

33rd International Symposium on Temporal Representation and Reasoning

TIME 2026, September 1–3, 2026, Cork, Ireland

Edited by

Andrea Orlandini
Sophie Pinchinat



Editors

Andrea Orlandini 

National Research Council, Rome, Italy
andrea.orlandini@istc.cnr.it

Sophie Pinchinat 

IRISA Laboratory/University of Rennes, France
sophie.pinchinat@irisa.fr

ACM Classification 2012

Computing methodologies → Temporal reasoning; Information systems → Spatial-temporal systems;
Theory of computation → Modal and temporal logics; Information systems → Temporal data; Mathematics
of computing → Time series analysis; Theory of computation → Timed and hybrid models; Computing
methodologies → Discrete-event simulation

ISBN 978-3-95977-448-2

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/978-3-95977-448-2>.

Publication date

October, 2026

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists all publications of this volume in the Deutsche Nationalbibliografie;
detailed bibliographic data are available in the Internet at <https://portal.dnb.de>.

License

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



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

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

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/OASlcs.TIME.2026.0

ISBN 978-3-95977-448-2

ISSN 1868-8969

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

OASlcs – OpenAccess Series in Informatics

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

Editorial Board

- Daniel Cremers (TU München, Germany)
- Barbara Hammer (Universität Bielefeld, Germany)
- Marc Langheinrich (Università della Svizzera Italiana – Lugano, Switzerland)
- Dorothea Wagner (*Editor-in-Chief*, Karlsruher Institut für Technologie, Germany)

ISSN 1868-8969

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

■ Contents

Preface	
<i>AndreA Orlandini and Sophie Pinchinat</i>	0:vii
TIME Steering Committee	
.....	0:ix
Program Committee Members	
.....	0:xi
List of Authors	
.....	0:xiii

Papers

Revisiting the Expressiveness of Metric Temporal Logic	
<i>Mohammed Aristide Foughali</i>	1:1–1:18
Core Fragments of Compass Logic	
<i>Pietro Casavecchia, Guido Sciavicco, and Leonardo Serrentino</i>	2:1–2:16
The Σ -Chain Product: A Succinct Model of Automata (De)Composition	
<i>Roberto Borelli, Davide Bresolin, Luca Geatti, Angelo Montanari,</i> <i>and Matteo Zavatteri</i>	3:1–3:17
Discrete Linear Ensemble Logic: Decidability, Expressiveness, and Axiomatization	
<i>Manfred Droste and Guo-Qiang Zhang</i>	4:1–4:18
Social Networks Through Time: Completeness of Temporal Network Logic	
<i>Saúl Fernández González, Mina Young Pedersen, and Sonja Smets</i>	5:1–5:21
Querying Interval-Based Temporal Data with SPARQL	
<i>Julien Corman, Roman Kontchakov, and Cem Okumus</i>	6:1–6:21
A Multi-Scale Process for Mining and Abstracting Environmental Spatio-Temporal Graphs	
<i>Assaad Zeghina, Aurélie Leborgne, Florence Le Ber, and Antoine Vacavant</i>	7:1–7:15
The Satisfiability Problem of Temporal-Spatial Logics over Quasi-Temporal Graphs	
<i>Eric Alsmann, Martin Lange, and Igor Semezie</i>	8:1–8:16
Dynamic Quantitative Skill Distribution in Multi-Agent Temporal Planning as SMT	
<i>Josselin Guéron and Frédéric Maris</i>	9:1–9:20
A Formal Model for Forensically Sound Timestamp Normalization	
<i>Lorenz Hornung, Carly Bakker, Bouke Timbermont, Roel van Dijk,</i> <i>Harm van Beek, and Natasha Alechina</i>	10:1–10:21
Decidable Reasoning About Time in Finite-Domain Situation Calculus Theories	
<i>Till Hofmann, Stefan Schupp, and Gerhard Lakemeyer</i>	11:1–11:22

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat

OpenAccess Series in Informatics



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

Earth Observation Satellite Constellation Planning with Matheuristics and
Metaheuristics Combination

Romain Barrault, Cedric Pralet, Gauthier Picard, and Eric Sawyer 12:1–12:17

Multi-Agent Path Finding with Tasks Under Time Uncertainty

Alexandre Albore, Christophe Grand, Noëlie Ramuzat, and Ajdin Sumic 13:1–13:21

■ Preface

We are delighted to welcome you to the 33rd International Symposium on Temporal Representation and Reasoning (TIME 2026). This year, TIME 2026 is co-located with the 37th International Conference on Principles of Diagnosis and Resilient Systems (DX'26) and is hosted from the 1st to the 3rd of September 2026 at the University College of Cork.

TIME has been for more than twenty years the only yearly multidisciplinary international event dedicated to the topic of time in computer science. The purpose of the symposium is to bring together active researchers in different scientific fields involving temporal and spatio-temporal data, information, and/or knowledge management. Such a concern arises in a number of different though often related research domains, namely Artificial Intelligence (both symbolic approaches based on explicit Logic or Constraint-based models, and numerical data-based approaches such as Deep Learning and Large Language Models), Databases and Data Mining, or System Specification and Verification. In an effort to broaden the symposium program we considered in TIME 2026 three types of submissions:

- Original long papers (15 pages), presenting unpublished theoretical (algorithms, models, proofs) or applied (systems, evaluations, applications) work;
- Survey papers (15 pages), offering concise overviews of established research areas;
- Extended abstracts (4 pages), initially intended to stimulate discussion and include work-in-progress, project reviews, PhD summaries, or summaries of papers published elsewhere.

This year, we received 26 submissions. Following a rigorous single blind peer-review process, 16 papers were accepted for presentation: 15 original long papers and 1 short paper (submitted extended abstracts). There were no survey papers among the submissions for this edition. Following requests from the respective authors, two papers accepted for presentation at the conference have been withdrawn from publication in these proceedings. These papers were presented as part of the conference program, but are intentionally omitted from this volume. Accordingly, the proceedings include all accepted contributions for which publication in the proceedings was authorized by the authors.

The program also features two invited talks by Kristin-Yvonne Rozier, Iowa State University, USA, and (shared with DX 2026) Alessandro Cimatti, Fondazione Bruno Kessler, Italy. The details of the talks are not available in this moment.

This year we also organized an open session where all participants may present interesting questions, open problems, highest priority locks that prevent the field from progressing, etc. This is for fostering interactions and possibly leading to new collaborations and synergies among attendees.

We are deeply grateful to the members of the program committee for their diligence and commitment to maintaining the scientific quality of the symposium, as well as to all authors for the care taken in their submissions and final versions of their articles and in their oral presentations, including our invited speakers.

In view of the high quality of these contributions, we have decided to give the best paper award is **Revisiting the Expressiveness of Metric Temporal Logic** which author is Mohammed Foughali. This paper provides a significant theoretical contribution by revisiting the expressiveness of Metric Temporal Logic, demonstrating the incomparability of

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat



OpenAccess Series in Informatics

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

its pointwise and interval-based semantics, and introducing a novel mixed semantics that unifies both approaches, offering a deeper understanding of temporal reasoning over timed executions.

We also thank the DX organizing team, Ingo Pill, Gregory Provan, and Marina Zanella, for their dedication in making this event a success, and express our appreciation to OASICs for publishing the proceedings and supporting open-access dissemination of scientific knowledge.

We hope that TIME 2026 provides a stimulating and collaborative environment for all participants and contributes meaningfully to the advancement of temporal representation and reasoning.

TIME 2026 Program Committee Chairs

Andrea Orlandini, National Research Council of Italy, Italy

Sophie Pinchinat, IRISA Laboratory/University of Rennes, France

■ TIME Steering Committee

Alexander Artikis

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

Carlo Combi (Chair)

University of Verona
Italy
carlo.combi@univr.it

Thomas Guyet

IRISA
France
thomas.guyet@irisa.fr

Martin Lange (Chair)

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

Shankara Narayanan Krishna (Krishna S.)

IIT Bombay
India
krishnas@cse.iitb.ac.in

Patricia Bouyer

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

Johann Eder

University of Klagenfurt
Austria
johann.eder@aau.at

Luke Hunsberger

Vassar College
United States
hunsberger@vassar.edu

Angelo Montanari

University of Udine
Italy
angelo.montanari@uniud.it

Mark Reynolds

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



■ Program Committee Members

Sundaraman Akshay

Indian Institute of Technology Bombay
India
akshayss@cse.iitb.ac.in

Alexandros Artikis

NCSR Demokritos & University of Piraeus
Greece
a.artikis@iit.demokritos.gr

Béatrice Bérard

Sorbonne University
France
beatrice.berard@lip6.fr

Jaewook Byun

Sejong University
South Korea
jwbyun@sejong.ac.kr

Nicola Gigante

Free University of Bozen-Bolzano
Italy
nicola.gigante@unibz.it

Peter Jonsson

Linköping University
Sweden
peter.jonsson@liu.se

Martin Lange

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

Periklis Mantenoglou

Örebro University
Sweden
periklis.mantenoglou@oru.se

Andrea Micheli

FBK, Trento
Italy
amicheli@fbk.eu

Andrea Orlandini

ISTC-CNR, Roma
Italy
andrea.orlandini@istc.cnr.it

Sophie Pinchinat

IRISA, Rennes
France
sophie.pinchinat@irisa.fr

Vladislav Ryzhikov

Birkbeck, University of London
United Kingdom
v.ryzhikov@bbk.ac.uk

Beatrice Amico

University of Verona
Italy
beatrice.amico@univr.it

Massimo Benerecetti

Università di Napoli "Federico II"
Italy
massimo.benerecetti@unina.it

Florian Bruse

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

Carlo Combi

University of Verona
Italy
carlo.combi@univr.it

Thomas Guyet

Inria-AIstroSight
France
thomas.guyet@inria.fr

Roman Kontchakov

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

Ruizhe Ma

University of Massachusetts Lowell
United States of America
ruizhe_ma@uml.edu

Maria Chiara Meo

"G. d'Annunzio" University of Chieti-Pescara
Italy
mariachiara.meo@unich.it

Angelo Montanari

University of Udine
Italy
angelo.montanari@uniud.it

Yannick Pencolé

LAAS-CNRS, Toulouse
France
yannick.pencole@laas.fr

Roberto Posenato

University of Verona
Italy
roberto.posenato@univr.it

Marco Saelzer

University of Kassel
Germany
marco.saelzer@uni-kassel.de

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: Andrea Orlandini and Sophie Pinchinat



OpenAccess Series in Informatics

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

Yakoub Salhi

Artois University
France
salhi@cril.fr

Guido Sciavicco

University of Ferrara
Italy
scvgdu@unife.it

Kathleen Stewart

University of Maryland
US
stewartk@umd.edu

Paolo Terenziani

University of Piemonte Orientale, Alessandria
Italy
paolo.terenziani@unipmn.it

Emanuel Sallinger

Technical University, Wien
Austria
emanuel.sallinger@tuwien.ac.at

Michael Sioutis

University of Montpellier
France
michael.sioutis@lirimm.fr

Ajdin Sumic

ONERA, Toulouse
France
aidin.sumic@onera.fr

Matteo Zavatteri

University of Padua
Italy
matteo.zavatteri@unipd.it

■ List of Authors

Alexandre Albore  (13)
ONERA, University of Toulouse, France

Natasha Alechina  (10)
Open Universiteit, Heerlen, The Netherlands;
Utrecht University, The Netherlands

Eric Alsmann  (8)
Theoretical Computer Science / Formal
Methods, University of Kassel, Germany

Carly Bakker (10)
Netherlands Forensic Institute,
The Hague, The Netherlands

Romain Barrault  (12)
DTIS, ONERA, Université de Toulouse, 31000,
Toulouse, France

Roberto Borelli  (3)
University of Padua, Italy

Davide Bresolin  (3)
University of Padua, Italy

Pietro Casavecchia  (2)
Department of Mathematics and Computer
Science, University of Ferrara, Italy

Julien Corman  (6)
Free University of Bozen-Bolzano, Italy

Manfred Droste  (4)
Institute of Computer Science, Leipzig
University, Germany

Saúl Fernández González  (5)
University of Oviedo, Spain

Mohammed Aristide Foughali  (1)
Université Paris Cité, CNRS, IRIF, F-75013,
Paris, France

Luca Geatti  (3)
University of Udine, Italy

Christophe Grand  (13)
ONERA, University of Toulouse, France

Josselin Guéron  (9)
Université Caen Normandie, ENSICAEN,
CNRS, Normandie Univ, GREYC UMR6072,
F-14000 Caen, France

Till Hofmann  (11)
RWTH Aachen University, Germany;
University of Oxford, UK

Lorenz Hornung  (10)
Open Universiteit, Heerlen, The Netherlands

Roman Kontchakov  (6)
Birkbeck, University of London, UK

Gerhard Lakemeyer  (11)
RWTH Aachen University, Germany

Martin Lange  (8)
Theoretical Computer Science / Formal
Methods, University of Kassel, Germany

Florence Le Ber  (7)
Université de Strasbourg, CNRS, ENGEEES,
ICube UMR 7357, Strasbourg, France

Aurélié Leborgne  (7)
Université de Strasbourg, CNRS, ENGEEES,
ICube UMR 7357, Strasbourg, France

Frédéric Maris  (9)
Université de Toulouse, IRIT, France

Angelo Montanari  (3)
University of Udine, Italy

Cem Okulmus  (6)
Paderborn University, Germany

Mina Young Pedersen  (5)
Institute for Logic, Language and Computation,
University of Amsterdam, The Netherlands

Gauthier Picard  (12)
DTIS, ONERA, Université de Toulouse, 31000,
Toulouse, France

Cedric Pralet  (12)
DTIS, ONERA, Université de Toulouse, 31000,
Toulouse, France

Noëlie Ramuzat  (13)
ONERA, University of Toulouse, France

Eric Sawyer  (12)
CNES, Toulouse, France

Stefan Schupp  (11)
TU Wien, Austria

Guido Sciavicco  (2)
Department of Mathematics and Computer
Science, University of Ferrara, Italy

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat



OpenAccess Series in Informatics

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

Igor Semeziez  (8)

Theoretical Computer Science / Formal
Methods, University of Kassel, Germany;
Laboratoire Méthodes Formelles, ENS
Paris-Saclay, France

Leonardo Serrentino  (2)

Department of Mathematics and Computer
Science, University of Ferrara, Italy

Sonja Smets  (5)

Institute for Logic, Language and Computation,
University of Amsterdam, The Netherlands

Ajdin Sumic  (13)

ONERA, University of Toulouse, France

Bouke Timmermont (10)

Open Universiteit, Heerlen, The Netherlands;
Netherlands Forensic Institute,
The Hague, The Netherlands

Antoine Vacavant  (7)

Université Clermont Auvergne, CNRS,
SIGMA Clermont, Institut Pascal, France

Harm van Beek  (10)

Open Universiteit, Heerlen, The Netherlands;
Netherlands Forensic Institute,
The Hague, The Netherlands

Roel van Dijk (10)

Netherlands Forensic Institute,
The Hague, The Netherlands

Matteo Zavatteri  (3)

Fondazione Bruno Kessler, Trento, Italy

Assaad Zaghina  (7)

Université de Strasbourg, CNRS, ENGEEES,
ICube UMR 7357, Strasbourg, France

Guo-Qiang Zhang  (4)

The University of Texas Health Science Center
at Houston, TX, USA

Revisiting the Expressiveness of Metric Temporal Logic

Mohammed Aristide Foughali 

Université Paris Cité, CNRS, IRIF, F-75013, Paris, France

Abstract

The expressiveness of Metric Temporal Logic (MTL) has been extensively studied throughout the last two decades. In particular, it has been shown that the *interval-based* semantics of MTL is strictly more expressive than the *pointwise* one. These results may suggest that enabling the evaluation of formulae at arbitrary time points *instead of* positions of timed events increases the expressive power of MTL. In this paper, we formally argue otherwise. We demonstrate that under standard models of finite or non-Zeno infinite (action-based) timed executions, the interval-based and the pointwise semantics are incomparable. We then propose a new *mixed* semantics that embeds both the pointwise and the interval-based ones.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Timed logics, Metric temporal logic

Digital Object Identifier 10.4230/OASICS.TIME.2026.1

Related Version *Full Version*: <https://arxiv.org/abs/2603.15379> [17]

Funding This work received partial support from the ANR project ERVoRS (ANR-25-CE48-6220).

1 Introduction

Metric Temporal Logic (MTL) [22] is a popular *timed* extension of the Linear-time Temporal Logic LTL [28]. The importance of MTL motivated a large body of research investigating its expressive power [14, 9, 10, 20, 27, 19]. In this paper, we focus on the expressiveness of the *pointwise* semantics of MTL as compared to its *interval-based* one. In order to present the existing results on such comparison, we first (informally) introduce the model of *timed words*.

A *timed event* is a couple containing a letter, representing an *action*, and a real number corresponding to its *timestamp*, i.e., the global time at which it occurred. A timed word is a finite or infinite sequence of timed events. Following the least constrained model of timed words [2, 25, 26], which we refer to hereafter as the *general model*, a (finite or infinite) timed word satisfies time monotonicity, i.e., timestamps are non-decreasing, and an infinite timed word obeys time divergence (also called non-Zenoness), that is global time does not converge towards some bounded value. For instance, over the alphabet $\{a, b, c\}$, the sequence $(a, 0)(b, 1)(c, 1)(a, 5.3)$ is a finite timed word whereas $(a, 0)(b, 1)(c, 0.8)$ is not (time monotonicity is violated). Similarly, the infinite timed word $(a, \frac{1}{2})(b, \frac{3}{4})(b, \frac{7}{8})(c, \frac{15}{16}) \dots$, i.e., where the difference between two consecutive timestamps is halved at every next action, is excluded (global time converges towards 1).

Under the pointwise semantics, formulae are evaluated at the *positions* of timed events within the timed word, whereas the interval-based semantics is given over continuous intervals within a *timed signal* [27]. To enable the comparison between both semantics, D’Souza & Prabhakar [14] interpret the interval-based semantics over timed words (instead of timed signals), allowing the evaluation of formulae at arbitrary time points (including those not corresponding to a timestamp). They demonstrate accordingly that the interval-based semantics



© Mohammed Aristide Foughali;

licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 1; pp. 1:1–1:18

OpenAccess Series in Informatics



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

is strictly more expressive than the pointwise one over finite timed words, using a constrained model where time monotonicity is strict (e.g., the timed word $(a, 0)(b, 1)(c, 1)(a, 5.3)$ used as an example above is excluded by design). We refer to this existing result as **ER**:

ER: The interval-based semantics is strictly more expressive than the pointwise one over strictly monotone finite timed words.

However, we argue that while strict monotonicity of time is reasonable in many cases, it is not justified in general. Indeed, simultaneous interleaving in timed systems, theorised since [13, 3], is of sheer importance when formally reasoning about concurrent behaviour, e.g., in multiprocessor real-time scheduling (see for instance [16, 11, 18]), and distributed applications [24, 15, 23]. There is therefore a need to complete the picture of **ER** and, in case of incomparability of the interval-based and the pointwise semantics, unify them in order to benefit from their respective advantages.

In a remotely related work, Bouyer et al. proved in two seminal papers [9, 10] that TPTL [4] is strictly more expressive than MTL. These two publications [9, 10] are of particular interest to us for the following two reasons:

- They use a clever encoding of timed words into *timed state sequences* to reason over the interval-based semantics. This encoding will inspire our incomparability proofs (contributions **C1**, **C2** and **C3** below).
- While considering the general model of infinite timed words [9, Sect. 2][10, Sect. 2], they make the following side claim [10, Section 1][9, Sect. 1]:

“As side results, we get that MTL is strictly more expressive under the interval-based semantics than under the pointwise one”

We formally show that this claim does not hold (Theorem 3.27).

Contributions. In this paper, we (i) answer the comparability question in the negative under the general model of timed words and (ii) devise a new mixed semantics that remedies such incomparability. More precisely, we provide the following contributions:

- **C1** The interval-based and the pointwise semantics are incomparable under the general model of finite timed words,
- **C2** The interval-based and the pointwise semantics are incomparable under the general model of infinite timed words,
- **C3** The incomparability holds even under reasonable restrictions such as *non-stuttering*,
- **C4** A mixed semantics embedding both the interval-based and the pointwise ones is proposed.

C1, **C2** and **C3** complete the picture of **ER**, showing that it does not hold under the more general (non-strict) monotonicity, regardless of whether timed words are finite or infinite. In addition, the mixed semantics (**C4**) conjugates the expressive powers of the interval-based and the pointwise ones, while embedding both semantics through automatic compilers.

Outline. Section 2 is dedicated to preliminaries, whereas the core of our contributions is detailed in Section 3 (contributions **C1**, **C2** and **C3**) and Section 4 (contribution **C4**). Section 5 wraps up with concluding remarks and future work directions.

2 Preliminaries

2.1 Binary relations and functions

Let X and Y be two non-empty sets.

► **Definition 2.1** (Binary relation). A binary relation $\mathcal{R}$ over X and Y is a subset of the cartesian product $X \times Y$. The inverse relation $\mathcal{R}^{-1}$ of $\mathcal{R}$ is defined as $\mathcal{R}^{-1} = \{(y, x) \in Y \times X \mid (x, y) \in \mathcal{R}\}$.

For any $x \in X$, we let $\mathcal{R}[x] = \{y \in Y \mid (x, y) \in \mathcal{R}\}$ denote the set of all images of x by $\mathcal{R}$. $\mathcal{R}$ is said left total iff every element of X has at least an image by $\mathcal{R}$, i.e., $\forall x \in X : \mathcal{R}[x] \neq \emptyset$, right total iff $\mathcal{R}^{-1}$ is left total, and total if it is both right and left total.

► **Definition 2.2** (Function). A binary relation $\mathcal{R}$ over X and Y is called a function from X to Y iff every element in X has exactly one image in Y , i.e., $\forall x \in X : |\mathcal{R}[x]| = 1$. We use the standard notation for functions, i.e., $\mathcal{R} : X \rightarrow Y$ for “ $\mathcal{R}$ a function from X to Y ” and $\mathcal{R}(x) = y$ (instead of $\mathcal{R}[x] = \{y\}$) for “ y is the image of x by $\mathcal{R}$ ”.

► **Definition 2.3** (Injective, surjective and bijective functions). A function $\mathcal{R}$ is said:

- injective iff all images are distinct: $\forall y \in Y : |\mathcal{R}^{-1}[y]| \leq 1$,
- surjective iff it is right total ($\mathcal{R}^{-1}$ is therefore a total relation),
- bijective iff it is both injective and surjective ($\mathcal{R}^{-1}$ is therefore a bijective function).

2.2 Timed words and timed state sequences

Let s be a sequence over a non-empty set X . We let sequences start at position 0, and $|s|$ denote the length of s defined in the usual way, with $|s| = \omega$ if s is infinite.

2.2.1 Timed words

The definitions provided in this subsection follow the general model of timed words, they can be seen as a generalisation of those given in [9] to both finite and infinite timed words.

► **Definition 2.4** (Time sequence). A time sequence τ is a non-empty sequence over $\mathbb{R}_{\geq 0}$ satisfying the following conditions:

- Monotonicity: $\forall i \in \mathbb{N}, 0 < i < |\tau| : \tau_{i-1} \leq \tau_i$,
- Time divergence: $|\tau| = \omega \Rightarrow (\forall t \in \mathbb{R}_{\geq 0} \exists i \in \mathbb{N} : \tau_i > t)$.

► **Remark 2.5.** In Definition 2.4, time divergence is only required for infinite time sequences. This is because every finite time sequence satisfying monotonicity is a prefix of a time-divergent infinite time sequence [5].

Let Σ be a non-empty, finite set of letters, Σ^+ and Σ^ω denote, respectively, the set of non-empty finite and infinite words over Σ , and $\Sigma^\infty = \Sigma^+ \cup \Sigma^\omega$.

► **Definition 2.6** (Timed word). A timed word is a pair $\rho = (\sigma, \tau)$ with $\sigma \in \Sigma^\infty$ and τ a time sequence such that $|\sigma| = |\tau|$. The length of ρ is $|\rho| = |\sigma|$, and each (σ_i, τ_i) is called a timed event. For some $i \in \mathbb{N}, i < |\rho|$, τ_i is called the timestamp of σ_i .

Accordingly, if $\rho = (\sigma, \tau)$ is finite, say of length $n \in \mathbb{N}_{>0}$ (resp. infinite), $\sigma = \sigma_0 \sigma_1 \dots \sigma_{n-1}$ is a finite word in Σ^+ and $\tau = \tau_0 \tau_1 \dots \tau_{n-1}$ is a finite time sequence (resp. $\sigma = \sigma_0 \sigma_1 \dots$ is an infinite word in Σ^ω and $\tau = \tau_0 \tau_1 \dots$ is an infinite time sequence). The duration of ρ , denoted $\mu(\rho)$, equals the timestamp of the last letter if ρ is finite, i.e., $\mu(\rho) = \tau_{|\rho|-1}$. Otherwise, $\mu(\rho) = \infty$ thanks to time divergence (Definition 2.4). We sometimes use the alternative notation of sequences of timed events (e.g., $(\sigma_0, \tau_0)(\sigma_1, \tau_1) \dots (\sigma_{|\rho|-1}, \tau_{|\rho|-1})$ instead of (σ, τ) for some finite timed word). We denote by ${}^m\rho, m < |\rho|$, the prefix of $\rho = (\sigma, \tau)$ up to and including position m , i.e., ${}^m\rho = (\sigma_0, \tau_0) \dots (\sigma_m, \tau_m)$.

► **Remark 2.7.** A timed word $\rho = (\sigma, \tau)$ inherits the properties of the time sequence τ , i.e., time monotonicity and, if ρ is infinite, time divergence (Definition 2.4).

► **Example 2.8.** Let $\Sigma = \{a, b, c\}$. The sequence $\rho_1 = (a, 1)(b, 1.3)(c, 3.5)(b, 3.5)$ is a timed word of length 4 and duration 3.5. The sequence $\rho_2 = (a, 1)(b, 1.3)(c, 1.2)(b, 3.5)$ is not a timed word (violation of time monotonicity). The sequence $\rho_3 = (\sigma_i, \tau_i)$, $i \in \mathbb{N}$ defined next is an infinite timed word: $\sigma_{2i} = a$, $\sigma_{2i+1} = b$; $\tau_0 = 0$ and for all i in $\mathbb{N}_{>0}$: $\tau_i = \tau_{i-1} + 1$ if $i \bmod 3 = 0$ and $\tau_i = \tau_{i-1}$ otherwise. Its prefix ${}^6\rho_3$ equals $(a, 0)(b, 0)(a, 0)(b, 1)(a, 1)(b, 1)(a, 2)$.

We now define some subclasses of timed words. A strongly stutter-free (resp. strictly monotone) timed word is exempt of the behaviour where two occurrences of the same letter (resp. of any two letters) happen at different positions with the same timestamp.

► **Definition 2.9** (Stutter freeness and strict monotonicity). *A timed word $\rho = (\sigma, \tau)$ is said:*

1. stutter free *iff* $\forall i \neq j \in \mathbb{N}, i, j < |\rho| : \tau_i = \tau_j \Rightarrow \sigma_i \neq \sigma_j$.
2. strictly monotone *iff* τ is strictly monotone: $\forall i \in \mathbb{N}, 0 < i < |\tau| : \tau_{i-1} < \tau_i$.

► **Remark 2.10.** While strict monotonicity implies stutter freeness, none of them is related to time-divergence. For instance, the infinite timed word ρ_3 of Example 2.8 is time divergent but not stutter free (and therefore not strictly monotone).

► **Example 2.11.** $(a, 0)(c, 3)$ is strictly monotone (and therefore stutter free), $(a, 0)(b, 1)(a, 1)(c, 3)$ is stutter free, and $(a, 0)(b, 1)(a, 1)(b, 1)(c, 3)$ is not stutter free.

2.2.2 Timed state sequences

The definitions given in this subsection generalise the model of infinite timed state sequences given in [9, 10] to both the finite and infinite cases.

A *non-empty interval* I over $\mathbb{R}_{\geq 0}$ has the form $[l, u]$, $(l, u]$, $[l, u)$ or (l, u) with its left bound $I^l = l \in \mathbb{R}_{\geq 0}$, right bound $I^u = u \in \mathbb{R}_{\geq 0} \cup \{\infty\}$, $I^l \leq I^u$ if I is right- and left-closed (i.e., has the form $[l, u]$) and $I^l < I^u$ otherwise, and ∞ excluded; let $\mathcal{I}$ be the set of all such intervals. An element of $\mathcal{I}$ denotes a non-empty convex subset of $\mathbb{R}_{\geq 0}$ defined uniquely in the usual way. Two intervals $I_1, I_2 \in \mathcal{I}$ are *adjacent* iff $I_1 \cap I_2 = \emptyset$, $I_1 \cup I_2 \in \mathcal{I}$ and $x < y$ for every $x \in I_1$ and $y \in I_2$. An interval I is *punctual* iff it denotes a singleton set, i.e., it has the form $[l, l]$.

► **Definition 2.12** (Interval sequence). *An interval sequence ι is a non-empty sequence over $\mathcal{I}$ satisfying:*

- *Adjacency:* $\forall i \in \mathbb{N}, 0 < i < |\iota| : \iota_{i-1}$ and ι_i are adjacent,
- *Progress:* $|\iota| = \omega \Rightarrow \forall t \in \mathbb{R}_{\geq 0} \exists i \in \mathbb{N} : t \in \iota_i$.

► **Definition 2.13** (Timed state sequence). *Let AP be a finite set of atomic propositions. A timed state sequence over AP is a pair $\kappa = (\eta, \iota)$ where η is a sequence over $\mathcal{P}(AP)$ (the powerset of AP), ι an interval sequence and $|\eta| = |\iota|$.*

Similarly to timed words, given a timed state sequence $\kappa = (\eta, \iota)$, we (i) sometimes use the alternative sequence notation (η_i, ι_i) , $i \in \mathbb{N}$, $i < |\kappa|$ instead of (η, ι) and (ii) define $\mu(\kappa)$, the duration of κ as $\iota_{|\kappa|-1}^u$ if κ is finite (requiring, without loss of generality, $\iota_{|\kappa|-1}$ to be right closed), and ∞ otherwise (thanks to the progress condition, Definition 2.12). Moreover, given t in $\mathbb{R}_{\geq 0}$ (satisfying $t \leq \mu(\kappa)$ if κ is finite), we denote by $\kappa(t)$ the set η_i where $t \in \iota_i$ (such ι_i is uniquely defined thanks to the adjacency property).

► **Example 2.14.** Let $AP = \{p, q\}$. The sequence $(\{p\}, [1, 2])(\{p, q\}, [2, 3])(\{p\}, [3, 3.4])$ is not a timed state sequence (violation of interval adjacency). The sequence $\kappa = (\{p\}, [1, 2])(\{p, q\}, [2, 3])(\{q\}, [3, 3]), (\{p\}, (3, 3.4])$ is a timed state sequence with length 4 and duration 3.4, and we have $\kappa(1.2) = \kappa(3.1) = \{p\}$, $\kappa(2.2) = \{p, q\}$ and $\kappa(3) = \{q\}$.

2.3 Timed words vs. timed state sequences

A timed word is a natural representation of an *action-based* execution of a timed system. For example, $(a, 2)(b, 2)(c, 4.1)$ corresponds to an execution where the system waited for two time units, performed action a then immediately b , then waited for 2.1 time units and finally performed c . A timed state sequence, on the other hand, is suitable for signals: e.g., $(\{p\}, [0, 2])(\{q, r\}, [2, 3])$ represents a signal where p held for two time units (excluding 2), then q and r held for one time unit. While there is a two-way translation between the untimed versions of these models modulo bisimulation [12], this is not the case in the timed setting, see e.g., the discussion on signals vs. timed words in [7] (this heterogeneity led to the proposition of signal-event frameworks as in [8]). We adopt an action-based view for the simple reason that it is the only way to enable any comparison involving the pointwise semantics. We still need timed state sequences as their use to encode timed words [9, 10] inspire our incomparability proofs.

2.4 Metric Temporal Logic (MTL)

Let $\mathcal{I}_q$ be the set of all non-empty intervals over $\mathbb{R}_{\geq 0}$ with rational endpoints, i.e., defined similarly to $\mathcal{I}$ (Section 2.2.2) but with $l \in \mathbb{Q}_{\geq 0}$, $u \in \mathbb{Q}_{\geq 0} \cup \{\infty\}$. In order to define clean syntax and semantics of MTL under both the pointwise and the interval-based semantics, we equate hereafter the sets of atomic propositions AP and letters (actions) Σ and refer to both as Σ . The syntax of MTL is given below with $a \in \Sigma$ and $I \in \mathcal{I}_q$.

$$\phi, \psi ::= a \mid \neg\phi \mid \phi \wedge \psi \mid \phi \mathcal{U}_I \psi$$

where $\mathcal{U}_I$ denotes the *timed Until* operator (note that we use its standard strict version [9, 27, 25]). The syntactic sugar is defined as usual: $\phi \vee \psi \equiv \neg(\neg\phi \wedge \neg\psi)$, $\top \equiv a \vee \neg a$, $\bigcirc_I \phi \equiv (\neg \bigvee_{a \in \Sigma} a) \mathcal{U}_I \phi$ (*next* ϕ), $\Diamond_I \phi \equiv \top \mathcal{U}_I \phi$ (*eventually* ϕ) and $\Box_I \phi \equiv \neg \Diamond_I \neg\phi$ (*globally* ϕ). We will use often the handy formula $\varphi_\Sigma = \bigvee_{a \in \Sigma} a$ (utilised to define the next operator above) hereafter. We omit the subscript I when $I = [0, \infty)$.

Let $\rho = (\sigma, \tau)$ be a timed word and $\kappa = (\eta, \iota)$ a timed state sequence. Following [9, 14], we force τ_0 to equal 0 and ι_0 to be left closed at 0 without loss of generality. We let P and K be, respectively, the set of all such timed words and timed state sequences. We give below the pointwise semantics (over elements of P) and the two versions of the interval-based semantics (one over P as in [14], and one over K following [9]).

2.4.1 Pointwise semantics

Formulae are evaluated over a timed word $\rho = (\sigma, \tau) \in P$ and a position $i \in \mathbb{N}$ with $i < |\rho|$. To avoid unnecessary cluttering of notation, $j < |\rho|$ is assumed in the semantics of $\phi \mathcal{U}_I \psi$.

$$\begin{array}{ll} \rho, i \models_{pw} a & \text{iff } \sigma_i = a \\ \rho, i \models_{pw} \neg\phi & \text{iff } \rho, i \not\models_{pw} \phi \\ \rho, i \models_{pw} \phi \wedge \psi & \text{iff } \rho, i \models_{pw} \phi \text{ and } \rho, i \models_{pw} \psi \\ \rho, i \models_{pw} \phi \mathcal{U}_I \psi & \text{iff } \exists j > i : \tau_j - \tau_i \in I \text{ and } \rho, j \models_{pw} \psi \text{ and } \forall i < k < j : \rho, k \not\models_{pw} \phi \end{array}$$

2.4.2 Interval-based semantics

Over timed words. Formulae are evaluated over a timed word $\rho = (\sigma, \tau) \in P$ and a timepoint $t \in \mathbb{R}_{\geq 0}$ (with $t \leq \mu(\rho)$ if ρ is finite). To avoid heavy notation, $|\rho| \neq \omega \Rightarrow t' \leq \mu(\rho)$ is assumed in the semantics of $\phi \mathcal{U}_I \psi$.

$$\begin{aligned}
\rho, t \models_{itw} a & \quad \text{iff} \quad \exists i < |\rho| : \sigma_i = a \text{ and } \tau_i = t \\
\rho, t \models_{itw} \neg \phi & \quad \text{iff} \quad \rho, t \not\models_{itw} \phi \\
\rho, t \models_{itw} \phi \wedge \psi & \quad \text{iff} \quad \rho, t \models_{itw} \phi \text{ and } \rho, t \models_{itw} \psi \\
\rho, t \models_{itw} \phi \mathcal{U}_I \psi & \quad \text{iff} \quad \exists t' > t : t' - t \in I \text{ and } \rho, t' \models_{itw} \psi \text{ and } \forall t < t'' < t' : \rho, t'' \models_{itw} \phi
\end{aligned}$$

Over timed state sequences. Formulae are interpreted over a timed state sequence $\kappa = (\eta, \iota) \in \mathbf{K}$ and a timepoint $t \in \mathbb{R}_{\geq 0}$ (with $t \leq \mu(\kappa)$ if κ is finite). We give the definition for $\phi = a$ only, the remaining cases can be obtained from $\models_{itw}$ by replacing ρ with κ .

$$\kappa, t \models_{its} a \quad \text{iff} \quad a \in \kappa(t)$$

We say that ρ satisfies ϕ , written $\rho \models_{sem} \phi$ iff $\rho, 0 \models_{sem} \phi$ with $sem \in \{pw, itw\}$ and similarly $\kappa \models_{its} \phi$ iff $\kappa, 0 \models_{its} \phi$. Let $sem \in \{pw, itw, its\}$. We write $\llbracket \phi \rrbracket_{sem}^f$ (resp. $\llbracket \phi \rrbracket_{sem}^{inf}$) to denote the set of all finite (resp. infinite) sequences s such that $s \models_{sem} \phi$, and let $\llbracket \phi \rrbracket_{sem} = \llbracket \phi \rrbracket_{sem}^f \cup \llbracket \phi \rrbracket_{sem}^{inf}$. For instance, $\llbracket \phi \rrbracket_{pw} = \{\rho \in \mathbf{P} \mid \rho \models_{pw} \phi\}$. When $sem \in \{pw, itw\}$, each of $\llbracket \phi \rrbracket_{sem}^f$, $\llbracket \phi \rrbracket_{sem}^{inf}$ and $\llbracket \phi \rrbracket_{sem}$ defines a *timed language*.

► **Example 2.15.** Let $\rho_1 = (a, 0)(b, 1)(a, 1)(c, 3.3)$, $\zeta_1 = \Diamond(b \wedge \bigcirc_{[0,0]} a)$ and $\zeta_2 = \Diamond_{(0,1)} \Diamond_{[0,3.5]} c$. For the formula ζ_1 , that reads “eventually b will occur followed immediately by an a ”, we have $\rho_1 \models_{pw} \zeta_1$ but $\rho_1 \not\models_{itw} \zeta_1$ (the closest formula to ζ_1 under $\models_{itw}$ is $\Diamond(a \wedge b)$ where the order is lost, see below). On the other hand, for ζ_2 that reads “from some time within $(0, 1)$, c will occur between 0 and 3.5 time units” we have $\rho_1 \models_{itw} \zeta_2$ but $\rho_1 \not\models_{pw} \zeta_2$.

The above example is an excellent motivator for the remainder of this paper: its counterintuitive outcome (as ρ_1 “should” satisfy both formulae) owes to the fact that the pointwise semantics can reason only at positions of timed events, whereas the interval-based one loses the notion of order over simultaneous events as we will see throughout Section 3.

3 The Pointwise and Interval-Based Semantics are Incomparable

3.1 Main observation

Over finite words. To explain the intuition behind the incomparability of both semantics, an excellent starting point is the existing result **ER** (Section 1), since D’Souza and Prabhakar [14] use a clean version of pointwise and interval-based semantics both interpreted over finite timed words, i.e., the satisfaction relations $\models_{pw}$ and $\models_{itw}$ (Section 2.4) where $|\rho| < \omega$. To arrive at **ER**, the authors use the formula ϕ_{dp} defined below¹ over $\Sigma = \{a, b\}$ and prove formally that there exists no formula ψ such that $\llbracket \psi \rrbracket_{pw}^f = \llbracket \phi_{dp} \rrbracket_{itw}^f$.

$$\phi_{dp} = \neg \Diamond (a \wedge (\neg \varphi_{\Sigma} \mathcal{U} a) \wedge (\neg a \mathcal{U} (\neg a \wedge \Diamond_{[1,1]} (a \vee b))))$$

The language $\llbracket \phi_{dp} \rrbracket_{itw}^f$ contains all finite timed words such that for every two consecutive occurrences of the action a at times t and t' , there is no action that occurs within the interval $[t + 1, t' + 1]$. This is an important expressiveness result, yet we argue that half of the picture is missing: the authors require timed words to be strictly monotone [14][Sect. 1], which impairs the expressive power of the pointwise semantics. To explain this, let us fall back on the general model of finite timed words, i.e., consider the set $\{\rho \in \mathbf{P} \mid |\rho| < \omega\}$. Consider now

¹ We slightly rewrote this formula in terms of the standard (strict) Until used in this paper.

the formula $\zeta_1 = \Diamond(b \wedge \bigcirc_{[0,0]} a)$ of Example 2.15. In the pointwise semantics, this formula captures exactly its informal description (Example 2.15), i.e., $\llbracket \zeta_1 \rrbracket_{pw}^f$ defines the language containing all finite timed words such that there exists a position $i > 0$ with $\tau_i = \tau_{i+1}$, $\sigma_i = b$ and $\sigma_{i+1} = a$. Under $\models_{itw}$, however, the language $\llbracket \zeta_1 \rrbracket_{itw}^f$ is empty, and the closest language to $\llbracket \zeta_1 \rrbracket_{pw}^f$ is $\llbracket \varphi \rrbracket_{itw}^f$ with $\varphi = \Diamond(a \wedge b)$. Yet, $\llbracket \varphi \rrbracket_{itw}^f$ is more permissive than $\llbracket \zeta_1 \rrbracket_{pw}^f$. Let us show this through an example.

► **Example 3.1.** Consider $\rho_1 = (a, 0)(b, 1)(a, 1)(c, 3.3)$ from Example 2.15 and $\rho_2 = (a, 0)(a, 1)(b, 1)(c, 3.3)$. Then we have $\rho_1 \in \llbracket \zeta_1 \rrbracket_{pw}^f$ and $\rho_2 \notin \llbracket \zeta_1 \rrbracket_{pw}^f$, but both ρ_1 and ρ_2 belong to the language $\llbracket \varphi \rrbracket_{itw}^f$.

That is, ζ_1 can distinguish between ρ_1 and ρ_2 under the pointwise semantics, whereas φ cannot under the interval-based one. As we will demonstrate later, there is in fact no MTL formula that distinguishes between ρ_1 and ρ_2 under the interval-based semantics.

Over infinite timed words. The same observation above continues to hold for infinite timed words as illustrated in the following example.

► **Example 3.2.** Consider the two infinite timed words $\rho = (\sigma, \tau)$ and $\rho' = (\sigma', \tau')$ s.t. ${}^3\rho = \rho_1$, ${}^3\rho' = \rho_2$ and $\forall i > 3 : \sigma'_i = \sigma_i = c \wedge \tau'_i = \tau_i = \tau_{i-1} + 1$, with ρ_1 and ρ_2 as defined in Example 3.1. That is, ρ and ρ' have respectively ρ_1 and ρ_2 as a prefix and agree otherwise on an infinite suffix containing only c actions. The above observation carries on to ρ and ρ' as we have $\rho \in \llbracket \zeta_1 \rrbracket_{pw}^{inf}$, $\rho' \notin \llbracket \zeta_1 \rrbracket_{pw}^{inf}$ but no formula can distinguish between ρ and ρ' under $\models_{itw}$ (this will be proven formally later on).

We observe the same weakness of the interval-based semantics over timed state sequences, i.e., when considering the relation $\models_{its}$ instead of $\models_{itw}$. To explain this, we rely on the clever remark in Bouyer et al. [9][Sect. 2], [10][Example 1]:

“an [infinite] timed word can be seen as a special case of timed state sequences, e.g., $\rho = (a, 0)(a, 1.1)(b, 2) \dots$ corresponds to the timed state sequence $\kappa = (\{a\}, [0, 0])(\emptyset, (0, 1.1))(\{a\}, [1.1, 1.1])(\emptyset, (1.1, 2))(\{b\}, [2, 2]) \dots$ ”

Which clearly holds under strict monotonicity. However, under the general model, the clearly different ρ and ρ' of Example 3.2 will correspond to the same timed state sequence starting with $(\{a\}, [0, 0])(\emptyset, (0, 1))(\{a, b\}, [1, 1])(\emptyset, (1, 3.3))(\{c\}, [3.3, 3.3])$. That is, under non-strict monotonicity of time, the encoding from infinite timed words to infinite timed state sequences, suggested in Bouyer et al.’s remark above, cannot correspond to an injective function, and therefore timed words become no longer a special case of timed state sequences. If one tries to have two different images for ρ and ρ' , e.g., by enforcing an order for the counterpart of ρ , they would obtain the $\hat{\kappa} = (\{a\}, [0, 0])(\emptyset, (0, 1))(\{b\}, [1, 1])(\{a\}, [1, 1]) \dots$, where interval adjacency is violated and $\hat{\kappa}(1)$ is no longer defined.

