p, \oplus^2\ell_p\} = \mathtt{F}\ell_p$ but $\mathtt{F}\ell_p \not\equiv \oplus\ell_p \vee \oplus^2\ell_p$.

Now, we characterize the disjunctions of two literals such that there does not exist an equivalent literal.

▶ **Definition 12.** *Let $\ell_1, \ell_2 \in Lit(p) \cup Lit(\neg p)$.*

- *The wff $\ell_1 \vee \ell_2$ is named a* 2-clause on p.
- *$\ell_1 \vee \ell_2$ is said* reducible *if there exists $\ell \in Lit$ such that $\ell_1 \vee \ell_2 \equiv \ell$.*
- *2-$Cla_{irr}(p)$ is the set of irreducible 2-clauses on p.*

The following remark, which provides the cases where a 2-clause can be reduced, is simply a matter of computation from the semantics of $FNext$.

▶ Remark 13. Analyzing exhaustively all the possible pairs of literals and the lattice shown in Figure 1 it is easy to conclude that the cases in which a 2-clause is reducible are the following:

- If $\ell_1, \ell_2 \in Lit(\ell_p)$ with $\ell_1 \models \ell_2$, then $\ell_1 \vee \ell_2 \equiv \ell_2$.
- If $\ell_1 \in Lit(\ell_p)$ and $\ell_2 \in Lit(\neg\ell_p)$ with $\neg\ell_1 \models \ell_2$, then $\ell_1 \vee \ell_2 \equiv \top$.
- $\oplus^{k+1}\ell_p \vee \mathtt{F} \oplus^{k+1} \ell_p \equiv \mathtt{F} \oplus^k \ell_p$ for all $k \in \mathbb{N}$.

To give a characterization of irreducible 2-clauses on p, we introduce the following definitions

▶ **Definition 14.** *Let $\gamma_1, \ldots, \gamma_n \in \{F, G, \oplus\}$ and $\ell_p \in \{p, \neg p\}$. The opposite of the wff $\ell = \gamma_1, \ldots, \gamma_n\ell_p$ is defined as $\overline{\ell} = \overline{\gamma_1}, \ldots, \overline{\gamma_n}\overline{\ell_p}$ where: $\overline{F} = G$, $\overline{G} = F$, $\overline{\oplus} = \oplus$, $\overline{p} = \neg p$ and $\overline{\neg p} = p$.*

The following proposition follows from Remark 13

▶ **Proposition 15** (Characterization of irreducible 2-clauses on p)**.** *The elements of* 2-$Cla_{irr}(p)$ *are the following:*

1. $\oplus^n \ell_p \vee \oplus^m \ell_p$*, for all* $m, n \in \mathbb{N}$ *with* $m \neq n$ *and* $\ell_p \in \{p, \overline{p}\}$.
2. $\oplus^n \ell_p \vee \ell$*, for all* $n \in \mathbb{N}$, $\ell_p \in \{p, \overline{p}\}$ *and for all* $\ell \in \{\mathtt{F} \oplus^m \ell_p \mid m > n\} \cup \{\mathtt{GF}\ell_p, \mathtt{FG}\ell_p\} \cup \{\mathtt{G} \oplus^m \ell_p \mid m \geq n\}$
3. $\ell_1 \vee \ell_2$*, where* $\ell_1 \in Lit(\ell_p)$, $\ell_2 \in Lit(\overline{\ell_p})$ *such that* $\overline{\ell_1} \not\models \ell_2$.

We have just characterized irreducible 2-clauses on p and now we properly center on studying formulas $\mathtt{G}(\ell_1 \vee \ell_2)$ that cannot be simplified.

▶ **Definition 16.** *Let* p *be a propositional variable. Then*

- *A* 2-G-clause on p *is a wff* $\mathtt{G}A$ *where* $A \in$ 2-$Cla_{irr}(p)$.
- *A* 2-G-clause on p, $\mathtt{G}A$, *is said to be* reducible *if there exists* $\ell \in Lit$ *such that* $\mathtt{G}A \equiv \ell$ *or there exists* $B \in$ 2-$Cla_{irr}(p)$ *such that* $\mathtt{G}A \equiv B$.
- 2-$\mathtt{G}$-$Cla_{irr}(p)$ *is the set of irreducible 2-*G*-clauses on* p.

With the idea of simplifying formulas in induction schemes, we are interested in the reduction from 2-G-clauses to 2-clauses or literals, when it is possible. Thus, the equivalences provided in Proposition 2 and Proposition 11 are to be read from left to right. For example, since $\mathtt{G}(\mathtt{F}p \vee \mathtt{F} \oplus \overline{p}) \equiv \mathtt{GF}p \vee \mathtt{GF} \oplus \overline{p}$ because of Proposition 11 (both disjuncts are past projectable) and, by Proposition 2, $\mathtt{GF} \oplus \overline{p} \equiv \mathtt{GF}\overline{p}$, then $\mathtt{G}(\mathtt{F}p \vee \mathtt{F} \oplus \overline{p})$ can be reduced to $\mathtt{GF}p \vee \mathtt{GF}\overline{p}$. Moreover, $\mathtt{GF}p \vee \mathtt{GF}\overline{p} \equiv \top$ by Remark 13

Now, to give a characterization of irreducible 2-G-clauses on p, we introduce new reduction laws for the 2-G-clauses on p. The results will be established (if it is possible) in their broader extent.

▶ **Proposition 17.** *Let* $A \in \mathcal{L}$ *and* $n, m \in \mathbb{N}$. *The following equivalences hold:*

1. $\mathtt{G}(\mathtt{F} \oplus^n A \vee \oplus^m A) \equiv \mathtt{GF}A$.
2. *If* $n > m$ *then*

$$\mathtt{G}(\mathtt{F} \oplus^n A \vee \mathtt{G} \oplus^m \neg A) \equiv \mathtt{GF}A \vee \mathtt{G} \oplus^{m+1} \neg A \equiv \mathtt{G}(\mathtt{F} \oplus^n A \vee \oplus^{m+1} \neg A)$$

Observe that, item 2 in proposition above only considers the case in which $n > m$ because otherwise $\mathtt{G}(\mathtt{F} \oplus^n A \vee \mathtt{G} \oplus^m \neg A)$ and $\mathtt{G}(\mathtt{F} \oplus^n A \vee \oplus^{m+1} \neg A)$ are not 2-G-clauses. In fact, when $n \leq m$, $\mathtt{F} \oplus^n A \vee \mathtt{G} \oplus^m \neg A \equiv \mathtt{F} \oplus^n A \vee \oplus^{m+1} \neg A \equiv \top$.

Proof. For item 1, it is a trivial task to check that $\mathtt{F} \oplus^n A \vee \oplus^m A \models \mathtt{F}A$ and therefore, by monotonicity of G, we have that $\mathtt{G}(\mathtt{F} \oplus^n A \vee \oplus^m A) \models \mathtt{GF}A$. Conversely, $\mathtt{GF}A \equiv \mathtt{GF} \oplus^n A$ by Proposition 2, and $\mathtt{GF}A \models \mathtt{G}(\mathtt{F} \oplus^n A \vee \oplus^m A)$.

For the equivalence $\mathtt{G}(\mathtt{F} \oplus^n A \vee \mathtt{G} \oplus^m \neg A) \equiv \mathtt{GF}A \vee \mathtt{G} \oplus^{m+1} \neg A$ in item 2, it is trivial that $\mathtt{GF}A \vee \mathtt{G} \oplus^{m+1} \neg A \models \mathtt{G}(\mathtt{F} \oplus^n A \vee \mathtt{G} \oplus^m \neg A)$. Conversely, assume $n > m$ and $t \in \mathtt{G}(\mathtt{F} \oplus^n A \vee \mathtt{G} \oplus^m \neg A)$. We can distinguish two cases:

- If $(t, \infty) \subseteq h(\mathtt{F} \oplus^n A)$, then $t \in h(\mathtt{GF} \oplus^n A) = h(\mathtt{GF}A) \subseteq h(\mathtt{GF}A \vee \mathtt{G} \oplus^{m+1} \neg A)$.
- Otherwise, there are instants greater than t that do not belong to $h(\mathtt{F} \oplus^n A)$. Then, let t_0 be the smallest of them, i.e. $t_0 = min(\,(t, \infty) \cap h(\mathtt{G} \oplus^m \neg A)\,)$. Thus, $t_0 + m \in h(\mathtt{G}\neg A)\,(\dagger_2)$. By definition of t_0, we have $t_0 - 1 \in h(\mathtt{F} \oplus^n A)\ (\dagger_1)$, i.e. $t_0 - 1 + n \in h(\mathtt{F}A)$, and since $n > m$ we get $t_0 - 1 + n \geq t_0 + m$ and so, we should have that $(\dagger_2)$ contradicts to $(\dagger_1)$, which is not possible except that $t_0 = t + 1$, in which case $t \in h(\mathtt{G} \oplus^{m+1} \neg A) \subseteq h(\mathtt{GF}A \vee \mathtt{G} \oplus^{m+1} \neg A)$.

Finally, to prove $\mathtt{GF}A \vee \mathtt{G} \oplus^{m+1} \neg A \equiv \mathtt{G}(\mathtt{F} \oplus^n A \vee \oplus^{m+1} \neg A)$ when $n > m$, we have that $\mathtt{GF}A \vee \mathtt{G} \oplus^{m+1} \neg A \models \mathtt{G}(\mathtt{F} \oplus^n A \vee \oplus^{m+1} \neg A)$ is trivial.

Conversely, suppose $t \in h(\mathtt{G}(\mathtt{F} \oplus^n A \vee \oplus^{m+1} \neg A))$ $(\dagger)$. We have two cases:

- If $[t+1,\infty) \subseteq h(\oplus^{m+1}\neg A)$, then clearly we obtain $t \in h(\mathtt{GF}A \vee \mathtt{G}\oplus^{m+1}\neg A)$.
- Otherwise, $\Lambda = [t+1,\infty) \cap h(\oplus^{m+1}A) \neq \varnothing$. We shall prove that Λ is infinite, and therefore $t \in h(\mathtt{GF}A) \subseteq h(\mathtt{GF}A \vee \mathtt{G}\oplus^{m+1}\neg A)$. If Λ is finite, consider $t_0 = max\Lambda$. Then $[t_0+1,\infty) \subseteq h(\oplus^{m+1}\neg A)$ (††). On the other hand, $t_0 \in h(\oplus^{m+1}A)$, so by the hypothesis (†) we get $t_0 \in h(\mathtt{F}\oplus^{n} A)$, that is, $(t_0+1,\infty) \cap h(\oplus^{m+1}A) \neq \varnothing$ (since $n > m$), in contradiction with (††). ◀

In Table 1, we collect the laws previously obtained, which are denoted as $\mathtt{Red}_i$. These laws are going to be considered as rewriting rules to transform 2-G-clauses into 2-clauses and are always read from left to right.

Table 1 G-Reduction Laws.

Law	Name
$\mathtt{G}(A \vee B) \equiv \mathtt{G}A \vee \mathtt{G}B$, if A and B are past projectable, or A and B are future projectable	$\mathtt{Red}_1$
$\mathtt{G}(A \vee B) \equiv \mathtt{G}A \vee B$, if B is atemporal	$\mathtt{Red}_2$
$\mathtt{G}(\mathtt{F}\oplus^{n} A \vee \oplus^{m} A) \equiv \mathtt{GF}A$	$\mathtt{Red}_3$
$\mathtt{G}(\mathtt{F}\oplus^{n} A \vee \mathtt{G}\oplus^{m}\neg A) \equiv \mathtt{GF}A \vee \mathtt{G}\oplus^{m+1}\neg A$, if $n > m$	$\mathtt{Red}_4$
$\mathtt{G}(\mathtt{F}\oplus^{n} A \vee \oplus^{m+1}\neg A) \equiv \mathtt{GF}A \vee \mathtt{G}\oplus^{m+1}\neg A$, if $n > m$	$\mathtt{Red}_5$

Reductions $\mathtt{Red}_1$ and $\mathtt{Red}_2$ are due to Proposition 11. On the other hand, $\mathtt{Red}_3$, $\mathtt{Red}_4$ and $\mathtt{Red}_5$ will be applied only in 2-G-clauses and hence, reductions correspond with the equivalences given in Proposition 17.

These equivalences are those that ensure a 2-G-clause on p is irreducible only in the case that none of the reductions $\mathtt{Red}_i$ with $1 \leq i \leq 5$ is applicable to it. The following characterization theorem for irreducible 2-G-clauses on p is a consequence of this assertion.

▶ **Theorem 18** (Characterization of Irreducible 2-G-clauses on p). *The elements of the set 2-G-$Cla_{irr}(p)$ are the following:*

(a) $\mathtt{G}(\oplus^{n}\ell_p \vee \oplus^{m}\ell_p)$, *if* $n \neq m$
(b) $\mathtt{G}(\oplus^{n}\ell_p \vee \oplus^{m}\overline{\ell_p})$, *if* $n \neq m$
(c) $\mathtt{G}(\oplus^{n}\ell_p \vee \mathtt{G}\oplus^{m}\ell_p)$, *if* $n \leq m$
(d) $\mathtt{G}(\oplus^{n}\ell_p \vee \mathtt{G}\oplus^{m}\overline{\ell_p})$

Proof. The proof is divided in two parts. In the first, we discard such 2-G-clauses on p that previous results ensure are reducible. Thus, we obtain that formulas labeled *a),b),c),d)* in the theorem are those that cannot be reduced through previous results. In the second part, we prove that they are really irreducible.

From Definition 16 and Proposition 15, a 2-G-clause on p is a formula $\mathtt{G}A$ such that A is of one of the following forms:

(i) $\oplus^{n}\ell_p \vee \oplus^{m}\ell_p$, with $m, n \in \mathbb{N}$ and $m \neq n$.
(ii) $\oplus^{n}\ell_p \vee \ell$, where $\ell \in \{\mathtt{F}\oplus^{m}\ell_p \mid m > n\} \cup \{\mathtt{GF}\ell_p, \mathtt{FG}\ell_p\} \cup \{\mathtt{G}\oplus^{m}\ell_p \mid m \geq n\}$
(iii) $\ell_1 \vee \ell_2$, where $\ell_1 \in Lit(\ell_p)$, $\ell_2 \in Lit(\overline{\ell_p})$ and $\overline{\ell_1} \not\models \ell_2$.