3.2 Approach

3.2.1 Equating $\models_{its}$ and $\models_{itw}$

In order to prove the incomparability between the pointwise and interval-based semantics, we first need to deal with the heterogeneity between timed words and timed state sequences. In other words, there is no way to compare $\models_{pw}$ and $\models_{its}$ if we allow arbitrary timed state sequences, due to the notable difference in nature with timed words, as explained in Section 2.3. For example, the timed state sequence $(\{a, b\}, [0, 1])(\{c\}, [1, 1])$ does not make much sense in the context of action-based timed executions: actions a and b , atomic by

definition, cannot remain true continuously over the dense interval $[0, 1)$. We need therefore to isolate the largest subset of timed state sequences that faithfully represents action-based timed executions modulo the interval-based semantics.

We show that such subset, which we call K_a , can be obtained using natural properties of action-based timed executions. In mathematical terms, K_a is the largest set of images of elements of P by a total relation $\mathcal{F}$ guaranteeing that $\models_{its}$ and $\models_{itw}$ coincide.

Action-based timed state sequences. To construct K_a , we use the fact that actions occur only at punctual intervals. Then, since the first action occurs at 0 and the execution, if finite, terminates with an action, the first interval is $[0, 0]$ and the last interval, if exists, is punctual. Finally, between every two punctual intervals, we have a non-punctual one with no action associated. This gives us an alternation of punctual intervals with sets of actions at even positions and non-punctual ones with empty sets at odd positions.

► **Definition 3.3** (Action-based timed state sequences). *The set of action-based timed state sequences K_a is defined as follows:*

$$\begin{aligned} K_a = \{ \kappa = (\eta, \iota) \in K \mid & \iota_0 = [0, 0] \text{ and} \\ & |\kappa| \in \mathbb{N} \Rightarrow \iota_{|\kappa|-1} \text{ is punctual and} \\ & \forall i \in \mathbb{N}, i < |\kappa|, i \bmod 2 = 0 : \eta_i \neq \emptyset \text{ and } \iota_i \text{ is punctual and} \\ & \forall i \in \mathbb{N}, i < |\kappa|, i \bmod 2 = 1 : \eta_i = \emptyset \text{ and } \iota_i \text{ is not punctual} \} \end{aligned}$$

► **Example 3.4.** $(\{a\}, [0, 0])(\emptyset, (0, 1))(\{a, b\}, [1, 1])$ is an element of K_a , corresponding to an action-based execution where a occurred at time 0, and a and b at time 1. In contrast, $(\{a\}, [0, 0])(\{a, b\}, (0, 1))(\emptyset, [1, 1])$ and $(\emptyset, [0, 0])(\{b\}, (0, 1))(\{a, b\}, [1, 1])$ do not represent any action-based execution and are naturally not elements of K_a .

► **Lemma 3.5.** *The length of any finite element in K_a is an odd number.*

Relating P and K_a . In order to define a relation between P and K_a , and to avoid heavy notations, we use an intermediary representation that allows to group events happening simultaneously while preserving their order. We refer to such representation, which uses the notions of *flat time sequences* (Definition 3.6) and *maximal flat subsequences* (Definition 3.7), as *compact timed words* (Definition 3.8).

► **Definition 3.6** (Flat time sequence). *A finite time sequence τ is said flat iff $\forall i \in \mathbb{N}, 0 < i < |\tau| : \tau_{i-1} = \tau_i$.*

► **Definition 3.7** (Maximal flat subsequence). *Let τ be a time sequence and $i, j \in \mathbb{N}$ such that $i \leq j < |\tau|$. The finite sequence $\tau' = \tau_i \tau_{i+1} \dots \tau_j$ is called a maximal flat subsequence (MFS) of τ iff τ' is a flat sequence and:*

- $i > 0 \Rightarrow \tau_i > \tau_{i-1}$ and
- $j + 1 < |\tau| \Rightarrow \tau_j < \tau_{j+1}$.

Note that when $i = j$ the MFS is *trivial* (contains only one element). It is easy to see that every time sequence τ is a sequence of MFSs. The number of MFSs is finite if τ is finite and infinite otherwise. If τ is infinite, the finiteness of each MFS is guaranteed by time divergence. Examples will follow.

Let $s\tau$ be the sequence over $\mathbb{N}$ defined as follows:

- $s\tau_0 = j$ s.t. $\tau_0 \dots \tau_j$ is an MFS of τ ,
- $s\tau_{i+1} = k$ s.t. $\tau_{s\tau_i+1} \dots \tau_k$ is an MFS of τ .

That is, $s\tau_i$ corresponds to the index of τ delimiting its i^{th} MFS, and $|s\tau|$ corresponds to the number of MFSs in τ . Now we are ready to define the compact version of an arbitrary timed word. In the following, let $\tilde{P}$ denote the set of all pairs $(\tilde{\sigma}, \tilde{\tau})$ where $\tilde{\sigma}$ is a (finite or infinite) sequence of non-empty finite sequences over Σ , $\tilde{\tau}$ a *strictly monotone* time sequence starting at 0 and $|\tilde{\sigma}| = |\tilde{\tau}|$.

► **Definition 3.8** (Compact timed word). *We define the function $\mathcal{C} : P \rightarrow \tilde{P}$ as follows. $\mathcal{C}(\rho = (\sigma, \tau)) = (\tilde{\rho} = (\tilde{\sigma}, \tilde{\tau}))$ iff:*

1. $|\tilde{\rho}| = |s\tau|$,
2. $\forall 0 \leq i < |\tilde{\rho}| : \tilde{\tau}_i = \tau_{s\tau_i}$,
3. $\tilde{\sigma}_0 = \sigma_0 \dots \sigma_{s\tau_0}$ and $\forall 0 < i < |\tilde{\rho}| : \tilde{\sigma}_i = \sigma_{s\tau_{i-1}+1} \dots \sigma_{s\tau_i}$.

That is, $\mathcal{C}$ relates every $\rho = (\sigma, \tau) \in P$ to its compact version, i.e., where every $\tau_i \dots \tau_j$ MFS of τ is reduced into a single element associated with the sequence of actions $\sigma_i \dots \sigma_j$.

► **Example 3.9.** Consider $\rho_1 = (a, 0)(b, 1)(a, 1)(c, 3.3)$ and $\rho_2 = (a, 0)(a, 1)(b, 1)(c, 3.3)$ from Example 3.1. Then we have $\tilde{\rho}_1 = ((a), 0)((b, a), 1)((c), 3.3)$ and $\tilde{\rho}_2 = ((a), 0)((a, b), 1)((c), 3.3)$.

► **Lemma 3.10.** $\mathcal{C}$ is bijective.

Relating $\tilde{P}$ and K_a . We now move to define a relation between compact timed words and action-based timed state sequences, where we use the function $setof()$ that transforms a sequence over Σ to a set preserving its elements.

► **Definition 3.11** (Relation between $\tilde{P}$ and K_a). *Let $\mathcal{R} : \tilde{P} \rightarrow K_a$ be the function defined as follows. $\mathcal{R}(\tilde{\rho} = (\tilde{\sigma}, \tilde{\tau})) = (\kappa = (\eta, \iota))$ iff:*

1. $|\kappa| = |\tilde{\rho}| = \omega$ if $\tilde{\rho}$ is infinite, $|\kappa| = 2|\tilde{\rho}| - 1$ otherwise and
2. $\forall i \in \{1 \dots |\tilde{\rho}| - 1\} : \eta_{2i} = setof(\tilde{\sigma}_i), \iota_{2i} = [\tilde{\tau}_i, \tilde{\tau}_i], \eta_{2i-1} = \emptyset$ and $\iota_{2i-1} = (\tilde{\tau}_{i-1}, \tilde{\tau}_i)$ and
3. $\eta_0 = setof(\tilde{\sigma}_0), \iota_0 = [\tilde{\tau}_0, \tilde{\tau}_0]$.

► **Example 3.12.** Consider $\tilde{\rho}_1 = ((a), 0)((b, a), 1)((c), 3.3)$ and $\tilde{\rho}_2 = ((a), 0)((a, b), 1)((c), 3.3)$ from Example 3.9. We have $\mathcal{R}(\tilde{\rho}_1) = \mathcal{R}(\tilde{\rho}_2) = (\{a\}, [0, 0])(\emptyset, (0, 1))(\{a, b\}, [1, 1])(\emptyset, (1, 3.3))(\{c\}, [3.3, 3.3])$.

In other words, the “holes” between the timestamps in $\tilde{\rho}$ are “filled” by the timed state sequence $\mathcal{R}(\tilde{\rho})$ (i.e., thanks to the non-punctual intervals between the punctual ones).

Notice how $\mathcal{R}$ is *not* injective due to the first component of a timed state sequence being a sequence of sets. We will show that this captures exactly the weakness of the relations $\models_{its}$ and $\models_{itw}$ alike, hinted at in Section 3.1.

► **Lemma 3.13.** $\mathcal{R}$ is surjective.

► **Lemma 3.14.** $\mathcal{R}$ is not injective.

Proof. The proof is straightforward through the counterexample given in Example 3.12. ◀

In the following, we let $\mathcal{F} = \mathcal{R} \circ \mathcal{C}$.

► **Lemma 3.15.** $\mathcal{F}$ is total.

Proof. $\mathcal{F}$ is a surjective function since (i) $\mathcal{C}$ is a function and (ii) $\mathcal{R}$ is a surjective function (Lemma 3.13). Therefore $\mathcal{F}$ is right and left total and consequently total. ◀

$\models_{itw}$ and $\models_{its}$ are equivalent. We are now ready to equate $\models_{its}$ and $\models_{itw}$. The following proposition is central as it shows that $\models_{itw}$ and $\models_{its}$ enjoy exactly the same expressive power over *all* timed words and *all* action-based timed state sequences. Note that we do not need to prove the equivalence for the version with every $\kappa \in K_a$ and elements of $\mathcal{F}^{-1}[\kappa]$ since $\mathcal{F}$ is total (Lemma 3.15).

► **Proposition 3.16.** *For every $\phi \in MTL$, $\rho \in P$ and $t \in \mathbb{R}_{\geq 0}$ ($t \leq \mu(\rho)$ if $|\rho| \in \mathbb{N}$):*

$$\rho, t \models_{itw} \phi \Leftrightarrow \mathcal{F}(\rho), t \models_{its} \phi$$

The main takeaway is summed by the following theorem, a direct consequence of Proposition 3.16 and Lemma 3.15, stating that $\mathcal{F}$ preserves the full expressiveness of $\models_{its}$ and $\models_{itw}$ over all timed words and all action-based timed state sequences.

► **Theorem 3.17.** *For every $\phi \in MTL$:*

- $\llbracket \phi \rrbracket_{its}^{fi} \cap K_a = \{\mathcal{F}(\rho) \mid \rho \in \llbracket \phi \rrbracket_{itw}^{fi}\}$ and $\llbracket \phi \rrbracket_{itw}^{fi} = \bigcup \{\mathcal{F}^{-1}[\kappa] \mid \kappa \in \llbracket \phi \rrbracket_{its}^{fi} \cap K_a\}$ with $fi \in \{f, inf\}$,
- $\llbracket \phi \rrbracket_{its} \cap K_a = \{\mathcal{F}(\rho) \mid \rho \in \llbracket \phi \rrbracket_{itw}\}$ and $\llbracket \phi \rrbracket_{itw} = \bigcup \{\mathcal{F}^{-1}[\kappa] \mid \kappa \in \llbracket \phi \rrbracket_{its} \cap K_a\}$.

3.2.2 Proving Incomparability

Since $\models_{its}$ and $\models_{itw}$ are proven equivalent, we can devise our incomparability proofs for $\models_{pw}$ vs. $\models_{itw}$ only, while exploiting the equalities of Theorem 3.17.

We first formalise the notion of incomparability. Let $\llbracket MTL \rrbracket_{sem}^f$ denote the set of all languages over finite timed words (i.e., a set of sets) defined by MTL formulae under the semantics corresponding to $sem \in \{pw, itw\}$: $\llbracket MTL \rrbracket_{sem}^f = \{\llbracket \phi \rrbracket_{sem}^f \mid \phi \in MTL\}$. Similarly, $\llbracket MTL \rrbracket_{sem}^{inf} = \{\llbracket \phi \rrbracket_{sem}^{inf} \mid \phi \in MTL\}$ and $\llbracket MTL \rrbracket_{sem} = \llbracket MTL \rrbracket_{sem}^f \cup \llbracket MTL \rrbracket_{sem}^{inf}$. Let $\#$ be the binary operator defined over sets as: $S_1 \# S_2 \equiv S_1 \not\subseteq S_2 \wedge S_2 \not\subseteq S_1$, i.e., none of its arguments is included in the other. We can now formally define incomparability of $\models_{pw}$ and $\models_{itw}$ over MTL.

► **Definition 3.18** (Incomparability). *The MTL semantics defined by $\models_{pw}$ and $\models_{itw}$ are incomparable over:*

- Finite timed words iff $\llbracket MTL \rrbracket_{pw}^f \# \llbracket MTL \rrbracket_{itw}^f$,
- Infinite timed words iff $\llbracket MTL \rrbracket_{pw}^{inf} \# \llbracket MTL \rrbracket_{itw}^{inf}$.

The semantics defined by $\models_{pw}$ and $\models_{itw}$ are incomparable iff they are incomparable over finite and infinite timed words, i.e., $\llbracket MTL \rrbracket_{pw} \# \llbracket MTL \rrbracket_{itw}$.

In order to show that e.g., $\llbracket MTL \rrbracket_{pw}^f \# \llbracket MTL \rrbracket_{itw}^f$, it suffices to find two formulae $\phi_1, \phi_2 \in MTL$ such that for all $\psi \in MTL$, the language of finite timed words defined by ϕ_1 (resp. ϕ_2) under $\models_{pw}$ (resp. $\models_{itw}$) is different from the language defined by ψ under $\models_{itw}$ (resp. $\models_{pw}$) as generalised by the following proposition.

► **Proposition 3.19.** $\llbracket MTL \rrbracket_{pw}^f \# \llbracket MTL \rrbracket_{itw}^f$ iff $\exists \phi_1, \phi_2 \in MTL \forall \psi \in MTL$:

- $\llbracket \phi_1 \rrbracket_{pw}^f \neq \llbracket \psi \rrbracket_{itw}^f$ and
- $\llbracket \phi_2 \rrbracket_{itw}^f \neq \llbracket \psi \rrbracket_{pw}^f$.

(The equivalence with $\llbracket MTL \rrbracket_{pw}^{inf} \# \llbracket MTL \rrbracket_{itw}^{inf}$ is defined analogously).

Proving $\llbracket MTL \rrbracket_{pw}^f \# \llbracket MTL \rrbracket_{itw}^f$. We start with proving $\llbracket MTL \rrbracket_{pw}^f \not\subseteq \llbracket MTL \rrbracket_{itw}^f$. It suffices to find two distinct finite timed words that have the same image by $\mathcal{F}$ and a formula ϕ that distinguishes between them under $\models_{pw}$. By definition, no formula under $\models_{its}$ can distinguish between their images through $\mathcal{F}$ (since it is the same image) and we conclude that no formula can distinguish between ρ_1 and ρ_2 under $\models_{itw}$ since $\models_{its}$ and $\models_{itw}$ are equally expressive.

► **Lemma 3.20.** *Let $\zeta_1 = \Diamond(b \wedge \bigcirc_{[0,0]} a)$ from Example 2.15. The following statement holds:*

$$\forall \psi \in MTL : \llbracket \zeta_1 \rrbracket_{pw}^f \neq \llbracket \psi \rrbracket_{itw}^f$$

Proof. Let $\rho_1 = (a, 0)(b, 1)(a, 1)(c, 3.3)$ and $\rho_2 = (a, 0)(a, 1)(b, 1)(c, 3.3)$ from Example 3.1. We have $\rho_1 \in \llbracket \zeta_1 \rrbracket_{pw}^f$ and $\rho_2 \notin \llbracket \zeta_1 \rrbracket_{pw}^f$. Now, ρ_1 and ρ_2 have the same image by $\mathcal{F}$ that we will refer to as κ . By definition, for every $\psi \in MTL$, we have either $\kappa \in \llbracket \psi \rrbracket_{its}^f$, and by Theorem 3.17, $\rho_1 \in \llbracket \psi \rrbracket_{itw}^f$ and $\rho_2 \in \llbracket \psi \rrbracket_{itw}^f$, or $\kappa \notin \llbracket \psi \rrbracket_{its}^f$, and again by Theorem 3.17 $\rho_1 \notin \llbracket \psi \rrbracket_{itw}^f$ and $\rho_2 \notin \llbracket \psi \rrbracket_{itw}^f$. And therefore $\llbracket \zeta_1 \rrbracket_{pw}^f \neq \llbracket \psi \rrbracket_{itw}^f$. ◀

► **Remark 3.21.** Note that this weakness of the interval-based semantics has a direct consequence on the coherence between MTL and its untimed version, LTL. Consider e.g., ρ_1 and ρ_2 above. Their untimed counterparts are, respectively, the words *abac* and *aabc* which are temporally different (in terms of LTL). Yet, under $\models_{itw}$ or $\models_{its}$ alike, it is impossible to distinguish between them whenever the two letters in the middle occur simultaneously.

For the part $\llbracket MTL \rrbracket_{itw}^f \not\subseteq \llbracket MTL \rrbracket_{pw}^f$, we depart from

$$\phi_{dp} = \neg \Diamond(a \wedge (\neg(\varphi_\Sigma \mathcal{U} a) \wedge (\neg a \mathcal{U} (\neg a \wedge \Diamond_{[1,1]}(a \vee b))))$$

The formula proven to have no equivalent under $\models_{pw}$ over strictly monotone finite timed words [14] (Section 3.1). The idea here is rather simple: find a formula γ_1 that captures exactly the language of ϕ_{dp} under the general model of finite timed words then conclude.

► **Lemma 3.22.** *Let $\gamma_1 = \phi_{dp} \wedge \phi_s$ with² $\phi_s = (\neg \bigvee_{a \neq b \in \Sigma} (a \wedge b)) \wedge \Box (\neg \bigvee_{a \neq b \in \Sigma} (a \wedge b))$. The following statement holds: $\forall \psi \in MTL : \llbracket \gamma_1 \rrbracket_{itw}^f \neq \llbracket \psi \rrbracket_{pw}^f$*

Proof. The language $\llbracket \gamma_1 \rrbracket_{itw}^f$ contains the set of strictly monotone (satisfying ϕ_s) finite timed words satisfying ϕ_{dp} . The proof then follows directly from [14][Sect. 3]. ◀

► **Proposition 3.23.** *The pointwise and the interval based semantics are incomparable over finite timed words.*

Proof. From Lemma 3.20, Lemma 3.22 and Proposition 3.19 we get $\llbracket MTL \rrbracket_{pw}^f \# \llbracket MTL \rrbracket_{itw}^f$ and conclude by Definition 3.18. ◀

Proving $\llbracket MTL \rrbracket_{pw}^{inf} \# \llbracket MTL \rrbracket_{itw}^{inf}$. We first prove $\llbracket MTL \rrbracket_{pw}^f \not\subseteq \llbracket MTL \rrbracket_{itw}^f$, using the same formula from Lemma 3.20.

► **Lemma 3.24.** *Let $\zeta_1 = \Diamond(b \wedge \bigcirc_{[0,0]} a)$. The following statement holds:*

$$\forall \psi \in MTL : \llbracket \zeta_1 \rrbracket_{pw}^{inf} \neq \llbracket \psi \rrbracket_{itw}^{inf}$$

Proof. The proof is very similar to that of Lemma 3.20, with $\rho \in \llbracket \zeta_1 \rrbracket_{pw}^{inf}$ and $\rho' \notin \llbracket \zeta_1 \rrbracket_{pw}^{inf}$ (Example 3.2) having the same image by $\mathcal{F}$. ◀

² The first clause of ϕ_s is necessary due to the strict nature of the Until operator, from which $\Box$ is derived.

For $\llbracket MTL \rrbracket_{itw}^{inf} \not\subseteq \llbracket MTL \rrbracket_{pw}^{inf}$, we use a formula already proven in [9] [Prop. 1 and Sect. 3.2] to have a language over infinite words that is not expressible under $\models_{pw}$. This language is informally defined as “all infinite timed words containing an occurrence of b within the absolute interval $(0, 2)$ and an occurrence of c within the absolute interval $(1, 2]$, such that the latter happens *after* the former within the relative interval $(0, 1)$ ”.

► **Lemma 3.25.** *Let $\gamma_2 = \Diamond_{[0,1]} (\Diamond_{[0,1]} b \wedge \Diamond_{[1,1]} c)$. The following statement holds:*

$$\forall \psi \in MTL : \llbracket \gamma_2 \rrbracket_{itw}^{inf} \neq \llbracket \psi \rrbracket_{pw}^{inf}$$

► **Proposition 3.26.** *The pointwise and the interval-based semantics are incomparable over infinite timed words.*

Proof. From Lemma 3.24, Lemma 3.25 and Proposition 3.19 we get $\llbracket MTL \rrbracket_{pw}^{inf} \# \llbracket MTL \rrbracket_{itw}^{inf}$ and conclude by Definition 3.18. ◀

And finally we conclude with the following theorem, a direct consequence of Proposition 3.23 and Proposition 3.26.

► **Theorem 3.27.** *The pointwise and the interval-based semantics of MTL are incomparable.*

Incomparability under stutter freeness. Our incomparability results continue to hold under the reasonable restriction of stutter freeness (Definition 2.9, Item 1). For the finite case, the formula ζ_1 given in Lemma 3.20, as shown in the proof of the same lemma, distinguishes between two stutter-free finite timed words under $\models_{pw}$ whereas no formula under $\models_{itw}$ can do so (and the same goes for the infinite case, Lemma 3.24 and its proof). The only technical difference when disallowing stuttering is the fact that a finite timed state sequence, element of K_a , will then have an image of finite cardinality by the inverse relation $\mathcal{F}^{-1}$, which has no effect on the incomparability theorem.

4 Mixed Semantics

In this section, we present a new *mixed* semantics that embeds both the pointwise and the interval-based ones. We do so in three steps. First, we present an intuitive version of the mixed semantics over MTL that, leveraging the expressive power of the interval-based semantics, is strictly more expressive than the pointwise one (Section 4.1). Second, in Section 4.2, we conjecture that such version is incomparable with the interval-based one (due mainly to the fact that the latter is more suited for signals). Third, we show that extending the syntax of MTL with a single atomic formula unleashes the power of the mixed semantics leading to the embedding of both the pointwise and the interval-based semantics, conjugating therefore their full expressive powers (Section 4.3). The embedding is automated through syntax-driven compilers.

4.1 Dual reasoning on time and positions

In this subsection, we undertake the first step described above, i.e., we propose to increase the power of the pointwise semantics using the flexibility of the interval-based one. The reason we do it in this direction (rather than the other way around) is the fact that the interval-based semantics is more suitable for signals (Section 2.3) whereas we focus on action-based timed executions (another example on the counter-intuitiveness of the interval-based semantics over timed words will be given in Section 4.2). We still provide in Section 4.3 an extended version of the mixed semantics that embeds both the pointwise and the interval-based ones.

Given a compact timed word $\tilde{\rho} = (\tilde{\sigma}, \tilde{\tau})$, we define its duration in the obvious way, i.e., $\mu(\tilde{\rho}) = \mu(C^{-1}(\tilde{\rho}))$. For any $t \in \mathbb{R}_{\geq 0}$ (with $t \leq \mu(\tilde{\rho})$ if $\tilde{\rho}$ is finite), we let $\tilde{\rho}(t)$ equal $\tilde{\sigma}_i$ where $\tilde{\tau}_i = t$ (such $\tilde{\tau}_i$ is uniquely defined if it exists) and the singleton sequence $(\vdash)$ otherwise, where $\vdash$ is a “helper” symbol that represents “no action”, i.e., it will be semantically nothing else than equivalent to $\neg\varphi_\Sigma$.

The idea is to evaluate an MTL formula over a compact timed word $\tilde{\rho} = (\tilde{\sigma}, \tilde{\tau})$, a time point $t \in \mathbb{R}_{\geq 0}$ (with $t \leq \mu(\tilde{\rho})$ if $|\tilde{\rho}| < \omega$) and a position $0 \leq j < |\tilde{\rho}(t)|$. It is important to remark here that the condition $0 \leq j < |\tilde{\rho}(t)|$ is by no means impractical for verification purposes, as the satisfaction of some formula ϕ by some $\tilde{\rho}$ boils down to the satisfaction of ϕ by $\tilde{\rho}$, $t_0 = 0$ and $j_0 = 0$ (examples will follow).

Now we are ready to formalise the mixed semantics:

$$\begin{aligned} \tilde{\rho}, t, j \models_{mx} a & \quad \text{iff} \quad a = \tilde{\rho}(t)(j) \\ \tilde{\rho}, t, j \models_{mx} \neg\phi & \quad \text{iff} \quad \tilde{\rho}, t, j \not\models_{mx} \phi \\ \tilde{\rho}, t, j \models_{mx} \phi \wedge \psi & \quad \text{iff} \quad \tilde{\rho}, t, j \models_{mx} \phi \text{ and } \tilde{\rho}, t, j \models_{mx} \psi \\ \tilde{\rho}, t, j \models_{mx} \phi \mathcal{U}_I \psi & \quad \text{iff} \quad \exists (t', j') < (t', j') : t' - t \in I \text{ and } \tilde{\rho}, t', j' \models_{mx} \psi \text{ and} \\ & \quad \forall (t, j) < (t'', j'') < (t', j') : \tilde{\rho}, t'', j'' \models_{mx} \phi \end{aligned}$$

Where the relation $<$, used for the $\mathcal{U}_I$ operator, denotes a lexicographic order³ defined in the usual way. We say $\tilde{\rho}$ satisfies ϕ , written $\tilde{\rho} \models_{mx} \phi$, iff $\tilde{\rho}, 0, 0 \models_{mx} \phi$.

Let us explain, relying on the rules of the semantics above, how the relation $\models_{mx}$ increases the expressive power of $\models_{pw}$ (we will formally prove this soon). The real t enables the evaluation at any arbitrary time point, whereas the natural j allows to “access” the positions of ordered events occurring at t , if any. The semantics of $\mathcal{U}_I$ reflects this increased power of expressiveness: $\phi \mathcal{U}_I \psi$ is evaluated at any timepoint and any position of the sequence of events occurring at t . Such position is guaranteed to exist: in particular, if t belongs to a “hole” in $\tilde{\rho}$ (i.e., strictly comprised between two timestamps), then $\tilde{\rho}(t)(0)$ exists and is equal to the $\vdash$ symbol which plays otherwise a transparent role as, in such case, $\tilde{\rho}, t, 0 \models \neg\varphi_\Sigma$ and $\tilde{\rho}, t, 0 \models \top$ as expected. The following example illustrates the intuitive nature of the mixed semantics.

► **Example 4.1.** Consider the setting of Example 2.15 with $\rho_1 = (a, 0)(b, 1)(a, 1)(c, 3.3)$, $\zeta_1 = \Diamond(b \wedge \bigcirc_{[0,0]} a)$ and $\zeta_2 = \Diamond_{(0,1)} \Diamond_{[0,3.5]} c$. As explained within the same example, ρ_1 should, intuitively, satisfy both formulae, yet we have $\rho_1 \not\models_{itw} \zeta_1$ and $\rho_1 \not\models_{pw} \zeta_2$.

Under the mixed semantics, both formulae are satisfied thanks to the flexibility of $\models_{mx}$, taking both time points and positions into account. To illustrate this, let us first encode in the obvious way $\rho_1 = (a, 0)(b, 1)(a, 1)(c, 3.3)$ into its compact counterpart $\tilde{\rho}_1 = ((a), 0)((b, a), 1)((c), 3.3)$.

Then we have $\tilde{\rho}_1 \models_{mx} \zeta_1$ since $\tilde{\rho}_1, 0, 0 \models_{mx} \zeta_1$: we have $\tilde{\rho}_1, t', j'$ with $(t' = 1, j' = 0)$, greater than $(0, 0)$ in lexicographic order, satisfies b (since $\tilde{\rho}_1(1)(0) = b$) and in turn, $\tilde{\rho}_1, t'', j''$ with $(t'' = 1, j'' = 1)$, greater than $(1, 0)$ satisfies a within $[0, 0]$ and (i) for every $(0, 0) < (t_1, j_1) < (t', j')$, $\tilde{\rho}_1, t_1, j_1 \models_{mx} \top$ and (ii) for every $(t', j') < (t_2, j_2) < (t'', j'')$, $\tilde{\rho}_1, t_2, j_2 \models_{mx} \neg\varphi_\Sigma$ (vacuous truth).

Dually, $\tilde{\rho}_1, 0, 0 \models_{mx} \zeta_2$ because for any couple $(t', j' = 0)$ with $t' \in (0, 1)$, we have $\tilde{\rho}_1, t', j' \models_{mx} \Diamond_{[0,3.5]} c$ since at $(t'' = 3.3, j'' = 0)$ we have $\tilde{\rho}_1, t'', j'' \models_{mx} c$ (since $\tilde{\rho}_1(t'')(j'') = c$) and $t'' - t' \in [0, 3.5]$.

³ Compact timed words are similar to *weakly-monotonic observations* [3] over which the authors also mention lexicographic order. While we were unaware of these similarities when realising this work, all our contributions summarised in Section 1 and developed in Section 3 and Section 4 are novel.

4.1.1 $\models_{mx}$ is strictly more expressive than $\models_{pw}$

The mixed semantics is strictly more expressive than the pointwise one. To show this, we proceed as follows. First, we devise a compiler $(\cdot)_{pw}$ over MTL such that $(\phi)_{pw}$ under $\models_{mx}$ is equivalent to ϕ under $\models_{pw}$ (using the bijective function $\mathcal{C}$ to move between timed words and their compact version). Second, we give an example of a formula not expressible in the pointwise semantics (from Section 3.2) and show that it is expressible in the mixed semantics.

From $\models_{pw}$ to $\models_{mx}$. We reason following the main observation that, in $\models_{pw}$, formulae are evaluated only at timepoints where actions occur. Thus, $(\phi)_{pw}$ must satisfy φ_Σ , regardless of its argument. The case for $\mathcal{U}_I$ is the involved one. Given a formula $\phi \mathcal{U}_I \psi$ interpreted under $\models_{pw}$, to get its equivalent formula under $\models_{mx}$, we need also to ensure that (i) ψ holds at a timepoint corresponding to an action occurrence and (ii) ϕ does not need to hold at timepoints where no action occurs, as formalised by $(\phi \mathcal{U}_I \psi)_{pw}$ in the following definition.

► **Definition 4.2** ($\models_{pw}$ -preserving compilation). *The compiler $(\cdot)_{pw}$ is defined inductively:*

- $(a)_{pw} = a$,
- $(\neg\phi)_{pw} = \varphi_\Sigma \wedge \neg(\phi)_{pw}$,
- $(\phi \wedge \psi)_{pw} = \varphi_\Sigma \wedge (\phi)_{pw} \wedge (\psi)_{pw}$,
- $(\phi \mathcal{U}_I \psi)_{pw} = \varphi_\Sigma \wedge ((\phi)_{pw} \vee \neg\varphi_\Sigma) \mathcal{U}_I ((\psi)_{pw} \wedge \varphi_\Sigma)$.

► **Proposition 4.3.** *For every $\phi \in \text{MTL}$ and $\rho \in \mathbf{P} : \rho \models_{pw} \phi \Leftrightarrow \mathcal{C}(\rho) \models_{mx} (\phi)_{pw}$.*

A language definable under $\models_{mx}$ but not under $\models_{pw}$. Consider the formula $\gamma_2 = \Diamond_{[0,1]} (\Diamond_{[0,1]} b \wedge \Diamond_{[1,1]} c)$ from Lemma 3.25. As states the same lemma, γ_2 has no equivalent under the pointwise semantics over infinite timed words. The following lemma states that γ_2 does have an equivalent under $\models_{mx}$ over infinite timed words.

► **Lemma 4.4.** *Let $\gamma_2 = \Diamond_{[0,1]} (\Diamond_{[0,1]} b \wedge \Diamond_{[1,1]} c)$ and $\gamma_3 = \Diamond_{(0,1]} (\Diamond_{[0,1]} b \wedge \Diamond_{[1,1]} c)$. Then:*

$$\llbracket \gamma_2 \rrbracket_{itw}^{inf} = \{\mathcal{C}^{-1}(\tilde{\rho}) \mid \tilde{\rho} \in \llbracket \gamma_3 \rrbracket_{mx}^{inf}\}$$

► **Remark 4.5.** Note the subtlety in γ_3 where the interval of the outer eventually operator is left-open at 0. Under $\models_{itw}$, γ_2 and γ_3 are equivalent because the time point satisfying the formula must be greater than 0 by the definition of the (strict) Until operator from which $\Diamond_{[0,1]}$ is derived. Under $\models_{mx}$, on the other hand, we have $\llbracket \gamma_3 \rrbracket_{mx}^{inf} \subsetneq \llbracket \gamma_2 \rrbracket_{mx}^{inf}$ since e.g., $(t=0, j'=2)$ is greater than $(t=0, j=0)$ (but t is not greater than 0); we have therefore the compact version of any infinite timed word starting with $(a,0)(c,0)(b,0)(c,1)(c,3)$ in $\llbracket \gamma_2 \rrbracket_{mx}^{inf}$ but not in $\llbracket \gamma_3 \rrbracket_{mx}^{inf}$. Hence the explicit exclusion of 0 from the interval.

► **Theorem 4.6.** *The mixed semantics is more expressive than the pointwise one.*

Proof. Direct from Lemma 4.4 and Proposition 4.3. ◀

4.2 Mixed semantics vs. interval-based semantics

We now explain the fundamental difference between $\models_{itw}$ and our $\models_{mx}$ and why we argue that $\models_{mx}$ is more suitable for action-based executions. Let $\phi = (a \vee \neg\varphi_\Sigma) \mathcal{U}_{[1,2]} b$ and $\rho = (a,0)(a,0.5)(c,0.5)(c,1.5)(b,1.5)$. The (action-based) intuition would suggest that ρ should not satisfy ϕ , two occurrences of c at different dates take place (which correspond to neither an a nor “no action”) before b does, and indeed $\tilde{\rho}$ does not satisfy ϕ under the mixed

semantics. In contrast, we have $\rho \models_{itw} \phi$, since the interval-based semantics is more suitable for signals. If ρ was a signal, it would naturally make sense for it to satisfy ϕ since at every time point, either the proposition a held or no proposition did until b within $[1, 2]$.

Because of this “permissive” nature of the $\mathcal{U}_I$ operator at its first argument under $\models_{itw}$ (and its more “restrictive” nature at its second argument, see the second example below), we conjecture that the satisfaction relations $\models_{mx}$ and $\models_{itw}$ are incomparable over timed words. The proof of this conjecture is already halfway since we know that the $\models_{mx}$ is strictly more expressive than $\models_{pw}$ (Theorem 4.6), and given that $\models_{itw}$ and $\models_{pw}$ are incomparable (Theorem 3.27), $\models_{mx}$ is either more expressive than or incomparable with $\models_{itw}$. Actually, we can relatively easily construct an MTL formula defining a language under $\models_{mx}$ (after un-compacting with C^{-1}) that cannot be characterised by any MTL formula under $\models_{itw}$ or $\models_{pw}$ alike, as states the following lemma.

► **Lemma 4.7.** *Let $\gamma_3 = \Diamond_{(0,1]} (\Diamond_{[0,1]} b \wedge \Diamond_{[1,1]} c)$ from Lemma 4.4, $\zeta_1 = \Diamond (b \wedge \bigcirc_{[0,0]} a)$ from Lemma 3.20, $\gamma_4 = \Box_{(0,2]} \neg \varphi_\Sigma$ and $\gamma_5 = \gamma_3 \vee (\zeta_1 \wedge \gamma_4)$. Then for all $\phi \in MTL$:*

- $\llbracket \phi \rrbracket_{pw}^{inf} \neq \{C^{-1}(\tilde{\rho}) \mid \tilde{\rho} \in \llbracket \gamma_5 \rrbracket_{mx}^{inf}\}$
- $\llbracket \phi \rrbracket_{itw}^{inf} \neq \{C^{-1}(\rho) \mid \tilde{\rho} \in \llbracket \gamma_5 \rrbracket_{mx}^{inf}\}$

To have a less biased view, let us now look at a trickier example that rather pleads in favour of $\models_{itw}$. Let $\rho = (c, 0)(c, 0.5)(c, 1.5)(b, 1.5)$ and $\phi = (c \vee \neg \varphi_\Sigma) \mathcal{U}_{[1,2]} (b \wedge \neg c)$. Here, we have $\tilde{\rho} \models_{mx} \phi$ but $\rho \not\models_{itw} \phi$. This is because $b \wedge \neg c$ is trivially satisfied at $(t, j) = (1.5, 1)$ under $\models_{mx}$, whereas it is not at $t = 1.5$ under $\models_{itw}$. It is however legitimate for a user to mean by this formula “verify that only occurrences of c are seen until an occurrence of b takes place between times 1 and 2 and c does not occur simultaneously with such b ”. The mixed semantics fails to capture this, and at first glance the naive way to remedy this issue would be to use an immediate past operator that allows to look *behind* $(t, j) = (1.5, 1)$ while maintaining the same timepoint, i.e., at $(1.5, 0)$. In the next subsection, we show that this “immediate past” is an overkill as we can instead use a single atomic formula to extend the syntax of MTL in order to strictly embed the full expressive power of $\models_{itw}$ into $\models_{mx}$.

4.3 Extending mixed semantics

Following the examples given in Section 4.2, and in order to embed $\models_{itw}$, the satisfaction relation $\models_{mx}$ needs to be able to force the evaluation of a formula at index 0 within any $\tilde{\rho}(t)$, looking forward at the whole sequence returned by $\tilde{\rho}(t)$ (instead of complexifying the syntax of MTL, and $\models_{mx}$ thereof, with an immediate past operator). Let us consider the example $(a \vee \neg \varphi_\Sigma) \mathcal{U}_I b$ satisfied by some ρ, t under $\models_{itw}$, and let t correspond to no timestamp of ρ to simplify the explanation. To satisfy the same formula at $\tilde{\rho}, t, 0$ under $\models_{mx}$, intuitively we need to check for $(a \vee \Diamond_{[0,0]} a)$ only at position 0 for every $\tilde{\rho}(t') \neq (\vdash), t' > t$ until $(b \vee \Diamond_{[0,0]} b)$ becomes true at position 0 of some $\tilde{\rho}(t'')$ satisfying $t'' - t \in I$.

To do this in a simple and efficient way, the idea is to interpret the mixed semantics over $\Sigma \cup \{\beta\}$, where β is an atomic proposition not in Σ . The syntax of MTL is accordingly extended with the atomic formula β , we call such extended syntax MTL_β . The semantics of β is defined simply to hold only at the beginning of a sequence $\tilde{\rho}(t)$:

$$\tilde{\rho}, t, j \models_{mx} \beta \text{ iff } j = 0$$

This seemingly minuscule addition comes with a major expressiveness gain. The mixed semantics now embeds the interval-based one using the compiler $(\llbracket - \rrbracket)_{itw} : MTL \rightarrow MTL_\beta$ defined inductively as follows:

- $\langle a \rangle_{itw} = a \vee \Diamond_{[0,0]} a,$
- $\langle \neg \phi \rangle_{itw} = \neg \langle \phi \rangle_{itw},$
- $\langle \phi \wedge \psi \rangle_{itw} = \langle \phi \rangle_{itw} \wedge \langle \psi \rangle_{itw},$
- $\langle \phi \mathcal{U}_I \psi \rangle_{itw} = (\beta \Rightarrow \langle \phi \rangle_{itw}) \mathcal{U}_I (\beta \wedge \langle \psi \rangle_{itw}).$

The following proposition states that $\models_{mx}$ over MTL_β embeds $\models_{itw}$ over MTL.

► **Proposition 4.8.** *For all ϕ in MTL and $\rho \in P$: $\rho \models_{itw} \phi \Leftrightarrow C(\rho) \models_{mx} \langle \phi \rangle_{itw}$.*

Strict expressiveness superiority follows directly from Proposition 4.8 and Theorem 4.6.

► **Remark 4.9.** The embedding of $\models_{itw}$ into $\models_{mx}$ (Proposition 4.8) looks asymmetrical since we allow an extended syntax for $\models_{mx}$ but not for $\models_{itw}$. However, adding a straightforward interpretation of β under $\models_{itw}$, i.e., $\rho, t \models_{itw} \beta$ iff $t = 0$ is effectless: since we have $\rho \models_{itw} \phi$ iff $\rho, 0 \models_{itw} \phi$, we get $\rho \models_{itw} \beta$ iff $\rho, 0 \models_{itw} \beta$, and therefore $\llbracket \beta \rrbracket_{itw} = P = \llbracket \top \rrbracket_{itw}$. In other words, the expressive power of $\models_{itw}$ over MTL and MTL_β would be identical.

We conclude with showing that the set of languages characterised by $\models_{mx}$ over MTL_β is strictly larger than that characterised by the union of the two sets of languages characterised by $\models_{itw}$ and $\models_{pw}$ over MTL.

► **Theorem 4.10.**

$$\llbracket MTL \rrbracket_{pw} \cup \llbracket MTL \rrbracket_{itw} \subsetneq \{ \{ C^{-1}(\tilde{\rho}) \mid \tilde{\rho} \in \llbracket \phi \rrbracket_{mx} \} \mid \phi \in MTL_\beta \}$$

Proof. Direct from Proposition 4.3, Proposition 4.8 and Lemma 4.7. ◀

Note that the witness formula for the strict inclusion in Theorem 4.10 is γ_5 in Lemma 4.7, which does not use the atomic formula β . This is an interesting observation: $\models_{mx}$ over MTL already characterises at least one language that can be expressed neither in $\models_{itw}$ nor in $\models_{pw}$.

5 Conclusion

In this paper, we revisited an important aspect of the extensively studied expressiveness of MTL, namely that of its pointwise semantics as opposed to the interval-based one. We have demonstrated that the two semantics are incomparable over the general models of action-based timed executions, giving thereby a clear answer, in the negative, to the question on whether reasoning over arbitrary time points *instead of* ordered positions of timed events increases the expressive power of MTL. To get the better of both worlds over action-based executions, we proposed a new mixed semantics that embeds the interval-based and the pointwise ones, therefore conjugating their full expressive powers. For future work, the results presented within this paper call for further research on at least two fronts.

On the complexity front, an interesting future direction would be to investigate that of monitoring under the mixed semantics, already shown to be tractable for the pointwise semantics e.g., in [21]. In particular, offline monitoring is a promising venue to reason on executions of distributed systems where the order over simultaneous events is a key aspect [24, 15, 23]. On the expressiveness front, it would be interesting to consider extending our mixed semantics to reason on mixed signal-event executions while maintaining reasonable complexity. This would require to define a mixed structure, e.g., $((a, b), [0, 0])(\{p, q\}, (0, 1))(c, [1, 1])$ where a, b and c are in the actions set and p, q in the atomic propositions one, then further extend our mixed semantics to both sets. In the same vein, we aim to investigate relating the mixed semantics to the different semantics of more expressive logics such as TPTL [4]

and T^{lin} [6]. Finally, our incomparability results are likely to hold when restricting MTL to its decidable fragment MITL [1], where only non-punctual intervals are allowed in formulae. Surprisingly, the non-containments $\llbracket MITL \rrbracket_{pw}^f \not\subseteq \llbracket MITL \rrbracket_{itw}^f$ and $\llbracket MITL \rrbracket_{pw}^{inf} \not\subseteq \llbracket MITL \rrbracket_{itw}^{inf}$ are easy to prove: Lemma 3.20 and Lemma 3.24 still hold if we replace the interval $[0, 0]$ in ζ_1 by any other interval $I \in \mathcal{I}_q$ such that I is left-closed at 0. We leave it open whether $\llbracket MITL \rrbracket_{itw}^f \not\subseteq \llbracket MITL \rrbracket_{pw}^f$ and $\llbracket MITL \rrbracket_{itw}^{inf} \not\subseteq \llbracket MITL \rrbracket_{pw}^{inf}$.

References

- 1 Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996. doi:10.1145/227595.227602.
- 2 Rajeev Alur, Limor Fix, and Thomas A Henzinger. A determinizable class of timed automata. In *International conference on computer aided verification (CAV)*, pages 1–13. Springer, 1994. doi:10.1007/3-540-58179-0_39.
- 3 Rajeev Alur and Thomas A Henzinger. Logics and models of real time: A survey. In *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*, pages 74–106. Springer, 1991. doi:10.1007/BFB0031988.
- 4 Rajeev Alur and Thomas A Henzinger. A really temporal logic. *Journal of the ACM (JACM)*, 41(1):181–203, 1994. doi:10.1145/174644.174651.
- 5 Rajeev Alur and Thomas A Henzinger. Modularity for timed and hybrid systems. In *International Conference on Concurrency Theory (CONCUR)*, pages 74–88. Springer, 1997. doi:10.1007/3-540-63141-0_6.
- 6 Mouloud Amara, Giovanni Bernardi, Mohammed Aristide Foughali, and Adrian Francalanza. A theory of (linear-time) timed monitors. In *European Conference on Object-Oriented Programming (ECOOP)*, volume 333, pages 1:1–1:30, 2025. doi:10.4230/LIPIcs.ECOOP.2025.1.
- 7 Eugene Asarin, Paul Caspi, and Oded Maler. Timed regular expressions. *Journal of the ACM*, 49(2):172–206, 2002. doi:10.1145/506147.506151.
- 8 Béatrice Bérard, Paul Gastin, and Antoine Petit. Refinements and abstractions of signal-event (timed) languages. In *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, pages 67–81. Springer, 2006. doi:10.1007/11867340_6.
- 9 Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. In *International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 432–443. Springer, 2005. doi:10.1007/11590156_35.
- 10 Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. *Information and Computation*, 208(2):97–116, 2010. doi:10.1016/J.IC.2009.10.004.
- 11 Bjorn B Brandenburg. *Scheduling and locking in multiprocessor real-time operating systems*. PhD thesis, The University of North Carolina at Chapel Hill, 2011.
- 12 Rocco De Nicola and Frits Vaandrager. Action versus state based logics for transition systems. In *LITP Spring School on Theoretical Computer Science*, pages 407–419. Springer, 1990. doi:10.1007/3-540-53479-2_17.
- 13 David L Dill. Timing assumptions and verification of finite-state concurrent systems. In *International Conference on Computer Aided Verification (CAV)*, pages 197–212. Springer, 1989. doi:10.1007/3-540-52148-8_17.
- 14 Deepak D’Souza and Pavithra Prabhakar. On the expressiveness of MTL in the pointwise and continuous semantics. *International Journal on Software Tools for Technology Transfer*, 9(1):1–4, 2007. doi:10.1007/S10009-005-0214-9.
- 15 Colin Fidge. Logical time in distributed computing systems. *Computer*, 24(8):28–33, 2002.
- 16 Mohammed Foughali, Pierre-Emmanuel Hladik, and Alexander Zuepke. Compositional verification of embedded real-time systems. *J. Syst. Archit.*, 142:102928, 2023. doi:10.1016/J.SYSARC.2023.102928.

- 17 Mohammed Aristide Foughali. Revisiting the expressiveness of metric temporal logic: A tale of "je t'aime, moi non plus.". *arXiv preprint arXiv:2603.15379*, 2026. doi:10.48550/arXiv.2603.15379.
- 18 Mohammed Aristide Foughali, Marius Mikucionis, and Maryline Zhang. Scalable computation of inter-core bounds through exact abstractions. In *Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 61–70, 2024. doi:10.1109/COMPSAC61105.2024.00019.
- 19 Carlo Alberto Furia and Matteo Rossi. On the expressiveness of MTL variants over dense time. In *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, pages 163–178. Springer, 2007.
- 20 Christoph Haase, Joël Ouaknine, and James Worrell. On process-algebraic extensions of metric temporal logic. *Reflections on the Work of CAR Hoare*, pages 283–300, 2010. doi:10.1007/978-1-84882-912-1_13.
- 21 Hsi-Ming Ho, Joël Ouaknine, and James Worrell. Online monitoring of metric temporal logic. In *International Conference on Runtime Verification (RV)*, pages 178–192. Springer, 2014. doi:10.1007/978-3-319-11164-3_15.
- 22 Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299, 1990. doi:10.1007/BF01995674.
- 23 Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. In *Concurrency: the Works of Leslie Lamport*, pages 179–196. Association for Computing Machinery, 2019. doi:10.1145/3335772.3335934.
- 24 Christoph Liebig, Mariano Cilia, and Alejandro Buchmann. Event composition in time-dependent distributed systems. In *International Conference on Cooperative Information Systems (COOPIS)*, pages 70–78. IEEE, 1999. doi:10.1109/COOPIS.1999.792159.
- 25 Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In *Symposium on Logic in Computer Science (LICS)*, pages 188–197. IEEE, 2005. doi:10.1109/LICS.2005.33.
- 26 Joël Ouaknine and James Worrell. On the decidability and complexity of metric temporal logic over finite words. *Logical Methods in Computer Science*, 3, 2007. doi:10.2168/LMCS-3(1:8)2007.
- 27 Joël Ouaknine and James Worrell. Some recent results in metric temporal logic. In *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, pages 1–13. Springer, 2008. doi:10.1007/978-3-540-85778-5_1.
- 28 Amir Pnueli. The temporal logic of programs. In *Annual Symposium on Foundations of Computer Science (SFCS)*, pages 46–57. iee, 1977. doi:10.1109/SFCS.1977.32.

Core Fragments of Compass Logic

Pietro Casavecchia ✉ 

Department of Mathematics and Computer Science, University of Ferrara, Italy

Guido Sciavicco ✉ 

Department of Mathematics and Computer Science, University of Ferrara, Italy

Leonardo Serrentino ✉ 

Department of Mathematics and Computer Science, University of Ferrara, Italy

Abstract

Compass Logic is arguably the most intuitive modal spatial logic designed for a point-based ontology, featuring exactly one unary modal operator for each cardinal direction, and interpreted over bi-dimensional structures; it can also be seen as a very natural bi-temporal logic, allowing independent reasoning in terms of future and past in two orthogonal temporal dimensions. As such, Compass Logic finds applications in several areas of modern artificial intelligence, from spatial reasoning to temporal databases. Its introduction is due to Marx and Reynolds in 1999 (inspired by previous work of Venema, 1990), when the authors proved that its satisfiability problem is undecidable regardless of the properties of the underlying structure. More recently, the idea of exploring sub-propositional fragments of unary modal logic emerged, and was applied to point-based and interval-based temporal logic. In the same spirit, in this paper we explore sub-propositional fragments of Compass Logic in the finite/discrete case, and we prove that while its satisfiability problem remains undecidable in very weak conditions, such as for the core fragment, characterized by having only formulas that are conjunctions of binary Horn clauses, it becomes surprisingly decidable for the same core fragment devoid of existential quantifiers.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Proof theory

Keywords and phrases spatial logic, bi-temporal logic, Horn fragment, core fragment, undecidability, decidability

Digital Object Identifier 10.4230/OASICS.TIME.2026.2

Related Version *Full Version:* <https://drive.google.com/file/d/10Za315mjSRp51SrK621EVQfEj8PaK9Q/view>

Funding This research was partially funded by INdAM – GNCS Project *Semantic neuro-symbolic contract for the rigorous reachability and CPS certifications* (CUP: E53C25002010001); Guido Sciavicco is a member of INdAM.

1 Introduction

Reasoning about time in terms of future and past, as well as about space in terms of directional relations is a natural idea.

Several qualitative temporal and spatial calculi have been developed for formalizing and reasoning about direction or ordering relations. At the algebraic level, these include the Point Calculus [19], which defines three ordering relations for points in a single-dimension Euclidean space, Allen’s Interval Calculus [2], with thirteen relations between intervals on a linear order, the Cardinal Direction Calculus [14], extending Point Calculus in two dimensions, and the Rectangle Algebra [6], extending, in turn, Allen’s algebra to the two dimensional case. Similarly, at the logical level the main separation among different proposals corresponds to points versus extended areas (in this case, intervals or rectangles) as atomic objects. The most important contributions generally come from the realm of propositional modal logic, and, on



© Pietro Casavecchia, Guido Sciavicco, and Leonardo Serrentino;
licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 2; pp. 2:1–2:16

OpenAccess Series in Informatics



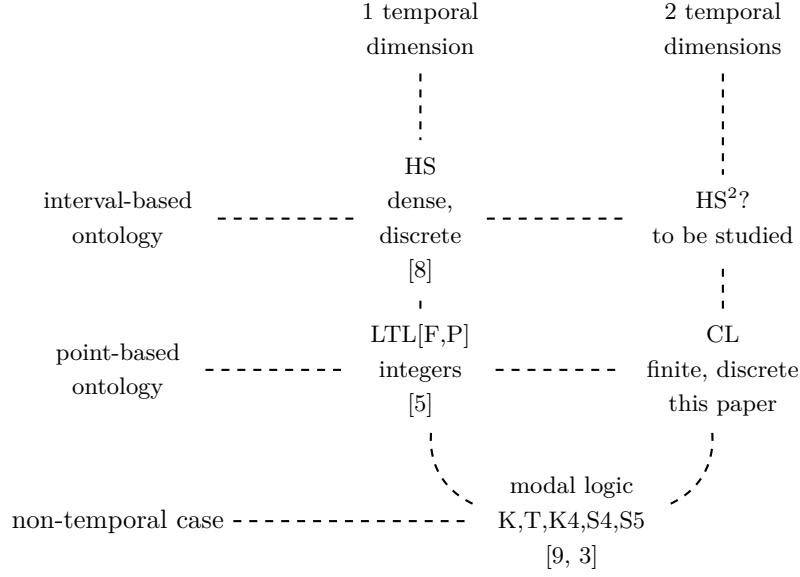
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a single temporal dimension, include Linear Temporal Logic with Future and Past (LTL[F,P], often presented as fragment of LTL, the Linear Temporal Logic with Since and Until) for the point-based case, and Halpern and Shoham's Modal Logic of Allen's Relations (HS) [12] for the interval-based case, and, in two temporal dimensions, Compass Logic (CL) [16], that is, the logical counterpart of Cardinal Direction Calculus for the point-based case. The immediate logical counterpart to Rectangle Algebra is the full product (in the sense of [11]) of HS with itself, encompassing 169 relations, and probably not very natural; the fragment of the Logic of Topological Relations (L_{RCC8}) [15] restricted to axes-parallel rectangles can also be mentioned in this context. Many other temporal and spatial logical frameworks exist, in fact too many to be reviewed here; however, for the purposes of this paper, we focus on logical languages inspired by, or that can assimilated to bi-temporal relations.

Sub-propositional fragments of modal logic are obtained by limiting the applicability of Boolean operators in formulas, in the spirit of Horn propositional logic. By focusing on binary clauses only, one obtains the so-called Krom fragments; binary Horn fragments are known as core fragments. In general, Horn and core fragments disallow the usual existential/universal duality, effectively generating different languages, with and without either of them, that can be partially ordered by expressive power. It is natural to ask how temporal and bi-temporal logics behave, computationally speaking, when restricted to sub-propositional fragments.

This line of investigation has been pursued in the case of Modal Logic interpreted in the classes of general (K), transitive (K4), reflexive (T), and transitive and reflexive frames (S4) [9], and revealed that the satisfiability problem reacts positively in several combinations of restrictions and class of frames: in all mentioned cases the satisfiability problem, which is PSPACE in the case of the full language, remains so in the Horn fragment, becomes *P*-complete in the case of the Horn fragment without diamonds, is between *NLOGSPACE* and *P* in the case of core fragment without diamonds for in all cases except K, which is *NLOGSPACE*-complete; the case of Horn S5 had already been mentioned in [3], and proved to be *NP*-complete. As for the temporal case, the sub-propositional fragments of LTL[F,P], with and without an additional universal operator, have been studied in [5], over the integers: as it happens, the satisfiability problem, which is *NP* for the full language, reduces to *P*-complete in the case of Horn without diamonds, and to *NLOGSPACE*-complete in the case of core without diamonds. This same approach in the case of linear temporal logic of intervals was attempted in [8] by tackling sub-propositional fragments of HS, whose satisfiability problem is generally undecidable, and revealed a more involuted situation. Indeed, HS interpreted on irreflexive frames (the usual interpretation) is still undecidable (*i*) under the core restriction even without boxes, and (*ii*) the Horn restriction even without diamonds, in the discrete case; it is *PSPACE*-hard under the core restriction, even without boxes, in the discrete case; it is *P* (resp., *P*-complete) under the core (resp., Horn) restriction without boxes. Moreover, HS interpreted on reflexive frames (a less intuitive and common interpretation) is undecidable under the core restriction even without boxes; *PSPACE*-hard under the Horn restriction, and in *P* (resp., *P*-complete) under the core (resp., Horn) restriction even without diamonds. Finally, both the reflexive and irreflexive cases still present several open problems.

In light of the above, stepping from one to two orthogonal temporal dimensions seems very natural; the situation can be graphically represented as in Fig. 1. Compass Logic, originally presented in [16] as a logic to reason about spatial situations (although Venema suggested the idea of Compass Logic and its notation in [18]), can be seen as the full product of LTL[F,P] with itself, allowing independent reasoning on both directions. Satisfiability in CL is undecidable, regardless of the properties of the underlying frame, and the quest for decidable and well-behaved fragments includes two relevant contributions. One of them is Cone Logic [17], which is a decidable weakening of CL obtained by enforcing the restriction



■ **Figure 1** A representation of the current status of research on sub-propositional temporal and bi-temporal logics. Bi-temporal logics can also be seen as spatial logics: in this regard, other prominent frameworks could show computational behaviour improvements under the Horn and core restrictions.

that quantifications along one axis can be used only after a similar quantification along the other axis; in this way, the modalities can be seen as north-east, north-west, south-east, and south-west, and the expressive power of the resulting language lowers to the point that its satisfiability problem becomes decidable. The other one is in fact the collection of all results concerning decidable fragments of HS [1, 7], which can be geometrically interpreted over two dimensions; such a geometric interpretation gives rise to (somewhat unnatural) operators that can be defined within Compass Logic, effectively converting any fragment of the former in a fragment of the latter. In both cases, however, the bi-temporal (and sometimes even the spatial) nature of the logic is lost. Finally, CL itself can be seen as a fragment of the bi-temporal logic proposed in [10], the fragments of the former, even those studied here, can be seen as fragments of the latter. Here, we prove that the satisfiability problem for CL, which is undecidable in the Boolean case, remains so for its core fragment with both universal and existential operators, and becomes P (and $NLOGSPACE$ -hard) without existential operators, in the finite (i.e., *has a formula φ a finite model?*), the infinite discrete (*has φ a model with each axes isomorphic to the integers?*), and the general discrete case (*has φ a discrete model?*), assuming, as usual, irreflexivity of the frames (i.e., the relation $<$ is strict).

All proofs can be found in the extended version of this paper.

2 Compass Logic

Let $\mathcal{P}$ be a (possibly infinite, but countable) set of *proposition letters* (or, simply, *propositions*).

► **Definition 1.** *The well-formed formulas of (propositional) Compass Logic (CL) are generated by the following grammar:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi,$$

where $p \in \mathcal{P}$ and $\Diamond \in \{\Diamond, \Diamond, \Diamond, \Diamond\}$. The size of φ , denoted by $|\varphi|$, is the number of its symbols.

Intuitively, formulas of CL will be interpreted on bi-dimensional Kripke structures, and $\diamondleft$ (resp., $\diamondright$, $\diamonddown$, $\diamondup$) will be interpreted as *exists to the left* (resp., *right*, *down*, *up*). The standard abbreviations apply: $\top$, $\vee$, and $\rightarrow$ respectively for true, Boolean disjunction, and Boolean implication, and one box operator for each diamond, defined as $\Box p \equiv \neg \diamond \neg p$, for $\Box \in \{\Boxleft, \Boxright, \Boxdown, \Boxup\}$. In the following, we use $|\varphi|$ to denote the length of φ , that is, the number of its symbols.

► **Definition 2.** Let $\mathbb{D} = \langle D, < \rangle$ be a linearly ordered set. A bi-dimensional Kripke frame (or, simply, a frame) is an object of the type $\mathcal{F} = \mathbb{D} \times \mathbb{D}$. Fixed a set $\mathcal{P}$ of propositional letters, a bi-dimensional Kripke structure (structure) $\mathcal{M}$ is a bi-dimensional Kripke frame enriched with a valuation function $\mathcal{V} : D \times D \rightarrow 2^{\mathcal{P}}$, that is, $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$. The truth of a formula φ on a bi-dimensional model $\mathcal{M}$ at a point (i, j) is given by the following recursive rules:

$$\begin{aligned} \mathcal{M}, (i, j) \models p &\text{ iff } p \in \mathcal{V}(i, j); \\ \mathcal{M}, (i, j) \models \neg \varphi &\text{ iff } \mathcal{M}, (i, j) \not\models \varphi; \\ \mathcal{M}, (i, j) \models \varphi \wedge \psi &\text{ iff } \mathcal{M}, (i, j) \models \varphi \text{ and } \mathcal{M}, (i, j) \models \psi; \\ \mathcal{M}, (i, j) \models \diamondleft \varphi &\text{ iff there is } k < i \text{ s.t. } \mathcal{M}, (k, j) \models \varphi; \\ \mathcal{M}, (i, j) \models \diamondright \varphi &\text{ iff there is } k > i \text{ s.t. } \mathcal{M}, (k, j) \models \varphi; \\ \mathcal{M}, (i, j) \models \diamonddown \varphi &\text{ iff there is } k < j \text{ s.t. } \mathcal{M}, (i, k) \models \varphi; \\ \mathcal{M}, (i, j) \models \diamondup \varphi &\text{ iff there is } k > j \text{ s.t. } \mathcal{M}, (i, k) \models \varphi. \end{aligned}$$

CL has an obvious geometric interpretation, in which the operators can be interpreted as *to the north*, *south*, *east*, and *west*; it also has an equally obvious bi-temporal interpretation in which the operators can be interpreted as *in the future* and *past* along two orthogonal temporal dimensions. Satisfiability of formulas of CL can be studied for several classes of bi-dimensional frames. Here, we are interested in the classes of (strongly) discrete frames, such as the class of all finite frames and the class of all frames based on the set of natural/integer numbers. In general, we shall assume that D is either finite of cardinality $2 \cdot N + 1$ (from $-N$ to N), and in this case we say that $\mathcal{M}$ has *size* N (denoted $|\mathcal{M}| = N$), or $D = \mathbb{Z}$. From now on, we focus on the satisfiability of a CL -formula at the point $(0, 0)$ of some discrete model, and we say that φ is (*finitely*) *satisfiable* if and only if it is satisfied on some (finite) discrete structure $\mathcal{M}$ at the point $(0, 0)$ (we also refer to this case as the *finite/discrete* case); the *satisfiability problem* for CL is the problem of establishing if a given formula is satisfiable, and it is denoted by $SAT(CL)$.

There are several additional operators that can be expressed in the language of basic CL , and that will be useful later. First, four *partial universal* modalities, ensuring a formula to be true everywhere up-right (resp., up-left, bottom-right, bottom-left) of the current point, excluding it (*irreflexive* version):

$$\begin{aligned} \Boxup \varphi &\equiv \Boxright \varphi \wedge \Boxdown \varphi \wedge \Boxleft \varphi, & \Boxright \varphi &\equiv \Boxup \varphi \wedge \Boxdown \varphi \wedge \Boxleft \varphi, \\ \Boxdown \varphi &\equiv \Boxleft \varphi \wedge \Boxright \varphi \wedge \Boxup \varphi, & \Boxleft \varphi &\equiv \Boxup \varphi \wedge \Boxdown \varphi \wedge \Boxright \varphi, \end{aligned}$$

and including it (*reflexive* version):

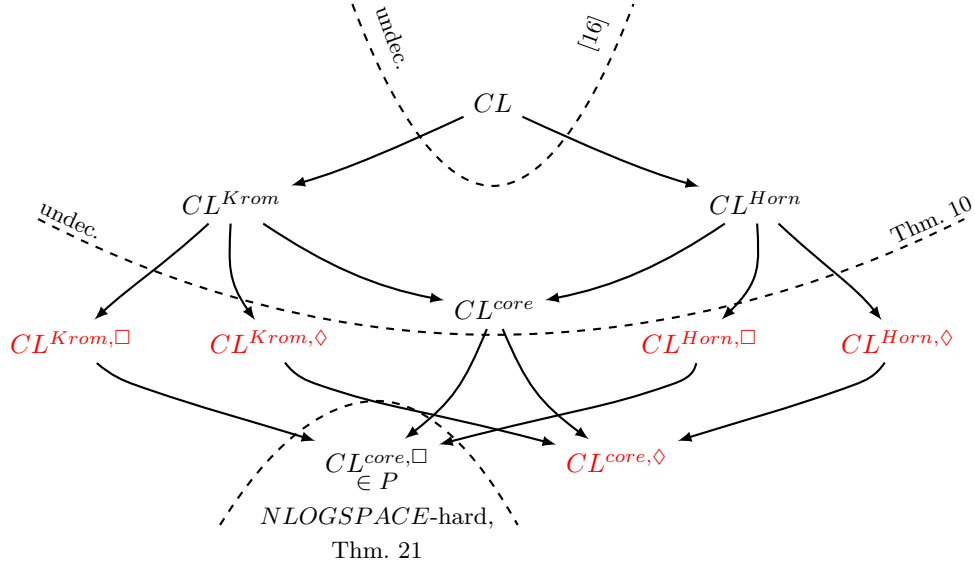
$$\begin{aligned} \Boxup \varphi &\equiv \varphi \wedge \Boxup \varphi, & \Boxright \varphi &\equiv \varphi \wedge \Boxright \varphi, \\ \Boxdown \varphi &\equiv \varphi \wedge \Boxdown \varphi, & \Boxleft \varphi &\equiv \varphi \wedge \Boxleft \varphi. \end{aligned}$$

Then, a *bidirectional* universal operator, placing φ everywhere on the same horizontal line, vertical line, and cross, excluding the current point (*irreflexive* version):

$$\Box \varphi \equiv \Boxup \varphi \wedge \Boxdown \varphi, \quad \Box \varphi \equiv \Boxright \varphi \wedge \Boxleft \varphi, \quad \Box \varphi \equiv \Boxup \varphi \wedge \Boxdown \varphi,$$

and including it (*reflexive* version):

$$\Box \varphi \equiv \varphi \wedge \Box \varphi, \quad \Box \varphi \equiv \varphi \wedge \Box \varphi, \quad \Box \varphi \equiv \varphi \wedge \Box \varphi.$$



■ **Figure 2** Relative expressive power by containment and known complexity results of fragments of CL in the finite/discrete case. The status of red fragments is unknown.

Finally, the classical (*full*) *universal* operator, which ensures that φ holds everywhere in a model:

$$\blacksquare\varphi \equiv \square\varphi \wedge \square\varphi \wedge \square\varphi \wedge \square\varphi.$$

All newly defined operators have, of course, an existential counterpart.

3 Sub-Propositional Fragments of Compass Logic

One way to define sub-propositional fragments of modal logics, that is, fragments of modal logic whose propositional expressive power is lower than that of standard Boolean logic, has been suggested in [9].

► **Definition 3.** A formula of CL is said to be in clausal form if and only if it is produced by the following grammar:

$$\begin{aligned} \lambda &::= \top \mid p \mid \Diamond\lambda \mid \square\lambda \\ C &::= \neg\lambda_1 \vee \dots \vee \neg\lambda_n \vee \lambda_{n+1} \vee \dots \vee \lambda_{n+m} \mid \square C \\ \varphi &::= C \mid \varphi \wedge \varphi, \end{aligned}$$

where $\square \in \{\square, \square, \square, \square\}$, and $n, m \in \mathbb{N}$. We use $\square^*$ in front of a clause to denote an arbitrary finite number of repetitions of some symbol $\square$.

Objects of the type λ and $\neg\lambda$ are called *literals* (this concept in fact generalizes the idea of literal in the propositional language) and objects of the type $\square^*(\dots)$ are called *clauses*. Given a CL -formula φ , let Λ_φ be the set of all literals that are sub-formulas of φ and their negations. Observe that, when in the clausal form, a formula φ is essentially a set (i.e., a conjunction) of clauses; a clause C can always be represented in its implicative form, which we shall always adopt. While there exist alternative definitions of modal clausal forms, as argued in [9] the previous one is the most general one. Some authors prefer clauses to be always universally

quantified, in the spirit of logic programming; since the universal operator is definable in the language, and since by adding enough fresh propositional letters we can always simulate the behaviour of a non-universal clause with universal ones, the two definitions are in fact equivalent up to satisfiability.

The clausal form of modal logic is Boolean-complete; however, it allows one to identify sub-Boolean, or *sub-propositional*, fragments, as follows.

► **Definition 4.** A *CL-clause*

$$\Box^*(\neg\lambda_1 \vee \dots \vee \neg\lambda_n \vee \lambda_{n+1} \vee \dots \vee \lambda_{n+m})$$

is said to be *Horn* if and only if $m \leq 1$, *Krom* if and only if $n + m \leq 2$, and *core* if it is both *Horn* and *Krom*.

The Horn, Krom, and core fragments of CL are denoted, respectively, CL^{Horn} , CL^{Krom} , and CL^{core} . While in the full Boolean language universal and existential operators are mutually definable, this, in general, is no longer true in sub-propositional ones. Thus, for $\spadesuit \in \{Horn, Krom, core\}$ it makes sense to define and study the fragments $CL^{\spadesuit, \Box}$, allowing only the universal version of the operators, and $CL^{\spadesuit, \Diamond}$, allowing only the existential one. These restrictions give rise to the natural taxonomy in Fig. 2. For a fragment L of CL , the corresponding satisfiability problem is denoted by $SAT(L)$.

Sub-propositional fragments may come across as too weak to be useful, especially when devoid of existential operators. However, when CL is seen as a bi-temporal logic, one could take advantage of its ability of expressing statements on two orthogonal dimensions, such as valid and transaction time in a temporal database, and formalize interesting statements. Just to limit ourselves to a single example, consider a temporal database with employees' information, and the case of a fired employee, for which the information *fired* should be persistent, that is, once someone is fired he/she is never hired back, and the system should never forget about it:

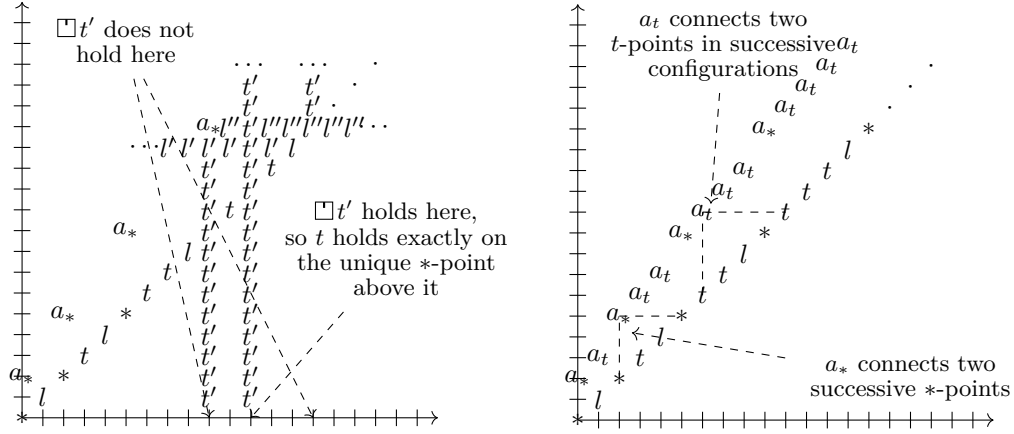
$$\blacksquare(fired \rightarrow (\Box fired \wedge \Box fired)).$$

In fact, one could express also interesting situations such as those modeled in [8, 10], at least in some reduced form. More in general, the fragment $L^{Horn, \Box}$ of a propositional modal logic L , and therefore its fragment $L^{core, \Box}$, translates to the Horn fragment of first-order logic via standard translation, underlying its naturalness, and, as it turns out, on top of the classical fields of automatic theorem proving and logic programming Horn fragments are often considered in recent contributions that focus on machine learning [4, 13], and having well-behaved fragments of bi-dimensional logic under Horn-like restrictions allows one to imagine similar applications specialized to the bi-dimensional case.

In the rest of this paper, focusing on finite/discrete structures only, we study the decidability of the satisfiability problem of sub-propositional fragments of CL, and we show that for CL^{core} this problem is still undecidable, while it becomes decidable for $CL^{core, \Box}$.

4 Undecidability of CL^{core}

The idea of the proof stems from the original undecidability argument for the non-clausal case [16]; we shall use a similar schema, and partially inherit the notation. Unlike the original result, however, we prove that the satisfiability of CL^{core} is undecidable via reduction from the acceptance problem of a deterministic Turing machine with empty input, rather than



■ **Figure 3** To the **left**: a partial representation of a model satisfying $(1) \wedge \dots \wedge (15)$. Observe, in particular, how l'' holds to the right of a a_* -point, and l' to the left of a l -point; since the former implies that t' holds everywhere above it (and including it), and the latter implies that t' holds everywhere below it (and including it), every point at height 0 whose horizontal coordinate is strictly between that of an a_* and that of an l is such that t' holds everywhere above it, and this, in turn, implies that t holds somewhere above it. To the **right**: a partial representation of a model satisfying $(1) \wedge \dots \wedge (24)$. By way of example, observe how a $*$ -point is connected to the next one and how a t -point is connected to the next one.

Finally, taking advantage of the sequence of p -points so defined, we can also use:

$$\Diamond z \equiv \Diamond z' \wedge \Box(z' \rightarrow u) \wedge \Box(z' \rightarrow \Diamond z) \wedge \Box(z \rightarrow p)$$

and

$$\Diamond z \equiv \Diamond z' \wedge \Box(z' \rightarrow u) \wedge \Box(z' \rightarrow \Diamond z) \wedge \Box(z \rightarrow p)$$

on p -points to place a propositional letter z on the next (resp., previous) p -point.

Preparing the sequence of configurations. Now, we define the notion of machine configuration as a sequence of p -points between two consecutive $*$ -points. We use the letter a_* to connect one $*$ -point to the next one, using e and the new letters $*_e$ and e_* to avoid forcing an infinite sequence. Thus, we have:

$$* \wedge \blacksquare(\Diamond e \rightarrow e), \quad (5)$$

$$\Box(e \rightarrow \Box e_*) \wedge \Box\Box(e \rightarrow \Box e_*) \wedge \blacksquare(* \rightarrow \Box e_*), \quad (6)$$

$$*_e \wedge \Box(\Box e_* \rightarrow \Diamond *_e) \wedge \blacksquare(*_e \rightarrow p), \quad (7)$$

$$\blacksquare(*_e \rightarrow \Diamond a_*) \wedge \blacksquare(a_* \rightarrow \Diamond *), \quad (8)$$

$$\Box(* \rightarrow \Diamond a_*) \wedge \blacksquare(a_* \rightarrow \Diamond *), \quad (9)$$

$$\blacksquare(a_* \rightarrow \Box\Box\neg*) \wedge \blacksquare(a_* \rightarrow \Box\Box\neg a_*) \wedge \blacksquare(a_* \rightarrow \Box\neg a_*), \quad (10)$$

$$\Diamond l \wedge \Box(* \rightarrow \Diamond l) \wedge \blacksquare(l \rightarrow \Diamond *), \quad (11)$$

$$\blacksquare(l \rightarrow \Box l') \wedge \blacksquare(a_* \rightarrow \Box l''), \quad (12)$$

$$\blacksquare(l' \rightarrow \Box t') \wedge \blacksquare(l'' \rightarrow \Box t'), \quad (13)$$

$$\blacksquare(l' \rightarrow t') \wedge \blacksquare(l'' \rightarrow t'), \quad (14)$$

$$\Box(\Box t' \rightarrow \Diamond t). \quad (15)$$

Fig. 3 (left-hand side) shows a partial representation of this part of the construction.

The first configuration is always constituted by a single tape cell, and the s -th configuration must be exactly one tape cell longer than the $(s - 1)$ -th. To achieve this, similarly to what already done with of $*$ -points, t - and l -points belonging to a configuration too will be connected to their corresponding ones in the next configuration, and this connection is mediated by the letter a_t . As before, such connections cannot force a model to be infinite; thus, t - (resp., l -) points that are below the level of e are marked by t_e (resp., l_e), and such a construction makes use of the auxiliary letter e_t (resp., e_l) to carry the information of the position of a t (resp., l); from an abstract point of view, the construction that makes this possible is quite similar to that used before to connect successive $*$ -points. This is realized by saying:

$$\blacksquare(e \rightarrow \Box e_t) \wedge \blacksquare(t \rightarrow \Box e_t), \quad (16)$$

$$\blacksquare(e \rightarrow \Box e_l) \wedge \blacksquare(l \rightarrow \Box e_l), \quad (17)$$

$$\Box(\Box e_t \rightarrow \Diamond t_e) \wedge \blacksquare(t_e \rightarrow t), \quad (18)$$

$$\Box(\Box e_l \rightarrow \Diamond l_e) \wedge \blacksquare(l_e \rightarrow l), \quad (19)$$

$$\blacksquare(t_e \rightarrow \Diamond a_t) \wedge \blacksquare(l_e \rightarrow \Diamond a_t) \wedge \blacksquare(a_t \rightarrow \Diamond t), \quad (20)$$

$$\blacksquare(t \rightarrow \Diamond a_t), \quad (21)$$

$$\blacksquare(a_t \rightarrow \Box \neg a_t) \wedge \blacksquare(a_t \rightarrow \Box \neg a_t), \quad (22)$$

$$\blacksquare(a_* \rightarrow \Box \neg a_t) \wedge \blacksquare(a_t \rightarrow \Box \neg a_*) \wedge \blacksquare(a_t \rightarrow \Diamond \Diamond *), \quad (23)$$

$$\bigwedge_{z_1, z_2 \in \{l, t, *\}, z_1 \neq z_2} \blacksquare(z_1 \rightarrow \neg z_2). \quad (24)$$

Fig. 3 (right-hand side) shows a partial representation of this part of the construction.

► **Lemma 6.** *Suppose that $\mathcal{M}, (0, 0) \models (1) \wedge \dots \wedge (24)$, and let $(i_0, i_0), \dots, (i_k, i_k)$ be the sequence of p -points whose existence is guaranteed by Lem. 5. Then:*

- (i) *there exists a finite sequence $(j_0, j_0), (j_1, j_1), \dots$ of $*$ -points on top of the sequence $(i_0, i_0), \dots, (i_k, i_k)$ whose existence is guaranteed by Lem. 5, no other points are $*$ -points, and $j_s \leq k$ for every s ;*
- (ii) *for each $s \geq 2$, $(j_s - 1, j_s - 1)$ is an l -point, and no other points are l -points;*
- (iii) *for each $s \geq 1$, all points between $(j_s + 1, j_s + 1)$ and $(j_{s+1} - 2, j_{s+1} - 2)$, included, are t -points, and no other points are t -points, and*
- (iv) *for each $s \geq 2$, $j_{s+1} - j_s = j_s - j_{s-1} + 1$.*

We say that every $$ -point starts a configuration.*

Introducing the transition function. Now we encode the transition function. To this end, we re-label tape cells in order to separate the first tape cell of each configuration (denoted with f) from any other tape cell (denoted by g). This construction is no different from the one already used to label all p -points between a $*$ -point and its corresponding l -point, and makes use of the auxiliary symbols f' , f'' , g' , and b_* :

$$\blacksquare(* \rightarrow \Diamond f), \quad (25)$$

$$\blacksquare(* \rightarrow \Diamond b_*) \wedge \blacksquare(b_* \rightarrow \Diamond *) \wedge \blacksquare(b_* \rightarrow \Box \neg *), \quad (26)$$

$$\blacksquare(f \rightarrow \Box f') \wedge \blacksquare(b_* \rightarrow \Box f''), \quad (27)$$

$$\blacksquare(f' \rightarrow \Box g') \wedge \blacksquare(f'' \rightarrow \Box g'), \quad (28)$$

$$\Box(\Box g' \rightarrow \Diamond g) \wedge \blacksquare(f \rightarrow \neg g) \wedge \blacksquare(g \rightarrow \neg f). \quad (29)$$

the auxiliary letters q'_c , that are forced to appear somewhere below each occurrence of a q_c , and particularly strictly above the horizontal line for first positions and strictly below it for non-first positions, and depending on where they actually occur, the final combined letters of the type $q_{c,f}$ and $q_{c,g}$ are built and placed on the same t -point. This construction is exemplified in Fig. 4 (left-hand side). We proceed by asserting:

$$\bigwedge_{z \in Q \cup \{r\}} (\Box(\Diamond 0 \rightarrow \neg z') \wedge \Box(\Diamond 0 \rightarrow \Box \neg z')), \quad (30)$$

$$\bigwedge_{z \in Q \cup \{r\}} (\Box(\Diamond 1 \rightarrow \neg z') \wedge \Box(\Diamond 1 \rightarrow \Box \neg z')), \quad (31)$$

$$\bigwedge_{z \in Q \cup \{r\}} (\Box(\Diamond \sqcup \rightarrow \Box \neg z') \wedge \Box(\Diamond \sqcup \rightarrow \Box \neg z')), \quad (32)$$

$$\bigwedge_{z \in Q \cup \{r\}} \blacksquare(z \rightarrow \Diamond z'), \quad (33)$$

$$\bigwedge_{z \in Q \cup \{r\}} (\Box(z' \rightarrow \Diamond z_\sqcup) \wedge \Box(\Diamond z' \rightarrow \Diamond z_0) \wedge \Box(\Diamond z' \rightarrow \Diamond z_1)), \quad (34)$$

$$\bigwedge_{q \in Q, c \in \{0,1,\sqcup\}} (\Box \neg q'_c \wedge \Box(\Diamond f \rightarrow \Box \neg q'_c) \wedge \Box(\Diamond g \rightarrow \Box \neg q'_c)), \quad (35)$$

$$\bigwedge_{q \in Q, c \in \{0,1,\sqcup\}} (\blacksquare(q_c \rightarrow \Diamond q'_c)), \quad (36)$$

$$\bigwedge_{q \in Q, c \in \{0,1,\sqcup\}} (\Box(\Diamond q'_c \rightarrow q_{c,f}) \wedge \Box(\Diamond q'_c \rightarrow q_{c,g})). \quad (37)$$

► **Lemma 8.** *If $\mathcal{M}, (0,0) \models (1) \wedge \dots \wedge (37)$ and $(i_0, i_0), (i_1, i_1), \dots, (i_k, i_k)$ is the sequence of p -points whose existence is guaranteed by Lem. 5, then:*

- (i) *if (i_s, i_s) satisfies q for some $q \in Q$, c for some $c \in \{0,1,\sqcup\}$, and f (resp., g), then it also satisfies $q_{c,f}$ (resp., $q_{c,g}$);*
- (ii) *if (i_s, i_s) satisfies r and c for some $c \in \{0,1,\sqcup\}$, then it also satisfies r_c .*

The last part of the reduction requires forcing configurations to behave according to the transition function δ . This implies making sure that the first configuration represents the Turing machine $\mathcal{A}$ starting with a 1-symbol tape holding $\sqcup$, the reading head currently on the first position, and the machine itself in state q_i . Reasoning by induction, and assuming that a particular configuration holds a particular sequence of tape symbols (on t -points) and that a particular letter q (for $q \in Q$) holds on some t -point (indicating that the reading head is currently on that position), the previous lemma ensures that all t -points consistently satisfy the letters $q_{c,f}$, $q_{c,g}$, and r_c , for $c \in \{0,1,\sqcup\}$. Our task is now that of transferring the information to the next configuration, in terms of symbols $0,1,\sqcup$ and q . This is achieved by means of the auxiliary letter a_t (introduced in Lem. 6) and the auxiliary letters of the type $q'_{c,f,M}$, with $M \in \{R,L\}$ (to denote a right or left movement of the reading head) and r'_c , that are forced to hold on a a_t point above some t -point and transfer the information on the corresponding t -point on its right, as well as $q''_{c,f,M}$, r''_c , that hold on t -points. Separating the situations in which the reading head is sitting, or not, on the first position in a configuration allows us to respond correctly to the particular case in which δ requires a left movement, which, in that case, must be ignored. An example is shown in Fig. 4 (right-hand side). Thus, we set:

$$\Diamond q_i \wedge \Diamond \sqcup \wedge \bigwedge_{q \in Q} \blacksquare(q \rightarrow \neg *), \quad (38)$$

$$\bigwedge_{(q,c) \rightarrow (\bar{q}, \bar{c}, M) \in \delta, v \in \{f, g\}} \blacksquare(q_{c,v} \rightarrow \Diamond \bar{q}'_{\bar{c}, v, M}) \wedge \bigwedge_{c \in \{0, 1, \sqcup\}} \blacksquare(r_c \rightarrow \Diamond \bar{r}'_c), \quad (39)$$

$$\bigwedge_{q \in Q, v \in \{f, g\}, M \in \{R, L\}, c \in \{0, 1, \sqcup\}} (\blacksquare(\bar{q}'_{\bar{c}, v, M} \rightarrow \Diamond \bar{q}''_{\bar{c}, v, M}) \wedge \blacksquare(\bar{r}'_c \rightarrow \Diamond \bar{r}''_c)), \quad (40)$$

$$\bigwedge_{q \in Q, v \in \{f, g\}, M \in \{R, L\}, c \in \{0, 1, \sqcup\}} (\blacksquare(\bar{q}'_{\bar{c}, v, M} \rightarrow a_t) \wedge \blacksquare(\bar{r}'_c \rightarrow a_t)), \quad (41)$$

$$\bigwedge_{q \in Q, v \in \{f, g\}, M \in \{R, L\}, c \in \{0, 1, \sqcup\}} (\blacksquare(\bar{q}''_{\bar{c}, v, M} \rightarrow t) \wedge \blacksquare(\bar{r}''_c \rightarrow t)), \quad (42)$$

$$\bigwedge_{q \in Q, v \in \{f, g\}, c \in \{0, 1, \sqcup\}} (\blacksquare(\bar{q}''_{\bar{c}, v, R} \rightarrow \Diamond \bar{q}) \wedge \blacksquare(\bar{q}''_{\bar{c}, v, R} \rightarrow \bar{c})), \quad (43)$$

$$\bigwedge_{q \in Q, v \in \{f, g\}, c \in \{0, 1, \sqcup\}} (\blacksquare(\bar{q}''_{\bar{c}, g, L} \rightarrow \Diamond \bar{q}) \wedge \blacksquare(\bar{q}''_{\bar{c}, g, L} \rightarrow \bar{c})), \quad (44)$$

$$\bigwedge_{q \in Q, v \in \{f, g\}, c \in \{0, 1, \sqcup\}} (\blacksquare(\bar{q}''_{\bar{c}, f, L} \rightarrow \bar{q}) \wedge \blacksquare(\bar{q}''_{\bar{c}, f, L} \rightarrow \bar{c})), \quad (45)$$

$$\bigwedge_{c \in \{0, 1, \sqcup\}} \blacksquare(\bar{r}''_c \rightarrow c), \quad (46)$$

$$\bigwedge_{c_1, c_2 \in \{0, 1, \sqcup\}} \blacksquare(c_1 \rightarrow \neg c_2) \wedge \bigwedge_{q_1, q_2 \in Q} \blacksquare(q_1 \rightarrow \neg q_2). \quad (47)$$

► **Lemma 9.** *If $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_i, q_a, q_r)$ is a deterministic Turing machine, $\mathcal{M}, (0, 0) \models (1) \wedge \dots \wedge (47)$, and $(j_0, j_0), (j_1, j_1), \dots$ is the sequence of $*$ -points whose existence is guaranteed by Lem. 6, then, for every $n \geq 0$ and $m \geq 1$:*

- (i) *if $(j_s + d, j_s + d)$ satisfies q for some $q \in Q$, c for some $c \in \{0, 1, \sqcup\}$, and f (resp., g), and if $(q, c) \rightarrow (\bar{q}, \bar{c}, R) \in \delta$, then $(j_{s+1} + d, j_{s+1} + d)$ satisfies $\bar{c}$ and $(j_{s+1} + d + 1, j_{s+1} + d + 1)$ satisfies $\bar{q}$;*
- (ii) *if $(j_s + d, j_s + d)$ satisfies q for some $q \in Q$, c for some $c \in \{0, 1, \sqcup\}$, and g , and if $(q, c) \rightarrow (\bar{q}, \bar{c}, L) \in \delta$, then $(j_{s+1} + d, j_{s+1} + d)$ satisfies $\bar{c}$ and $(j_{s+1} + d - 1, j_{s+1} + d - 1)$ satisfies $\bar{q}$;*
- (iii) *if $(j_s + d, j_s + d)$ satisfies q for some $q \in Q$, c for some $c \in \{0, 1, \sqcup\}$, and f , and if $(q, c) \rightarrow (\bar{q}, \bar{c}, L) \in \delta$, then $(j_{s+1} + d, j_{s+1} + d)$ satisfies both $\bar{c}$ and $\bar{q}$;*
- (iv) *if $(j_s + d, j_s + d)$ does not satisfy q for any $q \in Q$, and satisfies c for some $c \in \{0, 1, \sqcup\}$, then $(j_{s+1} + d, j_{s+1} + d)$ satisfies c .*

Completing the reduction. Consider now the formula:

$$(1) \wedge \dots \wedge (47) \wedge \Diamond \Diamond q_a. \quad (48)$$

► **Theorem 10.** *Given a Turing machine $\mathcal{A}$, the CL^{core} -formula (48) is satisfiable on a finite/discrete structure if and only if $\mathcal{A}$ accepts on the empty input. Therefore, the satisfiability problem for CL^{core} in the finite/discrete case is undecidable.*

5 Decidability of $CL^{core, \square}$

In this section, we prove that excluding diamonds from the grammar of the core fragment of CL entails regaining the decidability of its satisfiability problem in classes of discrete frames, as an effect of the low expressive power of this language.