If $\mathtt{G}A$ is a 2-G-clause such that A is a clause of the type described in item i) then $\mathtt{G}A$ cannot be reduced through previous results (item *a)* of the theorem).

Now, let $\mathtt{G}A$ be 2-$\mathtt{G}$-clause such that A is a clause of the type described in item ii). Then, we discard such 2-$\mathtt{G}$-clauses on p which previous results ensured are reducible.

- if $\ell \in \{\mathtt{GF}\ell_p, \mathtt{FG}\ell_p\}$ then from Proposition 11 (reduction $\mathtt{Red}_2$), $\mathtt{G}A$ is reducible, and
- if $\ell \in \{\mathtt{F} \oplus^m \ell_p \mid m > n\}$ then from Proposition 17 (reduction $\mathtt{Red}_3$), $\mathtt{G}A$ is reducible.
- if $\ell \in \{\mathtt{G} \oplus^m \ell_p \mid m \geq n\}$ then $\mathtt{G}A$ cannot be reduced through previous results (item *c)* of the theorem).

For item iii) we have to analyze the 11 elements stated in the following table:

(1)	$\mathtt{G}(\oplus^n \ell_p \vee \oplus^m \overline{\ell}_p)$ $n \neq m$	(2)	$\mathtt{G}(\oplus^n \ell_p \vee \mathtt{G} \oplus^m \overline{\ell}_p)$	(3)	$\mathtt{G}(\oplus^n \ell_p \vee \mathtt{F} \oplus^m \overline{\ell}_p)$
(4)	$\mathtt{G}(\oplus^n \ell_p \vee \mathtt{GF}\overline{\ell}_p)$	(5)	$\mathtt{G}(\oplus^n \ell_p \vee \mathtt{FG}\overline{\ell}_p)$	(6)	$\mathtt{G}(\mathtt{G} \oplus^n \ell_p \vee \mathtt{G} \oplus^m \overline{\ell}_p)$
(7)	$\mathtt{G}(\mathtt{G} \oplus^n \ell_p \vee \mathtt{F} \oplus^m \overline{\ell}_p)$	(8)	$\mathtt{G}(\mathtt{G} \oplus^n \ell_p \vee \mathtt{GF}\overline{\ell}_p)$	(9)	$\mathtt{G}(\mathtt{G} \oplus^n \ell_p \vee \mathtt{FG}\overline{\ell}_p)$
(10)	$\mathtt{G}(\mathtt{F} \oplus^n \ell_p \vee \mathtt{F} \oplus^m \overline{\ell}_p)$	(11)	$\mathtt{G}(\mathtt{FG}\ell_p \vee \mathtt{FG}\overline{\ell}_p)$		

In the previous table, from Proposition 11 (reduction $\mathtt{Red}_1$) items (6), (8), (9), (10) and (11) are eliminated (in each of them the two literals of 2-clauses are past projectable or the two literals of 2-clauses are future projectable).

From Proposition 11 (reduction $\mathtt{Red}_2$) items (4) and (5) are eliminated because in each of them the 2-clauses have atemporal literals.

From Proposition 17 item (7) is eliminated (reduction $\mathtt{Red}_4$) and item (3) is eliminated (reduction $\mathtt{Red}_5$).

Finally, items (1) and (2) cannot be removed because these formulas cannot be reduced by previous results of the theorem: (items *b)* and *d)* respectively).

At this moment, we have proved that 2-$\mathtt{G}$-clauses on p different to those that are described in the theorem can be reduced by using previous results. Now, we focus on proving that formulas $\mathtt{G}A$ described in items *a)*, *b)*, *c)* and *d)* are really irreducible. Thus, we prove that neither $\mathtt{G}A \not\equiv \ell$ for a literal ℓ, nor $\mathtt{G}A \not\equiv \ell_1 \vee \ell_2$ for $\ell_1 \vee \ell_2 \in 2\text{-}Cla_{irr}(p)$. Note that, collecting literals and clauses, there are 40 formulas that could be equivalent to $\mathtt{G}A$. Obviously, $\top$ and $\bot$ are discarded.

First, we prove $\mathtt{G}A$ is not equivalent to any of the following formulas: $\oplus^k \ell_p$, $\mathtt{F} \oplus^k \ell_p$, $\mathtt{GF}\ell_p$. Consider $k \in \mathbb{N}$ and $r = \max\{k+1, n+1, m+1\}$. For any interpretation h such that $h(\ell_p) = \bigcup_{i \in \mathbb{N}}[2ir, 2ir+1)$, we have that $0 \in h(\oplus^k \ell_p)$, $0 \in h(\mathtt{F} \oplus^k \ell_p)$ and $0 \in h(\mathtt{GF}\ell_p)$ but $0 \notin h(\mathtt{G}A)$. Therefore, $\oplus^k \ell_p \not\models \mathtt{G}A$, $\mathtt{F} \oplus^k \ell_p \not\models \mathtt{G}A$ and $\mathtt{GF}\ell_p \not\models \mathtt{G}A$.

With a similar reasoning, it can be proved that $\oplus^k \overline{\ell_p} \not\models \mathtt{G}A$, $\mathtt{F} \oplus^k \overline{\ell_p} \not\models \mathtt{G}A$ and $\mathtt{GF}\overline{\ell_p} \not\models \mathtt{G}A$. Moreover, any 2-clause $\ell_1 \vee \ell_2$ with ℓ_1 or ℓ_2 being one of the previous cases satisfies $\ell_1 \vee \ell_2 \not\models \mathtt{G}A$.

The rest of the formulas are:

$\mathtt{G} \oplus^k \ell_p$	$\mathtt{G} \oplus^k \overline{\ell_p}$	$\mathtt{FG}\ell_p$	$\mathtt{FG}\overline{\ell_p}$
$\mathtt{G} \oplus^{k_1} \ell_p \vee \mathtt{G} \oplus^{k_2} \overline{\ell_p}$	$\mathtt{G} \oplus^k \ell_p \vee \mathtt{FG}\overline{\ell_p}$	$\mathtt{FG}\ell_p \vee \mathtt{FG}\overline{\ell_p}$	

Now we prove that none of the formulas described in the theorem is equivalent to any of them.

(a) Let us consider h with $h(\ell_p) = \mathbb{Z} \setminus \{(|m-n|+1)^r \mid r \in \mathbb{N} \setminus \{0\}\}$, where $|m-n|$ denotes the absolute value of $m-n$. It is just a matter of computation to check that $0 \in h(\mathtt{G}(\oplus^n \ell_p \vee \oplus^m \ell_p))$ but it is not a model for formulas in the previous table.

(b) Let us suppose, without loss of generality, that $n < m$ and consider h with $h(\ell_p) = \bigcup_{i \in \mathbb{N}}[n + 2(m-n)i, m + 2(m-n)i)$. In this case, $0 \in h(\mathtt{G}(\oplus^n \ell_p \vee \oplus^m \overline{\ell_p}))$ but is not model for formulas in the previous table.

(c) For $\mathtt{G}(\oplus^n \ell_p \vee \mathtt{G} \oplus^m \ell_p)$ when $n \leq m$, the following cases are going to be considered:

- Any interpretation h such that $h(\ell_p) = [k+1, \infty)$ satisfies $0 \in h(\mathtt{FG}\ell_p)$ and $0 \in h(\mathtt{G} \oplus^k \ell_p)$. However, $0 \notin h(\mathtt{G}(\oplus^n \ell_p \vee \mathtt{G} \oplus^m \ell_p))$ when $m < k$. Therefore, $\mathtt{FG}\ell_p \not\models \mathtt{G}(\oplus^n \ell_p \vee \mathtt{G} \oplus^m \ell_p)$ and, when $m < k$, we have that $\mathtt{G} \oplus^k \ell_p \not\models \mathtt{G}(\oplus^n \ell_p \vee \mathtt{G} \oplus^m \ell_p)$.
- For any interpretation such that $0 \in h(\mathtt{G} \oplus^k \overline{\ell_p})$ satisfies $0 \in h(\mathtt{FG}\overline{\ell_p})$, but $0 \notin h(\mathtt{G}(\oplus^n \ell_p \vee \mathtt{G} \oplus^m \ell_p))$.
- If $\{\ell_1, \ell_2\} \cap \left(\{\mathtt{FG}\ell_p, \mathtt{FG}\overline{\ell_p}\} \cup \{\mathtt{G} \oplus^k \ell_p \mid m < k\} \cup \{\mathtt{G} \oplus^k \overline{\ell_p} \mid k \in \mathbb{N}\}\right) \neq \varnothing$ then $\ell_1 \vee \ell_2 \not\models \mathtt{G}(\oplus^n \ell_p \vee \mathtt{G} \oplus^m \ell_p)$.
- Finally, $\mathtt{G} \oplus^k \ell_p$ with $k \leq m$ is the last formula to be considered. In this case, any interpretation h such that $h(\ell_p) = \mathbb{Z} \setminus \{k+2\}$ satisfies that $0 \in h(\mathtt{G}(\oplus^n \ell_p \vee \mathtt{G} \oplus^m \ell_p))$ but $0 \notin h(\mathtt{G} \oplus^k \ell_p)$.

(d) With a similar reasoning to the above, it can be proved that the formula $\mathtt{G}(\oplus^n \ell_p \vee \mathtt{G} \oplus^m \overline{\ell_p})$ is not equivalent to any formula of the last table. ◀

5 Inductive Schemes

Observe that, from the kind of irreducible 2-G-clauses $\mathtt{G}(\neg\ell_1 \vee \ell_2)$ characterized in Theorem 18, only the items b) and d) could satisfy condition $\ell_2 \models \mathtt{F}\ell_1$. The following definition sums up all the conditions step by step previously introduced relative to inductive schemes.

▶ **Definition 19.** *An* ***inductive scheme on*** ℓ_p *is a formula* $\ell_1 \wedge \mathtt{G}(\bar{\ell}_2 \vee \ell_3)$*, where* $\ell_1, \ell_2, \ell_3 \in Lit(\ell_p)$ *which satisfies the following three conditions:*

Ind-1: $\mathtt{G}(\bar{\ell}_2 \vee \ell_3) \in 2\text{-}\mathtt{G}\text{-}Cla_{irr}(p)$*,*

Ind-2: $\ell_1 \models \mathtt{F}\ell_2$ *and* $\ell_3 \models \mathtt{F}\ell_2$*.*

▶ **Example 20.** $\mathtt{F} \oplus^5 p \wedge \mathtt{G}(\neg \oplus^2 p \vee \oplus^4 p)$ is an inductive scheme on p.

We introduce the main result that characterises those inductive formulas that cannot be simplified. Specifically, among of the 125 kind of formulas $\ell_1 \wedge \mathtt{G}(\bar{\ell}_2 \vee \ell_3)$ (i.e., 5^3, corresponding to non-trivial possible selections ℓ_1, ℓ_2 and ℓ_3 from the types of literals $\oplus^k \ell_p$, $\mathtt{F} \oplus^k \ell_p$, $\mathtt{G} \oplus^k \ell_p$, $\mathtt{FG}\ell_p$ and $\mathtt{GF}\ell_p$), only 15 satisfy `Ind-1`. Condition `Ind-2` only imposes restrictions relative to the super-index of the $\oplus$ connectives but the number of kind of formulas remains at being 15, which are enumerated in Table 2.

6 Conclusions and future works

In the framework of the propositional linear discrete temporal logic $FNext$, we have studied those formulas with a single propositional variable that involve the well-known idea of induction. Indeed, we have classified these formulas into those that can be expressed by equivalent ones without loops and the rest (inductive formulas). The main issue of this paper has been to study the set of inductive formulas in order to determine schemes that characterize them.

The starting point is the set of expressions $\ell_1 \wedge \mathtt{G}(\bar{\ell}_2 \vee \ell_3)$ which includes 125 kinds of formulas (i.e., 5^3, corresponding to non-trivial possible selections ℓ_1, ℓ_2 and ℓ_3 from the types of literals $\oplus^k \ell_p$, $\mathtt{F} \oplus^k \ell_p$, $\mathtt{G} \oplus^k \ell_p$, $\mathtt{FG}\ell_p$ and $\mathtt{GF}\ell_p$).

In Section 4, we have characterized those formulas as $\mathtt{G}(\bar{\ell}_2 \vee \ell_3)$ for which no simplest equivalent formulas exist. These formulas have been named irreducible 2-G-clauses. Specifically, Theorem 18 provides four schemes that cover all the irreducible 2-G-clauses and allows us to reduce the initial number to 15 kinds of formulas.

Table 2 Formulas $\ell_1 \wedge \mathtt{G}(\overline{\ell}_2 \vee \ell_3)$ satisfying `Ind-1` and `Ind-2`.