Minimal extended structures. Let us start by introducing the concept of minimal extended structure.

► **Definition 11.** *Given a bi-dimensional Kripke frame $\mathcal{F}$, a bi-dimensional extended Kripke structure (or, simply, extended structure) $\mathcal{M}^+$ is a bi-dimensional Kripke frame enriched with an extended valuation function $\mathcal{V}^+ : D \times D \rightarrow 2^{\Lambda_\varphi}$, that is, $\mathcal{M}^+ = \langle \mathcal{F}, \mathcal{V}^+ \rangle$, such that, for every point $(i, j) \in \mathcal{F}$:*

- $\neg\lambda \in \mathcal{V}^+(i, j)$ iff $\lambda \notin \mathcal{V}^+(i, j)$,
- $\Box\lambda \in \mathcal{V}^+(i, j)$ iff $\lambda \in \mathcal{V}^+(k, j)$ for every $k < i$,
- $\Box\lambda \in \mathcal{V}^+(i, j)$ iff $\lambda \in \mathcal{V}^+(k, j)$ for every $k > i$,
- $\Box\lambda \in \mathcal{V}^+(i, j)$ iff $\lambda \in \mathcal{V}^+(i, k)$ for every $k > j$,
- $\Box\lambda \in \mathcal{V}^+(i, j)$ iff $\lambda \in \mathcal{V}^+(i, k)$ for every $k < j$.

The notion of a structure $\mathcal{M}$ satisfying a formula φ can be lifted to a notion of an extended structure $\mathcal{M}^+$ satisfying the formula φ by simply replacing the base case in Def. 2 with:

$$\mathcal{M}^+, (i, j) \models \lambda \text{ iff } \lambda \in \mathcal{V}^+(i, j).$$

The following result is an immediate consequence of the definition.

► **Lemma 12.** *Let φ be a $CL^{core, \Box}$ -formula. Then, φ is satisfiable if and only if it is satisfied on some extended structure.*

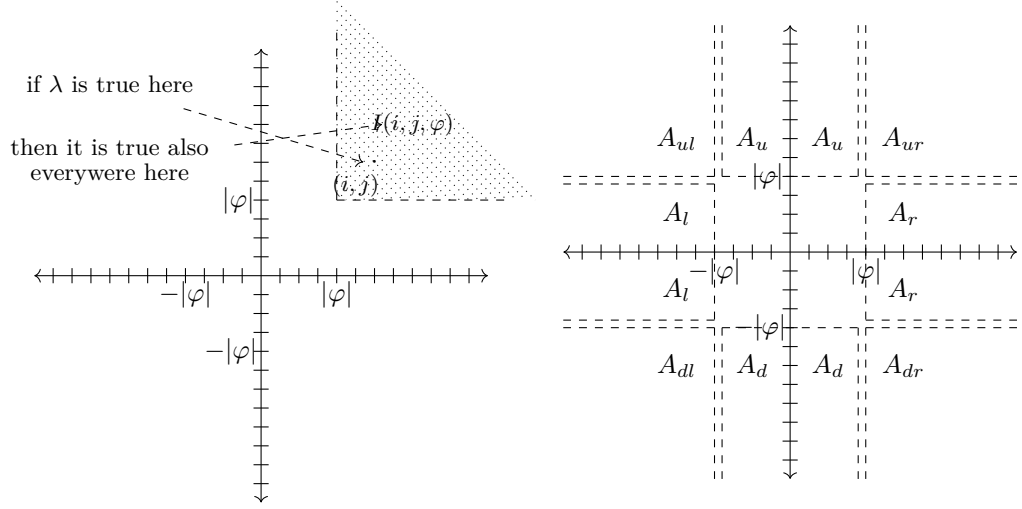
Since satisfiability of $CL^{core, \Box}$ -formulas can be reduced to satisfiability on extended structures, we use the same notation and terminology for structures (resp., models) and extended structures (resp., models).

► **Definition 13.** *Given a structure $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$, a point (i, j) , and a literal λ such that $\lambda \in \mathcal{V}(i, j)$, we say that a (i, j, λ) -reduct of $\mathcal{M}$ is any structure $\mathcal{M}' = \langle \mathcal{F}, \mathcal{V}' \rangle$ such that $\lambda \notin \mathcal{V}'(i, j)$. A structure $\mathcal{M}$ is said to be minimal for a $CL^{core, \Box}$ -formula φ if and only if $\mathcal{M}, (0, 0) \models \varphi$ and for every (i, j, λ) -reduct $\mathcal{M}'$ of $\mathcal{M}$ it is the case that $\mathcal{M}', (0, 0) \not\models \varphi$.*

► **Lemma 14.** *Let φ be a $CL^{core, \Box}$ -formula. Then, φ is satisfiable if and only if it is satisfied by some minimal structure.*

Minimal structures are a standard notion in Horn-like fragments of propositional (modal) logic, based on the observation that in such fragments there cannot be a disjunction of positive literals; this allows one to partially order structures by containment, which, in spatial logic, becomes pointwise containment. Since satisfiability of $CL^{core, \Box}$ -formulas can be reduced to satisfiability on minimal structures, again, we make the notation and terminology uniform by using structure (resp., model) instead of minimal extended structure (resp., minimal extended model).

Ultimately constant structures. In order to understand the properties of (minimal extended) structures that satisfy $CL^{core, \Box}$ -formulas, first we prove that positive literals cannot exist in isolated points except on $(0, 0)$.



■ **Figure 5** To the **left**: a pictorial representation of influence area and Lem. 16. To the **right**: a pictorial representation of an ultimately constant model and Lem. 19.

► **Definition 15.** Let φ be a $CL^{core, \square}$ -formula, $\mathcal{F}$ a structure, $(i, j) \in \mathcal{F}$ a point. The influence area of (i, j) w.r.t. φ , denoted $I(i, j, \varphi)$, is defined by the following rule. For every (i', j') , it is the case that $(i', j') \in I(i, j, \varphi)$ if and only if:

$$\begin{array}{ll}
 i' \geq \min\{i, |\varphi|\}, j' \geq \min\{j, |\varphi|\}, & \text{if } i > 0, j > 0, \\
 i' = 0, j' \geq \min\{j, |\varphi|\}, & \text{if } i = 0, j > 0, \\
 i' \leq \max\{i, -|\varphi|\}, j' \geq \min\{j, |\varphi|\}, & \text{if } i < 0, j > 0, \\
 i' \leq \max\{i, -|\varphi|\}, j' = 0, & \text{if } i < 0, j = 0, \\
 i' \leq \max\{i, -|\varphi|\}, j' \leq \max\{j, -|\varphi|\}, & \text{if } i < 0, j < 0, \\
 i' = 0, j' \leq \max\{j, -|\varphi|\}, & \text{if } i = 0, j < 0, \\
 i' \geq \min\{i, |\varphi|\}, j' \leq \max\{j, -|\varphi|\}, & \text{if } i > 0, j < 0, \\
 i' \geq \min\{i, |\varphi|\}, j' = 0, & \text{if } i > 0, j = 0.
 \end{array}$$

► **Lemma 16.** Let φ be a $CL^{core, \square}$ -formula, $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$ a model for φ , and λ a literal. Then, if $\mathcal{M}, (i, j) \models \lambda$ for some point $(i, j) \neq (0, 0)$, then $\mathcal{M}, (i', j') \models \lambda$ for every $(i', j') \in I(i, j, \varphi)$.

A pictorial representation of the effects of Lem. 16 can be found in Fig. 5 (left-hand side).

Now, when we look at points far away from the point $(0, 0)$ we see that structures have a very well-defined shape.

► **Definition 17.** Let φ be a $CL^{core, \square}$ -formula, and $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$ a model for φ . The up (resp., up right, right, down right, down, down left, left, up left) area, denoted $A_{\clubsuit}$, for $\clubsuit \in \{u, ur, r, dr, d, dl, l, ul\}$, is defined as follows:

$$\begin{array}{l}
 A_u = \{(i, j) \mid -|\varphi| < i < |\varphi|, j \geq |\varphi|\}; \\
 A_{ur} = \{(i, j) \mid i \geq |\varphi|, j \geq |\varphi|\}; \\
 A_r = \{(i, j) \mid i \geq |\varphi|, -|\varphi| < j < |\varphi|\}; \\
 A_{dr} = \{(i, j) \mid i \geq |\varphi|, j \leq -|\varphi|\}; \\
 A_d = \{(i, j) \mid -|\varphi| < i < |\varphi|, j \leq -|\varphi|\}; \\
 A_{dl} = \{(i, j) \mid i \leq -|\varphi|, j \leq -|\varphi|\}; \\
 A_l = \{(i, j) \mid i \leq -|\varphi|, -|\varphi| < j < |\varphi|\}; \\
 A_{ul} = \{(i, j) \mid i \leq -|\varphi|, j \geq |\varphi|\}.
 \end{array}$$

► **Definition 18.** Let φ be a $CL^{core, \square}$ -formula and $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$ a model for φ with size $|\mathcal{M}| > |\varphi|$. We say that $\mathcal{M}$ is ultimately constant if and only if for every pair of points $(i, j), (i', j')$ and every $\clubsuit \in \{ur, dr, dl, ul\}$:

1. if $(i, j), (i', j') \in A_{\clubsuit}$, then $\mathcal{V}(i, j) = \mathcal{V}(i', j')$;
2. if $(i, j) \in A_u$, then $\mathcal{V}(i, j) = \mathcal{V}(i, j + 1)$;
3. if $(i, j) \in A_r$, then $\mathcal{V}(i, j) = \mathcal{V}(i + 1, j)$;
4. if $(i, j) \in A_d$, then $\mathcal{V}(i, j) = \mathcal{V}(i, j - 1)$, and
5. if $(i, j) \in A_l$, then $\mathcal{V}(i, j) = \mathcal{V}(i - 1, j)$.

► **Lemma 19.** Let φ be a $CL^{core, \square}$ -formula. Then, if φ is satisfied on a structure $\mathcal{M}$ with size $|\mathcal{M}| > |\varphi|$, then $\mathcal{M}$ is ultimately constant.

A pictorial representation of the effects of Lem. 19 can be found in Fig. 5 (right-hand side).

Satisfiability of $CL^{core, \square}$ is in P . Decidability of satisfiability in $CL^{core, \square}$ in the discrete case is an immediate consequence of Lem. 19 and the fact that ultimately constant models are finitely representable. Decidability in the finite case is proven by observing that a given a $CL^{core, \square}$ -formula φ is finitely satisfiable if and only if it is satisfied on some minimal model of size $N = |\varphi|$.

► **Lemma 20.** Let φ be a $CL^{core, \square}$ -formula. Then, if φ is finitely satisfiable, then there exists a structure $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$, where $\mathcal{F} = \mathbb{D} \times \mathbb{D}$ and $\mathbb{D} = \langle D, < \rangle$, such that, first, $\mathcal{M}, (0, 0) \models \varphi$ and, second, $|\mathcal{M}| \leq |\varphi|$.

► **Theorem 21.** The satisfiability problem for $CL^{core, \square}$ in the class of all finite structures, the class of all infinite structures, and the class of all discrete structures, is in P and $NLOGSPACE$ -hard.

6 Conclusions

Despite the naturalness of Compass Logic (CL), its properties have been scarcely investigated, probably because it was introduced at the same time that its undecidability was proven, discouraging its use. Curiously, the same happened with Halpern and Shohams' Modal Logic of Allen's Relations (HS), which was immediately shown to be undecidable; yet, in the latter case a number of syntactical, semantical, and also clausal fragments were studied, contributing to the identification of a very complex landscape. In the case of CL, there have been only two previous attempts at finding well-behaved sub-languages.

In this paper we initiated the analysis of clausal fragments of CL, inspired by similar previous investigations. We unveiled a small, still undecidable fragment of CL, which turns out to be technically interesting due to the complexity of the undecidability proof, and an even smaller fragment of it, which, on the contrary, is decidable in deterministic polynomial time. Despite CL and HS being both bi-dimensional, their behaviour under Horn-like restrictions is different; in the case of CL, in particular, the core fragment turns out to be decidable in the discrete irreflexive case, which is precisely the most outstanding case left open in the case of the core fragment of HS. Whether the decidability of the satisfiability problem of core fragment of CL can be pushed further, say, to the Horn fragment, or whether its complexity can be pinpointed in a tighter way, are open problems at the moment.

Modal logics for qualitative reasoning have a wide range of real-life applications in modern artificial intelligence and machine learning. Decidable and efficient languages for reasoning about non-tabular data, such as temporal and spatial data, can be applied in machine learning-based applications, and good computational properties enable useful reasoning tasks, such as minimization of learned rules and checking expert-knowledge compliance.

References

- 1 Luca Aceto, Dario Della Monica, Valentin Goranko, Anna Ingólfssdóttir, Angelo Montanari, and Guido Sciavicco. A complete classification of the expressiveness of interval logics of Allen's relations: the general and the dense cases. *Acta Informatica*, 53(3):207–246, 2016. doi:10.1007/S00236-015-0231-4.
- 2 James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983. doi:10.1145/182.358434.
- 3 Linh A.Nguyen. On the complexity of fragments of modal logics. In *Proc. of the 5th Conference on Advances in Modal Logic*, pages 249–268. King's College Publications, 2004. URL: <http://www.aiml.net/volumes/volume5/Nguyen.ps>.
- 4 Marta Arias, José L. Balcázar, and Cristina Tirnăuță. Learning definite Horn formulas from closure queries. *Theoretical Computer Science*, 658:346–356, 2017. doi:10.1016/J.TCS.2015.12.019.
- 5 Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyashev. The complexity of clausal fragments of LTL. In *Proc. of the 19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 8312 of *Lecture Notes in Computer Science*, pages 35–52. Springer, 2013. doi:10.1007/978-3-642-45221-5_3.
- 6 Philippe Balbiani, Jean-François Condotta, and Luis Fariñas del Cerro. A model for reasoning about bidimensional temporal relations. In *Proc. of the 6th International Conference on Principles of Knowledge Representation and Reasoning*, pages 124–130, Trento, Italy, 1998. Morgan Kaufmann.
- 7 Davide Bresolin, Dario Della Monica, Angelo Montanari, Pietro Sala, and Guido Sciavicco. Interval temporal logics over strongly discrete linear orders: Expressiveness and complexity. *Theoretical Computer Science*, 560:269–291, 2014. doi:10.1016/J.TCS.2014.03.033.
- 8 Davide Bresolin, Agi Kurucz, Emilio Muñoz-Velasco, Vladislav Ryzhikov, Guido Sciavicco, and Michael Zakharyashev. Horn fragments of the Halpern-Shoham interval temporal logic. *ACM Transactions on Computational Logic*, 18(3):22:1–22:39, 2017. doi:10.1145/3105909.
- 9 Davide Bresolin, Emilio Muñoz-Velasco, and Guido Sciavicco. On sub-propositional fragments of modal logic. *Logical Methods in Computer Science*, 14(2), 2018. doi:10.23638/LMCS-14(2:16)2018.
- 10 Marcelo Finger and Mark Reynolds. Two-dimensional executable temporal logic for bitemporal databases. In Howard Barringer, Michael Fisher, Dov Gabbay, and Graham Gough, editors, *Advances in Temporal Logic*, pages 393–411. Springer, 2000.
- 11 Dov M. Gabbay and Valentin B. Shehtman. Products of modal logics, part 1. *Logic Journal of the IGPL*, 6(1):73–146, 1998. doi:10.1093/JIGPAL/6.1.73.
- 12 Joseph Y. Halpern and Yoav Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962, 1991. doi:10.1145/115234.115351.
- 13 Miki Hermann and Gernot Salzer. Efficient learning of Horn formulas over finite totally ordered domains. In *Proc. of the 18th International Conference on Multi-disciplinary Trends in Artificial Intelligence*, volume 16354 of *Lecture Notes in Computer Science*, pages 80–92, 2026. doi:10.1007/978-981-95-4960-3_7.
- 14 Gerard Ligozat. Reasoning about cardinal directions. *Journal of Visual Languages and Computation*, 9(1):23–44, 1998. doi:10.1006/JVLC.1997.9999.
- 15 Carsten Lutz and Frank Wolter. Modal logics of topological relations. *Logical Methods in Computer Science*, 2(2), 2006. doi:10.2168/LMCS-2(2:5)2006.
- 16 Maarten Marx and Mark Reynolds. Undecidability of compass logic. *Journal of Logic and Computation*, 9(6):897–914, 1999. doi:10.1093/LOGCOM/9.6.897.
- 17 Angelo Montanari, Gabriele Puppis, and Pietro Sala. A decidable weakening of compass logic based on cone-shaped cardinal directions. *Logical Methods in Computer Science*, 11(4), 2015. doi:10.2168/LMCS-11(4:7)2015.
- 18 Yde Venema. Expressiveness and completeness of an interval tense logic. *Notre Dame Journal of Formal Logic*, 31(4):529–547, 1990. doi:10.1305/NDJFL/1093635589.
- 19 Marc Vilain and Henry Kautz. Constraint propagation algorithms for temporal reasoning. In *Proc. of the 5th National Conference on Artificial Intelligence*, pages 377–382. Morgan Kaufmann, 1986. URL: <http://www.aaai.org/Library/AAAI/1986/aaai86-063.php>.

The Σ -Chain Product: A Succinct Model of Automata (De)Composition

Roberto Borelli  

University of Padua, Italy

Davide Bresolin   

University of Padua, Italy

Luca Geatti   

University of Udine, Italy

Angelo Montanari   

University of Udine, Italy

Matteo Zavatteri   

Fondazione Bruno Kessler, Trento, Italy

Abstract

The cascade product is a fundamental construction in automata theory, enabling hierarchical composition of automata and playing a central role in decomposition results such as the Krohn–Rhodes theorem. However, its use is limited by the exponential size required to represent cascades, which stems from the fact that each component may depend on all preceding ones, leading to exponentially large alphabets.

To address this issue, we introduce the Σ -chain product, a restricted variant in which each component depends only on the input alphabet and the component immediately preceding it. We show that Σ -chains achieve linear-size representations and can be exponentially more succinct than cascades. We prove that Σ -chains and cascades are expressively equivalent even when restricting the components to specific classes of automata, such as permutation-reset automata. As a consequence, we derive that a language is regular if and only if it is recognized by a Σ -chain of permutation-reset automata. Finally, we analyze structural properties of Σ -chains of reset automata, including a relation with well-known subclasses of star-free languages.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Automata extensions; Theory of computation $\rightarrow$ Modal and temporal logics; Theory of computation $\rightarrow$ Regular languages

Keywords and phrases Automata, Cascade Product, Formal Languages, Krohn-Rhodes Theory

Digital Object Identifier 10.4230/OASICS.TIME.2026.3

Related Version *Extended Version*: <https://arxiv.org/abs/2607.16884> [2]

Funding Davide Bresolin and Luca Geatti acknowledge financial support from the INdAM – GNCS Project “Neuro-symbolic semantic contract for rigorous reachability and certificates in CPS” (CUP E53C25002010001). Luca Geatti acknowledges financial support from the *Programma Operativo Complementare* of the *Regione Autonoma Friuli Venezia Giulia*, under CUP G23C25000880002, Operation Code 2025/13846, Project Code 2025/13846/16.

1 Introduction

The *cascade product* of deterministic automata generalises the direct product by arranging the component automata in a linear, hierarchical structure [12, 18, 13]. While in the direct product every component reads symbols from a fixed alphabet Σ , in the cascade product the component at level i reads symbols from Σ and, additionally, observe the state of every component at level $j < i$. Formally, the automaton at level i reads symbols from the Cartesian



© Roberto Borelli, Davide Bresolin, Luca Geatti, Angelo Montanari, and Matteo Zavatteri; licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 3; pp. 3:1–3:17

OpenAccess Series in Informatics



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

product $\Sigma \times Q_1 \times \cdots \times Q_{i-1}$, where Q_j denotes the state set of the automaton at level j ; the first component, as in the direct product, reads symbols from Σ alone. A transition labelled $(a, q_1, \dots, q_{i-1})$ is taken if and only if the current input symbol is $a \in \Sigma$ and the automaton at level j is currently in state q_j , for every $1 \leq j < i$. Crucially, the automaton obtained by applying the cascade product is defined over the original alphabet Σ alone.

The cascade product was introduced in the context of semigroup and semiautomaton decomposition theory, and lies at the heart of the Krohn–Rhodes Decomposition Theorem [12]. This theorem states that every deterministic semiautomaton (*i.e.* an automaton without initial and accepting states) can be decomposed into a sequence of permutation-reset semiautomata whose cascade product *covers* the original semiautomaton – that is, the original is a homomorphic image of a sub-semiautomaton of the product.¹ In other words, the cascade product can simulate all the behaviours of the original semiautomaton using only very simple components, namely permutation-reset semiautomata. A semiautomaton over a state set Q is called *permutation-reset* if every symbol in its alphabet induces a transformation $Q \rightarrow Q$ that is either a permutation (*i.e.* a bijection) or a reset (a map whose image has cardinality one). A semiautomaton is called *reset* if every symbol induces either a reset function or the identity (a special case of a permutation). A central insight of the Krohn–Rhodes theorem is that every counter-free semiautomaton – corresponding to star-free regular expressions – can be decomposed into a cascade product of two-state reset semiautomata.

Beyond its theoretical significance in automata theory and semigroup algebra, the Krohn–Rhodes Decomposition Theorem has found applications in the translation of automata to Linear Temporal Logic formulas [14, 13] and in the transformation of finite-word automata into formulas of pure-past Linear Temporal Logic – formulas whose temporal operators refer exclusively to the past – in what is known as the *pastification* problem [3, 1]. An application of the cascade product in formal verification appears in [7], where the approach is to decompose the specification to be model-checked against a system into a sequence of components arranged in a cascade structure, and to execute the model-checking algorithm one component at a time, starting from the first. The rationale is that component-by-component verification is generally more efficient than the monolithic verification of the full system. More recently, the cascade product has been extended from semiautomata to full automata in [3], where the relationship between cascades of a given number of reset automata and the corresponding classes of regular expressions has been studied. It has been shown that the cascade product of permutation-reset automata (respectively, of reset automata) captures exactly the class of regular (respectively, star-free) languages.

A significant drawback that limits the applicability of the cascade product is the size of its representation. On one hand, the automaton obtained by computing the cascade product of n (semi)automata $\mathcal{A}_1, \dots, \mathcal{A}_n$ has a number of states exponential in n , a phenomenon shared with the direct product. On the other hand, and more problematically, even representing the *sequence of components* $\langle \mathcal{A}_1, \dots, \mathcal{A}_n \rangle$ – which we call a *cascade* of (semi)automata – without computing the actual product requires exponential space. This is a consequence of the component alphabets: in particular, the alphabet of the last component $\mathcal{A}_n$ has cardinality $|\Sigma| \times |Q_1| \times \cdots \times |Q_{n-1}|$, which is already exponential in n . Since we consider deterministic automata, the number of transitions of $\mathcal{A}_n$ is also exponential in n . While this represents a significant drawback from the perspective of space complexity, it is also a key source of expressive power, allowing $\mathcal{A}_n$ to distinguish every possible configuration of the preceding

¹ A stronger version of the theorem exists, where the decomposition consists of permutation automata and two-state reset automata. We use the weaker formulation to keep all components of the same type.

components together with the current input symbol, and to select a different successor state for each such configuration. By contrast, representing the sequence of components of a direct product requires only space linear in n and in the size of the largest automaton.

Motivated by the desire to overcome this representational inefficiency while surpassing the expressive power of the direct product (which only captures closure under intersection), we introduce in this paper the notion of Σ -chains and the corresponding Σ -chain product. A Σ -chain is a sequence of automata in which the first component reads symbols from Σ , while each subsequent component depends only on Σ and on the immediately preceding component, that is, where the component at level i reads symbols from the Cartesian product $\Sigma \times Q_{i-1}$, where Q_{i-1} is the state set of the automaton at level $i-1$. The Σ -chain product is the resulting product automaton, defined analogously to the cascade product: the i -th automaton takes a transition labeled (a, q_{i-1}) if and only if the current input symbol is a and the preceding component is in state q_{i-1} . The Σ -chain product is thus a restriction of the cascade product in which each component's dependency is confined to Σ and the component immediately preceding it, and a generalization of the direct product in that it admits a hierarchical dependency structure among components.

We begin by analyzing the representational complexity of Σ -chains. We show that, while the Σ -chain product of n automata is an automaton with a number of states exponential in n (as with both the cascade and the direct product), the Σ -chain itself – *i.e.* the sequence of components – has size linear in n (as with the direct product, but unlike the cascade). Furthermore, we demonstrate that Σ -chains can be exponentially more succinct than cascades: there exists a family of languages recognisable by Σ -chains of n reset automata (thus representable in space polynomial in n) such that any cascade of reset automata for those languages requires at least n components, and hence at least exponential space.

We then turn to the expressive power of Σ -chains. We show that Σ -chains and cascades of automata are expressively equivalent. We first observe that every Σ -chain is, by construction, a special case of a cascade: the dependency restriction imposed on each component is simply a syntactic constraint, and lifting it yields a valid cascade with identical components. The converse direction is less immediate. We show that every cascade can be transformed into an equivalent Σ -chain via a *chainification* technique: intuitively, the dependence of a component – say, the third – on the states of a non-adjacent predecessor – say, the first – can be eliminated by encoding the relevant state information into the state space of the intervening component. We prove, moreover, that for the classes of permutation-reset, pure-reset (all symbols induce a reset), and permutation automata (all symbols induce a permutation), the resulting Σ -chain has the same number of components as the original cascade, and each component belongs to the same class. This comes at the cost that, at level i , the number of states becomes exponential in i . A corollary of this transformation is that Σ -chains of permutation-reset automata recognise exactly the class of regular languages. This means that, for permutation-reset automata, maximal expressive power can be achieved without requiring the i -th level to depend on all previous levels. Rather, it only needs to depend on the immediately preceding level and the input alphabet. For reset automata, the proposed transformation does not preserve the component type: a reset automaton may become a permutation-reset automaton. While the number of components is preserved, the resulting Σ -chain may no longer consist entirely of reset automata. Therefore, we can only establish that languages recognized by Σ -chains of reset automata are included in the star-free languages, leaving the converse open. Table 1 summarizes and compares the expressiveness results for Σ -chains against known results on cascades and direct products.

■ **Table 1** Summary of expressiveness results on direct products, Σ -chains and cascades of automata, for the classes of pure-reset ($\mathcal{R}^-$), reset ($\mathcal{R}$) and permutation-reset ($\mathcal{R}^+$) automata. SF denotes the class of star-free languages and REG denotes the class of all regular languages. [†] Some preliminary results on the expressiveness of Σ -chains of reset automata are discussed in Section 5.

	Direct Product	Σ -chains	Cascades
$\mathcal{R}^-$	$\bigcap_i (\Sigma^* R_i \cup J_i)$ $J_i = \varepsilon \text{ or } J_i = \emptyset$ $R_i, I_i \subseteq \Sigma, R_i \cap I_i = \emptyset$ [3]	$\subseteq$ Lemma 11	$=$ Corollary 14
$\mathcal{R}$	$\bigcap_i (\Sigma^* R_i I_i^* \cup J_i)$ $J_i = I_i^* \text{ or } J_i = \emptyset$ $R_i, I_i \subseteq \Sigma, R_i \cap I_i = \emptyset$ [3]	$\subseteq$ Lemma 11	$\subseteq$ Corollary 14 SF [4]
$\mathcal{R}^+$	$open$	$open$	$=$ Corollary 14 REG [4]

We next investigate Σ -chains of reset automata, showing that they can recognize several natural and well-studied classes of star-free languages, including Piecewise Testable languages [16] and R-trivial languages [5]. We also consider two hierarchies of languages obtained by restricting Σ -chains of reset automata by height (number of components) and by width (number of states per level). We prove that the height hierarchy is strict – for every h , there exist languages recognisable by a Σ -chain of reset automata of height h but not of height $h - 1$ – whereas the width hierarchy collapses: every Σ -chain of reset automata is equivalent to one in which the number of states of each component depends only on the size of the alphabet Σ . Finally, we address the pastification problem for Σ -chains of reset automata, namely the transformation of such structures into equivalent formulas in pure past Linear Temporal Logic. A straightforward approach would consist in first translating a Σ -chain into a cascade and then applying the procedure of [4] to obtain the corresponding formula. Instead, we adapt the construction of [4] so as to directly translate a Σ -chain of reset automata into a formula of quadratic size, thereby avoiding the exponential blow-up caused by the intermediate transformation from Σ -chains to cascades.

The paper is structured as follows. Related work is discussed in Section 2. In Section 3, we provide some background knowledge. In Section 4, we introduce the notion of Σ -chains and Σ -chain product of automata, we state some basic results and we discuss succinctness and expressiveness through the use of the chainification (the transformation of cascades into Σ -chains) technique. In Section 5 we focus on Σ -chains of reset automata, in particular we study height and width hierarchies, we compare Σ -chains of reset automata to piecewise-testable and R-trivial languages and we discuss an efficient pastification procedure for Σ -chains. Finally, in Section 6 we assess the work done, and we outline some directions for future works. Proofs of all results are provided in the Extended Version [2].

2 Related Work

The composition of (semi)automata has been extensively studied in automata theory and semigroup theory. In particular, the cascade product plays a central role in decomposition results such as the Krohn–Rhodes Theorem [12], which shows that every finite semiautomaton can be decomposed into a cascade of permutation-reset semiautomata. Since then, several works have investigated structural, algebraic, and logical properties of cascades of automata and semiautomata, while others have also considered restrictions of the cascade product.

In [9], Gelderie studies structural properties of cascade products and introduces the notion of scope: a cascade has scope k if each transition of each component depends on the states of at most the previous k components. Cascades of scope 1 have a strictly linear dependency structure and can be readily transformed into Σ -chains. Gelderie further shows that piecewise-testable languages are recognized by cascades of reset semiautomata of scope 1, and R-trivial languages by cascades of scope 2.

Our work differs in two ways. First, we consider full automata rather than semiautomata. Second, instead of studying cascades of bounded-scope, we introduce Σ -chains, where the i -th component depends only on the input letter and the state of component $i - 1$. We exhibit Σ -chains of reset automata for generalizations of both piecewise-testable and R-trivial languages. As a byproduct, since R-trivial languages are recognized by finite unions of Σ -chains of reset automata and cascades are closed under union, R-trivial languages are also recognized by cascades of reset semiautomata of scope 1, improving Gelderie's bound.

In [8], Gecseg surveys a wide range of results on homomorphically and isomorphically complete classes of semiautomata with respect to α_j -products [11]. Informally, the α_0 -product corresponds to a cascade product in which the dependency structure among components is acyclic, i.e., no loops are allowed between the dependencies of the automata. In [6], Domosi introduces the notion of v_i -product, where each semiautomaton is allowed to depend on at most i other semiautomata. The α_0 - v_1 product [10] combines these two restrictions: the dependency structure is acyclic and each component depends on at most one other component.

This construction is closely related to the Σ -chain product introduced in this paper. Indeed, both frameworks restrict the dependency structure among components to be essentially linear. In particular, every Σ -chain product defines an α_0 - v_1 product whose dependency graph is connected. Conversely, an α_0 - v_1 product may consist of several independent connected components; once an initial state and a set of final states are fixed, these components can be sequentially concatenated (in any order) to obtain a corresponding Σ -chain. In [10], it is shown that the α_0 - v_1 product is equivalent, in terms of homomorphic representations, to the general product, i.e., a product in which each component may depend on all the others. At first glance, this may seem to contradict the results of the present paper, since general products of reset automata can recognize languages that are not star-free, suggesting that Σ -chain of reset automata products might also exceed the expressive power of the star-free class. However, this apparent discrepancy is resolved by observing that the equivalence result of [10] relies on infinite products. Furthermore, the same work shows that, when restricting to finite products of finite automata, both the α_0 -product and the v_1 -product are strictly more expressive than the α_0 - v_1 product with respect to homomorphic representations. The expressiveness results for Σ -chain products we establish in this paper can thus be seen as an extension of the finite-product results of [10], providing a more refined understanding of the expressive power of finite α_0 - v_1 products.

3 Background

In this section, we introduce the necessary background on automata and cascades of automata.

3.1 Automata

A deterministic *semiautomaton* $\mathcal{A}$ is a tuple (Σ, Q, δ) such that: (i) Σ is the alphabet, i.e. a finite set of symbols; (ii) Q is the set of states; (iii) $\delta : Q \times \Sigma \rightarrow Q$ is the transition function. An deterministic *automaton* $\mathcal{A} = (\Sigma, Q, \delta, s_0, F)$ is a semiautomaton augmented

with an initial state s_0 and a set F of final states. From now on, the terms automaton and semiautomaton refer exclusively to deterministic ones. With δ^* we denote the Kleene closure of δ . The *size* of $\mathcal{A}$, denoted with $|\mathcal{A}|$, is defined as $|\delta|$.

Given an automaton $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ and a (finite) word $\sigma := \langle \sigma_0, \dots, \sigma_n \rangle \in \Sigma^*$, the run $\tau \in Q^+$ induced by σ is a sequence $\langle q_0, q_1, \dots, q_{n+1} \rangle$ starting from the initial state q_0 and such that $\delta(q_i, \sigma_i) = q_{i+1}$, for all $0 \leq i \leq n$. We say that τ is *accepting* iff $q_{n+1} \in F$. A word $\sigma \in \Sigma^*$ is *accepted* by $\mathcal{A}$ iff the run induced by σ is accepting. We define the *language of $\mathcal{A}$* (or *the language recognized by $\mathcal{A}$*), denoted by $\mathcal{L}(\mathcal{A})$, as the set of accepted words. Given a state $q \in Q$, let $\mathcal{L}_q(\mathcal{A})$ be the set of words inducing a run $\tau := \langle q_0, \dots, q_m \rangle$ with $q_m = q$.

Each symbol of the alphabet of an automaton $\mathcal{A}$ (with set of states Q) induces a transformation $Q \rightarrow Q$, defined as follows.

► **Definition 1** (Function induced by a symbol). *Let $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ be an automaton. For each symbol $a \in \Sigma$, we define the function induced by a in $\mathcal{A}$, denoted with τ_a^A or simply τ_a (when $\mathcal{A}$ is clear from the context), as the function $\tau_a : Q \rightarrow Q$ such that, for all $s \in Q$, we have that $\tau_a(s) = s'$ iff $\delta(s, a) = s'$.*

Let $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ be an automaton and let $a \in \Sigma$ be a symbol in the alphabet. We say that τ_a is a *reset function* (or simply a *reset*) iff there exists a $q' \in Q$ such that, for all $q \in Q$, we have that $\tau_a(q) = q'$; in this case, we say that τ_a is a *reset on q'* . If instead τ_a is bijective, we say that τ_a is a *permutation function* (or simply a *permutation*).

Reset automata and permutation automata are defined as follows.

► **Definition 2** (The classes of Pure-reset, Reset, Permutation, and Permutation-Reset automata). *Let $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ be an automaton. $\mathcal{A}$ is called:*

- a pure-reset automaton iff, for each $a \in \Sigma$, τ_a is a reset function;
- a reset automaton iff, for each symbol $a \in \Sigma$, τ_a is either the identity function or a reset function;
- a permutation automaton iff, for each symbol $a \in \Sigma$, τ_a is a permutation function.
- a permutation-reset automaton iff, for each symbol $a \in \Sigma$, τ_a is either a reset or a permutation function.

We denote with $\mathcal{R}^-$, $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{R}^+$ the classes of pure-reset, reset, permutation, and permutation-reset automata, respectively.

Let $A \in \{\mathcal{R}^-, \mathcal{R}, \mathcal{P}, \mathcal{R}^+\}$ be a class of automata. We denote with $\mathbb{DP}[A]$ the class of languages definable by a direct product of automata in the class A .

► **Proposition 3** (Size of direct products). *Let D be a set of automata $D = \{\mathcal{A}_1, \dots, \mathcal{A}_n\}$ such that $\mathcal{A}_i$ has alphabet Σ and set of states Q_i with $|Q_i| \geq 2$. Let $m = \max\{|Q_i| : 1 \leq i \leq n\}$. It holds that:*

- $|\mathcal{A}_1 \times \dots \times \mathcal{A}_n| \geq |\Sigma| \cdot 2^n$;
- $|D| = \sum_i |\mathcal{A}_i| \leq |\Sigma| \cdot n \cdot m$.

3.2 Cascades and the Cascade Product

The cascade product [14, 13] between two (semi)automata $\mathcal{A}$ and $\mathcal{B}$, is a generalization of the direct product in which $\mathcal{B}$ reads the Cartesian product of the input alphabet with the set of states of $\mathcal{A}$, and where the (semi)automaton $\mathcal{B}$ transitions from state s to state s' reading the symbol (a, q) if the current input symbol is a and the (semi)automaton $\mathcal{A}$ is in state q . The (semi)automaton obtained by applying the cascade product is defined over the Cartesian product of the set of states of $\mathcal{A}$ and $\mathcal{B}$ and the alphabet Σ , and is such that

transition function is applied component-wise. Additionally, for full automata [3], the initial state of the product automaton is the state comprising the initial states of $\mathcal{A}$ and $\mathcal{B}$ and the set of final states is the Cartesian product of the final states of $\mathcal{A}$ and $\mathcal{B}$.

► **Definition 4** (Cascade Product of automata). *Let Σ be a finite alphabet and let $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ and $\mathcal{B} = (\Sigma \times Q, Q', \delta', q'_0, F')$ be two automata over the alphabets Σ and $\Sigma \times Q$, respectively. The cascade product of $\mathcal{A}$ and $\mathcal{B}$, denoted by $\mathcal{A} \circ \mathcal{B}$, is the automaton $(\Sigma, Q \times Q', \delta'', (q_0, q'_0), F \times F')$ where δ'' is defined as:*

$$\delta''((q, q'), a) = (\delta(q, a), \delta'(q', (a, q)))$$

for all $(q, q') \in Q \times Q'$ and for all $a \in \Sigma$.

In the following, we will use the term *cascade* to denote a sequence of automata $\mathcal{C} = \langle \mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n \rangle$ to which the cascade product is applied, and we define the *size* of $\mathcal{C}$, denoted with $|\mathcal{C}|$, as $\sum_{i=1}^n |\mathcal{A}_i|$. Instead, with the term *cascade product*, we will denote the automaton $\mathcal{C} = \mathcal{A}_1 \circ \dots \circ \mathcal{A}_n$ obtained by applying the cascade product to the given sequence of automata. We define the *height* of the cascade $\mathcal{C} = \langle \mathcal{A}_1, \dots, \mathcal{A}_n \rangle$ as n . Given a cascade $\mathcal{C} = \langle \mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n \rangle$, we define the *language* of $\mathcal{C}$, denoted with $\mathcal{L}(\mathcal{C})$, as the language $\mathcal{L}(\mathcal{A}_1 \circ \dots \circ \mathcal{A}_n)$.

The proposition below shows that, contrary to the case of the direct product (cf. Proposition 3), representing a cascade (i.e. the *sequence* of components to which the cascade product is applied) requires no less space than representing the product automaton (i.e. the automaton obtained after applying the cascade product), which is exponential with respect to the number of components. This is due to the fact that, for cascades of automata, the alphabet of each component is the Cartesian product between Σ and the set of states of all previous components. In particular, in a cascade $\langle \mathcal{A}_1, \dots, \mathcal{A}_n \rangle$, the alphabet of the last component $\mathcal{A}_n$ has cardinality $|\Sigma| \times |Q_1| \times \dots \times |Q_{n-1}|$, which is exponential in n .

► **Proposition 5** (Size of cascades [4]). *For each cascade $\mathcal{C} = \langle \mathcal{A}_1, \dots, \mathcal{A}_n \rangle$, where Q_i is the set of states of $\mathcal{A}_i$, for each $1 \leq i \leq n$, it holds that:*

- $|\mathcal{A}_1 \circ \dots \circ \mathcal{A}_n| \geq |\Sigma| \cdot 2^n$;
- $|\mathcal{C}| \geq 2 \cdot |\Sigma| \cdot (2^n - 1)$.

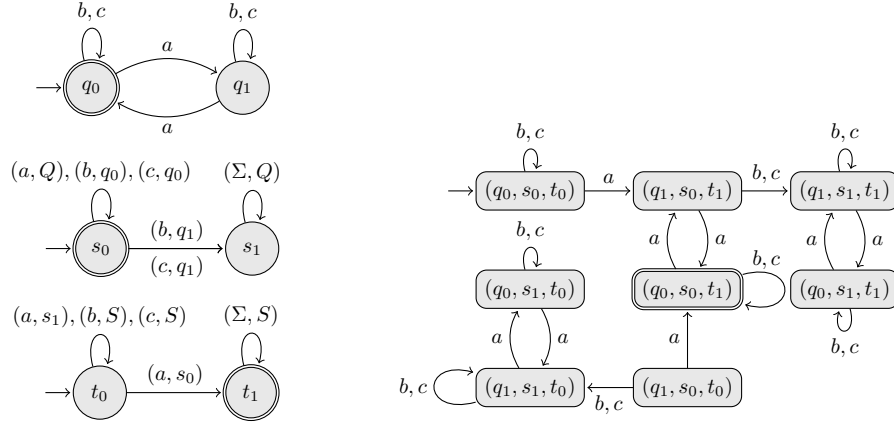
Let $A \in \{\mathcal{R}^-, \mathcal{R}, \mathcal{P}, \mathcal{R}^+\}$ be a class of automata. We denote with $\mathbb{C}[A]$ the class of languages recognized by cascades of automata of class A . A language $\mathcal{L} \subseteq \Sigma^*$ is regular iff it can be defined starting from finite subsets of Σ^* by the applications of union, negation, concatenation, and closure under Kleene star. $\mathcal{L}$ is *star-free* iff it can be generated starting from finite subset of Σ^* only by the application of union, negation, and concatenation. The following results characterizes the expressiveness of $\mathbb{C}[\mathcal{R}]$ and $\mathbb{C}[\mathcal{R}^+]$.

► **Theorem 6** ([4]). *The following characterizations hold:*

- A language $\mathcal{L}$ is regular if and only if $\mathcal{L} \in \mathbb{C}[\mathcal{R}^+]$;
- A language $\mathcal{L}$ is star-free if and only if $\mathcal{L} \in \mathbb{C}[\mathcal{R}]$.

4 Σ -Chains of automata

In this section we define Σ -chains of automata and the associated Σ -chain product. We then study their main properties in terms of succinctness and expressive power.



■ **Figure 1** On the left, starting from the top, the three components $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ of a Σ -chain $\mathcal{C}$, with $\Sigma = \{a, b, c\}$. On the right, the associated Σ -product. We denoted final states with double circles and with (a, Q) we denote the set of symbols $\{(a, q) \mid q \in Q\}$.

4.1 Definitions, Basic Properties and Succinctness

We define Σ -chains of automata as a restricted variant of cascades of automata, in which the alphabet of the first component is Σ (as in the case of cascades), while the alphabet of each successive component $\mathcal{A}_i$ is the Cartesian product between Σ and the set of states of the previous component $\mathcal{A}_{i-1}$.

► **Definition 7** (Σ -chain Product of automata). *Let Σ be an alphabet and let $\mathcal{A}_1, \dots, \mathcal{A}_n$ be n automata such that:*

- $\mathcal{A}_1 = \langle \Sigma, Q^1, \delta^1, q_0^1, F^1 \rangle$;
- $\mathcal{A}_i = \langle \Sigma \times Q^{i-1}, Q^i, \delta^i, q_0^i, F^i \rangle$, for each $i > 1$.

We define the Σ -chain product between $\mathcal{A}_1, \dots, \mathcal{A}_n$, denoted with $\mathcal{A}_1 \odot \dots \odot \mathcal{A}_n$, as the automaton $(\Sigma, Q^1 \times \dots \times Q^n, \delta, (q_0^1, \dots, q_0^n), F^1 \times \dots \times F^n)$ such that:

$$\delta((q^1, q^2, \dots, q^n), a) = (\delta^1(q^1, a), \delta^2(q^2, (a, q^1)), \dots, \delta^n(q^n, (a, q^{n-1})))$$

for all $(q^1, q^2, \dots, q^n) \in Q^1 \times Q^2 \times \dots \times Q^n$ and for all $a \in \Sigma$.

In the following, we will use the term Σ -chain to denote a sequence of automata to which the Σ -chain product is applied. To distinguish them from cascades, we will denote this sequence as $\mathcal{C} = [\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n]$. As with cascades, we define the *size* of the Σ -chain $\mathcal{C} = [\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n]$, denoted with $|\mathcal{C}|$, as $\sum_{i=1}^n |\mathcal{A}_i|$, and its *height* as n .

Given a Σ -chain $\mathcal{C} = [\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n]$, we define the *language of $\mathcal{C}$* , denoted with $\mathcal{L}(\mathcal{C})$, as the language $\mathcal{L}(\mathcal{A}_1 \odot \dots \odot \mathcal{A}_n)$. From now on, given a cascade (or a Σ -chain) $\mathcal{C}$ and a Σ -chain (or a cascade) $\mathcal{C}'$, we say that $\mathcal{C}$ is *equivalent* to $\mathcal{C}'$ iff $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{C}')$.

An example of a Σ -chain and of the corresponding Σ -chain product is shown in Figure 1. The three automata on the left constitute the components of the Σ -chain $\mathcal{C} = [\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3]$. Automaton $\mathcal{A}_1$ is a permutation-reset automaton, whereas $\mathcal{A}_2$ and $\mathcal{A}_3$ are reset automata. The language recognized by $\mathcal{C}$ consists of all words over the alphabet $\Sigma = \{a, b, c\}$ satisfying the following conditions: (i) the number of occurrences of the symbol “a” is even; (ii) for every i^{th} occurrence of “a” with odd index i , the immediately following symbol must also be “a”; and (iii) there is at least one occurrence of “a”. Each component of the Σ -chain enforces one of these conditions. In particular, since τ_a is a permutation in $\mathcal{A}_1$ and q_0 is its only

accepting state, $\mathcal{A}_1$ accepts precisely those inputs containing an even number of occurrences of “a”. The automaton $\mathcal{A}_2$ transitions to the sink state s_1 upon reading either (b, q_1) or (c, q_1) . This implies that if the most recently read “a” corresponds to an odd occurrence (i.e. $\mathcal{A}_1$ is in state q_1) and the current symbol is b or c , then the input word—and all its continuations—must be rejected. The automaton obtained by applying the Σ -chain product to $\mathcal{C}$ is depicted on the right-hand side of Figure 1.

Let $A \in \{\mathcal{R}^-, \mathcal{R}, \mathcal{P}, \mathcal{R}^+\}$ be a class of automata. We denote by $\text{SCH}[A]$ the class of languages recognized by Σ -chains whose components are automata in the class A . The following proposition shows that Σ -chains are a particular case of cascades. In particular, each Σ -chain of automata in a class $A \in \{\mathcal{R}^-, \mathcal{R}, \mathcal{P}, \mathcal{R}^+\}$ can be converted to an equivalent cascade of automata of the same class A with the same height and with the same number of states in each component.

► **Proposition 8.** *Let $\mathcal{C} = [\mathcal{A}_1, \dots, \mathcal{A}_n]$ be a Σ -chain where, for every $i = 1, \dots, n$, $\mathcal{A}_i$ is of class $A_i \in \{\mathcal{R}^-, \mathcal{R}, \mathcal{P}, \mathcal{R}^+\}$ and has m_i states. Then, there exists a cascade $\mathcal{C}' = \langle \mathcal{A}'_1, \dots, \mathcal{A}'_n \rangle$ such that $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{C}')$, where each $\mathcal{A}'_i$ is of class A_i and has m_i states (for every $i = 1, \dots, n$).*

From Proposition 8 and Theorem 6, it directly follows that the language of any Σ -chain of reset automata is a star-free language and that the language of any Σ -chain of permutation-reset automata is a regular language.

We analyze the size of Σ -chains with respect to the number of its components and we compare it to the size of its associated Σ -chain product. The next proposition shows that, as for direct products (cf. Proposition 3), the size of a Σ -chain is always linear in the number of its components, while the size of its associated Σ -chain product is exponential (in the number of components). This is a key difference with respect to the case of cascades, in which *both* the size of a cascade and the size of its cascade product are exponential in the number of the components (Proposition 5).

► **Proposition 9** (Size of Σ -chains and Σ -chain products). *Let $\mathcal{C} = [\mathcal{A}_1, \dots, \mathcal{A}_n]$ be a Σ -chain where, for every $i = 1, \dots, n$, Q_i is the set of states of $\mathcal{A}_i$, and let $m = \max\{|Q_i| : 1 \leq i \leq n\}$. The following statements hold:*

- $|\mathcal{A}_1 \odot \dots \odot \mathcal{A}_n| \geq |\Sigma| \cdot 2^n$;
- $|\mathcal{C}| \leq |\Sigma| \cdot n \cdot m^2$.

The preceding proposition raises the following question: *do there exist languages that admit a compact representation via Σ -chains but not via cascades?* The next theorem answers this positively for the class of reset automata, by exhibiting a family of languages $\{\mathcal{L}_n\}_{n=1}^\infty$ witnessing an exponential succinctness gap. Each $\mathcal{L}_n$ can be recognized by a Σ -chain of n reset automata of polynomial size (cf. Proposition 9), while any cascade of reset automata recognizing $\mathcal{L}_n$ must have at least n components, and therefore exponential size (cf. Proposition 5).

► **Theorem 10** (Succinctness of Σ -chains). *Let Σ be a finite alphabet. For every $h > 0$, let $\mathcal{L}_h = \Sigma^h \cdot \Sigma^*$. The following holds:*

- the smallest cascade of reset automata $\mathcal{C}_h$ recognizing $\mathcal{L}_h$ has size $|\mathcal{C}_h| \in \Omega(2^h \cdot |\Sigma|)$;
- there exists a Σ -chain of two-state reset automata $\mathcal{C}'_h$ recognizing $\mathcal{L}_h$ and of size $|\mathcal{C}'_h| \in O(h \cdot |\Sigma|)$.

Proof (sketch). It is possible to construct a Σ -chain $\mathcal{C}'_h = [\mathcal{A}_1, \dots, \mathcal{A}_h]$ recognizing $\mathcal{L}_h$ as follows (for every $1 \leq i \leq h$, $\mathcal{A}_i$ has only two states, the initial state q_0^i and the only final state q_1^i): (1) in $\mathcal{A}_1$, each symbol in Σ induces a reset on q_1^1 ; (2) for every $1 \leq i \leq h$, $\mathcal{A}_i$

transitions from its initial state to its final state only when the preceding component is in its final state (regardless of the symbol read from Σ), *i.e.* each symbol (a, q_1^{i-1}) induces a reset on q_1^i and all the others induce resets on q_0^i . This ensures that the automaton $\mathcal{A}_1 \odot \dots \odot \mathcal{A}_h$ reaches its final state $(q_1^1, \dots, q_1^h)$ only after consuming at least h symbols from Σ . The lower bound on the size of the smallest cascade for $\mathcal{L}_h$ is proved by Theorem 5.8 of [4]. ◀

Theorem 10 shows that, for certain languages, the use of Σ -chains can yield representations that are exponentially more succinct than those based on cascades. This motivates the study of the expressive power of Σ -chains, that is, the characterization of the languages recognizable via Σ -chains over a given class A of automata.

4.2 Expressiveness

We now investigate the expressive power of Σ -chains, focusing in particular on the case where the components are automata drawn from a fixed class $A \in \{\mathcal{R}^-, \mathcal{R}, \mathcal{P}, \mathcal{R}^+\}$ (*i.e.* pure-reset, reset, permutation, and permutation-reset automata).

First, we compare the Σ -chain product with the direct product, showing that the former is strictly more expressive than the latter.

Then, we prove that Σ -chains of permutation-reset automata capture exactly the class of regular languages. The result is established via a *chainification method*, namely, a construction that transforms any cascade of permutation-reset automata into an equivalent Σ -chain of permutation-reset automata. The expressive equivalence with regular languages then follows from the known equivalence between regular languages and cascades of permutation-reset automata (*cf.* Theorem 6).

We further show that the chainification method extends to the class $\mathcal{R}^-$, yielding that Σ -chains of pure-reset automata are expressively equivalent to cascades of pure-reset automata.

For the class $\mathcal{R}$ of reset automata, we show that Σ -chains of reset automata recognize only star-free languages. Whether the converse holds remains open. The natural proof strategy, namely applying chainification to cascades of reset automata, which are known to recognize all star-free languages (*cf.* Theorem 6), fails: chainification produces Σ -chains of permutation-reset automata rather than reset automata.

Comparison with the direct product

As already discussed in the introduction, the Σ -chain product can be viewed both as a restricted variant of the cascade product and as a generalization of the direct product. In fact, Σ -chains retain a key feature of direct products, namely their linear size with respect to the number of components (*cf.* Proposition 9). This naturally raises the question of their relative expressive power. We show that any direct product can be simulated by a Σ -chain product ($\text{DP}[A] \subseteq \text{SCH}[A]$, for each $A \in \{\mathcal{R}^-, \mathcal{R}, \mathcal{P}, \mathcal{R}^+\}$) and that the opposite is *not* possible: we exhibit two simple languages that cannot be recognized by any direct product of reset (resp. pure-reset) automata, while they are definable by a Σ -chain of reset (resp. pure-reset) automata.

► **Lemma 11.** *The following inclusions hold:*

- $\text{DP}[A] \subseteq \text{SCH}[A]$, for each $A \in \{\mathcal{R}^-, \mathcal{R}, \mathcal{P}, \mathcal{R}^+\}$;
- $\text{DP}[\mathcal{R}] \subsetneq \text{SCH}[\mathcal{R}]$ and $\text{DP}[\mathcal{R}^-] \subsetneq \text{SCH}[\mathcal{R}^-]$.

In particular, the strict inclusions are witnessed by the languages $\mathcal{L}_1 = \Sigma^ a \Sigma^* a \Sigma^*$ and $\mathcal{L}_2 = \Sigma^* a a$, over $\Sigma = \{a, b\}$, as we have that $\mathcal{L}_1 \in \text{SCH}[\mathcal{R}] \setminus \text{DP}[\mathcal{R}]$ and $\mathcal{L}_2 \in \text{SCH}[\mathcal{R}^-] \setminus \text{DP}[\mathcal{R}^-]$.*

Transformation of cascades into Σ -chains (Chainification)

We describe the transformation of cascades into equivalent Σ -chains starting from the simplest case, that is, the case of cascades of three automata, since every cascade of height two is trivially a Σ -chain.

Let $\mathcal{C} = \langle \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3 \rangle$ be a cascade of three automata: we shall construct an equivalent Σ -chain $\mathcal{C}' = [\mathcal{A}'_1, \mathcal{A}'_2, \mathcal{A}'_3]$ such that $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{C}')$ and if $\mathcal{A}_i$ is a permutation-reset (resp., permutation, pure-reset) automaton, then also $\mathcal{A}'_i$ is a permutation-reset (resp., permutation, pure-reset) automaton, for each $1 \leq i \leq 3$.

The main challenge in this translation lies in handling the dependence of the third component $\mathcal{A}_3$ on the first component $\mathcal{A}_1$, which is not permitted in a Σ -chain. To eliminate this dependence, we encode the required information into the states of the second component $\mathcal{A}'_2$. The construction proceeds as follows: (i) the second component $\mathcal{A}'_2$ is defined so as to combine the state information of $\mathcal{A}_1$ and $\mathcal{A}_2$; (ii) the third component $\mathcal{A}'_3$ then reads the state of $\mathcal{A}'_2$, from which it can recover the information about both $\mathcal{A}_1$ and $\mathcal{A}_2$. As stated by the following Lemma, if the second component $\mathcal{A}_2$ is permutation-reset (resp. permutation, pure-reset), the new component $\mathcal{A}'_2$ can be defined to be permutation-reset (resp. permutation, pure-reset).

► **Lemma 12.** *Let $\mathcal{C} = \langle \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3 \rangle$ be a cascade of permutation-reset (resp. permutation, pure-reset) automata. There exists a Σ -chain $\mathcal{C}' = [\mathcal{A}'_1, \mathcal{A}'_2, \mathcal{A}'_3]$ of permutation-reset (resp. permutation, pure-reset) automata such that $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{C}')$.*

Proof (sketch). Define $\mathcal{A}'_1$ as $\mathcal{A}_1$. To ensure that $\mathcal{A}'_2$ carries all the information needed by $\mathcal{A}'_3$, which reads only symbols in Σ and states of $\mathcal{A}'_2$, we define the state set S' of $\mathcal{A}'_2$ as the Cartesian product of the state sets of $\mathcal{A}_1$ and $\mathcal{A}_2$. Furthermore, $\mathcal{A}'_2$ must satisfy the following condition: for every state $(q, s) \in S'$ and every transition labeled (a, q) , where q matches the first component of the current state, the transition leads to a state (q', s') such that $\delta_1(q, a) = q'$ and $\delta_2(s, (a, q)) = s'$, i.e.

$$\forall (a, q) \in \Sigma \times Q. (\delta'_2((q, s), (a, q)) = (q', s') \Leftrightarrow \delta_1(q, a) = q' \wedge \delta_2(s, (a, q)) = s')$$

Moreover, it is possible to extend the definition of δ'_2 on all other input pairs (i.e. the pairs (a, q'') with $q'' \neq q$) in order to guarantee that $\tau_{(a, q)}^{\mathcal{A}'_2}$ is a permutation (when $\tau_{(a, q)}^{\mathcal{A}_2}$ is a permutation) or a reset (when $\tau_{(a, q)}^{\mathcal{A}_2}$ is a reset). The automaton $\mathcal{A}'_3$ is defined as $\mathcal{A}_3$ by transforming each transition $\tau_{(a, q, s)}$ into $\tau_{(a, (q, s))}$. Crucially, all states of $\mathcal{A}'_1 \odot \mathcal{A}'_2 \odot \mathcal{A}'_3$ of type $(q, (q', s), t)$ with $q \neq q'$ are not reachable from the initial state, having that the language of $\mathcal{A}'_1 \odot \mathcal{A}'_2 \odot \mathcal{A}'_3$ is equal to the language of $\mathcal{A}_1 \odot \mathcal{A}_2 \odot \mathcal{A}_3$. ◀

Figure 2 illustrates an example of chainification for a cascade $\langle \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3 \rangle$ over the alphabet $\Sigma = \{a, b\}$ (left side of Figure 2). The components $\mathcal{A}'_2$ and $\mathcal{A}'_3$ of the equivalent Σ -chain are shown on the right side of Figure 2. It is important to note that all transitions (a, q) in $\mathcal{A}'_2$ can be defined as either permutations or reset functions. Furthermore, we observe that symbols acting as identities in $\mathcal{A}_2$ may become non-identity permutations in $\mathcal{A}'_2$; this is the case, for instance, for the symbol (a, q_0) . Indeed, while $\mathcal{A}_2$ is a reset automaton, $\mathcal{A}'_2$ is a permutation-reset automaton.

Lemma 12 can be generalized to deal with cascades of unbounded height as stated by the following theorem.

► **Theorem 13 (Chainification Theorem).** *Let $\mathcal{C} = \langle \mathcal{A}_1, \dots, \mathcal{A}_n \rangle$ be a cascade of permutation-reset (resp. permutation, pure-reset) automata. There exists a Σ -chain $\mathcal{C}' = [\mathcal{A}'_1, \dots, \mathcal{A}'_n]$ of permutation-reset (resp. permutation, pure-reset) automata such that $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{C}')$ and $|\mathcal{C}'| \in \mathcal{O}(|\mathcal{C}|)$.*

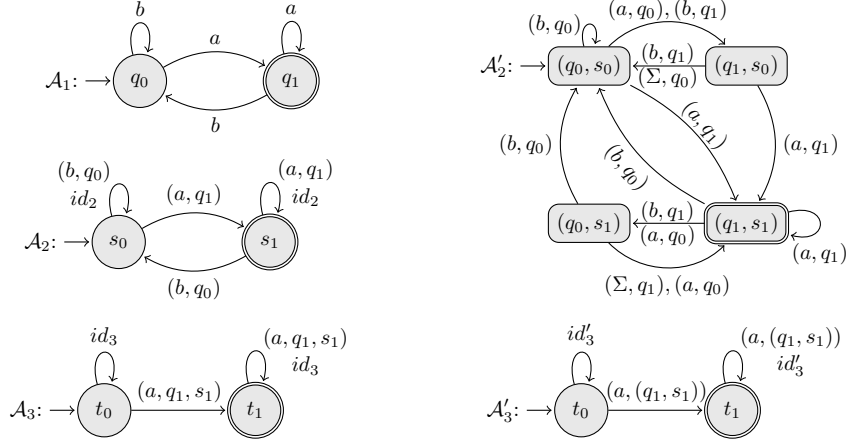


Figure 2 On the left, the cascade $\langle \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3 \rangle$ over $\Sigma = \{a, b\}$, where Q, S, T are the sets of states of $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$, respectively, $id_2 = \{(a, q_0), (b, q_1)\}$ and $id_3 = (\Sigma \times Q \times S) \setminus \{(a, q_1, s_1)\}$. On the right, the automata $\mathcal{A}'_2$ (with set of states S') and $\mathcal{A}'_3$ of the Σ -chain obtained through the chainification technique of Lemma 12, where $id'_3 = (Q \times S') \setminus \{(a, (q_1, s_1))\}$. The cascade on the left is equivalent to the Σ -chain $[\mathcal{A}_1, \mathcal{A}'_2, \mathcal{A}'_3]$.

Implications on the expressive power of Σ -chains

Proposition 8 and Theorem 13 imply that Σ -chains are expressively equivalent to cascades for the class $\mathcal{R}^-$ of pure-reset automata, for the class $\mathcal{P}$ of permutation automata and for the class $\mathcal{R}^+$ of permutation-reset automata. Moreover, Σ -chains of permutation-reset automata characterize the class of regular languages (cf. Theorem 6).

As for the class $\mathcal{R}$, we have that if a language $\mathcal{L}$ is recognized by a Σ -chain of reset automata then $\mathcal{L}$ is also recognized by a cascade of reset automata (Proposition 8) and $\mathcal{L}$ is star-free (Theorem 6).

Corollary 14. *It holds that:*

- $\text{SCH}[\mathcal{R}^+] = \mathbb{C}[\mathcal{R}^+]$ and thus a language $\mathcal{L}$ is regular if and only if $\mathcal{L} \in \text{SCH}[\mathcal{R}^+]$;
- $\text{SCH}[\mathcal{R}^-] = \mathbb{C}[\mathcal{R}^-]$;
- $\text{SCH}[\mathcal{R}] \subseteq \mathbb{C}[\mathcal{R}]$ and thus if $\mathcal{L} \in \text{SCH}[\mathcal{R}]$ then $\mathcal{L}$ is star-free;
- $\text{SCH}[\mathcal{P}] = \mathbb{C}[\mathcal{P}]$.

5 Σ -Chains of Reset Automata

In this section, we investigate Σ -chains of reset automata and the languages they recognize ($\text{SCH}[\mathcal{R}]$). First, we analyze how imposing structural constraints on Σ -chains – specifically, bounding their *height* (i.e. the number of components) and *width* (i.e. the number of states per component) – affects the class of recognizable languages.

Second, we compare Σ -chains of reset automata with Piecewise Testable [16] and R-trivial languages [5], showing that every language in these classes can be expressed as a Boolean combination and as a finite union of Σ -chains of reset automata, respectively. Moreover, we establish linear upper bounds on the size of such Σ -chain representations.

Finally, we address the *pastification* problem for Σ -chains of reset automata, namely, the transformation of these structures into equivalent formulas in pure past Linear Temporal Logic. We show a direct transformation yielding a formula of quadratic size when represented as a DAG (cf. dag-complexity [4]).

5.1 Height and Width Hierarchies

First, we show that increasing the height of a Σ -chain of reset automata (*i.e.* the number of its components) yields an infinite hierarchy of languages: for every level h , there exists a language at level h that is definable by a Σ -chain of h reset automata but is not recognizable by any Σ -chain of height $h - 1$. This result is analogous to what is known for cascades of reset automata (*cf.* Height-Hierarchy Lemma of [4]).

- **Proposition 15.** *Let Σ be a finite alphabet. For each $h > 0$, let $\mathcal{L}_h = \Sigma^h \cdot \Sigma^*$. It holds that:*
- $\mathcal{L}_h$ is definable by a Σ -chain of (two-state) reset automata $\mathcal{C}_h$ of height h ;
 - $\mathcal{L}_h$ cannot be defined by any Σ -chain of reset automata of height less than h .

As shown by Proposition 15 and by the Height-Hierarchy Lemma of [4], both Σ -chains and cascades of reset automata admit a height hierarchy: for every $h \geq 2$ there exist languages definable by a product of height h that cannot be defined by any product of smaller height. In contrast, such a hierarchy collapses for direct products, as formalized by the following lemma.

- **Lemma 16.** *Let $\Sigma = \{a, b\}$. Every language definable by a direct product of reset automata over Σ is also definable by a direct product of two reset automata over Σ .*

We now investigate the effect of increasing the number of states in each component of a Σ -chain of reset automata. The following proposition shows that this does not yield an infinite hierarchy of languages. In particular, every Σ -chain of reset automata is equivalent to a Σ -chain of the same height in which the number of states in each component depends only on the size of the alphabet Σ .

- **Proposition 17.** *Let Σ be a finite alphabet with $|\Sigma| = k \geq 2$ and let $\mathcal{L} \subseteq \Sigma^*$ be a language accepted by a Σ -chain of reset automata $\mathcal{C} = [\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_h]$, for some $h > 0$. It holds that $\mathcal{L}$ is accepted by some Σ -chain of reset automata $\mathcal{C}' = [\mathcal{A}'_1, \mathcal{A}'_2, \dots, \mathcal{A}'_h]$ where $\mathcal{A}'_i$ has at most $f(i) = \frac{k^{i+1}-1}{k-1}$ states (for $i = 1, \dots, h$).*

The $f(i) = \frac{k^{i+1}-1}{k-1}$ bound on the number of states in the i -th component of a Σ -chain, established in the previous proposition, is – owing to the restriction that each component depends only on its immediate predecessor – significantly smaller than the corresponding bound for cascades of reset automata (*cf.* Width-Collapse Lemma in [4]), namely $g^{i-1}(k+1)$ with $g(x) = x^2 - x + 1$.

5.2 Comparison with Piecewise Testable Languages

Piecewise testable languages were introduced by Simon [16] as a natural class of formal languages whose membership can be decided by inspecting only the scattered subsequences of bounded length occurring in a given word, regardless of the positions at which they appear. Intuitively, two words are indistinguishable with respect to a piecewise testable language whenever they share the same set of scattered subsequences up to a fixed length. This class has since attracted considerable attention in the theory of formal languages, owing in part to Simon's elegant algebraic characterization, which identifies piecewise testable languages with precisely those regular languages whose syntactic monoid is $\mathcal{J}$ -trivial [16], and in part to their natural logical characterization as exactly the languages definable by Boolean combinations of existential first-order sentences over words [17].

► **Definition 18** (Piecewise Testable Languages [16, 15]). Let Σ be a finite alphabet and let $\sigma = \langle a_1, \dots, a_n \rangle \in \Sigma^n$ be a word of length n , for some $n > 0$. We define $\mathcal{L}_\sigma \subseteq \Sigma^*$ as the language $\mathcal{L}_\sigma = \Sigma^* a_1 \Sigma^* \dots \Sigma^* a_n \Sigma^*$. A language $\mathcal{L}$ is k -piecewise testable if $\mathcal{L}$ is a Boolean combination of languages $\mathcal{L}_{\sigma_i}$ with $\sigma_i \in \Sigma^{\leq k}$. A language $\mathcal{L}$ is piecewise testable if it is k -piecewise testable for some $k > 0$.

To show that k -piecewise testable languages are recognizable by Boolean combinations of (languages recognized by) Σ -chains of k reset automata, we first establish a generalization of the fact that, for every $n > 0$ and every $\sigma \in \Sigma^+$, there exists a Σ -chain of n reset automata recognizing $\mathcal{L}_\sigma$. In particular, the following lemma shows that Σ -chains of reset automata can recognize languages of the form $\Sigma^* \sigma_1 \Sigma^* \dots \Sigma^* \sigma_n \Sigma^*$, where each σ_i is not restricted to a single symbol of the alphabet Σ (as in Definition 18), but may be an arbitrary word. This is achieved using a number of components equal to the sum of the lengths of all the σ_i .

► **Lemma 19.** Let Σ be a finite alphabet and let $\sigma_1, \dots, \sigma_m$ be m words in Σ^+ , for some $m > 0$. Let $\mathcal{L}_{\sigma_1, \dots, \sigma_m}$ be the language $\Sigma^* \sigma_1 \Sigma^* \dots \Sigma^* \sigma_m \Sigma^*$. There exists a Σ -chain $\mathcal{C}$ of (two-states) reset automata such that:

- $\mathcal{L}(\mathcal{C}) = \mathcal{L}_{\sigma_1, \dots, \sigma_m}$;
- the height of $\mathcal{C}$ is $|\sigma_1| + \dots + |\sigma_m|$.

Proof (sketch). We first construct, for each $1 \leq i \leq m$, a Σ -chain for the language $\Sigma^* \cdot \sigma_i \cdot \Sigma^*$. Let $\sigma_i = a_1 a_2 \dots a_n$. The Σ -chain consists of n two-state reset automata: for each $1 \leq j \leq n$, the component $\mathcal{A}_j$ transitions from its initial to its noninitial state (by means of a reset function) if and only if the current symbol is a_j and the previous component (if any) is in its noninitial state. Moreover, the last component ($\mathcal{A}_n$) must remain in its current state (by means of an identity function) for all combinations of symbols in $\Sigma \setminus \{a_n\}$ and states of $\mathcal{A}_{n-1}$, in order to ensure that the last Σ^* in $\Sigma^* \cdot \sigma_i \cdot \Sigma^*$. Finally, we set the final states of $\mathcal{A}_j$ to be the singleton containing only its noninitial state if $j = n$, otherwise we define it as the set of all the two states of $\mathcal{A}_j$.

We can then compose the Σ -chains for each σ_i obtained in this fashion to obtain a Σ -chain of $\sum_{i=1}^m |\sigma_i|$ components recognizing $\Sigma^* \sigma_1 \Sigma^* \dots \Sigma^* \sigma_m \Sigma^*$. ◀

As a straightforward corollary of Lemma 19, applied to the case where $|\sigma_i| = 1$ for every $1 \leq i \leq n$, we obtain the following result.

► **Corollary 20.** Let Σ be a finite alphabet and let $\mathcal{L} \subseteq \Sigma^*$. If $\mathcal{L}$ is a k -piecewise testable language, then $\mathcal{L}$ is a Boolean combination of languages, each of which is definable by a Σ -chain of (two-states) reset automata of height at most k .

5.3 Comparison with R-trivial Languages

R-trivial languages are those regular languages whose syntactic monoid is $\mathcal{R}$ -trivial and are dual to $\mathcal{L}$ -trivial languages [5]. Beyond their algebraic characterizations, R-trivial languages are deeply connected to automata theory. In particular, it holds that a language $\mathcal{L}$ is R-trivial if and only if the minimum automaton accepting $\mathcal{L}$ is partially ordered [5]. Surprisingly, there is also a characterization in terms of the cascade product. A language $\mathcal{L}$ is R-trivial if and only if the minimum automaton accepting $\mathcal{L}$ is covered by a cascade product of half-resets where an half-reset is a two-state semiautomaton $\mathcal{A} = \langle \Sigma, \{q_0, q_1\}, \delta \rangle$ such that Σ can be partitioned in two non-empty sets B_0 and B_1 such that $\delta(q_0, a_0) = q_0$ for every $a_0 \in B_0$, $\delta(q_0, a_1) = q_1$ for every $a_1 \in B_1$ and $\delta(a, q_1) = q_1$ for every $a \in \Sigma$ [5].

As with piecewise-testable languages, we start by defining R-trivial languages in terms of regular expressions.

► **Definition 21** (R-trivial Languages [5, 15]). Let Σ be a finite alphabet. A language $\mathcal{L} \subseteq \Sigma^*$ is R-trivial if $\mathcal{L}$ is a disjoint union of languages of the form $I_0^* a_1 I_1^* a_2 I_2^* \dots a_n I_n^*$ where: (i) $n \geq 0$; (ii) $a_i \in \Sigma$, for each $1 \leq i \leq n$; (iii) $I_i \subseteq \Sigma \setminus \{a_{i+1}\}$, for each $0 \leq i < n$; and (iv) $I_n \subseteq \Sigma$.

To show that R-trivial languages are recognizable by finite unions of (languages recognized by) Σ -chains of reset automata, we first establish a more general result. In particular, we show that Σ -chains of reset automata can recognize languages of the form $I_0^* B_1 I_1^* B_2 \dots B_n I_n^*$ where each I_i and each B_j is a subset of Σ without further restrictions. Moreover, the obtained Σ -chain is of height $n + 1$.

► **Lemma 22.** Let Σ be a finite alphabet. For some $m > 0$, let $I_0, \dots, I_m, B_1, \dots, B_m$ be subsets of Σ . Let $\mathcal{L}_{B_1, \dots, B_m}^{I_0, \dots, I_m} = I_0^* B_1 I_1^* \dots I_{m-1}^* B_m I_m^*$. There exists a Σ -chain of two-states reset automata $\mathcal{C}$ such that:

- $\mathcal{L}(\mathcal{C}) = \mathcal{L}_{B_1, \dots, B_m}^{I_0, \dots, I_m}$;
- $\mathcal{C}$ has height $m + 1$.

Proof (sketch). The proof follows an approach similar to the proof of Lemma 19. ◀

Applying Lemma 22 to the case where $|B_i| = 1$ (for every $1 \leq i \leq n$) and $I_i \subseteq \Sigma \setminus \{a_{i+1}\}$ (for $0 \leq i \leq n - 1$) leads to the following result.

► **Corollary 23.** Let Σ be a finite alphabet. Let $\mathcal{L} \subseteq \Sigma^*$ be an R-trivial language. $\mathcal{L}$ is recognized by a finite union of Σ -chains of two-states reset automata.

5.4 Σ -Chains of reset automata and ppLTL

In [14], Maler proposed a procedure to compile cascades of reset semi-automata into equivalent formulae of ppLTL, namely formulae of LTL with only past temporal operators. We call such a procedure a *pastification* procedure. In [4], this pastification procedure was applied to cascades of reset automata, providing bounds on the complexity of the resulting formula $\text{past}(\mathcal{C})$ (where $\mathcal{C}$ is a cascade of reset automata) under two measures:

- The tree-complexity (denoted $|\text{past}(\mathcal{C})|_t$): the size of the formula $\text{past}(\mathcal{C})$;
- The dag-complexity (denoted $|\text{past}(\mathcal{C})|_d$): the number of syntactically distinct sub-formulas of $\text{past}(\mathcal{C})$. The dag-complexity measures the space needed to represent the formula using the sub-term sharing technique.

Theorem 8.4 of [4] provides bounds on both measures: while the tree-complexity of the obtained ppLTL formula may be exponential in the size of $\mathcal{C}$, the dag-complexity has only a linear bound.

Given a Σ -chain $\mathcal{C}$ of n reset automata, Proposition 8 allows us to first transform it into an equivalent cascade $\mathcal{C}'$ of n reset automata, whose size may however be exponential in n . Applying Theorem 8.4 of [4] then yields an equivalent ppLTL formula, but whose dag-complexity is exponential in n . A direct translation, however, achieves a significantly better bound. Indeed, a straightforward extension of the techniques of [4] yields a transformation from a Σ -chain of n reset automata into an equivalent ppLTL formula with only quadratic dag-complexity.

► **Theorem 24.** Let $\mathcal{AP}$ be a set of atomic propositions and let $\mathcal{C} = [\mathcal{A}_1, \dots, \mathcal{A}_n]$ be a Σ -chain of reset automata over $\Sigma = 2^{\mathcal{AP}}$. There exists a ppLTL formula $\text{past}(\mathcal{C})$ such that:

- $\mathcal{L}(\text{past}(\mathcal{C})) = \mathcal{L}(\mathcal{C}) \setminus \{\epsilon\}$;
- $|\text{past}(\mathcal{C})|_d \in \mathcal{O}(|\mathcal{C}| \cdot (|\mathcal{AP}| + n)) \in \mathcal{O}(|\mathcal{C}| \cdot (|\mathcal{AP}| + |\mathcal{C}|))$.

By Theorem 10 and Theorem 24, the language $\Sigma^h \Sigma^*$ admits a pastification of dag-complexity $\mathcal{O}(h \cdot |\Sigma| \cdot (|\mathcal{AP}| + h))$ after a prior transformation into a Σ -chain of reset automata. In contrast, translating the same language into a cascade of reset automata yields a ppLTL formula of dag-complexity $\mathcal{O}(2^h \cdot |\Sigma| \cdot |\mathcal{AP}|)$. Therefore, Σ -chains can produce exponentially more succinct pastifications than cascades of reset automata.

6 Conclusions and Future Work

In this paper, we have introduced the notion of Σ -chain and the associated Σ -chain product operation, in which the component automata are combined in a feed-forward fashion, with each component depending solely on the initial alphabet Σ and the states of the immediately preceding component. We have investigated the expressive power of Σ -chains, establishing that every regular language can be decomposed into a Σ -chain of permutation-reset automata, and that every cascade of permutation-reset (pure-reset, and permutation) automata can be translated into an equivalent Σ -chain of the same class and height. While an analogous result for reset automata remains open, we have shown that (Boolean combinations of) Σ -chains of reset automata capture piecewise testable and R-trivial languages. Finally, we addressed the pastification problem, providing a translation of Σ -chains of reset automata into ppLTL formulas of quadratic dag-complexity.

A natural direction for future work is to determine whether every star-free language – that is, a language recognizable by a cascade of reset automata – is also recognizable by some Σ -chain of reset automata, which requires a complete characterization of the languages recognizable by Σ -chains. A further open problem is whether, as in the case of cascades of reset automata, every Σ -chain of reset automata can be translated into an equivalent one in which each component has only two states, possibly at the cost of increasing the height.

Several questions remain open regarding the pastification problem. It would be interesting to investigate whether the quadratic dag-complexity bound is tight, and whether the structural properties of Σ -chains can be exploited to obtain more efficient pastification procedures for LTL formulas in general.

Finally, the cascade product admits the wreath product as its algebraic counterpart. Identifying and studying the algebraic structure corresponding to the Σ -chain product is an interesting direction for future research.

References

- 1 Alessandro Artale, Luca Geatti, Nicola Gigante, Andrea Mazzullo, and Angelo Montanari. A Singly Exponential Transformation of LTL[X, F] into Pure Past LTL. In Pierre Marquis, Tran Cao Son, and Gabriele Kern-Isberner, editors, *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, pages 65–74, 2023. doi:10.24963/KR.2023/7.
- 2 Roberto Borelli, Davide Bresolin, Luca Geatti, Angelo Montanari, and Matteo Zavatterì. The σ -chain product: A succinct model of automata (de)composition (extended version), 2026. arXiv:2607.16884.
- 3 Roberto Borelli, Luca Geatti, Marco Montali, and Angelo Montanari. On Cascades of Reset Automata. In Olaf Beyersdorff, Michal Pilipczuk, Elaine Pimentel, and Kim Thang Nguyen, editors, *42nd International Symposium on Theoretical Aspects of Computer Science, STACS 2025, Jena, Germany, March 4-7, 2025*, LIPIcs, pages 20:1–20:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.STACS.2025.20.
- 4 Roberto Borelli, Luca Geatti, Marco Montali, and Angelo Montanari. Cascades of Reset Automata, 2026. Submitted to ACM Transactions on Computational Logic.

- 5 Janusz A. Brzozowski and Faith E Fich. Languages of R-trivial monoids. *Journal of Computer and System Sciences*, 20(1):32–49, 1980. doi:10.1016/0022-0000(80)90003-3.
- 6 Pál Dömösi and Balázs Imreh. On v_i -products of automata. *Acta Cybernetica*, 6(2):149–162, 1983.
- 7 Luca Geatti. Automata Cascades for Model Checking. In Angelo Montanari, Andrea Orlandini, Nicola Saccomanno, and Stefano Tonetta, editors, *Short Paper Proceedings of the 7th International Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, OVERLAY 2025, Bologna, Italy, October 26, 2025*, CEUR Workshop Proceedings, pages 49–57. CEUR-WS.org, 2025. URL: <https://ceur-ws.org/Vol-4142/paper6.pdf>.
- 8 Ferenc Gécseg. *Products of automata*, volume 7. Springer Science & Business Media, 2012.
- 9 Marcus Gelderie. Classifying regular languages via cascade products of automata. In *International Conference on Language and Automata Theory and Applications*, pages 286–297. Springer, 2011. doi:10.1007/978-3-642-21254-3_22.
- 10 F. Gécseg and H. Jürgensen. On α_0 - v_1 -products of automata. *Theoretical Computer Science*, 80(1):35–51, 1991. doi:10.1016/0304-3975(91)90204-F.
- 11 Balázs Imreh. On α_i -products of automata. *Acta Cybernetica*, 3(4):301–307, 1977. URL: <https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3156>.
- 12 Kenneth Krohn and John Rhodes. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116:450–464, 1965.
- 13 Oded Maler. On the Krohn-Rhodes Cascaded Decomposition Theorem. In Zohar Manna and Doron A. Peled, editors, *Time for Verification, Essays in Memory of Amir Pnueli*, volume 6200 of *Lecture Notes in Computer Science*, pages 260–278. Springer, 2010. doi:10.1007/978-3-642-13754-9_12.
- 14 Oded Maler and Amir Pnueli. Tight Bounds on the Complexity of Cascaded Decomposition of Automata. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 672–682. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89589.
- 15 Jean Eric Pin. *Varieties of formal languages*. Plenum Publishing Co., 1986.
- 16 Imre Simon. Piecewise testable events. In *Automata Theory and Formal Languages: 2nd GI Conference Kaiserslautern, May 20–23, 1975*, pages 214–222. Springer, 2005.
- 17 Wolfgang Thomas. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences*, 25(3):360–376, 1982. doi:10.1016/0022-0000(82)90016-2.
- 18 Karl-Heinz Zimmermann. On Krohn-Rhodes theory for semiautomata. *CoRR*, abs/2010.16235, 2020. arXiv:2010.16235.

Discrete Linear Ensemble Logic: Decidability, Expressiveness, and Axiomatization

Manfred Droste 

Institute of Computer Science, Leipzig University, Germany

Guo-Qiang Zhang¹ 

The University of Texas Health Science Center at Houston, TX, USA

Abstract

We study the discrete point-based fragment of Ensemble Logic $\text{EL}(\mathbb{N})$ over the natural numbers, a logic combining displacement φ_u , term-dependent metric modalities $\Box_t$ and $\Diamond_t$ with respect to an additive term t , Boolean connectives, and first-order quantification over $\mathbb{N}$. Motivated by the need for a unified symbolic representation of biomedical knowledge with temporal, spatial, and multimodal metric content, we develop the foundational discrete theory of the formalism. We give syntax and semantics, and prove a forward embedding of $\text{EL}(\mathbb{N})$ (over a finite proposition set $\mathcal{P}$) into first-order monadic Presburger arithmetic $\text{FO}(\mathbb{N}, <, +; \mathcal{P})$. This embedding yields the analytical upper bounds, while a reduction from nondeterministic two-counter Turing machines with recurring control states proves that satisfiability is Σ_1^1 -complete and validity is dually Π_1^1 -complete. Expressively, $\text{EL}(\mathbb{N})$ strictly extends the star-free ω -languages and is incomparable with the ω -regular languages: it defines the non- ω -regular counting language $\{a^m b^m c^m d^m \mid m \geq 1\} \cdot \Sigma^\omega$, whereas a delimited parity language remains outside the logic by a quantifier-bounding argument combined with classical circuit lower bounds for parity. On the proof-theoretic side, we present a sound Hilbert system $\mathcal{H}_{\text{EL}}$ and establish completeness relative to monadic Presburger validity as oracle, noting that completeness relative to plain Presburger arithmetic is impossible. We also prove that the existential fragment $\exists \text{EL}(\mathbb{N})$ has NP-complete satisfiability and coNP-complete unsatisfiability. Finally, we show that finite active-domain model-checking has PTIME data complexity and PSPACE-complete combined complexity. Together, these results give a precise decidability, expressiveness, proof-theoretic, and model-checking baseline for subsequent algorithmic applications of Ensemble Logic in biomedicine. This work is a part of the Symbolic Biomedicine program championed by the corresponding author.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Modal and temporal logics; Theory of computation $\rightarrow$ Logic and verification; Theory of computation $\rightarrow$ Algebraic language theory; Applied computing $\rightarrow$ Health informatics

Keywords and phrases Temporal logic, monadic Presburger arithmetic, descriptive complexity, electronic health records

Digital Object Identifier 10.4230/OASICS.TIME.2026.4

Funding *Guo-Qiang Zhang*: This work has been supported in part by the U.S. National Science Foundation Award IIS2500624 and the National Institutes of Health grants R01AG084236 and U24AG098157. The views of the paper are those of the authors and do not reflect those of the funding agencies.

Acknowledgements Part of this research was done during a research stay of MD in April 2026 at the University of Texas Health Science Center at Houston. He would like to thank his colleagues for a fruitful and most enjoyable time.

¹ Corresponding author.



© Manfred Droste and Guo-Qiang Zhang;

licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 4; pp. 4:1–4:18

OpenAccess Series in Informatics



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

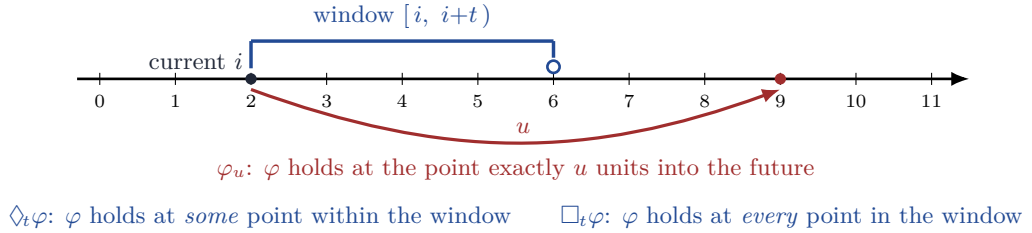
1 Introduction

Modern biomedicine has a formalism gap. Taxonomical ontologies such as SNOMED CT and UMLS [4] capture terminological structure but not metric time; pathway languages such as SBML [17] and BioPAX [8] capture biochemical network structure but not spatial containment; standard temporal logics [24] capture traces but not anatomical geometry; mathematical morphology captures shapes but not protocol-level quantifier scope [26]. A clinical constraint such as an HbA1c normalization condition that must hold continuously for 24 weeks, or a lymph-node tumour deposit large enough to contain a 1 mm spherical core, has a clear mathematical truth condition. Yet neither condition maps naturally to a single formal representation without external scripting in varied programming settings.

The central observation behind Ensemble Logic (EL) [28] is that biomedical reasoning repeatedly uses three non-Boolean, *term-dependent operations*: *exact displacement*, *bounded existence*, and *bounded universality*. In the discrete setting these are written φ_u , $\Diamond_t\varphi$, and $\Box_t\varphi$ (**Fig. 1**). Together with Boolean connectives and first-order quantifiers over an additive index domain, they yield the syntax

$$\varphi, \psi ::= p \mid \varphi_u \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \Diamond_t\varphi \mid \Box_t\varphi \mid \exists x\varphi \mid \forall x\varphi.$$

The interpretation domain of EL accommodates discrete time, dense continuous time, Euclidean space, discrete spatial coordinates, or mixed product spaces depending on the underlying monoid interpretation of the additive terms u and t [28], which in general come from different monoids. Over the discrete point-based domain $\mathbb{N}$, the focus of this paper, the interpretation of the three variable-dependent operations are illustrated below.



■ **Figure 1** An illustration of how the three operators are interpreted over the integer line.

Contributions

In this paper, we focus the mathematical investigation on the foundational discrete case $\text{EL}(\mathbb{N})$, originally introduced in [28], where $\mathbb{N}$ is interpreted temporally (modeling disease trajectories) or spatially (such as base coordinates in genomics). After brief motivational overview and comparison with existing temporal logic formalisms, we define the syntax and semantics of $\text{EL}(\mathbb{N})$ (§ 3). We show that satisfiability of $\text{EL}(\mathbb{N})$ formulas is undecidable, and in fact Σ_1^1 -hard, by a reduction from the recurring problem for nondeterministic two-counter machines. Building on the foundation laid out in [28], this paper establishes results in the directions of decidability, expressiveness, and axiomatization, with specific topics that lie at the heart of any new logical formalism:

1. We translate $\text{EL}(\mathbb{N})$ into monadic Presburger arithmetic, obtaining the analytical upper bounds Σ_1^1/Π_1^1 (§ 4).
2. We further prove Σ_1^1 -hardness by a two-counter-machine reduction tailored to EL's uniformly positive modal bounds (§ 5).