1. $\oplus^{n_1} \ell_p \wedge \mathtt{G}(\oplus^{n_2}\overline{\ell}_p \vee \oplus^{n_3}\ell_p)$, where $n_1, n_3 > n_2$	9. $\mathtt{FG}\ell_p \wedge \mathtt{G}(\oplus^{n_2}\overline{\ell}_p \vee \mathtt{G} \oplus^{n_3} \ell_p)$
2. $\mathtt{F} \oplus^{n_1} \ell_p \wedge \mathtt{G}(\oplus^{n_2}\overline{\ell}_p \vee \oplus^{n_3}\ell_p)$, where $n_1 \geq n_2$ and $n_3 > n_2$	10. $\mathtt{G} \oplus^{n_1} \ell_p \wedge \mathtt{G}(\oplus^{n_2}\overline{\ell}_p \vee \mathtt{G} \oplus^{n_3} \ell_p)$, where $n_1, n_3 \geq n_2$
3. $\mathtt{FG}\ell_p \wedge \mathtt{G}(\oplus^{n_2}\overline{\ell}_p \vee \oplus^{n_3}\ell_p)$, where $n_3 > n_2$	11. $\oplus^{n_1} \ell_p \wedge \mathtt{G}(\mathtt{G} \oplus^{n_2} \overline{\ell}_p \vee \oplus^{n_3}\ell_p)$, where $n_1, n_3 > n_2 + 1$
4. $\mathtt{F} \oplus^{n_1} \ell_p \wedge \mathtt{G}(\oplus^{n_2}\overline{\ell}_p \vee \mathtt{G} \oplus^{n_3} \ell_p)$, where $n_1 \geq n_2$	12. $\mathtt{F} \oplus^{n_1} \ell_p \wedge \mathtt{G}(\mathtt{G} \oplus^{n_2} \overline{\ell}_p \vee \oplus^{n_3}\ell_p)$, where $n_1 > n_2$ and $n_3 \geq n_2$
5. $\mathtt{GF}\ell_p \wedge \mathtt{G}(\oplus^{n_2}\overline{\ell}_p \vee \oplus^{n_3}\ell_p)$, where $n_3 > n_2$	13. $\mathtt{GF}\ell_p \wedge \mathtt{G}(\mathtt{G} \oplus^{n_2} \overline{\ell}_p \vee \oplus^{n_3}\ell_p)$, where $n_3 > n_2 + 1$
6. $\mathtt{G} \oplus^{n_1} \ell_p \wedge \mathtt{G}(\oplus^{n_2}\overline{\ell}_p \vee \oplus^{n_3}\ell_p)$, where $n_1 \geq n_2$ and $n_3 > n_2$	14. $\mathtt{FG}\ell_p \wedge \mathtt{G}(\mathtt{G} \oplus^{n_2} \overline{\ell}_p \vee \oplus^{n_3}\ell_p)$, where $n_3 > n_2 + 1$
7. $\oplus^{n_1} \ell_p \wedge \mathtt{G}(\oplus^{n_2}\overline{\ell}_p \vee \mathtt{G} \oplus^{n_3} \ell_p)$, where $n_1 > n_2$	15. $\mathtt{G} \oplus^{n_1} \ell_p \wedge \mathtt{G}(\mathtt{G} \oplus^{n_2} \overline{\ell}_p \vee \oplus^{n_3}\ell_p)$, where $n_3 > n_2 + 1$
8. $\mathtt{GF}\ell_p \wedge \mathtt{G}(\oplus^{n_2}\overline{\ell}_p \vee \mathtt{G} \oplus^{n_3} \ell_p)$	

Not all of these formulas correspond with the idea of induction because it is necessary that $\ell_1 \models \mathtt{F}\ell_2$ and $\ell_3 \models \mathtt{F}\ell_2$ (Condition `Ind-2`). This condition only imposes restrictions relative to the super-index. Thus, the number of kinds of formulas remains at 15, which are enumerated in Table 2.

To emphasize the interest of the theoretical results of this paper, as further work, we will study how to improve the definition of Temporal Negative Normal Form for the Temporal Logic introduced in [18]. It will have a significant relevance in the future design of efficient automated theorem provers.

In the midterm, the study of inductive schemes developed in this paper could be extended to a fully expressive temporal logic such as Hans Kamp's US logic or LN logic [4].

References

1 Kyungmin Bae and José Meseguer. Model checking linear temporal logic of rewriting formulas under localized fairness. *Science of Computer Programming*, 99:193–234, 2015. doi:10.1016/j.scico.2014.02.006.

2 Philippe Balbiani, Andreas Herzig, François Schwarzentruber, and Nicolas Troquard. DL-PA and DCL-PC: model checking and satisfiability problem are indeed in PSPACE. *CoRR*, abs/1411.7825, 2014. arXiv:1411.7825.

3 Nikolaj Bjorner, Anca Browne, Eddie Chang, Michael Colón, Arjun Kapur, Zohar Manna, Henny B. Sipma, and Tomás E. Uribe. STeP: Deductive-algorithmic verification of reactive and real-time systems: Deductive-algorithmic verification of reactive and real-time systems. In Rajeev Alur and Thomas A. Henzinger, editors, *Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 415–418. Springer Berlin Heidelberg, 1996. doi:10.1007/3-540-61474-5_92.

4 Alfredo Burrieza and Inma Pérez de Guzmán. A new algebraic semantic approach and some adequate connectives for computation with temporal logic over discrete time. *Journal of Applied Non-Classical Logics*, 2(2):181–200, 1992. doi:10.1080/11663081.1992.10510781.

5 E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-state Concurrent Systems Using Temporal Logic Specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, April 1986. doi:10.1145/5397.5399.

6 Pablo Cordero, Manuel Enciso, and Inma Pérez de Guzmán. Bases for closed sets of implicants and implicates in temporal logic. *Acta Inf.*, 38(9):599–619, 2002. doi:10.1007/s00236-002-0087-2.

7 Anatoli Degtyarev, Michael Fisher, and Boris Konev. A Simplified Clausal Resolution Procedure for Propositional Linear-Time Temporal Logic. In Uwe Egly and Chritian G. Fermüller, editors, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 2381 of *Lecture Notes in Computer Science*, pages 85–99. Springer Berlin Heidelberg, 2002. doi:10.1007/3-540-45616-3_7.

8 Stéphane Demri and Philippe Schnoebelen. The Complexity of Propositional Linear Temporal Logics in Simple Cases. *Information and Computation*, 174(1):84–103, 2002. doi:10.1006/inco.2001.3094.

9 Razvan Diaconescu. Structural induction in institutions. *Information and Computation*, 209(9):1197–1222, 2011. doi:10.1016/j.ic.2011.06.002.

10 Stephan Falke and Deepak Kapur. Rewriting Induction + Linear Arithmetic = Decision Procedure. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning*, volume 7364 of *Lecture Notes in Computer Science*, pages 241–255. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-31365-3_20.

11 Michael Fisher. *An Introduction to Practical Formal Methods Using Temporal Logic*. Wiley, 2011.

12 Michael Fisher, Clare Dixon, and Martin Peim. Clausal Temporal Resolution. *ACM Trans. Comput. Logic*, 2(1):12–56, 2001. doi:10.1145/371282.371311.

13 Nicoletta De Francesco, Antonella Santone, and Gigliola Vaglini. A user-friendly interface to specify temporal properties of concurrent systems. *Information Sciences*, 177(1):299–311, 2007. doi:10.1016/j.ins.2006.03.008.

14 Richard Jensen, Andrew Tuson, and Qiang Shen. Finding rough and fuzzy-rough set reducts with {SAT}. *Information Sciences*, 255(0):100–120, 2014. doi:10.1016/j.ins.2013.07.033.

15 Hans Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Ucla, 1968.

16 Savas Konur. A survey on temporal logics for specifying and verifying real-time systems. *Frontiers of Computer Science*, 7(3):370–403, 2013. doi:10.1007/s11704-013-2195-2.

17 Artur Meski, Wojciech Penczek, and Grzegorz Rozenberg. Model checking temporal properties of reaction systems. *Information Sciences*, 313:22–42, 2015. doi:10.1016/j.ins.2015.03.048.

18 Manuel Enciso Pablo Cordero and Inmaculada P. de Guzmán. From the poset of literals to a temporal negative normal form. *Reports on Mathematical Logic*, 36:3–53, 2002.

19 Amir Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57, 1977. doi:10.1109/SFCS.1977.32.

20 José Manuel Rodríguez-Jiménez, Pablo Cordero, Manuel Enciso, and Angel Mora. Negative Attributes and Implications in Formal Concept Analysis. In Fuad Aleskerov, Yong Shi, and Alexander Lepskiy, editors, *Proceedings of the Second International Conference on Information Technology and Quantitative Management, ITQM 2014, National Research University Higher School of Economics (HSE), Moscow, Russia, June 3-5, 2014*, volume 31 of *Procedia Computer Science*, pages 758–765. Elsevier, 2014. doi:10.1016/j.procs.2014.05.325.

21 A. P. Sistla and E. M. Clarke. The Complexity of Propositional Linear Temporal Logics. *J. ACM*, 32(3):733–749, 1985. doi:10.1145/3828.3837.

On Verifying Timed Hyperproperties

Hsi-Ming Ho
University of Cambridge, UK
hsi-ming.ho@cl.cam.ac.uk

Ruoyu Zhou
University of Cambridge, UK
ruoyu.zhou@cl.cam.ac.uk

Timothy M. Jones
University of Cambridge, UK
timothy.jones@cl.cam.ac.uk

Abstract

We study the satisfiability and model-checking problems for *timed hyperproperties* specified with HyperMTL, a timed extension of HyperLTL. Depending on whether interleaving of events in different traces is allowed, two possible semantics can be defined for timed hyperproperties: *synchronous* and *asynchronous*. While the satisfiability problem can be decided similarly as for HyperLTL regardless of the choice of semantics, we show that the model-checking problem for HyperMTL, unless the specification is alternation-free, is undecidable even when very restricted timing constraints are allowed. On the positive side, we show that model checking HyperMTL with quantifier alternations is possible under certain conditions in the synchronous semantics, or when there is a fixed bound on the length of the time domain.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Timed Automata, Temporal Logics, Cybersecurity

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.20

Related Version A full version of the paper is available at `https://arxiv.org/pdf/1812.10005`.

Funding This work was supported by the Engineering and Physical Sciences Research Council (EPSRC), through grant references EP/K026399/1 and EP/P020011/1.

1 Introduction

Background. One of the most popular specification formalisms for reactive systems is *Linear Temporal Logic* (LTL), first introduced into computer science by Pnueli [52] in the late 1970s. The success of LTL can be attributed to the fact that its satisfiability and model-checking problems are of lower complexity (PSPACE-complete, as compared with non-elementary for the equally expressive first-order logic of order) and it enjoys simple translations into automata and excellent tool support (e.g., [15, 35]).

While LTL is adequate for describing features of *individual* execution traces, many security policies in practice are based on relations between *two (or more)* execution traces. A standard example of such properties is *observational determinism* [37, 54, 59]: for every pair of execution traces, if the low-security inputs agree in both execution traces, then the low-security outputs in both execution traces must agree as well. Such properties are called *hyperproperties* [17]: a model of the property is not a single execution trace but a set of execution traces. HyperLTL [16], obtained from LTL by adding *trace quantifiers*, has been proposed as a specification formalism to express hyperproperties. For example, operational determinism can be expressed as the HyperLTL formula:

$$\forall \pi_a \, \forall \pi_b \; \mathbf{G}(I_a = I_b) \Rightarrow \mathbf{G}(O_a = O_b) \, .$$

HyperLTL inherits almost all the benefits of LTL; in particular, tools that support HyperLTL verification can be built by leveraging existing tools for LTL.

For many applications, however, in addition to the occurences and orders of events, *timing* has to be accounted for as well. For example, one may want to verify that in every execution trace of the system, whenever a request `req` is issued, the corresponding acknowledgement `ack` is received within the next 5 time units. *Timed automata* [4] and *timed logics* [5, 9, 39] are introduced exactly for this purpose. In the context of security, timing anomalies caused by different high-security inputs is a realistic attack vector that can be exploited to obtain sensitive information; this kind of *timing side-channel attacks* also play significant roles in high-profile exploits like Meltdown [45] and Spectre [38]. In order to detect such undesired characteristics of systems, one needs to reason about *timed hyperproperties*.

▶ **Example 1** ([55]). A piece of C code that selects between two variables `x` and `y` based on a secret selection bit `b` (i.e. the user gets the output – either `x` or `y` – but does not know which one was actually selected) may be written as follows:

```
uint32_t select_u32(uint32_t b, uint32_t x, uint32_t y)
{
    return b ? x : y;
}
```

This straightforward implementation, however, may result in a timing side channel – depending on what compiler optimisations are applied, the execution time can depend on which of `x` and `y` is returned. In sensitive applications like cryptography libraries and embedded smartcard software, such code snippets are usually replaced by obfuscated, functional-equivalent versions, with the hope of eliminating the potential leakage of secret information. In this case, one such version is as follows:

```
uint32_t ct_select_u32(uint32_t b, uint32_t x, uint32_t y)
{
    signed bit = 0 - b;
    return (x & bit) | (y & ~bit);
}
```

Nevertheless, such attempts of obfuscation can easily be wiped out by more agressive code optimisations. For instance, after compilation by `clang 3.3` (`-O2`), the C code above results in the following assembly code, which contains a jump instruction and may still reveal the truth value of `b` via differences in execution times due to branch prediction. The issue can, however, be detected by an analysis based on suitable instruction-level timing models.

```
ct_select_u32:
    mov    0x4(%esp),%al
    test   %al,%al
    jne    L
    lea    0xc(%esp),%eax
    mov    (%eax),%eax
    ret
L:  lea    0x8(%esp),%eax
    mov    (%eax),%eax
    ret
```

Given the highly-sophisticated cache hierarchies, pipeline stalls, etc. in contemporary real machines, the timing side channel in the example above may be difficult to realise and exploit in an actual attack; but such issues may also manifest themselves at lower levels (e.g., RTL), as illustrated by the following example.

▶ **Example 2** ([44]). An AND gate with two inputs A, B and an output C and respective delays T_A, T_B, and T_C can be modelled as the timed automaton with two clocks x, y in Figure 1 where $x = T_A$ checks if the value of clock x is T_A, $y := 0$ resets clock y to 0, etc.

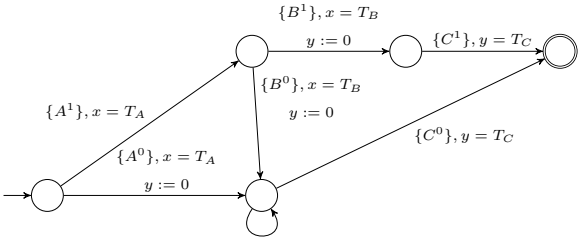


Figure 1 A timed automaton modelling an AND gate with inputs A, B and output C with respective delays T_A, T_B, and T_C.

(suppose that $T_A < T_B$ and $T_B - T_A < T_C$). Intuitively, the truth values of A and B are obtained after T_A and T_B respectively, and the output $C = A \wedge B$ has a delay of T_C from the point when its value is confirmed. Of course, once A turned out to be 0 (i.e. A^0 has happened), the output C must be 0 as well. But the *time* C^0 happens (assuming $C = 0$) also depends on the truth value of A. In other words, when $C = 0$, a low-security user (to whom A^0 and A^1 are non-observable), provided that he/she can measure time, can also infer the truth value of A while he/she should not be able to. The pair of traces with $C = 0$ that reveals A is depicted in Figure 2 and Figure 3. In this simple example, however, the timing side channel can be removed by adding $y := 0$ on the self-loop on the lower-right location.

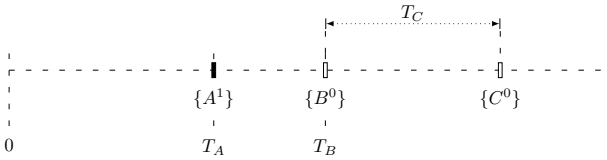


Figure 2 A trace ρ_1 with $A = 1$, $B = 0$, and $C = 0$.

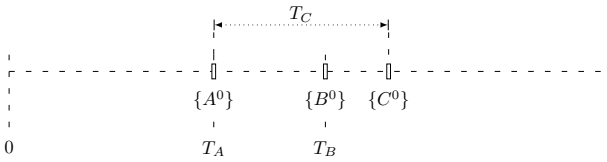


Figure 3 A trace ρ_2 with $A = 0$, $B = 0$, and $C = 0$.

Contributions. We propose HyperMTL, obtained by adding trace quantifiers to *Metric Temporal Logic* (MTL) [39], as a specification formalism for timed hyperproperties. We consider systems modelled as *timed automata*, and thus system behaviours are sequences of *events* that happen at different instants in time; this gives two possible pointwise semantics of HyperMTL: *asynchronous* and *synchronous* (this is in contrast to HyperLTL, for which a synchronous semantics is sufficient). We show that, as far as satisfiability is concerned, HyperMTL is similar to HyperLTL, i.e. satisfiability is decidable for fragments not containing $\forall\exists$, regardless of which semantics is assumed. However, in contrast with HyperLTL (whose model-checking problem is decidable), model checking HyperMTL is undecidable if there is at least one quantifier alternation in the specification, even when the timing constraints used in either the system or the specification are very restricted. Still, the alternation-free fragment of HyperMTL, which is arguably sufficient to capture many timed hyperproperties of practical interest, has a decidable model-checking problem. Finally, we identify several

subcases where HyperMTL model checking is decidable for larger fragments, such as when the synchronous semantics is assumed, the model is untimed, and the specification belongs to a certain subclass of one-clock timed automata, or when the time domain is bounded *a priori* by some $N \in \mathbb{N}_{>0}$.

Related work. Since the pioneering work of Clarkson and Schneider [17], there has been great interest in specifying and verifying hyperproperties in the past few years. The framework based on HyperLTL [16] is possibly the most popular for this purpose, thanks to its expressiveness, flexibility, and relative ease of implementation. In addition to satisfiability [23, 24] and model checking [16, 28], tools for monitoring HyperLTL also exist [3, 25, 26]. Notably, the complexity of monitoring HyperLTL, as well as model checking HyperLTL on restricted (tree-shaped or acyclic) Kripke structures, are studied in [12] and shown to be much lower than those of the general satisfiability and model-checking problems. These results, however, do not apply in the current timed setting – we will see in Section 4 that our main undecidability result holds even with these structural restrictions on the system.

Our formulation of HyperMTL is very closely related to HyperSTL [47] originally proposed in the context of quality assurance of cyber-physical systems. While [47] focusses on testing, we are concerned with the decidability of verification problems. On the other hand, the semantics of HyperSTL is defined over sets of continuous signals, i.e. state-based; as noted in [47], however, the price to pay for the extra generality is that implementing a model checker for HyperSTL is very difficult, especially for systems modelled in proprietary frameworks (such as Simulink®). Practical reasoning of HyperMTL, by contrast, can be carried out easily with existing highly optimised timed automata verification back ends, e.g., Uppaal [43].[1] Indeed, a prototype model checker based on Uppaal for the synchronous semantics of HyperMTL (with some restrictions) is reported in [32], although it does not consider the decidability of verification problems. Another relevant work [30], also based on Uppaal, checks noninterference in systems modelled as timed automata (similar to Example 4; see below). Their approach, however, is specifically tailored to noninterference and does not generalise. Some similar (but different) notions of noninterference for timed automata have been considered in [29, 58].

It is also possible to extend hyperlogics in other quantitative dimensions orthogonal to time. HyperPCTL [2] can express *probabilisitic hyperproperties*, e.g., the probability distribution of the low-security outputs are independent of the high-security inputs. In [27], specialised algorithms are developed for verifying *quantitative hyperproperties*, e.g., there is a bound on the number of traces with the same low-security inputs but different low-level outputs. The current paper is complementary to these works.

2 Timed hyperproperties

Timed words. A *timed word* (or a *trace*) over a finite alphabet Σ is a finite sequence of *events* $(\sigma_1, \tau_1) \dots (\sigma_n, \tau_n) \in (\Sigma \times \mathbb{R}_{\geq 0})^*$ with $\tau_1 \dots \tau_n$ an increasing sequence of non-negative real numbers ("*timestamps*"), i.e. $\tau_i < \tau_{i+1}$ for all i, $1 \leq i < n$.[2] For $t \in \mathbb{R}_{\geq 0}$ and a timed

[1] For more detailed accounts of the state-based and event-based semantics for timed automata and logics, see, e.g., [7, 51].

[2] To simplify the exposition, we focus on *finite* timed words in this paper (this assumption does not make the verification problems easier in general; e.g., HyperLTL satisfiability remains undecidable). All of our technical results carry over to the case of infinite timed words with some simple modifications. For example, in Section 3, suitable subformulae can be added to rule out the runs that get stuck in self-loops labelled with $\{p^\epsilon\}$.

word $\rho = (\sigma_1, \tau_1)\dots(\sigma_n, \tau_n)$, we write $t \in \rho$ iff $t = \tau_i$ for some i, $1 \le i \le n$. We denote by $T\Sigma^*$ the set of all timed words over Σ. A *timed language* (or a *trace property*) is a subset of $T\Sigma^*$.

Timed automata. Let X be a finite set of *clocks* ($\mathbb{R}_{\ge 0}$-valued variables). A *valuation* v for X maps each clock $x \in X$ to a value in $\mathbb{R}_{\ge 0}$. The set $G(X)$ of *clock constraints* (*guards*) g over X is generated by $g := \top \mid g \wedge g \mid x \bowtie c$ where $\bowtie \in \{\le, <, \ge, >\}$, $x \in X$, and $c \in \mathbb{N}_{\ge 0}$. The satisfaction of a guard g by a valuation v (written $v \models g$) is defined in the usual way. For $t \in \mathbb{R}_{\ge 0}$, we let $v + t$ be the valuation defined by $(v + t)(x) = v(x) + t$ for all $x \in X$. For $\lambda \subseteq X$, we let $v[\lambda \leftarrow 0]$ be the valuation defined by $(v[\lambda \leftarrow 0])(x) = 0$ if $x \in \lambda$, and $(v[\lambda \leftarrow 0])(x) = v(x)$ otherwise.

A *timed automaton* (TA) over Σ is a tuple $\mathcal{A} = \langle \Sigma, S, s_0, X, \Delta, F \rangle$ where S is a finite set of locations, $s_0 \in S$ is the initial location, X is a finite set of clocks, $\Delta \subseteq S \times \Sigma \times G(X) \times 2^X \times S$ is the transition relation, and F is the set of accepting locations. We say that $\mathcal{A}$ is *deterministic* iff for each $s \in S$ and $\sigma \in \Sigma$ and every distinct pair of transitions $(s, \sigma, g^1, \lambda^1, s^1) \in \Delta$ and $(s, \sigma, g^2, \lambda^2, s^2) \in \Delta$, $g^1 \wedge g^2$ is not satisfiable. A *state* of $\mathcal{A}$ is a pair (s, v) of a location $s \in S$ and a valuation v for X. A *run* of $\mathcal{A}$ on a timed word $(\sigma_1, \tau_1)\dots(\sigma_n, \tau_n) \in T\Sigma^*$ is a sequence of states $(s_0, v_0)\dots(s_n, v_n)$ where (i) $v_0(x) = 0$ for all $x \in X$ and (ii) for each i, $0 \le i < n$, there is a transition $(s_i, \sigma_{i+1}, g, \lambda, s_{i+1})$ such that $v_i + (\tau_{i+1} - \tau_i) \models g$ (let $\tau_0 = 0$) and $v_{i+1} = \big(v_i + (\tau_{i+1} - \tau_i)\big)[\lambda \leftarrow 0]$. A run of $\mathcal{A}$ is *accepting* iff it ends in a state (s, v) with $s \in F$. A timed word is *accepted* by $\mathcal{A}$ iff $\mathcal{A}$ has an accepting run on it. We denote by $[\![\mathcal{A}]\!]$ the timed language of $\mathcal{A}$, i.e. the set of all timed words accepted by $\mathcal{A}$. Two fundamental results on TAs are that the *emptiness* problem is decidable (PSPACE-complete), but the *universality* problem is undecidable [4].

Timed logics. The set of MTL formulae over a finite set of atomic propositions AP is generated by

$$\psi := \top \mid p \mid \psi_1 \wedge \psi_2 \mid \neg\psi \mid \psi_1 \,\overline{\mathbf{U}}_I\, \psi_2 \mid \psi_1 \,\overline{\mathbf{S}}_I\, \psi_2$$

where $p \in \mathsf{AP}$ and $I \subseteq \mathbb{R}_{\ge 0}$ is a *non-singular* interval with endpoints in $\mathbb{N}_{\ge 0} \cup \{\infty\}$.[3] We omit the subscript I when $I = [0, \infty)$ and sometimes write pseudo-arithmetic expressions for constraining intervals, e.g., '< 3' for $[0, 3)$. The other Boolean operators are defined as usual: $\bot \equiv \neg\top$ and $\psi_1 \vee \psi_2 \equiv \neg(\neg\psi_1 \wedge \neg\psi_2)$. We also define the dual temporal operators $\psi_1 \widetilde{\mathbf{U}}_I \psi_2 \equiv \neg\big((\neg\psi_1)\,\overline{\mathbf{U}}_I\,(\neg\psi_2)\big)$ and $\psi_1 \widetilde{\mathbf{S}}_I \psi_2 \equiv \neg\big((\neg\psi_1)\,\overline{\mathbf{S}}_I\,(\neg\psi_2)\big)$. Using these operators, every MTL formula ψ can be transformed into an MTL formula in *negative normal form*, i.e. $\neg$ is only applied to atomic propositions. To ease the presentation, we will also use the usual shortcuts like $\overline{\mathbf{F}}_I\,\psi \equiv \top\,\overline{\mathbf{U}}_I\,\psi$, $\overline{\mathbf{G}}_I\,\psi \equiv \neg\,\overline{\mathbf{F}}_I\,\neg\psi$, $\mathbf{X}_I\,\psi \equiv \bot\,\overline{\mathbf{U}}_I\,\psi$, and "weak-future" variants of temporal operators, e.g., $\mathbf{F}\,\psi \equiv \psi \vee \overline{\mathbf{F}}\,\psi$. Given an MTL formula ψ over AP in negative normal form, a timed word $\rho = (\sigma_1, \tau_1)\dots(\sigma_n, \tau_n)$ over $\Sigma_{\mathsf{AP}} = 2^{\mathsf{AP}}$, and $t \in \mathbb{R}_{\ge 0}$, we define the MTL satisfaction relation $\models$ as follows:[4]

[3] In the literature, this logic (with the requirement that constraining intervals must be non-singular) is usually referred to as MITL [5], but we simply call it MTL in this paper for notational simplicity. Also note that our undecidability results carry over to the fragment with only future operators.

[4] The formulation of the pointwise semantics of MTL here deviates slightly from the standard one (cf. [8, 50]) to enable a formal treatment of interleaving of events in different traces.

- $(\rho, t) \models \top$ iff $t \in \rho$;
- $(\rho, t) \models \bot$ iff $t \notin \rho$;
- $(\rho, t) \models p$ iff $t \in \rho$ and $p \in \sigma_i$;
- $(\rho, t) \models \neg p$ iff $t \in \rho$ and $p \notin \sigma_i$;
- $(\rho, t) \models \psi_1 \wedge \psi_2$ iff $(\rho, t) \models \psi_1$ and $(\rho, t) \models \psi_2$;
- $(\rho, t) \models \psi_1 \vee \psi_2$ iff $(\rho, t) \models \psi_1$ or $(\rho, t) \models \psi_2$;
- $(\rho, t) \models \psi_1 \,\overline{\mathbf{U}}_I\, \psi_2$ iff there exists $t' > t$ such that $t' - t \in I$, $(\rho, t') \models \top$, $(\rho, t') \models \psi_2$, and $(\rho, t'') \models \psi_1$ for all t'' such that $t'' \in (t, t')$ and $(\rho, t'') \models \top$;
- $(\rho, t) \models \psi_1 \widetilde{\overline{\mathbf{U}}}_I \psi_2$ iff for all $t' > t$ such that $t' - t \in I$ and $(\rho, t') \models \top$, either $(\rho, t') \models \psi_2$ or $(\rho, t'') \models \psi_1$ for some t'' such that $t'' \in (t, t')$ and $(\rho, t'') \models \top$;
- $(\rho, t) \models \psi_1 \,\overline{\mathbf{S}}_I\, \psi_2$ iff there exists t', $0 \leq t' < t$ such that $t - t' \in I$, $(\rho, t') \models \top$, $(\rho, t') \models \psi_2$, and $(\rho, t'') \models \psi_1$ for all t'' such that $t'' \in (t', t)$ and $(\rho, t'') \models \top$;
- $(\rho, t) \models \psi_1 \widetilde{\overline{\mathbf{S}}}_I \psi_2$ iff for all t', $0 \leq t' < t$ such that $t - t' \in I$ and $(\rho, t') \models \top$, either $(\rho, t') \models \psi_2$ or $(\rho, t'') \models \psi_1$ for some t'' such that $t'' \in (t', t)$ and $(\rho, t'') \models \top$.

We say that ρ *satisfies* ψ ($\rho \models \psi$) iff $(\rho, 0) \models \psi$, and we write $[\![\psi]\!]$ for the timed language of ψ, i.e. the set of all timed words satisfying ψ. It is well known that any MTL formula can be translated into a TA accepting the same timed language [6]; this implies that the satisfiability and model-checking problems for MTL are decidable (EXPSPACE-complete).