3. We place $\text{EL}(\mathbb{N})$ in the descriptive hierarchy: it contains all $\text{FO}(\mathbb{N}, <; \mathcal{P})$ languages, defines non- ω -regular counting languages, but does not contain all ω -regular languages (§ 6).
4. We give a sound Hilbert system and prove $\text{MONFO}_{\mathbb{N},+}$ -relative completeness (§ 7).
5. We prove NP-completeness of existential-fragment satisfiability, coNP-completeness of unsatisfiability, and give a complete proof system for satisfiability (§ 8).
6. We prove finite active-domain model checking is in PTIME for fixed formulas and PSPACE-complete (§ 9).

2 Background

2.1 Motivations from Biomedicine

Modern biomedicine depends on knowledge expressed across free-text discovery claims, longitudinal records, trial protocols, physiological signals, imaging, and omics. Data in different modalities are still analyzed with fragmented formalisms and ad hoc scripts. The same patient, lesion, molecular locus, or protocol event is represented differently across literature mining, EHR phenotyping, trial eligibility, image analysis, and genomics, even when the underlying scientific observation has a similar structure.

In prior work [20, 16], we demonstrated that the tri-modality of φ_u , $\Diamond_t \varphi$, and $\Box_t \varphi$ (together with standard Boolean and first-order quantifiers) suffices to articulate the entire temporal structure of clinical-trial protocols in a single closed formula, such as one for Alzheimer’s-disease trial NCT01266525 [27]. We illustrate the key modeling idea behind these works using a deliberately compact monitoring protocol example.

Whenever therapy starts on day x , a baseline measurement must occur during the four-day window beginning at x ; a dose (of medication) must be taken on every day of the fourteen-day window beginning one day later; and an outcome review must occur exactly twenty-eight days after the start.

With propositions **Start**, **Baseline**, **Dose**, and **Review**, the protocol is expressed as

$$\forall x \left(\text{Start}_x \rightarrow \left((\Diamond_4 \text{Baseline})_x \wedge (\Box_{14} \text{Dose})_{x+1} \wedge \text{Review}_{x+28} \right) \right). \quad (1)$$

The half-open-window convention ($[i, i+t)$ in **Fig. 1**) makes the three clauses precise:

Fragment	Plain reading	Truth condition
Review_{x+28}	exact displacement	Review takes place on day $x + 28$;
$(\Diamond_4 \text{Baseline})_x$	bounded existence	Baseline holds on day $x, x + 1, x + 2$, or day $x + 3$;
$(\Box_{14} \text{Dose})_{x+1}$	bounded universality	Dose is taken on every day in $[x + 1, x + 15)$.

The variable x refers to (a lapse of) x units from the time at which the closed formula is evaluated. Thus $\forall x$ checks all shifts into the future (i.e. displacement) from a given point for evaluation.

Quantification over metric parameters and logical inference become useful when the same unknown length must be reused. For example, the schematic rule

$$\forall x \left(\text{Start}_x \rightarrow \exists n \left[\Box_{n+1} \text{Stable} \wedge (\Box_{n+1} \text{Treatment})_{n+1} \wedge (\Diamond_{n+1} \text{Outcome})_{2(n+1)} \right]_x \right) \quad (2)$$

asks for a positive, record-dependent phase length $n + 1$ and reuses exactly that length for a stabilization phase, a treatment phase, and an outcome window. This reuse of a semantically chosen additive parameter is the most distinctive syntactic feature of **EL**.

A complete clinical-trial protocol is naturally a conjunction of many clauses of the form illustrated by Formula (1) above, together with domain-specific eligibility predicates and data-quality assumptions. Prior work shows how such clauses can be composed for trial protocols [20]; the present paper is concerned with the underlying discrete logic, not with validating any particular medical protocol.

The same approach extends to other medical domains. In sleep medicine [9], the formula φ_u places a finding φ at a specific offset from a reference time point; for example, snore occurs 6 seconds later can be formalized as Snore_6 . Similarly, $\Box_t \varphi$ states that the condition φ holds continuously for duration t ; for example, airflow reduction greater than 30% persisting for 10 seconds can be formalized as $\Box_{10} \text{FlowDrop30}$. The formula $\Diamond_t \varphi$ states that property φ holds somewhere within a bounded window of length t , so that a spindle occurring within a 3-second window can be formalized as $\Diamond_3 \text{Spindle}$.

The scientific content with respect to how long, how far, how soon, how continuously, is exactly the metric structure unaccounted-for by mainstream approaches. The *EL thesis* [28] is that three *variable-dependent operations*: *exact displacement*, *bounded existence*, and *bounded universality* are parts of a stable “lingua franca” for biomedical knowledge. The novelty is not that any one operator is new. Linear temporal logic and metric temporal logic already contain important special cases (see Section 2.2). The novelty is the cross-modal stability of the operator basis: the same tri-modality can be interpreted on sequences, claim windows, EHR days [20], dense waveforms [9, 15], Euclidean image regions, stranded genomic coordinates, and product spaces for spatial-temporal interactions.

2.2 Comparison with neighboring formalisms

The formalisms in Table 1 are selected according to three principled axes: the semantic carrier (points, intervals, or signals), the way a metric parameter is supplied (fixed constant, frozen timestamp, or quantified offset), and the domain over which first-order variables range. They are not intended as an exhaustive survey, but as an illustration of how EL thesis [28] is genuinely distinct from existing formalisms.

Propositional linear temporal logic (LTL) is the classical point-based logic of linear executions: formulas are evaluated at positions of a trace, and temporal operators such as next X and until U describe how truth evolves along that trace [18, 24]. LTL is qualitative: it can express that an event eventually occurs, but not that it occurs within a specified numerical delay.

Timed and metric temporal logics add quantitative time in different ways. Metric temporal logic (MTL [19, 22]) decorates temporal operators with fixed intervals, as in $\varphi U_{[a,b]} \psi$. Timed propositional temporal logic (TPTL [3]) instead introduces freeze variables: a formula may store the current timestamp and later compare it with other timestamps by arithmetic constraints [3]. First-order LTL combines temporal operators with quantification over an external data domain, while Halpern-Shoham logic evaluates formulas over intervals rather than points, using Allen-style interval relations [2, 12].

The logic EL is closest in spirit to metric temporal logics, but its metric parameters are not merely constants written into the syntax. Instead, EL quantifies over additive displacements of the timeline. Thus a formula can choose an offset n and then reason about positions such as $v+n$ or $v+n+1$. In discrete time, a bounded metric-until formula has the EL-style rendering $\varphi U_{[a,b]} \psi \equiv \bigvee_{r=a}^b (\psi_r \wedge \Box_r \varphi)$, up to the usual endpoint convention for until. Conversely, EL can quantify over the bound itself, as in formulas of the form $\exists n (\Box_{n+1} \varphi \wedge \dots)$, which is outside ordinary fixed-interval MTL. This ability to quantify over an unbounded domain of indices is also what conceptually brings EL closer to highly undecidable interval logics such as Halpern-Shoham logic over discrete linear orders [5, 12].

■ **Table 1** Comparison along semantic carrier, metric mechanism, and quantified domain.

Logic	Semantic carrier	Metric mechanism	Quantified domain
EL($\mathbb{N}$)	point of an ω -word	exact shifts and bounded windows whose additive parameters may contain variables	numerical offsets in $\mathbb{N}$
LTL	point of an infinite trace	next and qualitative until; no numerical parameter in the standard syntax	none in propositional LTL
MTL	timed position or time point	syntactically fixed metric intervals I on temporal operators	normally no quantification over interval bounds
TPTL	timed-word position with timestamp	freeze binder plus clock constraints	frozen timestamps
FO-LTL	trace position carrying a first-order structure	qualitative temporal operators plus first-order atoms	external objects, not temporal offsets

The most distinct feature of EL is that it is not one single logical formalism. EL represents a class of logics when its semantic carriers are instantiated in unlimited ways: discrete time, dense continuous time, Euclidean space, discrete spatial coordinates, or mixed product spaces, in which the underlying monoid interpretation of the additive terms u and t are fixed [28]. Each such instantiation results in a distinct logic (such as the focus of this paper – Discrete Linear Ensemble Logic), with likely distinct theoretical properties regarding decidability, expressiveness, complexity, and axiomatization.

Attributes of EL: Natural and Minimalist

An important distinction between discrete linear EL and the other formalisms mentioned above lies in its natural and minimalist character. A property of an infinite word is a yes/no question one can ask about the sequence $\alpha(0), \alpha(1), \alpha(2), \dots$. Classical temporal logic LTL answers such questions by stepping through the word one position at a time, using operators such as “next” and “until;” it can look arbitrarily far ahead, but it can only *count* by chaining single steps.

Discrete Linear Ensemble Logic keeps the temporal intuition of a point moving along the word, but it hands that point a ruler (**Fig. 1**). Instead of “sometime later” it can say “somewhere within the next t positions,” and instead of “the next position” it can say “the position exactly u steps further from here” (also called “displacement”) where t and u are *additive terms* built from numerical variables (integer variables in this paper). The single design decision that gives the logic its character is that these lengths are first-class objects: one variable may fix a distance that is then reused, combined, and referred to elsewhere in the same formula.

This is also the reason that complexity results based on LTL do not directly apply in the discrete linear EL context. Whereas the operator X encodes future positions in unary in LTL, the displacement φ_u refers to them through the underlying length measurement captured by u , typically yielding an encoding of logarithmic size. Moreover, nested applications of “until” are even less aligned with the natural-language patterns commonly found in biomedical discourse.

Natural and minimalist are also reflected in EL's general treatment of biomedical data records directly as semantic models (with existing annotated features). Other formalisms usually require a more elaborate semantic structure to work with. A full exposition, however, of how biomedical data records directly correspond to semantic models for EL, is beyond the scope of the current paper. For specific EL instantiations and applications, please see [9, 16, 15, 20].

3 Syntax and Semantics

We first recall basic definitions from [28]. Throughout, $\mathbb{N} = \{0, 1, 2, \dots\}$ and $\mathbb{N}_{>0} = \{1, 2, \dots\}$. We adopt a uniform convention: all first-order variables range over $\mathbb{N}$, while modal lengths must evaluate to numbers ≥ 1 .

► **Definition 1 (Terms).** We fix a countably infinite set of variables $V = \{x, y, z, \dots\}$. Additive terms are defined as $u, v, t, s ::= n \mid x \mid u + v$, where $n \in \mathbb{N}$ is a natural-number numeral and $x \in V$. Numerals are encoded in binary. The syntactic size of a numeral n is bounded by $|n| = \Theta(\log(n + 1))$, and the size $|u|$ of a term is given by the size of its syntax tree under this convention. Throughout the paper, $|\varphi|$ denotes the binary-encoded length of φ .

An assignment $\eta : V \rightarrow \mathbb{N}$ naturally extends to terms by homomorphic evaluation: $\eta(n) = n$ and $\eta(u + v) = \eta(u) + \eta(v)$. Whenever a term t appears as the metric subscript of a modality, its evaluation must be strictly positive in the current assignment ($\eta(t) \geq 1$). Syntactically, this can be ensured by writing modality bounds as $t + 1$.

► **Definition 2 (Formulas).** For a finite set $\mathcal{P}$ of atomic propositions, the formulas of $\text{EL}(\mathbb{N})$ are generated by the grammar

$$\varphi, \psi ::= p \mid \varphi_u \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \Diamond_t \varphi \mid \Box_t \varphi \mid \exists x \varphi \mid \forall x \varphi,$$

subject to the condition that every closed instance of a modal length evaluates to ≥ 1 . Standard abbreviations for $\top, \perp, \rightarrow, \leftrightarrow$ are derived as usual.

An interpretation is defined as $\alpha : \mathbb{N} \rightarrow 2^{\mathcal{P}}$, equivalently an ω -word over the alphabet $2^{\mathcal{P}}$. We denote the support of a proposition p by $P_p^\alpha = \{i \in \mathbb{N} \mid p \in \alpha(i)\}$.

► **Definition 3 (Satisfaction Relation).** For a given interpretation α , variable assignment $\bullet\eta$, and temporal/spatial origin point $i \in \mathbb{N}$, satisfaction is defined inductively by standard Boolean clauses paired with the metric primitives:

$$\begin{aligned} (\alpha, \eta, i) \models p &\iff p \in \alpha(i), \\ (\alpha, \eta, i) \models \varphi_u &\iff (\alpha, \eta, i + \eta(u)) \models \varphi, \\ (\alpha, \eta, i) \models \Diamond_t \varphi &\iff \exists j [i \leq j < i + \eta(t) \wedge (\alpha, \eta, j) \models \varphi], \\ (\alpha, \eta, i) \models \Box_t \varphi &\iff \forall j [i \leq j < i + \eta(t) \rightarrow (\alpha, \eta, j) \models \varphi], \\ (\alpha, \eta, i) \models \exists x \varphi &\iff \exists n \in \mathbb{N} (\alpha, \eta[x \mapsto n], i) \models \varphi, \\ (\alpha, \eta, i) \models \forall x \varphi &\iff \forall n \in \mathbb{N} (\alpha, \eta[x \mapsto n], i) \models \varphi. \end{aligned}$$

A formula is closed if it has no free first-order variables. Variables occurring inside additive subscripts are free unless they occur in the syntactic scope of an enclosing first-order quantifier; displacement and metric modalities do not bind variables. For a closed formula φ , $\alpha \models \varphi$ is a syntactic shorthand for $(\alpha, \emptyset, 0) \models \varphi$, and the corresponding language is defined as $L(\varphi) = \{\alpha \in (2^{\mathcal{P}})^\omega \mid \alpha \models \varphi\}$.

► **Proposition 4** ([28]). *For all formulas φ, ψ and additive terms u, v, s, t evaluating positively at modal lengths, the following semantic equivalences hold:*

- (i) $(\neg\varphi)_u \leftrightarrow \neg(\varphi_u)$, $(\varphi \wedge \psi)_u \leftrightarrow \varphi_u \wedge \psi_u$, and dually for $\vee$;
- (ii) $(\varphi_u)_v \leftrightarrow \varphi_{u+v}$;
- (iii) $\Box_t$ distributes over $\wedge$ and dually $\Diamond_t$ over $\vee$;
- (iv) $\neg\Box_t\varphi \leftrightarrow \Diamond_t\neg\varphi$ and dually for $\neg\Diamond_t$;
- (v) *Monotonicity: if $\eta(s) \leq \eta(t)$, then $\Box_t\varphi \rightarrow \Box_s\varphi$ and $\Diamond_s\varphi \rightarrow \Diamond_t\varphi$.*

As an immediate structural corollary, every $\text{EL}(\mathbb{N})$ formula is equivalent to a strict negation-normal form (NNF) where $\neg$ occurs exclusively as a terminal operator on propositional atoms.

4 Forward Embedding into Monadic Arithmetic

Let $\text{FO}(\mathbb{N}, <, +; \mathcal{P})$ represent first-order logic over the standard structure $(\mathbb{N}, <, +)$ expanded by unary predicates P_p for each $p \in \mathcal{P}$. Atomic formulas consist of $P_p(x)$ and standard equalities and inequalities for Presburger terms.

► **Theorem 5** (Forward embedding). *There exists a linear-time translation T assigning to every $\text{EL}(\mathbb{N})$ formula φ and every arithmetic term z denoting the current evaluation point an arithmetical first-order formula $T_z(\varphi)$ such that, for every interpretation α , assignment η , and point $m \in \mathbb{N}$,*

$$(\alpha, \eta, m) \models \varphi \iff (\mathbb{N}, <, +, (P_p^\alpha)_{p \in \mathcal{P}}), \eta[z \mapsto m] \models T_z(\varphi),$$

where bound variables are alpha-renamed to avoid capture of the distinguished variable z and of the fresh variables introduced by the modal clauses.

Proof. Define T_z by structural recursion:

$$\begin{aligned} T_z(p) &:= P_p(z), \\ T_z(\neg\varphi) &:= \neg T_z(\varphi), \\ T_z(\varphi \text{ op } \psi) &:= T_z(\varphi) \text{ op } T_z(\psi), \quad (\text{op} \in \{\wedge, \vee\}), \\ T_z(\varphi_u) &:= T_{z+u}(\varphi), \\ T_z(Qx\varphi) &:= Qx T_z(\varphi), \quad (Q \in \{\exists, \forall\}), \\ T_z(\Diamond_t\varphi) &:= \exists j (z \leq j \wedge j < z + t \wedge T_j(\varphi)), \\ T_z(\Box_t\varphi) &:= \forall j ((z \leq j \wedge j < z + t) \rightarrow T_j(\varphi)), \end{aligned}$$

with j fresh and with the harmless alpha-renaming convention just stated. Correctness is by structural induction on φ . The atomic, Boolean, and first-order quantifier cases are immediate from the semantics and the induction hypothesis. The displacement clause uses $(\alpha, \eta, m) \models \varphi_u$ iff $(\alpha, \eta, m + \eta(u)) \models \varphi$, which is exactly the induction hypothesis for $T_{z+u}(\varphi)$. The modal clauses match the half-open interval $[m, m + \eta(t))$ by quantifying over the fresh variable j with $z \leq j < z + t$. Each constructor contributes a constant-size translation node when the output is represented as a shared directed acyclic graph, so the translation is linear in $|\varphi|$ under the binary encoding convention of Definition 1. ◀

► **Corollary 6** (Analytical upper bound). *Satisfiability of arbitrary closed $\text{EL}(\mathbb{N})$ formulas resides in Σ_1^1 , and algorithmic validity resides in Π_1^1 .*

Proof. By Theorem 5, the satisfiability of an EL formula φ is equivalent to the satisfiability of the formula $T_0(\varphi)$. Hence deciding satisfiability means existentially quantifying the continuous array of possible monadic subset interpretations P_p^α . This can be expressed by a second-order existential prefix $\exists(P_p)_{p \in \mathcal{P}}$ appended to an arithmetical first-order formula, bounding the procedure within Σ_1^1 . Validity is obtained by the dual quantification $\forall(P_p)_{p \in \mathcal{P}}$, bounded in Π_1^1 . $\blacktriangleleft$

5 Σ_1^1 -Hardness via Two-Counter Machines

By Corollary 6, the EL($\mathbb{N}$)-satisfiability problem is in Σ_1^1 . Here we use a reduction to the recurring problem for nondeterministic two-counter machines to show its Σ_1^1 -hardness. A nondeterministic two-counter machine (2CM) [21] comprises a finite control (Q, q_0, q^*, I) alongside absolute counters $C_1, C_2 \in \mathbb{N}$ driving functional increment and test-and-decrement branches. The recurring problem asks whether there is an infinite computational sequence starting from $(q_0, 0, 0)$ and containing the loop state q^* infinitely often.

► **Lemma 7** (Harel-Pnueli-Stavi [13]). *The recurring problem for nondeterministic two-counter machines is Σ_1^1 -complete.*

Now we can show:

► **Theorem 8** (Σ_1^1 -hardness). *The satisfiability problem for closed EL($\mathbb{N}$) formulas is Σ_1^1 -hard.*

Proof. Fix a nondeterministic two-counter machine $M = (Q, q_0, q^*, I)$. We use propositions $Q \cup \{c_1, c_2, e\}$ and first define exact letter formulas

$$H_q := q \wedge \bigwedge_{r \in Q \setminus \{q\}} \neg r \wedge \neg e \wedge \neg c_1 \wedge \neg c_2,$$

$$E := e \wedge \neg c_1 \wedge \neg c_2 \wedge \bigwedge_{r \in Q} \neg r, \quad C_i := c_i \wedge \neg e \wedge \neg c_{3-i} \wedge \bigwedge_{r \in Q} \neg r \quad (i = 1, 2).$$

A configuration with state q and counter values (x, y) begins at position p and is encoded by

$$B_q(p, x, y) := (H_q)_p \wedge E_{p+1} \wedge (\Box_{x+1} C_1)_{p+2} \wedge E_{p+x+3} \\ \wedge (\Box_{y+1} C_2)_{p+x+4} \wedge E_{p+x+y+5}.$$

Thus a counter value x is represented by a C_1 -block of length $x + 1$, and similarly for y ; all modal lengths are positive. The next configuration begins immediately at

$$N(p, x, y) := p + x + y + 6.$$

Let $\Phi_{\text{init}} := B_{q_0}(0, 0, 0)$. For the transition formula, put the machine instructions into the usual finite normal form: increments are always enabled; a decrement/test instruction for counter C_i has one branch for value 0 and one branch for a positive value. For each state q and each zero/positive case $\chi \in \{0, +\}^2$, write $\bar{x}_0 = 0$, $\bar{x}_+ = x' + 1$, $\bar{y}_0 = 0$, and $\bar{y}_+ = y' + 1$. The clause for q, χ is

$$\forall p \forall x' \forall y' \left(B_q(p, \bar{x}_\chi, \bar{y}_\chi) \rightarrow \bigvee_{\iota \in I(q, \chi)} \text{Succ}_\iota(p, \bar{x}_\chi, \bar{y}_\chi) \right),$$

where $I(q, \chi)$ is the finite set of instructions enabled in that case; an empty disjunction is $\perp$. The successor formula is obtained by writing the corresponding next block at $N(p, \bar{x}_\chi, \bar{y}_\chi)$: for example,

$$\text{Succ}_{q:C_1:=C_1+1;q'}(p, x, y) = B_{q'}(N(p, x, y), x + 1, y),$$

and the positive decrement branch for C_1 is written with the current value $x' + 1$ in the antecedent and successor value x' in the consequent,

$$B_q(p, x' + 1, y) \rightarrow B_{q'}(N(p, x' + 1, y), x', y).$$

The zero branch is the analogous clause with current value 0. No arithmetic equality atom is needed; the same variables are reused in the block positions and lengths. Let Φ_{trans} be the conjunction of these finitely many clauses. Finally, recurrence of q^* is expressed by

$$\Phi_{\text{rec}} := \forall r \exists s \exists x \exists y B_{q^*}(r + s, x, y).$$

Set $\varphi_M := \Phi_{\text{init}} \wedge \Phi_{\text{trans}} \wedge \Phi_{\text{rec}}$.

If M has a recurring run, concatenate the corresponding exact blocks. The initial, transition, and recurrence clauses are then immediate. Conversely, suppose $\alpha \models \varphi_M$. From $B_{q_0}(0, 0, 0)$ and Φ_{trans} , induction on successor blocks yields an infinite sequence of adjacent blocks whose states and counter values follow an enabled run of M . Because the formulas H_q, E, C_1, C_2 are mutually exclusive and every successor starts exactly at the end of the current block, no state marker can occur inside a block of this chain; the adjacent blocks therefore tile $\mathbb{N}$. Hence every block satisfying $B_q(p, x, y)$ is a boundary of this run. The formula Φ_{rec} says that such q^* -boundaries occur beyond every cutoff, so the extracted run visits q^* infinitely often. Thus φ_M is satisfiable iff M has a recurring run. The construction uses one constant-size family of clauses per state, instruction, and zero/positive case, and all numerals are fixed small constants; hence $|\varphi_M| = \text{poly}(|M|)$. ◀

► **Corollary 9** (Analytical classification). *The satisfiability problem for closed $\text{EL}(\mathbb{N})$ formulas is Σ_1^1 -complete; the validity problem is Π_1^1 -complete.*

Proof. Upper limits are given by Corollary 6. The hardness of the satisfiability problem is given by Theorem 8. The hardness of the validity problems follows by considering $\neg\varphi_M$. ◀

► **Remark.** A reduction from the Post Correspondence Problem to the satisfiability problem for closed $\text{EL}(\mathbb{N})$ formulas was given in [28] earlier.

6 Position in the Descriptive Hierarchy

In this section we wish to compare the class of EL -definable languages with standard classes of ω -languages. As is well-known, the $\text{MSO}(\mathbb{N}, <; \mathcal{P})$ -definable languages coincide with the ω -regular languages, and the $\text{FO}(\mathbb{N}, <; \mathcal{P})$ -definable languages coincide with the ω -LTL-definable and the ω -star-free rational languages.

► **Example 10** (A non- ω -regular language in EL). We consider the standard alphabet of letters $\Sigma = \{a, b, c, d\}$. Since interpretations are subsets of $\mathcal{P}$ rather than singleton alphabet symbols, using a, b, c, d alone would also accept positions where several propositions hold simultaneously. Therefore, we let $\ell_a := a \wedge \neg b \wedge \neg c \wedge \neg d$ and define ℓ_b, ℓ_c, ℓ_d analogously. The closed formula

$$\exists n \left[\Box_{n+1} \ell_a \wedge (\Box_{n+1} \ell_b)_{n+1} \wedge (\Box_{n+1} \ell_c)_{(n+1)+(n+1)} \wedge (\Box_{n+1} \ell_d)_{(n+1)+(n+1)+(n+1)} \right]$$

defines, under the standard singleton-letter encoding over $\{a, b, c, d\}$, the language $L_{\text{abcd}} = \{a^m b^m c^m d^m \mid m \geq 1\} \cdot \Sigma^\omega$ (choose target index $m = n + 1$). By the pumping lemma, $L_{\text{abcd}} \notin \omega\text{-Reg}$.

► **Example 11** (An ω -regular language outside EL). Let $\Sigma := \{a, b, c, d\}$, and L_{par} over Σ be defined as $L_{\text{par}} := \{wcd^\omega \mid w \in \{a, b\}^*, |w|_a \equiv 0 \pmod{2}\}$. This language is ω -regular: a two-state automaton records the parity of the number of a 's before the unique delimiter c and then checks the d^ω tail. If L_{par} were definable in $\text{EL}(\mathbb{N})$, Theorem 5 would make it definable in $\text{FO}(\mathbb{N}, <, +; \mathcal{P})$. Restricting such a definition to ultimately constant words of the form wcd^ω would yield an $\text{FO}[<, +]$ definition of finite-word parity. This contradicts the classical lower bound that parity is not definable in first-order logic with addition on finite ordered words, equivalently parity is not in uniform AC^0 [10, 1, 14].

Now we obtain:

► **Theorem 12** (Hierarchy classification matrix). *We have:*

- (i) $L(\text{FO}(\mathbb{N}, <; \mathcal{P})) \subsetneq L(\text{EL}(\mathbb{N})) \subseteq L(\text{FO}(\mathbb{N}, <, +; \mathcal{P}))$.
- (ii) $L(\text{EL}(\mathbb{N})) \setminus \omega\text{-Reg} \neq \emptyset$ and $\omega\text{-Reg} \setminus L(\text{EL}(\mathbb{N})) \neq \emptyset$.

Proof. (i) By Kamp's theorem [7, 18], $\text{FO}(\mathbb{N}, <; \mathcal{P})$ -definability of ω -languages coincides with LTL-definability. It is straightforward to translate LTL-formulas into equivalent EL-formulas (i.e., preserving the languages): For the *Next* operator, replace $X\varphi$ by φ_1 , and for *strict Until*, replace $\varphi U_{>0} \psi$ by $\psi_1 \vee \exists x(\psi_{x+2} \wedge (\Box_{x+1} \varphi)_1)$. By Example 10, the inclusion is proper. For the second inclusion, we apply Theorem 5. (ii) Immediate by Examples 10 and 11. ◀

7 A Hilbert System and Oracle-Relative Completeness

In this section, we wish to obtain a Hilbert system for EL-logic and to consider completeness questions. Absolute completeness is impossible by Corollary 9; completeness relative only to Presburger arithmetic is also impossible, since Presburger arithmetic is decidable (Cooper [6], Presburger [25]) and any r.e. system using only $\text{Th}(\mathbb{N}, <, +)$ as oracle would have an r.e. proof relation, contradicting the Π_1^1 -completeness of EL-validity. Therefore we use the monadic Presburger validity theory as oracle.

The system $\mathcal{H}_{\text{EL}}$

The axioms below are given as schemas; rules are modus ponens, universal generalisation, and necessitation for $\varphi_u, \Box_t, \Diamond_t$ from valid formulas.

Group I (propositional and first-order):

- (A1) propositional tautologies;
- (A2) $\forall x \varphi(x) \rightarrow \varphi[u/x]$ when u free for x ;
- (A3) $\forall x(\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \forall x\psi)$ when $x \notin \text{free}(\varphi)$.

Group II (displacement):

- (B1) displacement distributes over $\wedge, \vee, \neg$;
- (B2) $(\varphi_u)_v \leftrightarrow \varphi_{u+v}$;
- (B3) if $u = v$ is arithmetically true, $\varphi_u \leftrightarrow \varphi_v$;
- (B4) $(Qx \varphi)_u \leftrightarrow Qx(\varphi_u)$ for $Q \in \{\exists, \forall\}$, $x \notin \text{free}(u)$.

Group III (metric modalities):

- (C1) $\neg \Box_t \varphi \leftrightarrow \Diamond_t \neg \varphi$ (dually);
- (C2) $\Box_t(\varphi \rightarrow \psi) \rightarrow (\Box_t \varphi \rightarrow \Box_t \psi)$;
- (C3) distribution of $\Box_t / \Diamond_t$ over $\wedge / \vee$;
- (C4) monotonicity: if $s \leq t$ holds in $(\mathbb{N}, <, +)$, then $\Box_t \varphi \rightarrow \Box_s \varphi$ and $\Diamond_s \varphi \rightarrow \Diamond_t \varphi$;
- (C5) $\Box_1 \varphi \leftrightarrow \varphi, \Diamond_1 \varphi \leftrightarrow \varphi$;

- (C6) $(\Box_t \varphi)_u \leftrightarrow \Box_t(\varphi_u)$ (dually for $\Diamond$);
 (C7) for s, t positive under every assignment, nested modalities compose: $\Diamond_s \Diamond_{t+1} \varphi \leftrightarrow \Diamond_{s+t} \varphi$ (dually for $\Box$);
 (C8) *additive decomposition*: for u, v positive under every assignment: $\Diamond_{u+v} \varphi \leftrightarrow \Diamond_u \varphi \vee (\Diamond_v \varphi)_u$ (dually for $\Box$).

Group IV (induction over displacement variables):

- (D1) $\exists x \varphi(x) \leftrightarrow \varphi(0) \vee \exists y \varphi(y+1)$; (dually for $\forall$);
 (D2) induction $\varphi(0) \wedge \forall x(\varphi(x) \rightarrow \varphi(x+1)) \rightarrow \forall x \varphi(x)$.

► **Theorem 13 (Soundness).** *If $\mathcal{H}_{EL} \vdash \varphi$ then $\models \varphi$.*

Proof. We verify the non-trivial axioms. **C7:** $(\alpha, \eta, i) \models \Diamond_s(\Diamond_t \varphi)$ iff $\exists r < s, q < t. (\alpha, \eta, i + r + q) \models \varphi$; the attainable sums $r + q$ form $[0, s + t - 1)$, giving $\Diamond_{s+t-1} \varphi$.

C8: for $a = \eta(u), b = \eta(v) \geq 1$, use the partition

$$[i, i + a + b) = [i, i + a) \cup [i + a, (i + a) + b).$$

D1: from $\mathbb{N} = \{0\} \cup \{y + 1 \mid y \geq 0\}$; **D2** is mathematical induction. Other axioms follow from Proposition 4 and direct inspection. ◀

The oracle

$$\text{MONFO}_{\mathbb{N},+} := \{ \Theta \in \text{FO}(\mathbb{N}, <, +; \mathcal{P}) \mid (\mathbb{N}, <, +, (P_p)) \models \Theta \text{ for every interpretation of the } P_p \}.$$

This is the standard-validity theory of monadic Presburger arithmetic.

► **Theorem 14 (Oracle-relative completeness).** *Extend $\mathcal{H}_{EL}$ with the rule: from $T_0(\varphi) \in \text{MONFO}_{\mathbb{N},+}$ infer φ (for closed φ). Then $\models \varphi \Leftrightarrow \mathcal{H}_{EL} + \text{MONFO}_{\mathbb{N},+} \vdash \varphi$.*

Proof. The proof is provided in the Appendix. ◀

By Halpern [11], $\text{MONFO}_{\mathbb{N},+}$ is itself Π_1^1 -complete; this is the correct level of relative completeness given Corollary 9. Axiom C8 (additive decomposition) is the central modal-arithmetic primitive: it states that inspection of a window of length $u + v$ factors through inspection of a window of length u followed by a displaced window of length v .

As an example for the power of this calculus, we just note that Barcan-type interaction laws can be derived. With the usual assumption of always-positive t , and $x \notin \text{free}(t)$, we have: $\vdash \forall x \Box_t \varphi \leftrightarrow \Box_t \forall x \varphi$; $\vdash \exists x \Diamond_t \varphi \leftrightarrow \Diamond_t \exists x \varphi$; $\vdash \exists x \Box_t \varphi \rightarrow \Box_t \exists x \varphi$; and $\vdash \Diamond_t \forall x \varphi \rightarrow \forall x \Diamond_t \varphi$.

8 The Existential Fragment $\exists \text{EL}(\mathbb{N})$

► **Definition 15 (Existential fragment).** *The existential fragment $\exists \text{EL}(\mathbb{N})$ consists of the closed formulas in negation normal form generated by*

$$\varphi, \psi ::= p \mid \neg p \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi_u \mid \Diamond_t \varphi \mid \exists x \varphi.$$

Thus $\forall$ and $\Box$ are not primitive in the fragment, and negation is allowed only in front of atomic propositions. As throughout the paper, metric bounds are required to be positive on every branch where the modality is used; in applications this is enforced syntactically by writing bounds as $t + 1$.

An explicit translation to existential Presburger arithmetic

We compile $\exists\text{EL}(\mathbb{N})$ -formulas into existential Presburger arithmetic with the standard binary numeral encoding. Up to inessential bookkeeping, the construction is determined by the following idea: the only semantic freedom in satisfiability is the choice of the monadic predicates $P_p^\alpha \subseteq \mathbb{N}$. This freedom can be eliminated as follows: each diamond chooses one witness inside its window, and each literal occurrence inspects *one* truth value of *one* proposition at *one* arithmetically computed position. If two such inspection positions happen to collide, the two truth values must agree. Once those constraints are written down, the satisfiability question reduces to a pure existential Presburger question.

Fix a closed formula $\varphi \in \exists\text{EL}(\mathbb{N})$. First alpha-rewrite all bound variables and assign a fresh variable h_δ to every occurrence δ of a modality $\Diamond_t$. Let $\bar{v}_\varphi$ be the tuple of all renamed first-order variables and all modal-witness variables h_δ . For every literal occurrence ρ introduce a fresh Boolean Presburger variable b_ρ . We define simultaneously a quantifier-free Presburger formula $B_\tau(\psi)$ and, for every literal occurrence reached during the recursion, its proposition p_ρ and absolute position term a_ρ :

$$\begin{aligned} B_\tau(p^\rho) &:= b_\rho = 1, & a_\rho &:= \tau, & p_\rho &:= p, \\ B_\tau((\neg p)^\rho) &:= b_\rho = 0, & a_\rho &:= \tau, & p_\rho &:= p, \\ B_\tau(\psi \wedge \chi) &:= B_\tau(\psi) \wedge B_\tau(\chi), & B_\tau(\psi \vee \chi) &:= B_\tau(\psi) \vee B_\tau(\chi), \\ B_\tau(\psi_u) &:= B_{\tau+u}(\psi), & B_\tau(\exists x \psi) &:= B_\tau(\psi), \\ B_\tau(\Diamond_t^\delta \psi) &:= (0 \leq h_\delta \wedge h_\delta < t) \wedge B_{\tau+h_\delta}(\psi). \end{aligned}$$

The clause for $\exists x$ is legitimate because all existential variables have already been alpha-renamed and are placed in the global existential prefix. Moving them to the front preserves truth, since only existential first-order quantifiers are present.

Let $\text{Occ}(\varphi)$ be the finite set of literal occurrences. Define

$$\begin{aligned} \text{Bool}_\varphi &:= \bigwedge_{\rho \in \text{Occ}(\varphi)} (b_\rho = 0 \vee b_\rho = 1), \\ \text{Cons}_\varphi &:= \bigwedge_{\substack{\rho, \sigma \in \text{Occ}(\varphi) \\ p_\rho = p_\sigma}} (a_\rho = a_\sigma \rightarrow b_\rho = b_\sigma), \\ \text{EP}(\varphi) &:= \exists \bar{v}_\varphi \exists (b_\rho)_{\rho \in \text{Occ}(\varphi)} (B_0(\varphi) \wedge \text{Bool}_\varphi \wedge \text{Cons}_\varphi). \end{aligned}$$

Here implication is read as the Presburger formula $a_\rho \neq a_\sigma \vee b_\rho = b_\sigma$.

The formula $\text{EP}(\varphi)$ is an existential Presburger formula of size polynomial in $|\varphi|$; the only quadratic blow-up is the pairwise consistency condition.

► **Lemma 16** (Correctness of the Presburger translation). *For every closed $\varphi \in \exists\text{EL}(\mathbb{N})$,*

$$\varphi \text{ is satisfiable over } \omega\text{-words} \iff (\mathbb{N}, <, +) \models \text{EP}(\varphi).$$

Proof. Suppose first that $\alpha \models \varphi$. Choose witnesses for the first-order existential quantifiers and, along one successful evaluation of each used diamond occurrence, choose h_δ to be the offset from the current point to the witnessing point. Variables belonging only to unused disjunctive branches may be assigned arbitrarily. For every literal occurrence ρ , set $b_\rho = 1$ iff $p_\rho \in \alpha(\eta(a_\rho))$. Then Bool_φ is immediate. If two occurrences ρ, σ mention the same proposition and $\eta(a_\rho) = \eta(a_\sigma)$, they inspect the same proposition at the same position of the same word, so $b_\rho = b_\sigma$; hence Cons_φ holds. A direct induction on the successful evaluation tree of φ gives $B_0(\varphi)$.

Conversely, assume $(\mathbb{N}, <, +) \models \text{EP}(\varphi)$ and fix satisfying values for $\bar{v}_\varphi$ and the Boolean variables b_ρ . Define an ω -word α as follows. For every proposition p and position n , if there exists an occurrence ρ with $p_\rho = p$ and $\eta(a_\rho) = n$, put $p \in \alpha(n)$ exactly when $b_\rho = 1$; if no such occurrence exists, choose the truth value of p at n arbitrarily, say false. The definition is independent of the chosen occurrence by Cons_φ . Now prove by structural induction on subformulas that whenever $B_\tau(\psi)$ is true under the fixed witnesses, then $(\alpha, \eta, \eta(\tau)) \models \psi$. The literal cases follow from the construction of α ; Boolean cases are immediate; displacement changes the current term from τ to $\tau + u$; the existential case uses the globally fixed value of the renamed variable; and the diamond case uses the fixed offset h_δ , whose conjunct $0 \leq h_\delta < t$ places the witnessing point inside the half-open interval required by Definition 3. Applying the induction to $B_0(\varphi)$ yields $\alpha \models \varphi$. ◀

► **Theorem 17** (Decidability and NP-completeness). *Satisfiability of closed $\exists\text{EL}(\mathbb{N})$ formulas is decidable and NP-complete.*

Proof. Membership in NP follows from Lemma 16 and the classical NP upper bound for satisfiability of existential Presburger arithmetic with binary numerals [23]. The translation $\varphi \mapsto \text{EP}(\varphi)$ is polynomial: it introduces one numeric witness for each diamond, one Boolean variable for each literal occurrence, and only a quadratic number of consistency implications.

For hardness, reduce propositional satisfiability. Given a propositional formula F over variables $X_1, \dots, X_m$, introduce propositions $p_1, \dots, p_m$ and replace each positive literal X_i by p_i and each negative literal $\neg X_i$ by $\neg p_i$. The resulting $\exists\text{EL}(\mathbb{N})$ formula contains no quantifier and no modality. A truth assignment satisfying F is realised by choosing the truth values of the propositions p_i at position 0, and conversely every satisfying word determines such a truth assignment from its letter at position 0. Thus F is satisfiable iff the translated $\exists\text{EL}(\mathbb{N})$ formula is satisfiable. ◀

► **Corollary 18** (Finite certificate and unsatisfiability complexity). *Every satisfiable closed $\exists\text{EL}(\mathbb{N})$ formula has a certificate consisting of polynomial-bit Presburger witnesses together with truth values for the finitely many literal occurrences. Consequently, unsatisfiability of closed $\exists\text{EL}(\mathbb{N})$ formulas is coNP-complete.*

Proof. The certificate is precisely a satisfying assignment for $\text{EP}(\varphi)$. The polynomial bit bound is the standard small-certificate property behind the NP upper bound for existential Presburger arithmetic. Since satisfiability is NP-complete by Theorem 17, its complement, unsatisfiability, is coNP-complete. ◀

A complete proof system for satisfiability of the existential fragment

Let PrA be any fixed sound and complete axiomatisation of Presburger arithmetic. For

$$\text{EP}(\varphi) = \exists \bar{y} \Theta_\varphi(\bar{y}),$$

where Θ_φ is quantifier-free, define a labelled calculus $\text{S}_{\exists\text{EL}}$ for satisfiability judgements by the two terminal rules

$$\frac{\text{PrA} \vdash \Theta_\varphi(\bar{n})}{\text{S}_{\exists\text{EL}} \vdash \text{Sat}(\varphi)} \quad \frac{\text{PrA} \vdash \neg \exists \bar{y} \Theta_\varphi(\bar{y})}{\text{S}_{\exists\text{EL}} \vdash \text{Unsat}(\varphi)},$$

where $\bar{n}$ ranges over tuples of numerals of the right arity. The first rule is a witness rule; the second is a Presburger refutation rule. In both cases Θ_φ is the matrix produced by the explicit translation above.

For standard validity proofs, let $\text{H}_{\text{rel}}^\exists$ denote the restriction of the oracle-extended Hilbert system of Theorem 14 to conclusions that belong to $\exists\text{EL}(\mathbb{N})$.

► **Theorem 19** (Complete proof systems for satisfiability of $\exists\text{EL}(\mathbb{N})$). *For every closed $\varphi \in \exists\text{EL}(\mathbb{N})$:*

- (i) $S_{\exists\text{EL}} \vdash \text{Sat}(\varphi)$ iff φ is satisfiable;
- (ii) $S_{\exists\text{EL}} \vdash \text{Unsat}(\varphi)$ iff φ is unsatisfiable;
- (iii) $H_{\text{rel}}^{\exists} \vdash \varphi$ iff $\models \varphi$.

Proof. For (i), if the witness rule derives $\text{Sat}(\varphi)$, then the Presburger proof of $\Theta_{\varphi}(\bar{n})$ makes $\text{EP}(\varphi)$ true, and Lemma 16 gives a satisfying word. Conversely, if φ is satisfiable, Lemma 16 gives a tuple $\bar{n}$ satisfying Θ_{φ} ; completeness of PrA proves the true ground Presburger sentence $\Theta_{\varphi}(\bar{n})$, so the witness rule applies.

For (ii), the refutation rule is sound because $\text{PrA} \vdash \neg \exists \bar{y} \Theta_{\varphi}$ implies that $\text{EP}(\varphi)$ is false, hence φ is unsatisfiable by Lemma 16. Conversely, if φ is unsatisfiable, then $\neg \text{EP}(\varphi)$ is a true Presburger sentence, so it is provable in PrA and the refutation rule applies.

For (iii), $\exists\text{EL}(\mathbb{N})$ is a syntactic subfragment of $\text{EL}(\mathbb{N})$. Therefore the soundness and completeness equivalence $\mathcal{H}_{\text{EL}} + \text{MONFO}_{\mathbb{N},+} \vdash \psi$ iff $\models \psi$ from Theorem 14 applies in particular to every closed existential formula $\psi = \varphi$; restricting the set of allowed conclusions does not change derivability of such conclusions. ◀

9 Finite Active-Domain Model Checking

In clinical applications, model checking is performed over a finite active domain. Padding a finite trace by a don't-care suffix would leave quantifiers ranging over an infinite tail and is a different problem.

► **Definition 20** (Active-domain semantics). *A finite trace of length N is $\sigma : D_N \rightarrow 2^{\mathcal{P}}$ for $D_N = \{0, \dots, N-1\}$. Active-domain satisfaction $\sigma, \eta, i \models_N \varphi$ is as in Definition 3, except that all positions and quantifier ranges are restricted to D_N : φ_u requires $i + \eta(u) \in D_N$; modalities range over $D_N \cap [i, i + \eta(t))$; quantifiers range over D_N .*

► **Theorem 21** (Finite model-checking complexity). *For finite trace σ of length N and closed $\varphi \in \text{EL}(\mathbb{N})$, deciding $\sigma \models_N \varphi$ has (i) PTIME data complexity for fixed φ , and (ii) PSPACE-complete combined complexity in $|\varphi| + \log N$, where $|\varphi|$ is the binary-encoded size from Definition 1 and $\log N$ is the bit-length of the active-domain bound.*

Proof.

Upper bound. Evaluate φ by DFS recursion. A configuration is (occurrence, $i, \eta \upharpoonright \text{scope}$); each value uses $O(\log N)$ bits, scope has $\leq |\varphi|$ variables, recursion depth is $\leq |\varphi|$. Quantifiers and modalities iterate over D_N (and over $D_N \cap [i, i + \eta(t))$); arithmetic evaluations $\eta(u), \eta(t)$ on binary terms are done in $\text{poly}(|\varphi| + \log N)$ time and $O(\log N)$ space. The algorithm uses polynomial space in $|\varphi| + \log N$. For fixed φ , nested iterations give time $O(N^c)$, $c = c(\varphi)$.