Adding trace quantifiers. Let V be an infinite supply of *trace variables*, the set of HyperMTL formulae over AP are generated by

$$\varphi := \exists \pi\, \varphi \mid \forall \pi\, \varphi \mid \psi$$
$$\psi := \top \mid \top_\pi \mid p_\pi \mid \psi_1 \wedge \psi_2 \mid \neg\psi \mid \psi_1 \,\overline{\mathbf{U}}_I\, \psi_2 \mid \psi_1 \,\overline{\mathbf{S}}_I\, \psi_2$$

where $\pi \in V$, $p \in \mathsf{AP}$, and $I \subseteq \mathbb{R}_{\geq 0}$ is a non-singular interval with endpoints in $\mathbb{N}_{\geq 0} \cup \{\infty\}$ (to ease the notation, we will usually write, e.g., p_a for p_{π_a}). Without loss of generality we forbid the reuse of trace variables, i.e. each trace quantifier must use a fresh trace variable. Syntactic sugar is defined as in MTL, e.g., $\overline{\mathbf{F}}_I\, \psi \equiv \top \,\overline{\mathbf{U}}_I\, \psi$. A HyperMTL formula is *closed* if it does not have free occurrences of trace variables. Following [22], we refer to fragments of HyperMTL by their quantifier patterns, e.g., $\exists^*\forall^*$-HyperMTL. Finally, note that trace quantifiers can be added to TAs in the same manner (in this case, quantified TAs operate over "stacked" traces; see the semantics for HyperMTL below).

In contrast with TAs and MTL formulae, which define *trace properties*, HyperMTL formulae define *(timed) hyperproperties*, i.e. sets of trace properties. Depending on whether one requires timestamps in quantified traces to match exactly (i.e. all quantified traces must *synchronise*), two possible semantics can be defined accordingly.

Asynchronous semantics. A *trace assignment* over Σ is a partial mapping from V to $T\Sigma^*$. We write $\Pi_\emptyset$ for the empty trace assignment and $\Pi[\pi \mapsto \rho]$ for the trace assignment that maps π to ρ and π' to $\Pi(\pi')$ for all $\pi' \neq \pi$. Given a HyperMTL formula φ over AP whose quantifier-free part is in negative normal form, a trace set T over Σ_{AP}, a trace assignment Π over Σ_{AP}, and $t \in \mathbb{R}_{\geq 0}$, we define the HyperMTL *asynchronous* satisfaction relation $\models$ as follows (we omit the cases where the definitions are obvious or exactly similar):

- $(T,t) \models_\Pi \top$ iff $t \in \rho$ for some $\rho \in \mathit{range}(\Pi)$;[5]
- $(T,t) \models_\Pi \top_\pi$ iff $t \in \rho$ for $\rho = \Pi(\pi)$;
- $(T,t) \models_\Pi p_\pi$ iff $t \in \rho$ for $\rho = \Pi(\pi)$ and $p \in \sigma_i$ for the event (σ_i, t) in ρ;
- $(T,t) \models_\Pi \psi_1 \,\overline{\mathbf{U}}_I\, \psi_2$ iff there exists $t' > t$ such that $t' - t \in I$, $(T,t') \models_\Pi \top$, $(T,t') \models_\Pi \psi_2$, and $(T,t'') \models_\Pi \psi_1$ for all t'' such that $t'' \in (t,t')$ and $(T,t'') \models_\Pi \top$;
- $(T,t) \models_\Pi \psi_1 \widetilde{\mathbf{U}}_I \psi_2$ iff for all $t' > t$ such that $t' - t \in I$ and $(T,t') \models_\Pi \top$, either $(T,t') \models_\Pi \psi_2$ or $(T,t'') \models_\Pi \psi_1$ for some t'' such that $t'' \in (t,t')$ and $(T,t'') \models_\Pi \top$;
- $(T,t) \models_\Pi \exists \pi\, \varphi$ iff there is a trace $\rho \in T$ such that $(T,t) \models_{\Pi[\pi \mapsto \rho]} \varphi$;
- $(T,t) \models_\Pi \forall \pi\, \varphi$ iff for all traces $\rho \in T$, $(T,t) \models_{\Pi[\pi \mapsto \rho]} \varphi$.

We say that T *satisfies* a closed HyperMTL formula φ in the asynchronous semantics ($T \models \varphi$) iff $(T,0) \models_{\Pi_\emptyset} \varphi$.

The asynchronous semantics for HyperMTL is (arguably) the most natural choice of semantics for the current event-based setting. As the examples below illustrate, allowing explicit interleaving of events may simplify the specification even when no quantitative timing constraint is involved.

▶ **Example 3.** Consider again the system in Example 2 and a low-security user u_L who can observe $\{B^0, B^1, C^0, C^1\}$ but not $\{A^0, A^1\}$. The property "if B^0 occurs in both π_a and π_b, then the corresponding C^0's must occur simultaneously in both π_a and π_b" (a variant of noninference [46]) can be specified with the following HyperMTL formula in the asynchronous semantics:

$$\varphi_1 = \forall \pi_a \forall \pi_b \left(\mathbf{F}\, B_a^0 \wedge \mathbf{F}\, B_b^0 \Rightarrow \mathbf{F}(C_a^0 \wedge C_b^0) \right).$$

In particular, $C_a^0 \wedge C_b^0$ holds only when the two $\{C^0\}$-events occur simultaneously in π_a and π_b. It is clear that the system does not satisfy φ_1, as there are two traces of the system where B^0 occurs in both, but the occurrences of C^0 are at different times; as we mentioned earlier, this allows u_L to infer A by timing C^0. If, on the other hand, the timing accuracy attainable by u_L is limited and thus it can only differentiate events that are d time units apart, the system can instead be checked against

$$\varphi_2 = \forall \pi_a \forall \pi_b \left(\mathbf{F}\, B_a^0 \wedge \mathbf{F}\, B_b^0 \Rightarrow \mathbf{F}\left(C_a^0 \wedge (\mathbf{F}_{\le d}\, C_b^0 \vee \mathbf{O}_{\le d}\, C_b^0)\right) \right)$$

where $\mathbf{O}$ is the past version of $\mathbf{F}$. This will be satisfied if $T_B - T_A \le d$, and since u_L will not be able to infer A, the system may be considered secure in this case. Finally, note that in the original (synchronous) semantics for HyperLTL [16], φ_1 is satisfied by the system, as events are synchronised by their positions rather than times of occurrence.

▶ **Example 4** (Noninterference in event-based systems [31])**.** A system operating on sequences of commands issued by different users can be modelled as a deterministic finite automaton $\mathcal{A}$ over $\Sigma = U \times C$ where U is the set of users and C is the set of commands. Additionally, let Obs be the set of observations and $\mathit{out} : S \times U \to \mathsf{Obs}$ be the observation function for what can be observed at each location by each user. Let there be a partition of U into two disjoint sets of users $U_H \subseteq U$ and $U_L \subseteq U$. *Noninterference* requires that for each $w \in \Sigma^*$ where w ends with a command issued by a user in U_L and $\mathcal{A}$ reaches s after reading w, the subsequence w' obtained by removing all the commands issued by the users in U_H results in a location s' such that the observation $\mathit{out}(s', u_L)$ of each user $u_L \in U_L$ is identical

[5] Note the dependency of the interpretation of $\top$ on Π; in particular, it is possible for a trace set with out-of-sync traces to satisfy $\forall \pi_b\, (p_b \,\overline{\mathbf{U}}\, q_b)$ but not $\forall \pi_a \forall \pi_b\, (p_b \,\overline{\mathbf{U}}\, q_b)$.

to $out(s, u_L)$. For our purpose, we can combine $\mathcal{A}$ and out (in the expected way) into an automaton $\mathcal{A}'$ over Σ_{AP} where $\mathsf{AP} = (U \times C) \uplus (U \times \mathsf{Obs})$ (atomic propositions in $U \times \mathsf{Obs}$ reflect the observations at the location that has just been entered). Checking noninterference then amounts to model checking $\mathcal{A}'$ (whose locations are all accepting) against the following HyperMTL formula in the asynchronous semantics:

$$\varphi_3 = \forall \pi_a \, \forall \pi_b \, \big(\mathbf{G}(\top_b \Rightarrow \psi_b^L \wedge \psi_{U,C}^{=}) \\ \wedge \mathbf{G}(\top_a \wedge \bot_b \Rightarrow \psi_a^H) \Rightarrow \mathbf{G}(\top_b \Rightarrow \psi_{out(U_L)}^{=}) \big)$$

(where ψ_b^L asserts that the command in π_b is issued by a user in U_L, $\psi_{U,C}^{=}$ says that the two synchronised commands in π_a and π_b agree on U and C, etc.). Specifically,

- $\mathbf{G}(\top_b \Rightarrow \psi_b^L \wedge \psi_{U,C}^{=})$ asserts that π_b only contains low commands and π_a also contains these commands at the exactly same times;
- $\mathbf{G}(\top_a \wedge \bot_b \Rightarrow \psi_a^H)$ asserts that all the commands that are only present in π_a are high commands;
- $\mathbf{G}(\top_b \Rightarrow \psi_{out(U_L)}^{=})$ ensures that, after each low command in π_b, the observation of each $u_L \in U_L$ is identical to the observation of u_L after the corresponding low command in π_a, regardless of the high commands that occur in the preceding "gaps".

We remark that while this example is essentially untimed, the asynchronous event-based formulation leads to a much simpler and clearer specification than the state-based one in [16].

Synchronous semantics. A less general semantics can be defined for HyperMTL formulae where each trace quantifier only ranges over traces that synchronise with the traces in the current trace assignment (this is the case in the original HyperLTL semantics [16]). For example, the second quantifier in $\exists \pi_a \, \exists \pi_b \, \psi$ requires π_b to satisfy $(\pi_a, t) \models \top_a \Leftrightarrow (\pi_b, t) \models \top_b$ for all $t \in \mathbb{R}_{\geq 0}$. The HyperMTL *synchronous* satisfaction relation $\models^{sync}$ can, in fact, be expressed in the asynchronous semantics by explicitly requiring newly quantified traces to synchronise in the quantifier-free part of the formula. More precisely, for a closed HyperMTL formula $\varphi = \mathcal{Q} \, \varphi'$ where $\mathcal{Q}$ denotes a block of quantifiers of the same type (i.e. all existential or all universal) and φ' is a possibly open HyperMTL formula, and a set V of trace variables, let (abusing notation slightly) $sync(\varphi, V) = \mathcal{Q} \, \big(\mathbf{G}(\bigwedge_{\pi \in \mathcal{Q} \cup V} \top_\pi) \wedge sync(\varphi', \mathcal{Q} \cup V) \big)$ when $\mathcal{Q}$ are existential, $sync(\varphi) = \mathcal{Q} \, \big(\mathbf{G}(\bigwedge_{\pi \in \mathcal{Q} \cup V} \top_\pi) \Rightarrow sync(\varphi', \mathcal{Q} \cup V) \big)$ when $\mathcal{Q}$ are universal, and $sync(\psi, V) = \psi$ when ψ is quantifier-free. The following lemma holds subject to rewriting the formula into prenex normal form.

▶ **Lemma 5.** *For any trace set T over Σ_{AP} and closed* HyperMTL *formula φ over* AP*, $T \models^{sync} \varphi$ iff $T \models sync(\varphi, \emptyset)$.*

While the synchronous semantics may seem quite restricted (intuitively, the chance that two random traces of a timed system have exactly the same timestamps is certainly slim!), one can argue that it already suffices for many applications if *stuttering steps* are allowed. We will see later that for alternation-free HyperMTL, the asynchronous semantics can be emulated in the synchronous semantics using a "weak inverse" of Lemma 5.

Satisfiability and model checking. Given a closed HyperMTL formula φ over AP, the *satisfiability* problem asks whether there is a *non-empty* trace set $T \subseteq T\Sigma_{\mathsf{AP}}^*$ satisfying it, i.e. $T \models \varphi$ (or $T \models^{sync} \varphi$, if the synchronous semantics is assumed). Given a TA $\mathcal{A}$ over Σ_{AP} and a closed HyperMTL formula φ over AP, the *model-checking* problem asks whether $[\![\mathcal{A}]\!] \models \varphi$ (or $[\![\mathcal{A}]\!] \models^{sync} \varphi$). Our focus in this paper is on the *decidability* of these problems, as their complexity (when they are decidable) follow straightforwardly from standard results on MTL [5] and HyperLTL [16, 22].

3 Satisfiability

To emulate interleaving of events (of a concurrent or distributed system, say) in a synchronous, state-based setting, it is natural and necessary to introduce stuttering steps. In the context of verification, it is often a desirable trait for a temporal logic to be *stutter-invariant* [41, 42] so that it cannot be used to differentiate traces that ought to be regarded as the same (e.g., in an iterative refinement process, an abstract component of a system may be replaced by a concrete implementation that simulates an abstract step with some additional internal actions). As a simple attempt to reconcile the asynchronous and synchronous semantics of HyperMTL, we can make use of *silent events* in the same spirit to enable synchronisation of interleaving traces while preserving the semantics. More precisely, let $\mathit{stutter}(\rho)$ for a trace $\rho \in T\Sigma^*_{\mathsf{AP}}$ be the maximal set of traces $\rho' \in T\Sigma^*_{\mathsf{AP}_\epsilon}$ ($\mathsf{AP}_\epsilon = \mathsf{AP} \cup \{p^\epsilon\}$) such that

- for every event (σ_i, τ_i) in ρ', either $\sigma_i = \{p^\epsilon\}$ or $p^\epsilon \notin \sigma_i$;
- ρ can be obtained from ρ' by deleting all the $\{p^\epsilon\}$-events.

This extends to trace sets $T \subseteq T\Sigma^*_{\mathsf{AP}}$ in the obvious way. For a closed alternation-free HyperMTL formula $\varphi = \mathcal{Q}\psi$ over AP, let $\mathit{stutter}(\varphi) = \mathcal{Q}\psi''$ be the HyperMTL formula over AP_ϵ obtained by replacing in ψ, e.g., all $\top_\pi$ with $\neg p^\epsilon_\pi$, to give ψ', and finally let $\psi'' = \mathbf{G}(\bigvee_{\pi\in\mathcal{Q}} \neg p^\epsilon_\pi) \wedge \big(\bigwedge_{\pi\in\mathcal{Q}} \mathbf{G}(p^\epsilon_\pi \Rightarrow \bigwedge_{p\in\mathsf{AP}} \neg p_\pi)\big) \wedge \psi'$ when $\mathcal{Q}$ are existential and $\psi'' = \mathbf{G}(\bigvee_{\pi\in\mathcal{Q}} \neg p^\epsilon_\pi) \wedge \big(\bigwedge_{\pi\in\mathcal{Q}} \mathbf{G}(p^\epsilon_\pi \Rightarrow \bigwedge_{p\in\mathsf{AP}} \neg p_\pi)\big) \Rightarrow \psi'$ when $\mathcal{Q}$ are universal. Intuitively, ψ'' ensures that the traces involved are *well-formed* (i.e. satisfy the first condition above), and its own satisfaction is insensitive to the addition of silent events. The following lemma follows from a simple structural induction.

▶ **Lemma 6.** *For any trace set T over Σ_{AP} and closed alternation-free* HyperMTL *formula $\varphi = \mathcal{Q}\psi$ over* AP *($\mathcal{Q}$ is either a block of existential quantifiers or universal quantifiers and ψ is quantifier-free), $T \models \varphi$ iff $\mathit{stutter}(T) \models^{sync} \mathit{stutter}(\varphi)$.*

The following two lemmas follow from Lemma 6 and the fact that for alternation-free HyperMTL formulae, satisfiability in the synchronous semantics can be reduced (in the same way as HyperLTL) to MTL satisfiability.

▶ **Lemma 7.** *The satisfiability problem for* $\exists^*$-HyperMTL *is decidable.*

▶ **Lemma 8.** *The satisfiability problem for* $\forall^*$-HyperMTL *is decidable.*

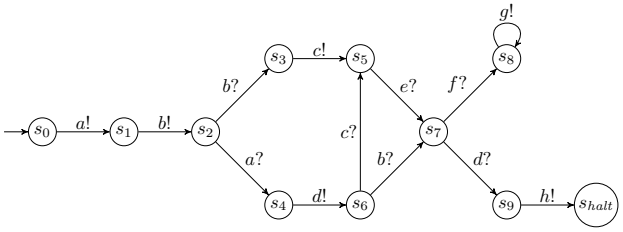
Lemma 6, however, does not extend to larger fragments of HyperMTL. For example, consider $T = \{(\{p\}, 1)(\{r\}, 3), (\{q\}, 2)\}$ and $\varphi = \exists\pi_a \forall\pi_b\, (\mathbf{F}\, p_a \wedge \neg\, \mathbf{F}\, q_b)$. Now it is obvious that $T \not\models \varphi$, but since $(\{p\}, 1)(\{r\}, 3) \in \mathit{stutter}(T)$, we have $\mathit{stutter}(T) \models^{sync} \mathit{stutter}(\varphi)$ (provided that the definition of $\mathit{stutter}(\cdot)$ is extended to general HyperMTL formulae, as in Lemma 5). Still, it is not hard to see that the crucial observation used in $\exists^*\forall^*$-HyperLTL satisfiability (if $\exists\pi_0 \ldots \exists\pi_k \forall\pi'_0 \ldots \forall\pi'_\ell\, \psi$ is satisfiable, then it is also satisfiable by the trace set $\{\pi_0, \ldots \pi_k\}$) extends to HyperMTL in the asynchronous semantics; the following lemma then follows from Lemma 7.

▶ **Lemma 9.** *The satisfiability problem for* $\exists^*\forall^*$-HyperMTL *is decidable.*

Finally, note that the undecidability of $\forall\exists$-HyperLTL carries over to HyperMTL: in the synchronous semantics, the reduction in [22] applies directly with some trivial modifications (as we work with finite traces); undecidability then holds for the case of asynchronous semantics as well, by Lemma 5.

▶ **Lemma 10.** *The satisfiability problem for* $\forall\exists$-HyperMTL *is undecidable.*

▶ **Theorem 11.** *The satisfiability problem for* HyperMTL *is decidable if the formula does not contain* $\forall\exists$.



$$(s_0, \epsilon) \to (s_1, a) \to (s_2, ab) \to (s_4, b) \to (s_6, bd) \to (s_7, d) \to (s_9, \epsilon) \to (s_{halt}, h)$$

Figure 4 A DCM and its unique halting computation.

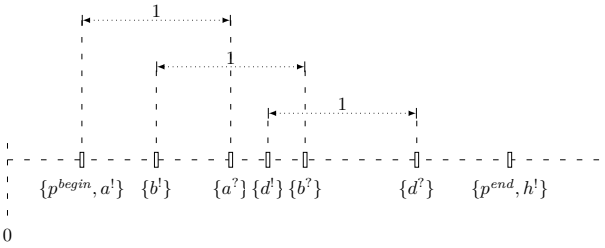


Figure 5 A trace encoding the halting computation of the DCM in Figure 4. Note that each $m^!$ is followed by a corresponding $m^?$ exactly 1 time unit later.

4 Model checking

We now turn to the model-checking problem, which behaves quite differently than in the case of HyperLTL.

The alternation-free case. Without loss of generality, we consider only the case of $\exists^*$-HyperMTL in the asynchronous semantics. By Lemma 6, checking $[\![\mathcal{A}]\!] \models \varphi$ (for a TA $\mathcal{A}$ over Σ_{AP} and a closed $\exists^*$-HyperMTL formula φ over AP) is equivalent to checking $\mathit{stutter}([\![\mathcal{A}]\!]) \models^{\mathit{sync}} \mathit{stutter}(\varphi)$. To this end, we define $\mathit{stutter}(\mathcal{A})$ as the TA over $\Sigma_{\mathsf{AP}_\epsilon}$ obtained from $\mathcal{A}$ by adding a self-loop labelled with $\{p^\epsilon\}$ to each location; it should be clear that $[\![\mathit{stutter}(\mathcal{A})]\!] = \mathit{stutter}([\![\mathcal{A}]\!])$. In this way, the problem reduces to model checking $\exists^*$-HyperMTL in the synchronous semantics which, as the model-checking problem for $\exists^*$-HyperLTL, can be reduced to MTL model checking.

▶ **Theorem 12.** *Model checking alternation-free* HyperMTL *is decidable.*

The general case. Recall that the model-checking problem for HyperLTL is decidable even when the specification involves arbitrary nesting of quantifiers. This is unfortunately not the case for HyperMTL: allowing only one quantifier alternation already leads to undecidability. To see this, recall that any TA can be written as a formula $\exists X\, \psi$ where X is a set of (new) atomic propositions and ψ is an MTL formula [33, 53]. The undecidable TA universality problem – given a TA $\mathcal{A}$ over Σ, deciding whether $[\![\mathcal{A}]\!] = T\Sigma^*$ – can thus be reduced to model checking HyperMTL: one simply checks whether there exists an X-labelling for every timed word over Σ so that ψ is satisfied. Here we show that model checking HyperMTL is essentially a harder problem: in the case of asynchronous semantics, model checking HyperMTL with quantifier alternations necessarily involves TAs with ϵ-transitions [11], and therefore remains

undecidable even when *both the model and the specification are deterministic and only one of them uses a single clock (i.e. the other is untimed)*; by contrast, (standard) TA universality over finite timed words is decidable when the TA uses only one clock [49].

We adapt the undecidability proof of the *reactive synthesis* problem for MTL in [14], which itself is by reduction from the halting problem for *deterministic channel machines* (DCMs), known to be undecidable [13]. Note that, in contrast to HyperMTL model checking, MTL reactive synthesis is decidable when the specification is deterministic [19]; in this sense, quantification over traces is more powerful than quantification over strategies (there is a winning strategy of the controller for all possible strategies of the environment).[6] For our purpose, we introduce the $\triangleleft_I$ operator, in which we allow I to be singular (note that this is merely syntactic sugar and does not increase the expressiveness of MTL [33, 53]):

- $(T,t) \models_\Pi \triangleleft_I \varphi$ iff there exists t', $0 \leq t' < t$ such that $t - t' \in I$, $(T,t') \models_\Pi \top$, $(T,t') \models_\Pi \varphi$, and $(T,t'') \not\models_\Pi \varphi$ for all t'' such that $t'' \in (t',t)$ and $(T,t'') \models_\Pi \top$.

Let $\mathsf{LTL}_\triangleleft$ be the fragment of MTL where all timed subformulae must be of the form $\triangleleft_I \varphi$, and all φ's in such subformulae must be "pure past" formulae; these requirements ensure that $\mathsf{LTL}_\triangleleft$, in which we will write the quantifier-free part of the specification, translates into deterministic TAs [18]. To ease the understanding, we will first do the proof for the case of asynchronous semantics and then adapt it to the case of synchronous semantics.

▶ **Theorem 13.** *Model checking* $\exists^*\forall^*$-HyperMTL *and* $\forall^*\exists^*$-HyperMTL *are undecidable in the asynchronous semantics.*

Proof. A DCM $\mathcal{S} = \langle S, s_0, s_{halt}, M, \Delta \rangle$ can be seen as a finite automaton equipped with an unbounded fifo channel: S is a finite set of locations, s_0 is the initial location, s_{halt} is the halting location (such that $s_{halt} \neq s_0$), M is a finite set of messages, and $\Delta \subseteq S \times \{m!, m? \mid m \in M\} \times S$ is the transition relation satisfying the following determinism hypothesis: (i) $(s,q,s') \in \Delta$ and $(s,q,s'') \in \Delta$ implies $s' = s''$; (ii) if $(s,m!,s') \in \Delta$ then it is the only outgoing transition from s. Without loss of generality, we further assume that there is no incoming transition to s_0, no outgoing transition from s_{halt}, and $(s_0,q,s') \in \Delta$ implies that $q \in \{m! \mid m \in M\}$ and $s' \neq s_{halt}$. The semantics of $\mathcal{S}$ can be described with a graph $G(\mathcal{S})$ with vertices $\{(s,x) \mid s \in S, x \in M^*\}$ and edges defined as follows: (i) $(s,x) \to (s', xm)$ if $(s,m!,s') \in \Delta$; (ii) $(s,mx) \to (s',x)$ if $(s,m?,s') \in \Delta$. In other words, $m!$ "writes" a copy of m to the channel and $m?$ "reads" a copy of m off the channel. We say that $\mathcal{S}$ *halts* if there is a path in $G(\mathcal{S})$ from (s_0,ϵ) to (s_{halt},x) (a *halting computation* of $\mathcal{S}$) for some $x \in M^*$. An example DCM and its unique halting computation are depicted in Figure 4.

The idea, as in many similar proofs (e.g., [50]), is to encode a halting computation of $\mathcal{S}$ as a trace where each $m?$ is preceded by a corresponding $m!$ exactly 1 time unit earlier, and each $m!$ is followed by an $m?$ exactly 1 time unit later if s_{halt} has not been reached yet. To this end, let the model $\mathcal{A}$ be an (untimed) finite automaton over $\Sigma = 2^{\mathsf{AP}}$ where $\mathsf{AP} = \{m^!, m^? \mid m \in M\} \cup \{p^{begin}, p^{end}, p^{read}, p^1, q^1\}$ and whose set of locations is $S \cup \{s_1\}$, where s_1 is a new non-accepting location. The transitions of $\mathcal{A}$ follow $\mathcal{S}$: for each $m \in M$, $s \xrightarrow{\{m^?\}} s'$ is a transition of $\mathcal{A}$ iff $(s,m?,s') \in \Delta$, and similarly for $m!$ – except for those going out of s_0 or going into s_{halt}, on which we further require p^{begin} or p^{end} to hold, respectively. Let s_0 be the initial location and s_{halt} be the only accepting location, and finally add transitions $s_0 \xrightarrow{\{p^{read}\}} s_{halt}$ and $s_0 \xrightarrow{\{p^1\}} s_1 \xrightarrow{\{q^1\}} s_{halt}$. It is clear that $\mathcal{A}$ is deterministic and it accepts only three types of traces:

[6] Indeed, the quantifier-free part ψ in the simpler encoding mentioned above (based on labelling timed words with propositions in X) is already in $\mathsf{LTL}_\triangleleft$ and thus is deterministic.

1. From s_0 through some other locations of $\mathcal{S}$ and finally s_{halt}, i.e. those respecting the transition relation, but not necessarily the semantics, of $\mathcal{S}$.
2. From s_0 to s_{halt} in a single transition (on which p^{read} holds).
3. From s_0 to s_1 and then s_{halt}.

It remains to write a specification φ such that $[\![\mathcal{A}]\!] \models \varphi$ exactly when $\mathcal{A}$ accepts a trace of type (1) that also respects the semantics of $\mathcal{S}$ (one such trace that corresponds to the unique halting computation of the DCM in Figure 4 is depicted in Figure 5). This is where the traces of types (2) and (3) come into play: for example, if a trace of type (1) issues a read $m?$ without a corresponding write $m!$, then a trace of type (3) can be used to "pinpoint" the error. More precisely, let $\varphi = \exists \pi_a \, \forall \pi_b \, (\psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4)$ where

- $\psi_1 = \mathbf{F}\, p_a^{end}$ ensures that π_a is of type (1);
- $\psi_2 = \mathbf{F}(p_b^{read} \wedge \psi_R) \Rightarrow \mathbf{F}(p_b^{read} \wedge \lhd_{\geq 1} p_a^{begin})$, where $\psi_R = \bigvee\{m_a^? \mid m \in M\}$, is a simple sanity check which ensures that in π_a, each $m^?$ must happen at time $\geq t+1$ if p^{begin} happens at t;
- $\psi_3 = \bigwedge_{m \in M} \Big(\mathbf{F}(q_b^1 \wedge m_a^?) \Rightarrow \big(\mathbf{F}(p_a^{begin} \wedge \mathbf{F}\, p_b^1) \wedge \mathbf{F}(q_b^1 \wedge \lhd_{=1} p_b^1) \Rightarrow \mathbf{F}(p_b^1 \wedge m_a^!) \big) \Big)$ ensures that each $m^?$, if it happens at t, is preceded by a corresponding $m^!$ at $t-1$ in π_a;
- $\psi_4 = \bigwedge_{m \in M} \Big(\mathbf{F}(p_b^1 \wedge m_a^!) \Rightarrow \mathbf{F}(p_a^{end} \wedge \lhd_{<1} p_b^1) \vee \big(\mathbf{F}(q_b^1 \wedge \lhd_{=1} p_b^1) \Rightarrow \mathbf{F}(q_b^1 \wedge m_a^?) \big) \Big)$ ensures that each $m^!$ at t is followed by a corresponding $m^?$ at $t+1$ (unless p^{end} happens first) in π_a.