Hardness. Reduce QBF on traces of length 2: $\mathcal{P} = \{T\}$, $\sigma(0) = \emptyset$, $\sigma(1) = \{T\}$; translate $X_i \mapsto T_{x_i}$, $\exists X_i \mapsto \exists x_i$, $\forall X_i \mapsto \forall x_i$. Each x_i ranges over $\{0, 1\}$ and T_{x_i} is true iff $x_i = 1$. The reduction is linear in any encoding. ◀

10 Discussion: Expressive Limits

We conjecture that the EL -definable languages form a proper subclass of the $\text{FO}(\mathbb{N}, <, +; \mathcal{P})$ -definable languages; a proof could proceed using Ehrenfeucht-Fraïssé-games. It would be interesting to explore suitable oracle boundaries for sub-hierarchies in $\mathcal{H}_{\text{EL}}$. Furthermore, it is open whether bounded sub-fragments of EL might map exactly to known classes of ω -recognizable languages.

11 Conclusion

This paper develops the foundational discrete theory of Ensemble Logic over $\mathbb{N}$. Starting from the three core operations of exact displacement, bounded existence, and bounded universality, we gave a syntax and semantics for $\text{EL}(\mathbb{N})$ that is expressive enough to represent metric constraints while remaining close to the symbolic forms used in temporal, spatial, genomic, and clinical reasoning. The formal development clarifies the precise role of additive terms, positive modal bounds, and first-order quantification over the discrete index domain.

The first main contribution is the translation of $\text{EL}(\mathbb{N})$ into monadic first-order Presburger arithmetic with unary predicates. The capture-avoiding embedding provides a uniform semantic bridge from Ensemble Logic to a classical arithmetic setting and yields sharp analytical upper bounds for satisfiability and validity. Complementing this upper bound, the two-counter-machine construction shows that these bounds are tight: satisfiability is Σ_1^1 -complete and validity is Π_1^1 -complete. Thus the full logic is highly expressive, and its undecidability is not an artefact of the presentation but an intrinsic consequence of combining metric displacement, bounded modalities, and quantification.

The second contribution is the placement of $\text{EL}(\mathbb{N})$ in the descriptive hierarchy. The logic contains the star-free ω -languages and can define counting patterns such as $\{a^m b^m c^m d^m \mid m \geq 1\} \cdot \Sigma^\omega$, showing that it reaches beyond ω -regularity. At the same time, the parity-separation argument shows that $\text{EL}(\mathbb{N})$ does not subsume all ω -regular behavior. This gives a more nuanced expressiveness profile: $\text{EL}(\mathbb{N})$ is neither simply a metric variant of linear temporal logic nor merely a notational fragment of regular-language theory.

The third contribution is proof-theoretic. We introduced the Hilbert system $\mathcal{H}_{\text{EL}}$, proved its soundness, and established an oracle-relative completeness theorem with respect to monadic Presburger validity. The relative theorem identifies the exact arithmetic oracle needed for full completeness and explains why a weaker oracle for plain Presburger arithmetic cannot suffice. For the existential fragment, the Presburger-based calculus and complexity analysis give a decidable and computationally meaningful subtheory, with NP-complete satisfiability and coNP-complete unsatisfiability.

The fourth contribution is algorithmic. Finite active-domain model checking has PTIME data complexity for fixed formulas and PSPACE-complete combined complexity. This separates the cost of evaluating a fixed biomedical rule over a finite trace from the cost of allowing the rule itself to vary, and it provides the appropriate complexity baseline for implementations that target electronic health records, clinical-trial protocols, sleep-event streams, genomic coordinates, or other finite symbolic data sources.

Overall, the results establish $\text{EL}(\mathbb{N})$ as a mathematically robust discrete metric logic: expressive enough to encode non-regular counting and analytically complete computation-theoretic phenomena, yet structured enough to admit exact embeddings, proof systems for meaningful fragments, and finite-trace model-checking algorithms. The remaining expressive-limit questions discussed above point toward a sharper classification of pure EL inside monadic Presburger arithmetic and toward practically useful bounded fragments for biomedical knowledge representation.

References

- 1 Miklós Ajtai. Σ_1^1 -formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983. doi:10.1016/0168-0072(83)90038-6.
- 2 James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983. doi:10.1145/182.358434.
- 3 Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–204, 1994. doi:10.1145/174644.174651.

- 4 Olivier Bodenreider. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32(Database issue):D267–D270, 2004. doi:10.1093/nar/gkh061.
- 5 Davide Bresolin, Angelo Montanari, Pietro Sala, and Guido Sciavicco. What’s decidable about Halpern and Shoham’s interval logic? The maximal fragment ABBL. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science (LICS 2011)*, pages 387–396. IEEE Computer Society, 2011. doi:10.1109/LICS.2011.35.
- 6 D. C. Cooper. Theorem proving in arithmetic without multiplication. In Bernard Meltzer and Donald Michie, editors, *Machine Intelligence*, volume 7, pages 91–99. Edinburgh University Press, Edinburgh, 1972.
- 7 S. Shelah D. Gabbay, A. Pnueli and J. Stavi. On the temporal analysis of fairness. In *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages (POPL)*, 1980, pages 163–173. ACM Press, 1980. doi:10.1145/567446.567462.
- 8 Emek Demir et al. The BioPAX community standard for pathway data sharing. *Nature Biotechnology*, 28(9):935–942, 2010. doi:10.1038/nbt.1666.
- 9 J. Fan, Xiaojin Li, Yan Huang, X. Hao, Licong Cui, and Guo-Qiang Zhang. Ensemble logic for symbolic representation of sleep medicine guidelines. In *Proceedings of the AMIA 2026 Annual Symposium*, 2026. In press, 10 pages. doi:10.48550/arXiv.2607.21331.
- 10 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. doi:10.1007/BF01744431.
- 11 Joseph Y. Halpern. Presburger arithmetic with unary predicates is Π_1^1 complete. *Journal of Symbolic Logic*, 56(2):637–642, 1991. doi:10.2307/2274706.
- 12 Joseph Y. Halpern and Yoav Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962, 1991. doi:10.1145/115234.115351.
- 13 David Harel, Amir Pnueli, and Jonathan Stavi. Propositional dynamic logic of nonregular programs. *Journal of Computer and System Sciences*, 26(2):222–243, 1983. doi:10.1016/0022-0000(83)90014-4.
- 14 Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC 1986)*, pages 6–20. ACM, 1986. doi:10.1145/12130.12132.
- 15 Yan Huang, Xiaojin Li, Licong Cui, and Guo-Qiang Zhang. A logic-based temporal cohort discovery engine: algorithms, indices, and experimental results on the National Sleep Research Resource. *Journal of Biomedical Informatics*, 2026. Under review. doi:10.48550/arXiv.2607.21377.
- 16 Yan Huang, Xiaojin Li, X. Hao, Rashmie Abeysinghe, Licong Cui, and Guo-Qiang Zhang. Scaling up formal representation of clinical trial protocols in ensemble logic using LLMs: a preliminary study. In *Proceedings of the AMIA 2026 Annual Symposium*, 2026. In press, 10 pages. doi:10.48550/arXiv.2607.21307.
- 17 Michael Hucka et al. The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003. doi:10.1093/bioinformatics/btg015.
- 18 Hans Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968. URL: <https://philpapers.org/rec/KAMTLA-2>.
- 19 Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990. doi:10.1007/BF01995674.
- 20 Xiaojin Li, Yan Huang, Rashmie Abeysinghe, Zenan Sun, Hongyu Chen, Pengze Li, Xing He, Shiqiang Tao, Cui Tao, Jiang Bian, Licong Cui, and Guo-Qiang Zhang. Temporal Ensemble Logic for Integrative Representation of the Entirety of Clinical Trials. In Thierry Vidal and Przemysław Andrzej Wałęga, editors, *32nd International Symposium on Temporal Representation and Reasoning (TIME 2025)*, volume 355 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:16, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.TIME.2025.13.

- 21 Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, NJ, 1967.
- 22 Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS 2005)*, pages 188–197. IEEE Computer Society, 2005. doi:10.1109/LICS.2005.33.
- 23 Christos H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM*, 28(4):765–768, 1981. doi:10.1145/322276.322287.
- 24 Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS 1977)*, pages 46–57, Providence, RI, 1977. IEEE. doi:10.1109/SFCS.1977.32.
- 25 Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du I Congrès des Mathématiciens des Pays Slaves*, pages 92–101, Warsaw, 1929. URL: <https://eudml.org/doc/212367>.
- 26 Jean Serra and Pierre Soille, editors. *Mathematical Morphology and Its Applications to Image Processing*, volume 2 of *Computational Imaging and Vision*. Kluwer Academic Publishers, Dordrecht, 1994. doi:10.1007/978-94-011-1040-2.
- 27 U.S. National Library of Medicine. Effect of Different Doses of SAR110894 on Cognition in Patients with Mild to Moderate Alzheimer’s Disease on Donepezil. ClinicalTrials.gov, 2011. ClinicalTrials.gov Identifier: NCT01266525. URL: <https://clinicaltrials.gov/study/NCT01266525>.
- 28 Guo-Qiang Zhang. Temporal Ensemble Logic. *CoRR*, abs/2408.14443, 2024. doi:10.48550/arXiv.2408.14443.

A Proof of Theorem 14

► **Theorem 14** (Oracle-relative completeness). *Extend $\mathcal{H}_{EL}$ with the rule: from $T_0(\varphi) \in \text{MONFO}_{\mathbb{N},+}$ infer φ (for closed φ). Then $\models \varphi \Leftrightarrow \mathcal{H}_{EL} + \text{MONFO}_{\mathbb{N},+} \vdash \varphi$.*

Proof.

Soundness of the extended system $\mathcal{H}_{EL} + \text{MONFO}_{\mathbb{N},+}$. We must show that for every closed $\varphi \in \text{EL}(\mathbb{N})$, if $\mathcal{H}_{EL} + \text{MONFO}_{\mathbb{N},+} \vdash \varphi$ then $\models \varphi$. The system $\mathcal{H}_{EL} + \text{MONFO}_{\mathbb{N},+}$ inherits all axioms and rules of $\mathcal{H}_{EL}$ and adds a single rule: from $T_0(\varphi) \in \text{MONFO}_{\mathbb{N},+}$, infer φ . We verify soundness for each kind of derivation step.

Inherited $\mathcal{H}_{EL}$ -axioms and rules. By Theorem 13, every $\mathcal{H}_{EL}$ -axiom is valid on every ω -word, and modus ponens, generalisation, and necessitation preserve validity. The verification there proceeds axiom-by-axiom: **A1** by the classical truth-table argument, **A2-A3** by standard first-order semantics, **A4** by definition of $\text{Th}(\mathbb{N}, <, +)$, **B1-B4** by the displacement clause of Definition 3, **C1-C8** by case analysis on the satisfaction intervals (with **C7, C8** explicitly checked in Theorem 13), **D1** from $\mathbb{N} = \{0\} \cup \{y + 1 \mid y \geq 0\}$, and **D2** from mathematical induction.

The oracle rule. Suppose the last step of the derivation is $T_0(\varphi) \in \text{MONFO}_{\mathbb{N},+} \Rightarrow \varphi$. By Theorem 5, for every ω -word α and every assignment η ,

$$(\alpha, \eta, 0) \models \varphi \iff (\mathbb{N}, <, +, (P_p^\alpha)_{p \in \mathcal{P}}, \eta) \models T_0(\varphi).$$

Since φ is closed, η is immaterial. The condition $T_0(\varphi) \in \text{MONFO}_{\mathbb{N},+}$ asserts that $T_0(\varphi)$ holds in $(\mathbb{N}, <, +)$ under *every* interpretation (P_p) of the unary predicates. In particular, for every ω -word α , the assignment $P_p \mapsto P_p^\alpha$ is one such interpretation, so $(\mathbb{N}, <, +, (P_p^\alpha)) \models T_0(\varphi)$, hence $(\alpha, 0) \models \varphi$. As α was arbitrary, $\models \varphi$.

Completeness of $\mathcal{H}_{\text{EL}} + \text{MONFO}_{\mathbb{N},+}$. Assume $\models \varphi$. By the equivalence in Theorem 5 applied in the *contrapositive* direction, for every ω -word α , $(\mathbb{N}, <, +, (P_p^\alpha)) \models T_0(\varphi)$. Crucially, the map $\alpha \mapsto (P_p^\alpha)_{p \in \mathcal{P}}$ is a bijection between ω -words and unary-predicate interpretations $\mathcal{P} \rightarrow 2^{\mathbb{N}}$: setting $P_p^\alpha = \{i \in \mathbb{N} : p \in \alpha(i)\}$ defines a bijection with inverse $\alpha(i) = \{p \in \mathcal{P} : i \in P_p\}$. Hence every interpretation (P_p) arises as (P_p^α) for some α , and

$$T_0(\varphi) \in \text{MONFO}_{\mathbb{N},+} \iff \forall \alpha (\alpha, 0) \models \varphi \iff \models \varphi.$$

The forward direction of this chain gives $T_0(\varphi) \in \text{MONFO}_{\mathbb{N},+}$, and the oracle rule then derives φ in a single step. Thus $\mathcal{H}_{\text{EL}} + \text{MONFO}_{\mathbb{N},+} \vdash \varphi$.

Axiom-usage. The completeness derivation produced above uses *only* the oracle rule and *none* of the structural axioms or rules of $\mathcal{H}_{\text{EL}}$. This is the standard content of an *oracle-relative completeness* theorem in modal logic: the underlying calculus is sound (and the oracle rule extends this soundly), but the completeness reduction itself is delegated to the oracle. The role of the $\mathcal{H}_{\text{EL}}$ -axioms is therefore:

- **Sub-derivations leading to the oracle.** While the completeness construction above invokes the oracle in one step, practical proof search would typically use the $\mathcal{H}_{\text{EL}}$ -axioms to *normalise* φ before consulting the oracle (e.g., **C8** unfolds bounded modalities, **B1-B2** flatten nested displacements, **D1** unfolds existentials), reducing the size of the formula handed to the oracle.
- **Oracle-free completeness on restricted fragments.** The structural axioms suffice without an oracle on the *existential fragment* (Theorem 19), where the existential counterparts **A1[∃]**-**E4[∃]** give a meta-complete system without an oracle, although with **A4[∃]** retained as a Presburger side-condition oracle.

Necessity of the oracle. By Corollary 9, validity in $\text{EL}(\mathbb{N})$ is Π_1^1 -complete, hence not recursively enumerable. No recursive extension of $\mathcal{H}_{\text{EL}}$ can therefore be complete; the relativisation to $\text{MONFO}_{\mathbb{N},+}$ (also Π_1^1 -complete, by Halpern [11]) is optimal in this analytical sense. ◀

Social Networks Through Time: Completeness of Temporal Network Logic

Saúl Fernández González ✉ 

University of Oviedo, Spain

Mina Young Pedersen ✉ 

Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands

Sonja Smets ✉ 

Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands

Abstract

The framework of Temporal Network Logic (TNL) – first introduced in [29] and further developed in [30, 28] – was construed to model how a social network of interacting agents changes through time, with a language that could reflect both agents’ actions, such as posting information and following each other, as well as temporal operators. This framework allows, among other things, to detect social bots, by modelling given bot behaviours as logical formulas and checking these formulas against a given model. While such applications were the main focus in [29, 30, 28], the complete axiomatic system of this language has remained an open problem. In this paper, we focus on new logic-technical results and give a complete axiomatisation for TNL as well as a partial completeness result for its hybrid extension. The proofs are nonstandard and mix techniques from temporal and hybrid logics with “orthogonal” models.

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

Keywords and phrases temporal logic, completeness, social network logic, hybrid logic, logics for multi-agent systems

Digital Object Identifier 10.4230/OASICS.TIME.2026.5

1 Introduction

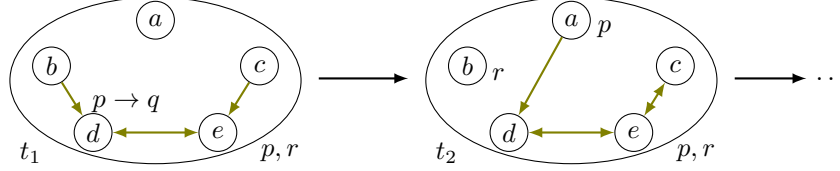
Temporal logics and multi-agent systems go hand in hand. Since the introduction of Alternating-time Temporal Logic (ATL) [1], much of the work in logics for multi-agent systems has been devoted to analysing how agents, their actions, strategies, and relations change through time. By multi-agent systems, we mean systems of multiple autonomous agents that may have diverging interests or diverging information but can interact, following [37], among others. By temporal logic, we mean logical frameworks that include modal operators such as *Past*, *Future*, *neXt-time*, *Yesterday*, *Since* or *Until* to reason about time. Temporal logics for multi-agent systems, in addition to ATL, include Epistemic-Temporal Logic (ETL) [26], for reasoning about agents’ knowledge of observable events through histories in time; Coalition Logic (CL) [27], used to model game-theoretic strategies of groups of agents;¹ and Strategy Logic (SL) [25] where strategies are treated as their own notions in the language, in contrast to ATL and CL where strategies are embedded in temporal operators. The BDI-architecture [31, 32] for modeling beliefs, desires and intentions in agents must also be mentioned as early work predating ATL in temporal reasoning of multi-agent systems.

A paradigm at the intersection of temporal logic and multi-agent systems concerns temporal logics to reason about *social networks*, which are graphs where the nodes represent agents and the relation between them represents a social relation (“follows”, “is friends

¹ It was proved in [20] that CL is in fact a fragment of ATL with only *neXt-time* as its temporal operator.



with”, etc.). Work in the field of temporal logics for social networks includes papers such as [24, 9, 16] and the recent work on *Temporal Network Logic (TNL)*, first introduced in [29] and further developed in [30, 28]. TNL is a temporal logic where the formulas are interpreted on a linear timeline with a beginning and no end. Within each time point, there is a social network, which lets us model how a network of agents changes through time. See Figure 1, taken from [28], for an example.



■ **Figure 1** A temporal network model.

This figure depicts the two first points of a temporal network model, modelling how a network of agents named from a to e develops from one step to another. The interpretation is that the model continues to the infinite. Agents can follow each other, denoted by an arrow: we can observe that, for instance, agent b follows agent d at t_1 . Agents can also post information in the network where the information posted is itself a logical formula. In the figure, posts are denoted by formulas next to the agent: for example, note that agent d posted $p \rightarrow q$ at t_1 . Propositional atoms are universally true or false at each time point, denoted in the figure by atoms on the right of the time point. Note that the absence of an atom means that its negation holds at that time point. The intuitive reading of the model is that it records snapshots of what is true in a network at given times; this means that we interpret posts that are there in one point but gone in the next as deleted from an agent’s profile. Similarly, we interpret that if an agent follows another in one point but not in the next, it has unfollowed the other. As an example, we observe that agent d has posted false information in t_1 , but deleted it in the next step: at t_1 agent d posted $p \rightarrow q$ while p is true but q is false, and at t_2 the post is gone.

The original work on TNL was developed to propose model checking as a detection method of *social bot* behaviour in temporal network models. Social bots are computer programs that act like humans on social media platforms and have been associated with certain behaviours such as sharing and amplifying misinformation online [36]. The proposal in [29] was to use the logical language of TNL to formalise bot behaviour; the global model checking problem was then interpreted as checking at what time points the behaviour occurred in the social network. In [30, 28], the language was extended with nominals and operators from hybrid logic [4]. In this extended hybrid-logical framework, formulas are checked against pairs of agents and time points, which meant that the output of the global model checking algorithm gave not only when, but also which agents in the network exhibited bot behaviour. For the most thorough explanation of TNL and its model checking procedure, we advise the reader to turn to [28, Chapter 4]. In the rest of the paper, we refer to this work for the original work on TNL.

This paper is about completeness of TNL, which until now has been an open problem, both in its standard form (for which we offer a complete axiomatics) and its extensions (for which we have partial completeness results). Proving completeness of the extended framework, including both temporal and “social” operators in the language, presents particular challenges, requiring techniques from *orthogonal frames* [6, 19] and *fusion logics* [14, 23]; we show completeness of a simplified bidimensional socio-temporal framework and highlight some of these difficulties towards eventually proving completeness of the full hybrid extension.

The paper is structured as follows. We begin in Section 2 by presenting Temporal Network Logic. We give an axiomatic system and show that it is weakly complete with respect to models of TNL. In Section 3 we introduce the hybrid extension of TNL, called Temporal Network Agent Logic (TNAL) and discuss steps towards proving completeness. These steps include the presentation of an intermediate logical framework that lies between TNL and TNAL, and a sketch to prove completeness for this logic. We conclude by sketching on-going work on an alternative framework with dynamic modalities instead of temporal operators.

2 Temporal Network Logic

We begin by presenting the syntax and semantics of Temporal Network Logic before introducing its axiomatic system.

2.1 Syntax and Semantics

► **Definition 1** (Syntax of TNL [28]). *Let $Prop$ be a finite set of atomic propositions and A be a finite set of agents. We define the well-formed formulas of the language of TNL, L_{TNL} , to be generated by the following grammar:*

$$\begin{aligned}\varphi &::= p \mid f_{a,b} \mid p_{a,\omega} \mid \neg\varphi \mid (\varphi \wedge \psi) \mid G\varphi \mid H\varphi \mid X\varphi \\ \omega &::= p \mid \neg\omega \mid (\omega \wedge \omega)\end{aligned}$$

where $p \in Prop$ and $a, b \in A$. We define propositional connectives like $\vee, \rightarrow$ and the formulas $\top, \perp$ as usual. Further, we define the duals $F := \neg G \neg$ and $P := \neg H \neg$.

Let Ω be the Boolean fragment of the language (i.e. the set of formulas given recursively by $\omega ::= p \mid \neg\omega \mid (\omega \wedge \omega)$). Intuitively, we read $f_{a,b}$ as “agent a follows agent b ”, $p_{a,\omega}$ as “agent a posted ω ” or “the post ω is on agent a ’s profile”. Note that an agent can only post information encoded in the Boolean fragment of the language. We read $G\varphi$ as “ φ is always the case in the future”, $H\varphi$ as “ φ is always the case in the past” and $X\varphi$ as “ φ is the case in the next time point”. The models of TNL are meant to emulate a discrete timeline with a beginning and no end; they are thus built from the natural numbers and their usual order $\mathbb{N}$.

► **Definition 2** (Temporal Network Frame and Model [28]). *A temporal network model is a tuple $M = (A, \mathbb{N}, <, (F_t)_{t \in \mathbb{N}}, (P_t)_{t \in \mathbb{N}}, V)$, where:*

- A is a finite set of agents;
- $(\mathbb{N}, <)$ is the set of natural numbers in their strict natural order;
- $P_t : A \rightarrow 2^\Omega$ is a function stating what each agent posts at each time $t \in \mathbb{N}$;
- $F_t \subseteq A^2$ is a “following” relation on the set of agents at each time $t \in \mathbb{N}$;
- $V : Prop \rightarrow 2^\mathbb{N}$ is a valuation.

We also define a pointed temporal network model (M, t) where M is a temporal network model and $t \in \mathbb{N}$ its distinguished point, at which the evaluation takes place.

We can now define the semantics of TNL.

► **Definition 3** (Semantics of TNL [28]). *Let $M = (A, \mathbb{N}, <, (F_t)_{t \in \mathbb{N}}, (P_t)_{t \in \mathbb{N}}, V)$ be a temporal network model on $\mathbb{N}$, $t \in \mathbb{N}$ be a time point, $a, b \in A$ be agents, and $\omega \in \Omega$ be a post. We inductively define the truth conditions for TNL as follows:*

$$\begin{aligned}
M, t &\models p \text{ iff } t \in V(p) \\
M, t &\models f_{a,b} \text{ iff } F_t ab \\
M, t &\models p_{a,\omega} \text{ iff } \omega \in P_t(a) \\
M, t &\models \neg\varphi \text{ iff } M, t \not\models \varphi \\
M, t &\models \varphi \wedge \psi \text{ iff } M, t \models \varphi \text{ and } M, t \models \psi \\
M, t &\models G\varphi \text{ iff } \forall s \in \mathbb{N} \text{ such that } t < s : M, s \models \varphi \\
M, t &\models H\varphi \text{ iff } \forall s \in \mathbb{N} \text{ such that } s < t : M, s \models \varphi \\
M, t &\models X\varphi \text{ iff } M, t+1 \models \varphi, \\
&\text{where } t+1 \text{ is the immediate successor of } t
\end{aligned}$$

2.2 Completeness

We continue by presenting the axiomatic system TNL and show that it is weakly complete with respect to temporal network frames on $\mathbb{N}$. As shown in [28], TNL is not compact and so therefore no axiomatisation for it can be strongly complete.

The logic TNL has the axioms and rules featured in Table 1.²

■ **Table 1** The axiomatic system TNL. Recall that $F := \neg G\neg$ and $P := \neg H\neg$.

(Taut)	All propositional tautologies	(HG)	$\varphi \rightarrow HF\varphi \wedge GP\varphi$
(K _G)	$G(\varphi \rightarrow \psi) \rightarrow (G\varphi \rightarrow G\psi)$	(wc _G)	$G(G\varphi \rightarrow \psi) \vee G(G\psi \rightarrow \varphi)$
(K _X)	$X(\varphi \rightarrow \psi) \rightarrow (X\varphi \rightarrow X\psi)$	(wc _H)	$H(H\varphi \rightarrow \psi) \vee H(H\psi \rightarrow \varphi)$
(sd)	$\neg X\varphi \leftrightarrow X\neg\varphi$	(T _G)	$G\varphi \rightarrow GG\varphi$
(XG)	$G\varphi \leftrightarrow XG\varphi \wedge X\varphi$	(T _H)	$H\varphi \rightarrow HH\varphi$
(ind)	$X\varphi \wedge G(\varphi \rightarrow X\varphi) \rightarrow G\varphi$	(XH)	$XH\varphi \leftrightarrow \varphi \wedge H\varphi$
(K _H)	$H(\varphi \rightarrow \psi) \rightarrow (H\varphi \rightarrow H\psi)$	(0 _H)	$H\perp \vee PH\perp$.
(MP)	From $\vdash \varphi \rightarrow \psi$ and $\vdash \varphi$, infer $\vdash \psi$.		
(Nec)	From $\vdash \varphi$, infer $\vdash X\varphi, \vdash G\varphi$ and $\vdash H\varphi$.		

Given a formula φ in the language of TNL we write $\models \varphi$ to denote that $M, t \models \varphi$ for every model M and every $t \in \mathbb{N}$; we write $\vdash_{\text{TNL}} \varphi$ (or simply $\vdash \varphi$) to denote that φ is derivable in the logic TNL. In this section we shall state:

► **Theorem 4** (Soundness and Weak Completeness). *For every formula φ in the language, $\models \varphi$ iff $\vdash \varphi$.*

Soundness ($\vdash \varphi$ implies $\models \varphi$) is routine; we shall devote the rest of this section to prove completeness ($\models \varphi$ implies $\vdash \varphi$). For this, it suffices to prove that any formula which is *consistent* (i.e. whose negation is not derivable) is made true in some temporal network model on $\mathbb{N}$ (for then if $\models \varphi$, it must be the case that $\vdash \varphi$, for otherwise its negation would be consistent and thus satisfied in some model).

The completeness proof we present in the sequel is non-standard: it is based on the completeness proof for *Temporal Logic with NeXt-time* [18], albeit we diverge from it substantially to offer a more simplified argument. The basic skeleton of this proof is as follows: starting from a consistent formula φ_0 , we take a finite set $\bar{T}$ of finite consistent sets of formulas, one

² Some of the axioms in Table 1 are redundant and included here for expositional clarity. In particular, the axioms T_G, T_H, wc_G, wc_H are derivable from the others (same proof as [18, Ch. 9]).

of which containing φ_0 , and we arrange the elements of $\bar{T}$ into a sequence resembling the natural numbers. After using this sequence for defining an appropriate valuation, we build a model $(\mathbb{N}, <, V)$ which we show satisfies all the formulas appearing in $\bar{T}$.

2.2.1 Maximal Consistent Sets

We say a set x of formulas in the language of TNL is *consistent* if there are no formulas $\varphi_1, \dots, \varphi_n \in x$ such that $\vdash_{\text{TNL}} \varphi_1 \wedge \dots \wedge \varphi_n \rightarrow \perp$. A formula φ is *consistent* if the set $\{\varphi\}$ is consistent. A *maximal consistent set* is a consistent set having no consistent supersets.

The proof of the following result is standard and can be found in, e.g., [10].

► **Lemma 5.**

- (1) *Given a TNL-consistent set y , there exists some TNL-maximal consistent set x such that $y \subseteq x$.*
- (2) *A TNL-maximal consistent set x enjoys the following properties: (i) For all $\varphi \in L_{\text{TNL}}$, either $\varphi \in x$ or $\neg\varphi \in x$ (but not both); (ii) $\varphi \wedge \psi \in x$ iff $\varphi, \psi \in x$; (iii) $\text{TNL} \subseteq x$.*

Let us now fix a TNL-consistent formula φ_0 ; let us find a model for it in order to prove completeness. Let us fix as well some maximal consistent set x_0 such that $\varphi_0 \in x_0$, whose existence is guaranteed by the above Lemma. Let MCS be the set of TNL-maximal consistent sets; in it, we define the relation $x < y$ iff $\{\varphi : G\varphi \in x\} \subseteq y$; we then consider the set

$$T = \{y \in \text{MCS} : x_0 = y \text{ or } x_0 < y \text{ or } y < x_0\}.$$

► **Lemma 6.**

- (i) $x < y$ iff $\{\varphi : H\varphi \in y\} \subseteq x$.
- (ii) $<$ is transitive and connected (when restricted to T).
- (iii) If $x \in T$ and $G\theta \notin x$, then there is some $y \in T$ such that $x < y, \theta \notin y$.
- (iv) If $x \in T$ and $H\theta \notin x$, then there is some $y \in T$ such that $y < x, \theta \notin y$.
- (v) If $x \in T$, then the set $x^+ := \{\theta : X\theta \in x\}$ is also in T ; moreover $x < x^+$.

Proof. The proof of this result is quite standard; we sketch it here. Item (i) is proven using the (HG) axiom, noting that $\neg\varphi \in x < y$ implies $P\neg\varphi \in y$ implies $H\varphi \notin y$. For item (ii), transitivity is a consequence of (T_G) ; for connectedness, one first uses the (wc_G) and (wc_H) axioms to prove *weak connectedness* in both directions:

if either $y > x < z$ or $y < x > z$, then either $y < z$ or $z < y$ or $y = z$;

connectedness follows since all elements in T are related to x_0 . For item (iii) (resp. (iv)), one proves consistency of the set $\{\varphi : G\varphi \in x\} \cup \{\neg\theta\}$ (resp. $\{\varphi : H\varphi \in x\} \cup \{\neg\theta\}$) and extends it to some $y \in \text{MCS}$. For item (v), one uses the (sd) axiom to show that $\vdash \varphi \rightarrow \neg\psi$ implies $\vdash X\varphi \rightarrow \neg X\psi$, whence consistency follows; maximality is also a consequence of (sd) . ◀

2.2.2 The Filtrated Tree

Building a TNL model for φ_0 requires transforming the tree-like structure $(T, <)$ into a linear order isomorphic to $\mathbb{N}$. A way to achieve this is to construct a sequence by picking out elements from some finite set $\bar{T}$, which we will obtain by *filtrating* our tree T .³

³ For other uses of this technique, see, e.g. [18, 2].

Let us consider the set of formulas

$$\Delta := \text{subf } \varphi_0 \cup \{XG\theta, X\theta : G\theta \in \text{subf } \varphi_0\} \cup \{XH\theta : H\theta \in \text{subf } \varphi_0\} \cup \{H\perp, \perp\},$$

where $\text{subf } \varphi_0$ denotes the set of subformulas of φ_0 . Note that Δ is a finite set closed under subformulas.

We filtrate T through Δ in order to obtain the finite set

$$\bar{T} := \{\bar{x} : x \in T\}, \text{ where } \bar{x} := x \cap \Delta,$$

and we define two relations on $\bar{T}$ as follows:

$$\bar{x} <^+ \bar{y} \text{ iff, for some } z \in T, \bar{z} = \bar{x} \text{ and } \bar{z}^+ = \bar{y}.$$

$$\bar{x} <^* \bar{y} \text{ iff there exists some path } \bar{x} = \bar{x}_0 <^+ \dots <^+ \bar{x}_n = \bar{y}.$$

Finally, given $x \in T$, let

$$\delta_x := \bigwedge_{\theta \in \Delta \cap x} \theta \wedge \bigwedge_{\theta \in \Delta - x} \neg \theta.$$

be the *characteristic formula of x* . Note that, by definition, $\delta_x \in z$ iff the formulas of Δ that appear in x and in z are the same, i.e. iff $\bar{x} = \bar{z}$.

► **Lemma 7.**

- (1) If $x < y$, then $\bar{x} <^* \bar{y}$.
- (2) $<^*$ is connected and transitive.

Proof.

- (1) Let $x < y$; let us see that $\bar{x} <^* \bar{y}$. For this we consider the formula $\delta := \bigvee_{\bar{x} <^* \bar{z}} \delta_z$: note that $\delta \in y$ iff $\bar{x} <^* \bar{y}$. We prove that $G(\delta \rightarrow X\delta) \in x$: indeed, suppose not. Then there is some $z > x$ with $\delta \rightarrow X\delta \notin z$. Then $\delta \in z$ and $X\delta \notin z$. The fact that $\delta \in z$ implies that $\bar{x} <^* \bar{z}$. But then consider $\bar{z}^+$: since it also holds that $\bar{x} <^* \bar{z}^+$, then $\delta \in \bar{z}^+$ which implies $X\delta \in z$: a contradiction. We also note that $\delta \in x^+$ (because $\bar{x} <^* \bar{x}^+$) and thus $X\delta \in x$. But since $G(\delta \rightarrow X\delta) \in x$ and $X\delta \in x$, we may apply the induction axiom to conclude $G\delta \in x$, which implies $\delta \in y$, which implies $\bar{x} <^* \bar{y}$.
- (2) Transitivity: if there exists a $<^+$ -chain from $\bar{x}$ to $\bar{y}$, and one from $\bar{y}$ to $\bar{z}$, then there is one from $\bar{x}$ to $\bar{z}$. Connectedness: if $x \neq y$, then either $x < y$ or $y < x$, and thus by (1), $\bar{x} <^* \bar{y}$ or $\bar{y} <^* \bar{x}$. ◀

► **Lemma 8.** Let $\bar{x} <^* \bar{y}$.

- (1) If $G\theta \in \bar{x}$, then $G\theta, \theta \in \bar{y}$.
- (2) If $H\theta \in \bar{y}$, then $H\theta, \theta \in \bar{x}$.

Proof.

- (1) By induction on the length of the chain $\bar{x} <^+ \dots <^+ \bar{y}$. If this chain is of length 1, we have $\bar{x} <^+ \bar{y}$; this means that there is some $z \in T$ with $z \cap \Delta = x \cap \Delta, z^+ \cap \Delta = y \cap \Delta$. Since $G\theta \in z < z^+$, it follows that $\theta \in z^+$; since $\vdash G\theta \rightarrow GG\theta$, it follows that $GG\theta \in z$ and thus $G\theta \in z^+$; but then we have $\theta, G\theta \in z^+ \cap \Delta = \bar{y}$. Now, assume it is true for chains of length $n - 1$ and let $\bar{x} <^+ \dots <^+ \bar{x}_{n-1} <^+ \bar{y}$; we simply apply our induction hypothesis to obtain $G\theta \in \bar{x}_{n-1}$ and we are back to the length 1 case.
- (2) Analogous to (1). ◀

2.2.3 Initial and Final Elements

We say an element $\bar{x} \in \bar{T}$ is *final* iff $\bar{y} <^* \bar{x}$ for all $\bar{y} \neq \bar{x}$; we say it is *initial* iff $\bar{x} <^* \bar{y}$ for all $\bar{y} \neq \bar{x}$. The fact that $<^*$ is a connected and transitive relation on a finite set guarantees the existence of both initial and final elements. These elements need not be unique, except in the case of the initial element:

► **Lemma 9.** $\bar{x} \in \bar{T}$ is initial iff $H\perp \in \bar{x}$. As a consequence, there is only one initial element.

Proof. From left to right, let $H\perp \in x$ and let $\bar{y} \neq \bar{x}$. Suppose towards contradiction that $\bar{x} \not<^* \bar{y}$. Then by connectedness it holds that $\bar{y} <^* \bar{x}$ and thus $\perp \in \bar{y}$ by Lemma 8, against consistency. From right to left, suppose $\bar{x} <^* \bar{y}$ for all $\bar{y} \neq \bar{x}$. But then in particular if we take some $x_1 \in T$ with $H\perp \in x_1$, it holds that $\bar{x}_1 \neq \bar{x}$, and thus $\bar{x} <^* \bar{x}_1$, which in turn implies $\perp \in x$: contradiction. There cannot be two initial elements because if $\bar{x} \neq \bar{y}$ are both initial, then we would have $\bar{x} <^* \bar{y}$ and, since $H\perp \in y$, by Lemma 8 we obtain $\perp \in \bar{x}$: a contradiction. ◀

The final elements need not be unique. Let us show however that we can put them all in a very convenient order.

► **Definition 10.** An *F-path* is a path of the form $\bar{z}_1 <^+ \dots <^+ \bar{z}_q$ satisfying $\bar{z}_q <^+ \bar{z}_1$ and wherein all (and only) final elements appear.

► **Lemma 11.** There exists some *F-path*.

Proof. Let $\bar{z}_1, \dots, \bar{z}_q$ be the final elements in $\bar{T}$. Since all are final, we have

$$\bar{z}_1 <^* \dots <^* \bar{z}_q <^* \bar{z}_1,$$

ensuring the existence of a $<^+$ -path as described above. ◀

2.2.4 The Final Model

Recall φ_0 is a consistent TNL formula and that $x_0 \in T$ is a MCS such that $\varphi_0 \in \bar{x}_0$. Let $\bar{y}_0$ be the initial element, and let $\bar{z}_1 <^+ \dots <^+ \bar{z}_q$ be some *F-path* containing all and only final elements. Since $\bar{y}_0 <^* \bar{x}_0 <^* \bar{z}_1$, there is some sequence

$$\bar{y}_0 <^+ \bar{y}_1 <^+ \dots <^+ \bar{y}_n = \bar{x}_0 <^+ \bar{x}_1 <^+ \dots <^+ \bar{x}_m = \bar{z}_1.$$

We shall construct an infinite sequence consisting of the above path concatenated with infinite repetitions of the *F-path*:

$$\bar{y}_0 <^+ \dots <^+ \bar{x}_0 <^+ \dots <^+ \bar{z}_1 <^+ \dots <^+ \bar{z}_q <^+ \bar{z}_1 <^+ \dots <^+ \bar{z}_q <^+ \bar{z}_1 <^+ \dots$$

More explicitly, the sequence in question can be written as $(\bar{a}_t)_{t \in \mathbb{N}}$ where

$$\bar{a}_t = \begin{cases} \bar{y}_t & \text{if } t = 0, \dots, n; \\ \bar{x}_{t-n} & \text{if } t = n+1, \dots, n+m; \\ \bar{z}_j & \text{iff } t > m+n \text{ and } t - (m+n) \equiv j \pmod{q} \end{cases}$$

Note that, by construction, this sequence satisfies for all $t \in \mathbb{N}$:

- i. $\bar{a}_t <^+ \bar{a}_{t+1}$,
- ii. $\bar{a}_t <^* \bar{a}_n$ whenever $t < n$,
- iii. for all final $\bar{z}$, there is some $n > t$ such that $\bar{a}_n = \bar{z}$,
- iv. the only initial element is $\bar{a}_0$.

Now we have to make $\mathbb{N}$ into a model. We define $V(p) = \{t \in \mathbb{N} : p \in \bar{a}_t\}$. For $t \in \mathbb{N}, a \in A$, we define the relation and set:

$$F_t = \{(a, b) : f_{a,b} \in a_t \cap \Delta\}, P_t(a) = \{\omega \in \Omega : p_{a,\omega} \in a_t \cap \Delta\}.$$

Let $M = (A, \mathbb{N}, <, V, (P_t)_{t \in \mathbb{N}}, (F_t)_{t \in \mathbb{N}})$. We shall prove that this is the model we are looking for; but let us first make an observation about the sequence $(\bar{a}_t)_{t \in \mathbb{N}}$:

► **Lemma 12** (Filtrated Existence Lemma).⁴ Let $t \in \mathbb{N}$.

- (1) If $X\theta \in \Delta$, then $X\theta \in \bar{a}_t$ iff $\theta \in \bar{a}_{t+1}$.
- (2) If $G\theta \in \Delta - \bar{a}_t$, then $\theta \notin \bar{a}_s$ for some $s > t$.
- (3) If $H\theta \in \Delta - \bar{a}_t$, then $\theta \notin \bar{a}_s$ for some $s < t$.

Proof.

- (1) Let $X\theta \in \Delta$. Since by construction $\bar{a}_t <^+ \bar{a}_{t+1}$, we have that there is some $z \in T$ such that $\bar{z} = \bar{a}_t$, $\bar{z}^+ = \bar{a}_{t+1}$. But then $X\theta \in \bar{a}_t$ iff $X\theta \in \bar{z}$ iff $\theta \in \bar{z}^+$ iff $\theta \in \bar{a}_{t+1}$.
- (2) Let $G\theta \in \Delta$, $G\theta \notin \bar{a}_t$; let us assume towards contradiction that $\theta \in \bar{a}_s$ for all $s > t$. We can prove by a simple induction that, in this case, $G\theta \notin \bar{a}_s$ for all $s > i$: indeed, note that $X\theta \wedge XG\theta \rightarrow G\theta$ is derivable by the (XG) axiom, and since $G\theta \notin \bar{a}_t$ and $X\theta \in \bar{a}_t$ (the latter because $\theta \in \bar{a}_{t+1}$ by hypothesis), then we must have $XG\theta \notin \bar{a}_t$, and thus $G\theta \notin \bar{a}_{t+1}$; by repeating this process taking $t+1$ instead of i , we obtain $G\theta \notin \bar{a}_{t+2}$, and so on. But then take some $m > t$ such that $\bar{a}_m$ is final (such an m exists by construction of the sequence): since $G\theta \notin \bar{a}_m$, we have $G\theta \notin a_m$, which entails there is some $y > a_m$ with $\theta \notin y$. But, since (by Lemma 7) $\bar{a}_m <^* \bar{y}$, it must then be that $\bar{y}$ is final and thus by construction there is some $s \geq m$ with $\bar{y} = \bar{a}_s$, and thus we have $t < s$ and $\theta \notin \bar{a}_s$, in contradiction with our hypothesis.
- (3) Let $H\theta \in \Delta$, $H\theta \notin \bar{a}_i$; note that we must have $i > 0$, for $\bar{a}_0$ is an initial element, which means by Lemma 9 that $H\perp \in \bar{a}_0$, which implies $H\theta \in \bar{a}_0$, noting that $H\perp \rightarrow H\theta$ is derivable. Let us assume towards contradiction that $\theta \in \bar{a}_j$ for all $j < i$. In particular, $H\theta, \theta \in \bar{a}_0$, which combined with the (XH) axiom gives us $XH\theta \in \bar{a}_0$. But this means $H\theta \in \bar{a}_1$, and since $\theta \in \bar{a}_1$ by hypothesis, we obtain with the same reasoning $H\theta \in \bar{a}_2$. Repeating this process i times we reach the contradiction $H\theta \in \bar{a}_i$. ◀

► **Proposition 13** (Truth Lemma). For any formula $\varphi \in \Delta$ and any $t \in \mathbb{N}$,

$$M, t \models \varphi \text{ iff } \varphi \in \bar{a}_t.$$

Proof. Induction on φ . The cases $\varphi = p, \theta_1 \wedge \theta_2, \neg\theta$ are routine.

Cases $\varphi = f_{a,b}, \varphi = p_{a,\omega}$: by definition of F_t and $P_t(a)$.

Case $\varphi = X\theta$: By Lemma 12, $X\theta \in \bar{a}_t$ iff $\theta \in \bar{a}_{t+1}$ iff (by induction hypothesis) $t+1 \models \theta$ iff $t \models X\theta$.

Case $\varphi = G\theta$: If $G\theta \in \bar{a}_t$, then by Lemma 8 and noting that $\bar{a}_t <^* \bar{a}_s$ whenever $t < s$, we obtain $\theta \in \bar{a}_s$ for all $s > t$, which by hypothesis gives $s \models \theta$ for all $s > t$, and thus $t \models G\theta$. Conversely if $G\theta \notin \bar{a}_t$, then by Lemma 12, we have $\theta \notin \bar{a}_s$ for some $s > t$, and thus by induction $s \not\models \theta$ for some $s > t$, which implies $t \not\models G\theta$.