Now observe that the only timed subformulae are $\lhd_{\geq 1} p_a^{begin}$, $\lhd_{=1} p_b^1$, and $\lhd_{<1} p_b^1$. As p^1 and p^{read} cannot happen in the same trace (π_b), it is not hard to see that the reduction remains correct if we replace these by $\lhd_{\geq 1}(p_a^{begin} \vee p_b^1)$, $\lhd_{=1}(p_a^{begin} \vee p_b^1)$, and $\lhd_{<1}(p_a^{begin} \vee p_b^1)$ (respectively) to obtain ψ_2', ψ_3', and ψ_4'. It follows that $\psi_1 \wedge \psi_2' \wedge \psi_3' \wedge \psi_4'$ can be translated into a one-clock deterministic TA. Finally, it is possible to move all the timing constraints into the model and use an untimed HyperLTL formula as the specification: in the model, ensure that p^1 and q^1 are separated by exactly 1 time unit, and add $s_0 \xrightarrow{\{p^2\}} s_1 \xrightarrow{\{q^2\}} s_{halt}$ such that p^2 and q^2 are separated by < 1 time unit; in the specification, use p^2, q^2 to rule out those π_a's with some $m^?$ at < 1 time unit from p^{begin}. ◀

Now we consider the synchronous semantics. The corresponding result is weaker in this case, as we will see in the next section that in several subcases the problem becomes decidable. Still, the reduction above can be made to work if the model has one clock and an extra trace quantifier is allowed.

▶ **Theorem 14.** *Model checking* $\exists^*\forall^*$-HyperMTL *and* $\forall^*\exists^*$-HyperMTL *are undecidable in the synchronous semantics.*

Proof of Theorem 14. We use a modified model $\mathcal{A}'$ whose set of locations is $S \cup \{s_1, s_2, s_3, s_4\}$; the transitions are similar to $\mathcal{A}$ in the proof of Theorem 13, but we now use a clock x in the path $s_0 \xrightarrow[x:=0]{\{p^1\}} s_1 \xrightarrow[x \geq 1, x:=0]{\{q^1\}} s_{halt}$, the paths $s_0 \xrightarrow[x:=0]{\{p^2\}} s_2 \xrightarrow[x \leq 1, x:=0]{\{q^2\}} s_{halt}$, $s_0 \xrightarrow[x:=0]{\{p^3\}} s_3 \xrightarrow[x > 1, x:=0]{\{q^3\}} s_{halt}$, $s_0 \xrightarrow[x:=0]{\{p^4\}} s_4 \xrightarrow[x < 1, x:=0]{\{q^4\}} s_{halt}$ are added, and $s_0 \xrightarrow{\{p^{read}\}} s_{halt}$ is removed. Moreover, a self-loop labelled with $\{p^\epsilon\}$ is added to each of s_0, s_1, s_2, s_3, s_4, and s_{halt}. The specification is $\varphi' = \exists \pi_a \, \forall \pi_b \, \forall \pi_c \bigwedge_{1 \leq i \leq 9} \psi_i'$ where $\bigwedge_{1 \leq i \leq 9} \psi_i'$ is the following untimed LTL formula:

- $\psi_1' = \mathbf{F}\, p_a^{end}$;
- $\psi_2' = \mathbf{F}(q_b^4 \wedge \psi_R) \Rightarrow \neg\, \mathbf{F}(p_b^4 \wedge p_a^{begin})$ where $\psi_R = \bigvee\{m_a^? \mid m \in M\}$;
- $\psi_3' = \bigwedge_{m \in M} \big(\mathbf{F}(q_b^1 \wedge q_c^2 \wedge m_a^?) \wedge \mathbf{F}(p_b^1 \wedge p_c^2) \Rightarrow \mathbf{F}(p_b^1 \wedge p_c^2 \wedge m_a^!) \big)$;

- $\psi'_4 = \mathbf{F}(q_b^3 \wedge \psi_R) \Rightarrow \neg\,\mathbf{F}(p_b^3 \wedge \mathbf{X}\, q_b^3)$;
- $\psi'_5 = \mathbf{F}(q_b^3 \wedge q_c^4 \wedge \psi_R) \Rightarrow \neg\,\mathbf{F}(p_b^3 \wedge \mathbf{X}\, p_c^4)$;
- $\psi'_6 = \mathbf{F}(p_b^4 \wedge \psi_W) \Rightarrow \neg\,\mathbf{F}(q_b^4 \wedge \neg\,\mathbf{X}\,\top)$ where $\psi_W = \bigvee\{m_a^! \mid m \in M\}$;
- $\psi'_7 = \bigwedge_{m\in M}\Big(\mathbf{F}(p_b^1 \wedge p_c^2 \wedge m_a^!) \wedge \mathbf{F}(q_b^1 \wedge q_c^2) \Rightarrow \mathbf{F}\left(q_b^1 \wedge q_c^2 \wedge (m_a^? \vee p_a^\epsilon)\right)\Big)$;
- $\psi'_8 = \mathbf{F}(p_b^3 \wedge \psi_W) \Rightarrow \neg\,\mathbf{F}(p_b^3 \wedge \mathbf{X}\, q_b^3)$;
- $\psi'_9 = \mathbf{F}(p_b^3 \wedge p_c^4 \wedge \psi_W) \Rightarrow \neg\,\mathbf{F}(q_b^3 \wedge \mathbf{X}\, q_c^4)$.

In this modified reduction, ψ'_1, ψ'_2 play similar roles as ψ_1, ψ_2 in the proof of Theorem 13. ψ'_3 ensures that if each $m^?$ at t is preceded by an event at $t-1$, then $m^!$ must hold there. ψ'_4 and ψ'_5 ensures that each $m^?$ at t is actually preceded by an event at $t-1$. The roles of ψ'_6, ψ'_7, ψ'_8, and ψ'_8 are analogous (note the use of silent events at the end of π_a). ◀

Restricted models. We conclude this section by showing that the undecidability results above can actually be obtained for trivial systems with only a single location. In particular, the structural restrictions considered in [12] have no effect on the decidability of HyperMTL model checking.

▶ **Corollary 15.** *Model checking* $\exists^*\forall^*$-HyperMTL *and* $\forall^*\exists^*$-HyperMTL *are undecidable in the asynchronous semantics for systems with only one location.*

▶ **Corollary 16.** *Model checking* $\exists^*\forall^*$-HyperMTL *and* $\forall^*\exists^*$-HyperMTL *are undecidable in the synchronous semantics for systems with only one location.*

5 Decidable subcases

While the negative results in the previous section may be disappointing, we stress again that model checking alternation-free HyperMTL is no harder than MTL model checking, and it can in fact be carried out with algorithms and tools for the latter. In any case, we now identify several subcases where model checking is decidable beyond the alternation-free fragment.

Untimed model + untimed specification. The first case we consider is when both the model and the specification are untimed, and the asynchronous semantics is assumed (note that, if instead, the synchronous semantics is assumed, then this case is simply HyperLTL model checking). Our algorithm follows the lines of [16] and is essentially based on *self-composition* (cf. [10], and many others; see the references in [16]) of the model; the difficulty here, however, is to handle interleaving of events. Let the model $\mathcal{A}$ be a finite automaton over Σ_{AP} and the specification be a (untimed) closed HyperMTL formula over AP. Without loss of generality, we assume the specification to be $\varphi = \exists\pi_1\,\forall\pi_2\,\ldots\exists\pi_{k-1}\,\forall\pi_k\,\psi$, which can be rewritten into $\exists\pi_1\,\neg\exists\pi_2\,\neg\ldots\exists\pi_{k-1}\,\neg\exists\pi_k\,\neg\psi$. We start by translating $\mathit{stutter}(\neg\psi)$ (in which we replace all occurrences of $\top_i$ with $\neg p_i^\epsilon$, i.e. regarded here simply as an MTL formula over $(\mathsf{AP}_\epsilon)^k = \{p_i \mid p \in \mathsf{AP}_\epsilon, 1 \le i \le k\}$) into the equivalent finite automaton over $\Sigma_{(\mathsf{AP}_\epsilon)^k}$, and take its product with (i) the automaton for $\mathbf{G}(\bigvee_{1\le i\le k}\neg p_i^\epsilon) \wedge \left(\bigwedge_{1\le i\le k}\mathbf{G}(p_i^\epsilon \Rightarrow \bigwedge_{p\in\mathsf{AP}}\neg p_i)\right)$ and (ii) the automaton obtained from $\mathit{stutter}(\mathcal{A})$ by extending the alphabet to $\Sigma_{(\mathsf{AP}_\epsilon)^k}$ and renaming all the occurrences of p to p_k, to obtain $\mathcal{B}$. Now let $\mathcal{C}$ be the projection of $\mathcal{B}$ onto $(\mathsf{AP}_\epsilon)^{k-1} = \{p_i \mid p \in \mathsf{AP}_\epsilon, 1 \le i \le k-1\}$ (this step corresponds to $\exists$ in $\neg\exists\pi_k$). By construction, $\mathcal{B}$ accepts only traces that are well-formed in dimensions 1 to $k-1$, and so does $\mathcal{C}$; but $\mathcal{C}$ may accept traces containing $\{p_i^\epsilon \mid 1 \le i \le k-1\}$-events. We replace these events by ϵ (the "real" silent event, which can be removed with the standard textbook constructions, e.g., [36]) to obtain $\mathcal{C}'$. Finally, we complement $\mathcal{C}'$ to obtain $\mathcal{C}''$ (this step corresponds to $\neg$ in $\neg\exists\pi_k$). We can then start over by taking the product of $\mathcal{C}''$, the automaton for

$\mathbf{G}(\bigvee_{1\le i\le k-1}\neg p_i^\epsilon) \wedge \big(\bigwedge_{1\le i\le k-1}\mathbf{G}(p_i^\epsilon \Rightarrow \bigwedge_{p\in\mathsf{AP}}\neg p_i)\big)$, and the automaton obtained from $\mathit{stutter}(\mathcal{A})$ by extending the alphabet to $\Sigma_{(\mathsf{AP}_\epsilon)^{k-1}}$ and renaming all the occurrences of p to p_{k-1}; the resulting automaton is the new $\mathcal{B}$. We continue this process until the outermost quantifier $\exists\pi_1$ is reached, when we test the emptiness of $\mathcal{B}$ (at this point, it is an automaton over $\Sigma_{\mathsf{AP}_\epsilon}$).

▶ **Proposition 17.** *Model checking* HyperMTL *is decidable when the model and the specification are both untimed.*

One clock + one alternation. The algorithm outlined in the previous case crucially depends on the fact that both $\mathcal{A}$ and φ are untimed, hence their product (in the sense detailed in the previous case) can be complemented. When the synchronous semantics is assumed and there is only one quantifier alternation in φ, it might be the case that we do not actually need complementation. For example, if $\mathcal{A}$ is untimed and $\varphi = \forall\pi_a\,\exists\pi_b\,\exists\pi_c\,\psi$ where ψ translates into a one-clock TA, the corresponding model-checking problem clearly reduces to universality for one-clock TAs, which is decidable but non-primitive recursive [1].[7] This observation applies to other cases as well, such as when $\mathcal{A}$ is a one-clock TA and $\varphi = \exists\pi_a\,\forall\pi_b\,\psi$ where ψ is untimed; here model checking reduces to language inclusion between two one-clock TAs.

Untimed model + MIA specification. The main obstacle in applying the algorithm above to larger fragments of HyperMTL, as should be clear now, is that universal quantifiers amount to complementations, which are not possible in general in the case of TAs. Moreover, we note that the usual strategy of restricting to deterministic models and specifications does not help, as the projection step in the algorithm necessarily introduces non-determinism. To make the algorithm work for larger fragments, we essentially need a class of automata that is both *closed under projection* and *complementable*. Fortunately, there is a subclass of one-clock TAs that satisfies these conditions. We consider two additional restrictions on one-clock TAs:

- Non-Singular (NS): a one-clock TA is NS if all the guards are non-singular (i.e. must be of the form $x \in I$ where x is the single clock and I is a non-singular interval).
- Reset-on-Testing (RoT): a one-clock TA is RoT if whenever the guard of a transition is not $\top$, x must be reset on that transition.

One-clock TAs satisfying both NS and RoT are called *metric interval automata* (MIAs), which are determinisable [21]. Since the projection operation cannot invalidate NS and RoT, the algorithm above can be applied when the synchronous semantics is assumed, $\mathcal{A}$ is untimed, ψ or $\neg\psi$ translates to a MIA, and only one complementation is involved; in this case it runs in elementary time.

▶ **Proposition 18.** *Model checking* $\forall^*\exists^*$-HyperMTL *(*$\exists^*\forall^*$-HyperMTL*) is decidable in the synchronous semantics when the model is untimed and* ψ *(*$\neg\psi$*) translates into a* MIA *in the specification* $\varphi = \forall\pi_1 \ldots \exists\pi_k\,\psi$ *(*$\varphi = \exists\pi_1 \ldots \forall\pi_k\,\psi$*).*

On the other hand, we can adapt the proof of Theorem 14 to show that model checking an untimed model against an $\exists^*\forall^*$-HyperMTL specification φ in the synchronous semantics, when the quantifier-free part ψ (instead of $\neg\psi$) translates into a MIA, remains undecidable.

[7] This case is undecidable in the asynchronous semantics by Theorem 13; as explained above, the algorithm may introduce ϵ-transitions in the asynchronous semantics, while universality for one-clock TAs with ϵ-transitions is undecidable [1].

▶ **Proposition 19.** *Model checking* $\exists^*\forall^*$-HyperMTL *is undecidable in the synchronous semantics when the model is untimed and* ψ *in the specification* $\varphi = \exists \pi_1 \ldots \forall \pi_k \, \psi$ *translates into a* MIA.

The decidability results in the synchronous semantics are summarised in Table 1.

Table 1 Decidability of model checking untimed or one-clock TAs against (one-clock) HyperMTL in the synchronous semantics; NS stands for Non-Singular constraints and RoT stands for Reset-on-Testing.

Model \ Spec.	untimed	NS+RoT	NS	RoT
untimed	Dec. (Proposition 17)	Dec. for $\forall^*\exists^*$ (Proposition 18)	Undec. for $\exists\forall\forall$ (Proposition 19)	Undec. for $\exists\forall\forall$ (Proposition 19)
NS+RoT	Undec. for $\exists\forall\forall$ (Theorem 14)	Undec.	Undec.	Undec.
NS	Undec.	Undec.	Undec.	Undec.
RoT	Undec.	Undec.	Undec.	Undec.

Bounded time domains. We end this section by showing that when there is an *a priori* bound N (where N is a positive integer) on the length of the time domain, the model-checking problem for full HyperMTL becomes decidable; in fact, in the case of synchronous semantics it reduces to the satisfiability problem for QPTL [56]. From a practical point of view, this implies that *time-bounded* HyperMTL verification (at least for the $\exists^*\forall^*$-fragment, say) can be carried out with highly efficient, off-the-shelf tools that work with LTL and (untimed) automata, such as SPOT [20], GOAL [57], and `Owl` [40].

We assume the asynchronous semantics. For a given N, we consider all traces in which all timestamps are less than N. Denote by $[\![\mathcal{A}]\!]_{[0,N)}$ the set of all such traces in $[\![\mathcal{A}]\!]$; the model-checking problem then becomes deciding whether $[\![\mathcal{A}]\!]_{[0,N)} \models \varphi$. As before, we assume φ to be $\exists \pi_1 \, \neg \exists \pi_2 \, \neg \ldots \exists \pi_{k-1} \, \neg \exists \pi_k \, \neg \psi$. Following [34, 48], we can use the *stacking construction* to obtain, from the conjunction ψ' of *stutter*$(\neg\psi)$ and $\mathbf{G}(\bigvee_{\pi\in\mathcal{Q}} \neg p^\epsilon_\pi) \wedge \big(\bigwedge_{\pi\in\mathcal{Q}} \mathbf{G}(p^\epsilon_\pi \Rightarrow \bigwedge_{p\in\mathsf{AP}} \neg p_\pi)\big)$, an equi-satisfiable untimed (QPTL) formula $\overline{\varphi} = \exists W \, \overline{\psi'}$ over the stacked alphabet $\overline{(\mathsf{AP}_\epsilon)^k} \cup \overline{Q}$ (where $\overline{(\mathsf{AP}_\epsilon)^k} = \{p_{i,j} \mid p \in \mathsf{AP}_\epsilon, 1 \le i \le k, 0 \le j < N\}$ and $\overline{Q} = \{q_j \mid 0 \le j < N\}$). We apply the following modifications to $\overline{\varphi}$ to obtain $\overline{\varphi'}$:

- introduce atomic propositions $\{p^\epsilon_i \mid 1 \le i \le k\}$ and add the conjunct

$$\wedge_{1\le i\le k} \mathbf{G}\left(\big(\wedge_{0\le j<N} (q_j \Rightarrow p^\epsilon_{i,j})\big) \Leftrightarrow p^\epsilon_i\right);$$

- introduce atomic propositions $\{q_{i,j} \mid 1 \le i \le k, 0 \le j < N\}$ and add the conjunct

$$\wedge_{1\le i\le k} \mathbf{G}\left(\wedge_{0\le j<N} (\neg p^\epsilon_i \wedge q_j \Leftrightarrow q_{i,j})\right);$$

- project away $\{p^\epsilon_{i,j} \mid 1 \le i \le k, 0 \le j < N\}$ and $\overline{Q}$;
- replace all occurrences of p^ϵ_i by $\bot_i$.

Now, as we mentioned earlier, we can write $\mathcal{A}$ as an (MSO[$<$, +1] [48]) formula $\varphi_\mathcal{A} = \exists X_\mathcal{A} \, \psi_\mathcal{A}$ where $X_\mathcal{A}$ is a set of atomic propositions such that $\mathsf{AP} \cap X_\mathcal{A} = \emptyset$ and $\psi_\mathcal{A}$ is an MTL formula over $\mathsf{AP} \cup X_\mathcal{A}$. Let $\overline{\varphi_\mathcal{A}}$ be its stacked counterpart $\exists \overline{X_\mathcal{A}} \, \exists Y \, \overline{\psi_\mathcal{A}}$; we translate $\overline{\varphi_\mathcal{A}}$ back into an untimed automaton $\overline{\mathcal{A}}$ over the stacked alphabet $\overline{\mathsf{AP}} \cup \overline{Q}$. The problem thus reduces to

untimed model checking of $\overline{\mathcal{A}}$ against $\exists\pi_1\,\forall\pi_2\,\ldots\exists\pi_{k-1}\,\forall\pi_k\,\overline{\varphi'}$ in the asynchronous semantics, which is decidable by Proposition 17 ($\overline{\varphi'}$ has outermost existential propositional quantifiers, but clearly the equivalent automaton can be used directly in the algorithm).

Finally, note that the proof is simpler for the case of synchronous semantics: we can simply work with a (non-stuttering) MSO[$<,+1$] formula in all the intermediate steps without translating it into an automaton, and then check the satisfiability of the final formula by stacking it into a QPTL formula.

▶ **Proposition 20.** *Model checking* HyperMTL *is decidable when the time domain is* $[0, N)$*, where* N *is a given positive integer.*

References

1 Parosh Aziz Abdulla, Johann Deneux, Joël Ouaknine, Karin Quaas, and James Worrell. Universality Analysis for One-Clock Timed Automata. *Fundam. Inform*, 89(4):419–450, 2008.

2 Erika Ábrahám and Borzoo Bonakdarpour. HyperPCTL: A temporal logic for probabilistic hyperproperties. In *QEST*, volume 11024 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2018.

3 Shreya Agrawal and Borzoo Bonakdarpour. Runtime Verification of k-Safety Hyperproperties in HyperLTL. In *CSF*, pages 239–252. IEEE Computer Society, 2016.

4 Rajeev Alur and David L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

5 Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The Benefits of Relaxing Punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

6 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, 1992.

7 Rajeev Alur and Thomas A. Henzinger. Logics and Models of Real Time: A Survey. In *REX*, volume 600 of *LNCS*, pages 74–106. Springer-Verlag, 1992.

8 Rajeev Alur and Thomas A. Henzinger. Real-Time Logics: Complexity and Expressiveness. *Information and Computation*, 104(1):35–77, 1993.

9 Rajeev Alur and Thomas A. Henzinger. A Really Temporal Logic. *Journal of the ACM*, 41(1):164–169, 1994.

10 Gilles Barthe, Pedro R. D'Argenio, and Tamara Rezk. Secure information flow by self-composition. *Mathematical Structures in Computer Science*, 21(6):1207–1252, 2011.

11 Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. Characterization of the Expressive Power of Silent Transitions in Timed Automata. *Fundam. Inform*, 36(2-3):145–182, 1998.

12 Borzoo Bonakdarpour and Bernd Finkbeiner. The Complexity of Monitoring Hyperproperties. In *CSF*, pages 162–174. IEEE Computer Society, 2018.

13 D. Brand and P. Zafiropulo. On communicating finite state machines. *Journal of the ACM*, 30:323–342, 1983.

14 Thomas Brihaye, Morgane Estiévenart, Gilles Geeraerts, Hsi-Ming Ho, Benjamin Monmege, and Nathalie Sznajder. Real-Time Synthesis is Hard! In *FORMATS*, volume 9884 of *LNCS*, pages 105–120. Springer, 2016.

15 Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. NuSMV2: An opensource tool for symbolic model checking. In *CAV*, volume 2404 of *LNCS*, pages 359–364. Springer, 2002.

16 Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. Temporal Logics for Hyperproperties. In *POST*, volume 8414 of *LNCS*, pages 265–284. Springer, 2014.

17 Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.

18 Laurent Doyen, Gilles Geeraerts, Jean-François Raskin, and Julien Reichert. Realizability of Real-Time Logics. In *FORMATS*, volume 5813 of *LNCS*, pages 133–148. Springer, 2009.

19 Deepak D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *STACS*, volume 2285 of *LNCS*, pages 571–582. Springer, 2002.

20 Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. Spot 2.0 - A Framework for LTL and ω-Automata Manipulation. In *ATVA*, volume 9938 of *LNCS*, pages 122–129. Springer, 2016.

21 Thomas Ferrère. The Compound Interest in Relaxing Punctuality. In *FM*, volume 10951 of *LNCS*, pages 147–164. Springer, 2018.

22 Bernd Finkbeiner and Christopher Hahn. Deciding Hyperproperties. In *CONCUR*, volume 59 of *LIPIcs*, pages 13:1–13:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

23 Bernd Finkbeiner, Christopher Hahn, and Tobias Hans. MGHyper: Checking Satisfiability of HyperLTL Formulas Beyond the $\exists^*\forall^*$ Fragment. In *ATVA*, volume 11138 of *Lecture Notes in Computer Science*, pages 521–527. Springer, 2018.

24 Bernd Finkbeiner, Christopher Hahn, and Marvin Stenger. EAHyper: Satisfiability, Implication, and Equivalence Checking of Hyperproperties. In *CAV*, volume 10427 of *Lecture Notes in Computer Science*, pages 564–570. Springer, 2017.

25 Bernd Finkbeiner, Christopher Hahn, Marvin Stenger, and Leander Tentrup. Monitoring Hyperproperties. In *RV*, volume 10548 of *LNCS*, pages 190–207. Springer, 2017.

26 Bernd Finkbeiner, Christopher Hahn, Marvin Stenger, and Leander Tentrup. RVHyper: A runtime verification tool for temporal hyperproperties. In *TACAS*, volume 10806 of *Lecture Notes in Computer Science*, pages 194–200. Springer, 2018.

27 Bernd Finkbeiner, Christopher Hahn, and Hazem Torfah. Model Checking Quantitative Hyperproperties. In *CAV*, volume 10981 of *Lecture Notes in Computer Science*, pages 144–163. Springer, 2018.

28 Bernd Finkbeiner, Markus N. Rabe, and César Sánchez. Algorithms for Model Checking HyperLTL and $HyperCTL^*$. In *CAV*, volume 9206 of *LNCS*, pages 30–48. Springer, 2015.

29 Guillaume Gardey, John Mullins, and Olivier H. Roux. Non-Interference Control Synthesis for Security Timed Automata. *Electr. Notes Theor. Comput. Sci*, 180(1):35–53, 2007.

30 Christopher Gerking, David Schubert, and Eric Bodden. Model Checking the Information Flow Security of Real-Time Systems. In *ESSoS*, volume 10953 of *LNCS*, pages 27–43. Springer, 2018.

31 J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *S&P*, pages 11–20. IEEE Computer Society, 1982.

32 Jens Heinen. Model Checking Timed Hyperproperties. Master's thesis, Saarland University, 2018.

33 Thomas A. Henzinger, Jean-François Raskin, and Pierre-Yves Schobbens. The Regular Real-Time Languages. In *ICALP*, volume 1443 of *LNCS*, pages 580–591. Springer, 1998.

34 Hsi-Ming Ho. On the Expressiveness of Metric Temporal Logic over Bounded Timed Words. In *RP*, volume 8762 of *Lecture Notes in Computer Science*, pages 138–150. Springer, 2014.

35 Gerard J. Holzmann. The Model Checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

36 John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

37 Marieke Huisman, Pratik Worah, and Kim Sunesen. A Temporal Logic Characterisation of Observational Determinism. In *CSFW*, page 3. IEEE Computer Society, 2006.

38 Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre Attacks: Exploiting Speculative Execution. *CoRR*, abs/1801.01203, 2018. `arXiv:1801.01203`.

39 Ron Koymans. Specifying Real-time Properties with Metric Temporal Logic. *Real-Time Systems*, 2(4):255–299, 1990.

40 Jan Kretínský, Tobias Meggendorfer, and Salomon Sickert. Owl: A Library for ω-Words, Automata, and LTL. In *ATVA*, volume 11138 of *Lecture Notes in Computer Science*, pages 543–550. Springer, 2018.

41 Antonín Kučera and Jan Strejček. The stuttering principle revisited. *Acta Informatica*, 41(7–8):415–434, 2005.

42 L. Lamport. What good is Temporal Logic? In R.E.A. Mason, editor, *IFIP Congress*, pages 657–667, Amsterdam, 1983. North-Holland.

43 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.

44 Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault Sensitivity Analysis. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 320–334. Springer, 2010.

45 Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading Kernel Memory from User Space. In *USENIX Security Symposium*, pages 973–990. USENIX Association, 2018.

46 John McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *S&P*, pages 79–93. IEEE Computer Society, 1994.

47 Luan Viet Nguyen, James Kapinski, Xiaoqing Jin, Jyotirmoy V. Deshmukh, and Taylor T. Johnson. Hyperproperties of real-valued signals. In *MEMOCODE*, pages 104–113. ACM, 2017.

48 Joël Ouaknine, Alexander Rabinovich, and James Worrell. Time-bounded verification. In *Proceedings of CONCUR 2009*, volume 5710 of *LNCS*, pages 496–510. Springer, 2009.

49 Joël Ouaknine and James Worrell. On the Language Inclusion Problem for Timed Automata: Closing a Decidability Gap. In *LICS*, pages 54–63. IEEE Computer Society, 2004.

50 Joël Ouaknine and James Worrell. On the Decidability and Complexity of Metric Temporal Logic over Finite Words. *Logical Methods in Computer Science*, 3(1), 2007.

51 Joël Ouaknine and James Worrell. Some Recent Results in Metric Temporal Logic. In *FORMATS*, volume 5215 of *LNCS*, pages 1–13. Springer, 2008.

52 Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.

53 Jean-François Raskin. *Logics, automata and classical theories for deciding real time.* PhD thesis, FUNDP (Belgium), 1999.

54 A. W. Roscoe. CSP and determinism in security modelling. In *S&P*, pages 114–127. IEEE Computer Society, 1995.

55 Laurent Simon, David Chisnall, and Ross J. Anderson. What You Get is What You C: Controlling Side Effects in Mainstream C Compilers. In *EuroS&P*, pages 1–15. IEEE, 2018.

56 A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. The Complementation Problem for Büchi Automata with Applications to Temporal Logic (Extended Abstract). In *ICALP*, volume 194 of *LNCS*, pages 465–474. Springer, 1985.

57 Ming-Hsien Tsai, Yih-Kuen Tsay, and Yu-Shiang Hwang. GOAL for Games, Omega-Automata, and Logics. In *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 883–889. Springer, 2013.

58 Panagiotis Vasilikos, Flemming Nielson, and Hanne Riis Nielson. Secure Information Release in Timed Automata. In *POST*, volume 10804 of *LNCS*, pages 28–52. Springer, 2018.

59 Steve Zdancewic and Andrew C. Myers. Observational Determinism for Concurrent Program Security. In *CSFW*, page 29. IEEE Computer Society, 2003.