Case $\varphi = H\theta$: Analogous to the previous case: $H\theta \in \bar{a}_t$ implies $t \models H\theta$ by Lemma 8, and the converse holds by Lemma 12. ◀

This ends the proof of completeness, for we have started with a consistent formula $\varphi_0 \in x_0$ and we have proven that $M, t \models \varphi_0$, where $t \in \mathbb{N}$ is the index such that $\bar{a}_t = \bar{x}_0$.

⁴ A previous version of this proof was significantly more involved; proving this lemma allows us to greatly simplify the construction of the sequence used to build the valuation for the final model. We thank Rob Goldblatt for this idea.

3 Social Relations Through Time

In [28], a multi-dimensional extension of TNL is proposed, with a more expressive language which allows us to talk about the evolution of an agent's social network through time. This is achieved by moving the set of agents A to the semantics: formulas are now read with respect to a pair (a, t) , where a is an agent and t is a time point, thus we can add formulas to our language which say, e.g., “every follower of a will satisfy p at some future time”. In this section, we present *Temporal Network Agent Logic* (TNAL) as it appears in [28], discuss a possible completeness proof and the foreseeable challenges thereof, and make a first step towards such proof by giving the complete axiomatics for a (modified) fragment of it.

3.1 Temporal Network Agent Logic

► **Definition 14** (Syntax of TNAL [28]). *The language of Temporal Network Agent Logic (henceforth, TNAL) is L_{TNAL} , built from a countable set Prop of propositional variables, a countable set Nom of nominals, and a countable set SVar of state variables with the following grammar:*

$$\begin{aligned}\varphi &::= p \mid n \mid x \mid f_{n,n} \mid p_{n,\omega} \mid \neg\varphi \mid (\varphi \wedge \psi) \mid G\varphi \mid H\varphi \mid X\varphi \mid \Box\varphi \mid \Box^{-1}\varphi \mid @_n\varphi \mid \downarrow x.\varphi \\ \omega &::= p \mid \neg\omega \mid (\omega \wedge \omega),\end{aligned}$$

where $p \in \text{Prop}$, $x \in \text{SVar}$, $n \in \text{Nom}$.

A formula $\Box\varphi$ (with dual $\Diamond\varphi$) is meant to be read “ φ holds at all of the agents that the current agent is following”, whereas $\Box^{-1}\varphi$ can be interpreted as “all those who follow the current agent φ ”. In TNAL, formulas are evaluated at pairs of agents and time points; thus, the follower relation can be interpreted in the language with a modal box. The operators $@_n\varphi$ and $\downarrow x.\varphi$, standing respectively for “agent n φ ” and “the current agent φ ”, are borrowed from *hybrid logic* [11]: each *nominal* n is a *name* that will have to be assigned to an agent by the model, whereas each *state variable* x will also refer to an agent but this reference can be determined within the formula via the $\downarrow x.$ operator. Using this language, one can express things like “All of my followers follow n ” ($\Box\Diamond^{-1}n$), “ n and m follow each other” ($@_n\Diamond m \wedge @_m\Diamond n$), n follows the current agent $\downarrow x.@_n\Diamond x$.

► **Definition 15** (Models of TNAL [28]). *Temporal network agent models (henceforth TNAL-models) are tuples $M = (A, \mathbb{N}, <, (\rightsquigarrow_t)_{t \in \mathbb{N}}, (P_t)_{t \in \mathbb{N}}, V)$*

- A is a finite set of agents;
- $(\mathbb{N}, <)$ are the natural numbers in their usual order;
- for each $t \in \mathbb{N}$, $\rightsquigarrow_t$ is a “follow” relation on the set A ;
- for each t , $P_t : A \rightarrow 2^\Omega$ is a function;
- $V : \text{Prop} \cup \text{Nom} \rightarrow 2^{A \times \mathbb{N}}$ is a valuation function satisfying the following requirement: for each $t \in \mathbb{N}$, $n \in \text{Nom}$, there exists a unique $a \in A$ such that $(a, t) \in V(n)$.

Given $a \in A$, $t \in \mathbb{N}$ we say that $n \in \text{Nom}$ names a at time t if a is the unique agent such that $(a, t) \in V(n)$; we write $\text{name}_t(n) = a$ to express this.

► **Definition 16** (Semantics of TNAL [28]). *Formulas are evaluated with respect to a temporal network agent model $M = (A, \mathbb{N}, <, (\rightsquigarrow_t)_{t \in \mathbb{N}}, (P_t)_{t \in \mathbb{N}}, V)$ and a triple (a, t, g) , where $a \in A$, $t \in \mathbb{N}$ and $g : \text{SVar} \rightarrow A$ is an assignment function, as follows:*

$M, a, t, g \models p$ iff $(a, t) \in V(p)$;	$M, a, t, g \models \Box\varphi$ iff $M, b, t, g \models \varphi$ whenever $a \rightsquigarrow_t b$;
$M, a, t, g \models n$ iff $a = \text{name}_t(n)$;	$M, a, t, g \models \Box^{-1}\varphi$ iff $M, b, t, g \models \varphi$ whenever $b \rightsquigarrow_t a$;
$M, a, t, g \models f_{nn'}$ iff $\text{name}_t(n) \rightsquigarrow_t \text{name}_t(n')$;	$M, a, t, g \models G\varphi$ iff $\forall s > t : M, s, a, g \models \varphi$;
$M, a, t, g \models x$ iff $g(x) = a$;	$M, a, t, g \models H\varphi$ iff $\forall s < t : M, s, a, g \models \varphi$;
$M, a, t, g \models p_{n\omega}$ iff $\omega \in P_t(\text{name}_t(n))$;	$M, a, t, g \models X\varphi$ iff $M, a, t+1, g \models \varphi$;
$M, a, t, g \models \neg\varphi$ iff $M, a, t, g \not\models \varphi$;	$M, a, t, g \models @_n\varphi$ iff $M, \text{name}_t(n), t, g \models \varphi$;
$M, a, t, g \models \varphi_1 \wedge \varphi_2$ iff $M, a, t, g \models \varphi_i, i = 1, 2$;	$M, a, t, g \models \downarrow x.\varphi$ iff $M, a, t, g_a^x \models \varphi$,
$M, a, t, g \models x$ iff $g(x) = a$;	where $g_a^x(x) = a, g_a^x(y) = g(y)$ for $y \neq x$.

In [28], a model checking algorithm for the TNAL framework is employed to check formulas against pairs (a, t) in the model instead of just time points t as in the non-hybrid TNL. The output of the global model checking algorithm given in [28] is the set of all pairs of agents and time points where the given formula is true; in effect, it gives in addition to at what time, also which agents exhibited the behaviour, characterised by the formula. Finding the complete logic of TNAL models is a matter of ongoing investigation. In the following subsection we offer partial results, in the form of completeness of a simplified fragment of TNAL (taking us on a journey through transfer properties of fusion logics [14, 23] and orthogonal frames [6]), we explain why standard techniques like filtration do not quite work in this framework and highlight some of the most interesting technical challenges of this framework.

3.2 Time and Friendship: A Complete Logic

Let us present a fragment of the language TNAL containing simply the temporal and “follower” modality, and define models and a semantics for it. We shall call this framework *Temporal Social Logic*, or TSL. We have to make some concessions: first, with the set of agents A no longer being part of the syntax, and in the absence of nominals, we will have to work without the $p_{a,\omega}$ and $f_{a,b}$ atomic formulas. Secondly, our models no longer have a temporal beginning and will now be based on the temporal structure on $\mathbb{Z}$, for keeping $\mathbb{N}$ would present important technical challenges, as we will later highlight. This change may be problematic if one is conceptually committed to the beginning of time, but we argue it is a harmless change for most use cases. Thirdly, we do not demand that the set of agents A be finite; finding this framework’s version of the *finite model property* is still a pending task. We make one further concession: we consider *symmetric* social relations; by doing so, we model the simplest case wherein the “follower” relation acts more like a “friendship” relation and the $\Box^{-1}$ operator becomes redundant. We note that the proof below works the same way if one removes symmetry, by simply removing the (*sym* $_{\Box}$) axiom.

► **Definition 17** (Syntax and Semantics of TSL). *Given a countable set Prop of propositional variables, we let L_{TSL} be the set of formulas defined inductively as follows:*

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \Box\varphi \mid G\varphi \mid H\varphi \mid X\varphi,$$

with $p \in \text{Prop}$. Models for TSL are tuples $M = (A, \mathbb{Z}, <, (\rightleftharpoons_t)_{t \in \mathbb{Z}}, V)$, where A is a set of agents, $(\mathbb{Z}, <)$ is the set of integers in its natural order, $\rightleftharpoons_t \subseteq A^2$ is a symmetric relation for each t , and $V : \text{Prop} \rightarrow 2^{A \times \mathbb{Z}}$ is a valuation.

Formulas of L_{TSL} are read on a model M with respect to a pair $(a, t) \in A \times \mathbb{Z}$ as follows:

$M, a, t \models p$ iff $(a, t) \in V(p)$;	$M, a, t \models G\varphi$ iff $M, a, s \models \varphi$ for all $s > t$;
$M, a, t \models \varphi_1 \wedge \varphi_2$ iff $M, a, t \models \varphi_i$ for $i = 1, 2$;	$M, a, t \models H\varphi$ iff $M, a, s \models \varphi$ for all $s < t$;
$M, a, t \models \neg\varphi$ iff $M, a, t \not\models \varphi$;	$M, a, t \models X\varphi$ iff $M, a, t+1 \models \varphi$;
	$M, a, t \models \Box\varphi$ iff $M, b, t \models \varphi$ for all $b \rightleftharpoons_t a$.

We shall show that the complete logic for this framework is simply the fusion of the logics corresponding to the “temporal” and “social” components of the framework. We thus further divide the language into two fragments, only to then combine the resulting logics.

3.2.1 The Temporal Fragment

The *temporal fragment* of this language, L_T , is comprised of formulas of the shape

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg \varphi \mid G\varphi \mid H\varphi \mid X\varphi.$$

Note that this, minus the $p_{a,\omega}$ and $f_{a,b}$ atoms, is almost identical to the language of TNL. And so are its models: a T-model is a structure $(\mathbb{Z}, <, V)$, where $V : Prop \rightarrow 2^{\mathbb{Z}}$ is a valuation and formulas are otherwise read as in Definition 3. The logic of these models based on $\mathbb{Z}$ is exactly like TNL minus the axiom establishing a beginning. Indeed, let $TNL_{\leftrightarrow}$ be the least normal L_T -modal logic having the same axioms and rules as TNL except (0_H) . We have:

► **Proposition 18.** $TNL_{\leftrightarrow}$ is the logic of T-models.

Proof. Very similar to the proof of completeness of TNL; see the Appendix for a sketch. ◀

3.2.2 The Social Fragment

The *social fragment* of this language is L_S , given by $\varphi ::= p \mid \varphi \wedge \varphi \mid \neg \varphi \mid \Box \varphi$, with $p \in Prop$. Models for L_S (henceforth *S-models*) are of the shape $(A, \rightleftharpoons, V)$, where A is a set of agents and $\rightleftharpoons$ is a symmetric relation. The sound and complete logic of these models is the normal modal logic whose axioms are

$$(K_{\Box}) \Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi) \quad (sym_{\Box}) \varphi \rightarrow \Box\Diamond\varphi,$$

with Modus Ponens (from φ and $\varphi \rightarrow \psi$, infer ψ) and Necessitation (from φ , infer $\Box\varphi$) as inference rules. Let us call this logic SL. The proof of the following result is standard in modal logic (see, e.g., [10]) and not included:

► **Proposition 19.** SL is the logic of (finite) S-models.

3.3 Combining Both Fragments

The sound and complete logic of TSL models is the fusion of the above logics, i.e., the least normal modal logic containing the axioms of TNL and SL, as shown in Table 2. Let us refer to this logic as TSL.

Let us prove this is indeed the complete logic of TSL models. For this, we will have to visit the proofs of some well-known transfer results for fusion logic; we start by describing a new class of frames:

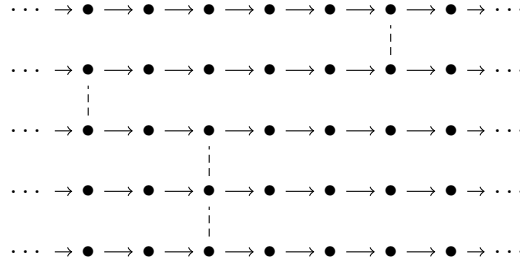
- **Definition 20 (Sproutings).** A sprouting⁵ is a birelational model $(X, <, \rightleftharpoons, V)$ satisfying:
- (S1) Every connected component⁶ of the $<$ relation (henceforth a timeline) is a T-frame (i.e. isomorphic to $(\mathbb{Z}, <)$).
 - (S2) Every connected component of the $\rightleftharpoons$ relation is an S-frame.

⁵ We borrow this nomenclature from Fine and Schurz [14].

⁶ Connected component: equivalence class of the least equivalence relation containing $<$.

■ **Table 2** The logic TSL.

(<i>Taut</i>)	All propositional tautologies	(<i>wc_G</i>)	$G(G\varphi \rightarrow \psi) \vee G(G\psi \rightarrow \varphi)$
(<i>K_G</i>)	$G(\varphi \rightarrow \psi) \rightarrow (G\varphi \rightarrow G\psi)$	(<i>wc_H</i>)	$H(H\varphi \rightarrow \psi) \vee H(H\psi \rightarrow \varphi)$
(<i>K_X</i>)	$X(\varphi \rightarrow \psi) \rightarrow (X\varphi \rightarrow X\psi)$	(<i>T_G</i>)	$G\varphi \rightarrow GG\varphi$
(<i>sd</i>)	$\neg X\varphi \leftrightarrow X\neg\varphi$	(<i>T_H</i>)	$H\varphi \rightarrow HH\varphi$
(<i>XG</i>)	$G\varphi \leftrightarrow XG\varphi \wedge X\varphi$	(<i>XH</i>)	$XH\varphi \leftrightarrow \varphi \wedge H\varphi$
(<i>ind</i>)	$X\varphi \wedge G(\varphi \rightarrow X\varphi) \rightarrow G\varphi$	(<i>K_□</i>)	$\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$
(<i>K_H</i>)	$H(\varphi \rightarrow \psi) \rightarrow (H\varphi \rightarrow H\psi)$	(<i>sym_□</i>)	$\varphi \rightarrow \Box\Diamond\varphi$
(<i>HG</i>)	$\varphi \rightarrow HF\varphi \wedge GP\varphi$		
(MP)	From $\vdash \varphi \rightarrow \psi$ and $\vdash \varphi$, infer $\vdash \psi$		
(Nec)	From $\vdash \varphi$, infer $\vdash \Box\varphi$, $\vdash X\varphi$, $\vdash G\varphi$ and $\vdash H\varphi$		



■ **Figure 2** A finite sprouting with five timelines. The dashed lines represent the $\equiv$ relation.

(S3) Given a closed loop of timelines $t_0 \equiv t_1 \equiv \dots \equiv t_n \equiv t_0$, there exist unique elements $x_i \in t_i$ such that $x_0 \equiv \dots \equiv x_n \equiv x_0$, where we define

$$t \equiv t' \text{ iff } x \equiv x' \text{ for some } x \in t, x' \in t'.$$

Note that this entails that two timelines are connected by $\equiv$ at no more than one point, and, while points are allowed to be reflexive, no two points in the same timeline are connected by $\equiv$. A sprouting is said to be finite (resp. countable) if it has a finite (resp. countable) number of timelines. Given a sprouting $(X, <, \equiv, V)$, and given $x \in X$, let

$$t_x := \{y \in X : y < x \text{ or } y > x \text{ or } y = x\}$$

denote the timeline of x , and let

$$x^+ := \min\{y \in X : x < y\}, \quad x^- := \max\{y \in X : y < x\}$$

denote respectively the successor and predecessor of x . Note that, by definition, every point of X has a successor and predecessor, and $y \in t_x$ iff there is some n such that either $y = x^{+n}$ or $y = x^{-n}$ (where the $\pm n$ superindices denote the concatenation of the $\pm$ operators n times).

We read formulas of TSL on sproutings as follows:

► **Definition 21** (Semantics for Sproutings.). Given a sprouting $S = (X, <, \equiv, V)$ and formula φ in the language of TSL, we say φ is true at $x \in X$, denoted $S, x \models \varphi$, if:

$$\begin{array}{llll}
 S, x \models p & \text{iff } x \in V(p); & S, x \models G\varphi & \text{iff } S, y \models \varphi \text{ for all } y > x; \\
 S, x \models \neg\varphi & \text{iff } S, x \not\models \varphi; & S, x \models H\varphi & \text{iff } S, y \models \varphi \text{ for all } y < x; \\
 S, x \models \varphi \wedge \psi & \text{iff } S, x \models \varphi \text{ and } S, x \models \psi; & S, x \models X\varphi & \text{iff } S, x^+ \models \varphi; \\
 & & S, x \models \Box\varphi & \text{iff } S, y \models \varphi \text{ for all } y \equiv x.
 \end{array}$$

As stated earlier, our goal here is to use these auxiliary structures in order to prove that TSL is the complete logic of TSL models. This will be done in two steps: first, we shall show that TSL is the logic of countable sproutings. Then we prove that countable sproutings and TSL models satisfy the same formulas, whence completeness follows.

Let us show these results in order. The first is already proven in the literature: it appears (albeit hidden) in the proof of a well-known result on fusion logics (see e.g. [14, 15, 23]).

► **Proposition 22** (TSL is the logic of sproutings). *If a formula φ is TSL-consistent, then there is a countable sprouting satisfying φ .*

Proof sketch. This result is proven in [14, 23]; see the Appendix for a proof sketch. ◀

All that remains to finish our completeness proof is to transform a countable sprouting into a regular TSL model.

► **Proposition 23** (Sproutings and TSL models satisfy the same formulas.). *If a formula φ_0 is satisfied on a countable sprouting, then there is some TSL model satisfying φ_0 .*

Proof. Let $S = (X, <, \rightleftharpoons, V)$ be a countable sprouting, and let $x_0 \in X$ such that $S, x_0 \models \varphi_0$. Let t_0 be the timeline with $x_0 \in t_0$. We may assume without loss of generality that, for any two timelines t_i and t_j , there is some path $t_i \rightleftharpoons \dots \rightleftharpoons t_j$ (for otherwise we could just take the submodel of S consisting of timelines reachable by t_0 , and a trivial induction shows this submodel also satisfies φ). We shall now take a function $f : X \rightarrow \mathbb{Z}$ that assigns an integer to each point in X satisfying the following properties, for all $x, y \in X$:

$$(F1) \ f(x^+) = f(x) + 1, f(x^-) = f(x) - 1; \quad (F2) \ x \rightleftharpoons y \text{ implies } f(x) = f(y).$$

In the Appendix, we show that:

► **Lemma 24.** *There is some function $f : X \rightarrow \mathbb{Z}$ satisfying (F1) and (F2).*

Take some such function f ; note that by (F1), the restriction of f to any timeline is an order isomorphism. For any timeline t and integer n , let $f_t^{-1}(n)$ denote the unique $x \in t$ such that $f(x) = n$. Finally, consider the tuple $M = (A, \mathbb{Z}, <, (\rightleftharpoons_n)_{n \in \mathbb{Z}}, V^*)$, where:

- A is the set of timelines of S ;
- For each $n \in \mathbb{Z}$, $t_i \rightleftharpoons_n t_j$ iff there are $x \in t_i, y \in t_j$ such that $x \rightleftharpoons y$ and $f(x) = f(y) = n$.
- $V^*(p) = \{(t, n) : f_t^{-1}(n) \in V(p)\}$.

This is a TSL model (indeed, each $\rightleftharpoons_n$ relation is symmetric); therefore the following Lemma (proven in the Appendix) finishes our proof:

► **Lemma 25.** *For any formula ψ , $S, x \models \psi$ iff $M, (t_x, f(x)) \models \psi$.*

This proves that M validates φ_0 at $(t_0, 0)$, as we desired to show. ◀

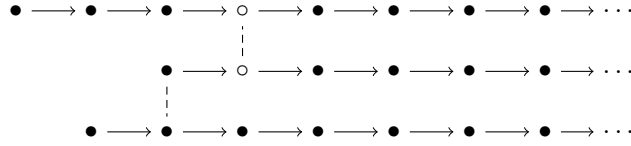
3.4 Towards a Complete Logic for TNAL

In the present and previous sections we have axiomatised fragments of L_{TNAL} ; finding a complete logic for the whole framework remains an open problem. It is not an easy problem: many parts of this framework are non-standard and require rather esoteric proof techniques. The proof we showed in this section illustrates some of these difficulties, while shedding light on some interesting properties of our models. We highlight some of these below:

Finite Agents. The TNAL framework is reasonably defined to have finitely many agents; in this first foray into its completeness proof we have allowed for up to countably many. The reason is that this framework is resistant to the usual technique of *filtration*: models for TNAL are infinite bidimensional models which are countably infinite in one dimension (the timeline) yet finite in the other (the agents); this is in contrast to frameworks such as [22], where the agents are defined at the syntax level. Now, of course, any filtration in our models would undesirably make the whole model finite, and any careless attempt to unravel our finite filtrated model back into a timeline risks adding back infinite agents. We do not know of many bidimensional frameworks with such a cardinality requirement. Some novel filtration technique will most likely have to be developed for this.

Timelines. Famously, models based on timelines require care in their completeness proofs, since a canonical model does not usually have the desired structure of a model: being countable (for the set), being irreflexive (for $<$) are not modally definable, and the functionality of the successor relation is usually not preserved through filtration. The technique of “pre-filtrating” the canonical model and constructing a sequence from the finite filtrated set (see [18], [2] and Section 2 of this paper) does not translate well into our bidimensional framework. For this technique to work, some non-trivial adaptations are in order.

The Beginning of Time. As we noted earlier, removing the requirement that our timelines have a beginning simplifies the proof. Here is how: Suppose we take the framework TSL_0 , defined exactly as TSL except with models based on $\mathbb{N}$. One would expect the logic of this framework to be the fusion of TNL with SL. However, if one tries to prove this using the same techniques in this section, one can easily get sproutings with the following kind of shape:



Our numbering trick from Prop. 23 does not work here: there is no way to number the elements in this sprouting so that (i) each timeline is properly numbered and (ii) the two white dots in the drawing have the same number assigned. It is certainly not straightforward to avoid these models wherein an agent finds herself at time t yet has a friend who finds himself at time $s \neq t$. This is not a problem in $\mathbb{Z}$, where given a discrete sequence infinite in both directions, we can take any element to be its zero, and adjust the numbering of its neighbour timelines accordingly. Is there some trick to redefine sproutings in the framework based on $\mathbb{N}$ so that this numbering works? We dare to conjecture there is not: a crucial difference between the sociotemporal models based on $\mathbb{Z}$ and those based on $\mathbb{N}$ is that, in the former, no meaningful theorems express any interaction between the social and temporal dimensions (recall that TSL has no axioms combining social and temporal modalities), whereas in the latter some meaningful multimodal formulas will be theorems of the logic: for instance, the formula $H\perp \rightarrow \Box H\perp$, expressing “if I am at time $t = 0$, then all my friends are at time $t = 0$ ” is clearly valid in TNAL (and thus any sociotemporal subframework thereof). We thus expect the logic of sociotemporal models based on $\mathbb{N}$ to include some interaction axioms between the modalities H, G, X on one hand and $\Box$ on the other. This would take us away from the theory of fusion logics and require a different proof technique. It is most

curious how this seemingly simple change in the structure of the models creates a significant difference in expressive power: in the models based on $\mathbb{N}$, our agents apparently have a way to keep track of time, something impossible in models based on $\mathbb{Z}$.⁷

Hybrid Operators. Languages with hybrid modalities have historically been resistant to proofs based on canonical models; see for example [34], where a completeness proof based on tree sequent calculus is given for Epistemic Friendship Logic [35]. A recent technique [7] involves considering maximal consistent sets of *named formulas*, i.e., formulas preceded by the $@_n$ modality; noting that the $@_n$ operator is self-dual ($\neg @_n \varphi \leftrightarrow @_n \neg \varphi$ is valid in TNAL), one has that, if x is a maximal consistent set and n is a fixed nominal, then $x_n = \{\varphi : @_n \varphi \in x\}$ is also a maximal consistent set. The aforementioned technique uses this fact to encode the name of an agent directly onto the canonical model.

4 Conclusion and Future Work

In this paper, we solved the open problem of completeness of Temporal Network Logic and discussed steps towards proving completeness of its hybrid extension Temporal Network Agent Logic. These steps included introducing the intermediate logic Temporal Social Logic as well as its completeness proof. The aim for future work is to build on techniques from the completeness proof for TSL to develop a proof for the full hybrid extension of TNL. As discussed in the previous section, this is a task that will likely require mixing techniques from fusion logics [14, 23], hybrid logics [11, 7], temporal logics [18, 2] and orthogonal frames [6], and perhaps coming up with a few novel ones.

Decidability remains an open problem; we conjecture TNAL to be undecidable due to the undecidability of the hybrid logic $\mathcal{H}(@, \downarrow)$ [3] and certain epistemic-temporal logics [22, 39].

In on-going work, we are also modelling the bot behaviour that one observes in specific multi-agent systems, not in temporal models as in this paper, but by using the techniques coming from Dynamic Epistemic Logic (DEL) [8, 40]. This allows us to model the dynamics of posting-events and (un)following social actions in tandem, using two types of events each of which comes with its own dynamic operators. Such models would be two-dimensional, similar to those that have operators involving both behaviour change and social network changes in [38, 5]. DEL is particularly well suited to make the dynamic mechanisms of this type of information change explicit in such multi-agent systems. Moreover, the DEL framework will make it easier to include also propositions about the epistemic information states of the agents in the language and to allow for ‘posts’ that go beyond the Boolean fragment of the logic. In DEL, learning a piece of incoming information from a highly trusted source or from a source that is considered to be unreliable, will lead to a different result in the updated information state of the learner. These information states can then feature as the precondition for an agent to follow or unfollow another agent in the network. Formally, such a setting will require us to model the temporal flow of the model as a sequence of step-wise generated models, each of which is the result of a specific event that acts as a model-transformer on the model of the previous step. Such sequences of event-generated models are typically only future oriented in standard DEL but extensions exist so that one can also look back into the past using the language and framework described in [33].

⁷ Note as well that a formula such as $P(p \wedge \neg q) \wedge P(q \wedge \neg p)$ can only be true at timepoints $t \geq 2$ in models based on $\mathbb{N}$, yet can be true anywhere in $\mathbb{Z}$!

Another venue for future work includes other extensions to the framework for reasoning about further phenomena on social networks. A path left to explore is to add agents' opinions to the network; this could let us model that certain posts by one agent could trigger opinion change in another. Existing work using logic to model opinion and opinion change in social networks includes [13, 12, 17, 21].

References

- 1 Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002. doi:10.1145/585265.585270.
- 2 Hajnal Andréka, Valentin Goranko, Szabolcs Mikulas, István Németi, and Ildikó Sain. Effective temporal logics of programs. In *Time & Logic*, pages 51–129. Routledge, December 1995. doi:10.4324/9780429321047-2.
- 3 Carlos Areces, Patrick Blackburn, and Maarten Marx. Hybrid logics: characterization, interpolation and complexity. *Journal of Symbolic Logic*, 66(3):977–1010, 2001. doi:10.2307/2695090.
- 4 Carlos Areces and Balder ten Cate. Hybrid logics. In Johan van Benthem, Patrick Blackburn, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 821–868. Elsevier, Amsterdam, 2007. doi:10.1016/S1570-2464(07)80017-6.
- 5 Edoardo Baccini, Zoé Christoff, and Rineke Verbrugge. Dynamic logics of diffusion and link changes on social networks. *Studia Logica*, 2024. doi:10.1007/s11225-024-10126-0.
- 6 Philippe Balbiani and Saúl Fernández González. Orthogonal frames and indexed relations. In *International Workshop on Logic, Language, Information, and Computation*, pages 219–234. Springer, 2021. doi:10.1007/978-3-030-88853-4_14.
- 7 Philippe Balbiani and Saúl Fernández González. Indexed frames and hybrid logics. In *International Conference on Advances in Modal Logic (AiML 2020)*, volume 13, pages 56–72. College Publications, 2020.
- 8 Alexandru Baltag and Bryan Renne. Dynamic Epistemic Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2016 edition, 2016.
- 9 Francesco Belardinelli and Davide Grossi. On the formal verification of diffusion phenomena in open dynamic agent networks. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 237–245, 2015. URL: <http://dl.acm.org/citation.cfm?id=2772912>.
- 10 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, Cambridge, 2001. doi:10.1017/CB09781107050884.
- 11 Patrick Blackburn and Balder ten Cate. Pure extensions, proof rules, and hybrid axiomatics. *Studia Logica*, 84(2):277–322, 2006. doi:10.1007/S11225-006-9009-6.
- 12 Zoé Christoff and Davide Grossi. Stability in binary opinion diffusion. In Alexandru Baltag, Jeremy Seligman, and Tomoyuki Yamada, editors, *Logic, Rationality, and Interaction*, pages 166–180, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg. doi:10.1007/978-3-662-55665-8_12.
- 13 Zoé Christoff, Jens Ulrik Hansen, and Carlo Proietti. Reflecting on social influence in networks. *Journal of Logic, Language and Information*, 25(3-4):299–333, 2016. doi:10.1007/s10849-016-9242-y.
- 14 Kit Fine and Gerhard Schurz. Transfer theorems for multimodal logics. *Logic and reality: essays on the legacy of Arthur Prior*, pages 169–213, 1996.
- 15 Dov M. Gabbay, Agi Kurucz, Frank Wolter, and Michael Zakharyashev. Many-dimensional modal logics: theory and applications. *Studia Logica*, 81(1), 2005.
- 16 Rustam Galimullin, Munyque Mittelman, and Laurent Perrussel. Formal verification of diffusion auctions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(23):19099–19107, 2026. doi:10.1609/aaai.v40i23.38983.

- 17 Rustam Galimullin and Mina Young Pedersen. Visibility and exploitation in social networks. *Mathematical Structures in Computer Science*, 34(7):615–644, 2024. doi:10.1017/S0960129523000397.
- 18 Robert Goldblatt. *Logics of time and computation*. Center for the Study of Language and Information, USA, 1987.
- 19 Saúl Fernández González. *Logics for social networks: Asynchronous announcements in orthogonal structures*. PhD thesis, Université Paul Sabatier-Toulouse III, 2021.
- 20 Valentin Goranko. Coalition games and alternating temporal logics. In *Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '01, pages 259–272, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
- 21 Umberto Grandi, Emiliano Lorini, Arianna Novaro, and Laurent Perrussel. Strategic disclosure of opinions on a social network. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 1196–1204, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems. URL: <http://dl.acm.org/citation.cfm?id=3091292>.
- 22 Joseph Y Halpern and Moshe Y Vardi. The complexity of reasoning about knowledge and time. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 304–315, 1986.
- 23 Marcus Kracht and Frank Wolter. Properties of independently axiomatizable bimodal logics. *The Journal of Symbolic Logic*, 56(4):1469–1485, 1991. doi:10.2307/2275487.
- 24 Vitor Machado and Mario Benevides. Temporal logic for social networks. *Journal of Logic and Computation*, 32(6):1088–1108, 2022. doi:10.1093/logcom/exac023.
- 25 Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Logic*, 15(4), 2014. doi:10.1145/2631917.
- 26 Rohit Parikh and Ramaswamy Ramanujam. A knowledge based semantics of messages. *Journal of Logic, Language and Information*, 12(4):453–467, 2003. doi:10.1023/A:1025007018583.
- 27 Marc Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002. doi:10.1093/LOGCOM/12.1.149.
- 28 Mina Young Pedersen. *Malicious Agents and the Power of the Few: On the Logic of Abnormality in Social Networks*. PhD thesis, Department of Information Science and Media Studies, Universitetet i Bergen, Bergen, Norway, 2024. URL: <https://hdl.handle.net/11250/3151733>.
- 29 Mina Young Pedersen, Marija Slavkovik, and Sonja Smets. Social bot detection as a temporal logic model checking problem. In Sujata Ghosh and Thomas Icard, editors, *Logic, Rationality, and Interaction*, pages 158–173, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-88708-7_13.
- 30 Mina Young Pedersen, Marija Slavkovik, and Sonja Smets. Detecting bots with temporal logic. *Synthese*, 202(3):79, 2023. doi:10.1007/s11229-023-04264-6.
- 31 Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a BDI-architecture. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991. URL: <http://jmvidal.cse.sc.edu/library/rao91a.pdf>.
- 32 Anand S. Rao and Michael P. Georgeff. Bdi agents: From theory to practice. In *Proceedings of the 1st Int. Conf. on Multi-Agent Systems (ICMAS95)*, 1995, pages 312–319, 1995.
- 33 Joshua Sack. *Adding Temporal Logic to Dynamic Epistemic Logic*. PhD thesis, Indiana University, 2007.
- 34 Katsuhiko Sano. Axiomatizing epistemic logic of friendship via tree sequent calculus. In *International Workshop on Logic, Rationality and Interaction*, pages 224–239. Springer, 2017. doi:10.1007/978-3-662-55665-8_16.

- 35 Jeremy Seligman, Fenrong Liu, and Patrick Girard. Logic in the community. In *Indian conference on logic and its applications*, pages 178–188. Springer, 2011. doi:10.1007/978-3-642-18026-2_15.
- 36 Chengcheng Shao, Giovanni Luca Ciampaglia, Onur Varol, Kai-Cheng Yang, Alessandro Flammini, and Filippo Menczer. The spread of low-credibility content by social bots. *Nature Communications*, 9(1):4787, 2018.
- 37 Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, Cambridge, UK, 2009.
- 38 Sonja Smets and Fernando R. Velázquez-Quesada. A logical analysis of the interplay between social influence and friendship selection. In Luís Soares Barbosa and Alexandru Baltag, editors, *Dynamic Logic. New Trends and Applications - Second International Workshop, DaLi 2019, Porto, Portugal, October 7-11, 2019, Proceedings*, volume 12005 of *Lecture Notes in Computer Science*, pages 71–87, Cham, 2020. Springer. doi:10.1007/978-3-030-38808-9_5.
- 39 Edith Spaan. Nexttime is not necessary. In *Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 241–256, 1990.
- 40 Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*. Springer, 2008.

A

 Appendix

Proof sketch of Prop. 18. This proof is almost identical to the proof of completeness for TNL in the Section 2, with the following changes:

- There is no longer a unique initial element in the filtrated tree $\bar{T}$. Instead, we have multiple initial elements $\bar{i}_1, \dots, \bar{i}_p$ and we can arrange them into an *I-path*, $\bar{i}_1 <^+ \dots <^+ \bar{i}_p$, where $\bar{i}_p <^+ \bar{i}_1$. The existence of such a path is proven just like in Lemma 11.
- Given $\bar{x}_0 \in \bar{T}$ such that $\varphi_0 \in \bar{x}_0$, we build a sequence of elements of $\bar{T}$ much like in Section 2.2.4, except it is now isomorphic to $\mathbb{Z}$, i.e. infinite in both directions: first we take a finite $<^+$ -path that goes from some initial element (say $\bar{i}_p$) to some final element (say $\bar{f}_1$), going through $\bar{x}_0$; we then plug a infinite repetition of an *F-path* to its right, and an infinite repetition of the *I-path* to the left:

$$\dots <^+ \bar{i}_p <^+ \underbrace{\bar{i}_1 <^+ \dots <^+ \bar{i}_p}_{I\text{-path}} <^+ \dots <^+ \bar{x}_0 <^+ \dots <^+ \underbrace{\bar{f}_1 <^+ \dots <^+ \bar{f}_q}_{F\text{-path}} <^+ \bar{f}_1 <^+ \dots$$

- We assign integers to the elements of this sequence, assigning 0 to $\bar{x}_0$ and numbering the rest according to their relative position to $\bar{x}_0$; we use this to build a valuation on $(\mathbb{Z}, <)$ and the rest of the proof proceeds like in Lemma 12 and Proposition 13, noting one no longer needs the (0_H) axiom to prove item (3) of Lemma 12, which is instead proven exactly like item (2). ◀

Proof sketch of Prop. 22. This result is proven in [14] and [23] for sproutings based on arbitrary complete normal modal logics L_1 and L_2 . The notion of “sprouting” in the present paper is simplified to fit our particular logics, but generally a *sprouting* is a birelational frame (W, R_1, R_2) than can be built as the union of L_1 -frames and L_2 -frames in a way such that the different L_i -frames have no points in common and, for $i \neq j$, any of the L_i -frames and any of the L_j -frames have at most one point in their intersection.

One shows that, for any formula φ which is consistent in the fusion of both logics $L_1 \oplus L_2$, there is a sprouting satisfying φ . This is used to prove a weaker result: namely that the fusion $L_1 \oplus L_2$ is complete with respect to frames (W, R_1, R_2) where (W, R_i) is an L_i -frame for $i = 1, 2$.

Let us see a sketch of this construction for our particular pair of logics:

Given a formula φ in the language of TSL, a *T-constituent* of φ is any maximal subformula of φ which is preceded by one of the temporal operators G, H or X (or their duals); whereas an *S-constituent* is a maximal subformula preceded by $\Box$ (or $\Diamond$).

The *T-projection* of φ , denoted φ^T , is the formula in the language of TNL resulting from substituting the *S*-constituents of φ with “surrogate” propositional variables; its *S-projection*, φ^S , is the formula in the language of SL obtained by replacing the *T*-constituents with surrogates.

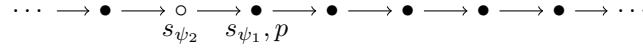
(For example, take the formula $\varphi = F(p \wedge \Diamond Xq) \wedge \Diamond Pr$. Its *S*-constituents are $\psi_1 = \Diamond Xq$ and $\psi_2 = \Diamond Pr$; its *T*-projection can be written as $\varphi^T = F(p \wedge s_{\psi_1}) \wedge s_{\psi_2}$, where s_{ψ_1} and s_{ψ_2} are fresh “surrogate” variables.)

It holds that, if φ is a consistent formula in the fusion logic TSL, then both its projections φ^T and φ^S are consistent in their respective fragments (note that if, e.g., φ^T were inconsistent in TNL, then $\neg\varphi^T$ would be a theorem of TSL since $\text{TNL} \subseteq \text{TSL}$, and so would $\neg\varphi$, obtained from φ^T via uniform substitution; likewise for ψ^S).

The process of building the sprouting that satisfies φ is done in a number of steps no larger than the modal depth of φ ; it goes as follows:

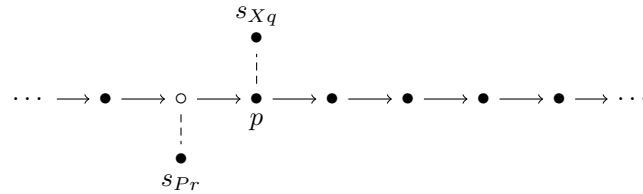
- First, one constructs a T-model satisfying the *T*-projection φ^T . Some of the points in this model will satisfy the “surrogate” variables of this formula.

Using our example formula $\varphi = F(p \wedge \Diamond Xq) \wedge \Diamond Pr$ to illustrate this, we have the following model satisfying φ^T at the white point:



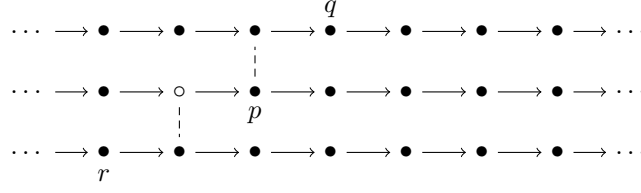
- We now take each of these points satisfying some surrogate variable s_{ψ} and we make it into the root of a finite S-model satisfying ψ^S . Note that the (possibly infinitely many) models obtained in this step have no points in common amongst themselves and at most one point in common with the model built in the previous step.

In our example, we compute the projections, $\psi_1^S = (\Diamond Xq)^S = \Diamond s_{Xq}$ and $\psi_2^S = (\Diamond Pr)^S = \Diamond s_{Pr}$, and we make a finite S-model for ψ_2^S sprout from the white point and one for ψ_1^S sprout from its successor:



- Again, some of the points in these S-models will satisfy some surrogate variables p_{θ}^T . For each of these, we construct a T-model satisfying θ^T at some point and we “plug it” into the point satisfying p_{θ}^T . Again, we note that the models obtained in this step are disjoint among themselves, have no points in common with any T-models obtained in previous steps, and each of them has at most one point in common with at most one S-model obtained in the previous step.

In our running example:



- We repeat this process, alternating T-models and S-models that hang off each other, until the formula is exhausted.

Some caveats are necessary to make sure this construction does not backfire: for instance, the authors of [23] work with partial valuations and have some restrictions on the types of models one can use in the alternating sequence, whereas those of [14] circumvent this by skipping the step-by-step construction and using Zorn's Lemma to postulate the existence of a maximal sprouting; likewise, one has to be careful with Boolean combinations of surrogate variables. We point the reader to the aforementioned texts to fill out the finer details.

What is relevant for our purposes is that, at the end of this construction, we are left with a birelational model $M = (X, <, \rightleftharpoons, V)$ that satisfies φ and, by construction, is a countable sprouting. ◀

Proof of Lemma 24. We construct f by defining it first on $D_0 = t_0$ and successively augmenting its domain to the sets $D_1, D_2, \dots$, where

$$D_{n+1} = D_n \cup \{x : t \rightleftharpoons t_x \text{ for some } t \in D_n\}.$$

(In other words, D_n is the union of all timelines separated from t_0 by a $\rightleftharpoons$ -path of length at most n ; note that every timeline is a subset of some D_n , and that $X = \bigcup_{n \in \mathbb{N}} D_n$).

- First, assign $f(x_0) = 0$. Then, for every element $x \in t_0$, assign a number corresponding to the relative position of this element to x_0 in the timeline (i.e. $f(x_0^+) = 1, f(x_0^{++}) = 2, f(x_0^-) = -1$, etc.)
- Once all the elements of D_n have a number assigned, take a timeline $t \subseteq D_{n+1} \setminus D_n$; there is some timeline $t' \subseteq D_n$ such that $t' \rightleftharpoons t$; there is thus a unique pair of points $x \in t, x' \in t'$ such that $x \rightleftharpoons x'$. Set $f(x) := f(x')$ and assign values to the rest of the elements in t according to their relative position to x ($f(x^+) = f(x) + 1, f(x^-) = f(x) - 1, \dots, f(x^{\pm n}) = f(x) \pm n$).

Note that this assignment is well-defined on account of property (S3) of the definition of *sprouting*: if $t'' \subseteq D_n$ is another timeline such that $t'' \rightleftharpoons t$, then, remembering that all timelines have a symmetric $\rightleftharpoons$ -path to t_0 , we take a closed loop

$$t_0 \rightleftharpoons \dots \rightleftharpoons t' \rightleftharpoons t \rightleftharpoons t'' \rightleftharpoons \dots \rightleftharpoons t_0$$

and we apply (S3) to conclude that t'' and t are also connected at point x and that $f(x') = f(x'')$, where x'' is the unique point in t'' such that $x'' \rightleftharpoons x$.

By this inductive process, we are left with a function $f : X \rightarrow \mathbb{Z}$ which, by construction, satisfies (F1) and (F2). ◀

Proof of Lemma 25. Simple induction on ψ ; the case $\psi = p$ follows from the definition of V^* , whereas the cases $\psi = \neg\theta, \psi = \theta_1 \wedge \theta_2$ are trivial.

Case $\psi = G\theta$: $S, x \models G\theta$ iff $S, y \models \theta$ for all $y > x$, iff (by induction hypothesis) $M, (t_y, f(y)) \models \theta$ for all $y > x$, iff (noting that for all $m > f(x)$ there is some $y > x$ with $f(y) = m, t_y = t_x$), iff $M, (t_x, m) \models \theta$ for all $m > f(x)$, iff $M, (t_x, f(x)) \models G\theta$.

Cases $\psi = H\theta, \psi = X\theta$: analogous to the previous case.

Case $\psi = \Box\theta$: $S, x \models \Box\theta$ iff $S, y \models \theta$ for all $y \rightleftharpoons x$, iff (by hypothesis) $M, (t_y, f(y)) \models \theta$ for all $y \rightleftharpoons x$ iff (remembering that $f(x) = f(y)$ when $x \rightleftharpoons y$) $M, (t_y, f(x)) \models \theta$ whenever $t_y \rightleftharpoons_{f(x)} t_x$, iff $M, (t_x, f(x)) \models \Box\theta$. ◀

Querying Interval-Based Temporal Data with SPARQL

Julien Corman ✉ 

Free University of Bozen-Bolzano, Italy

Roman Kontchakov ✉ 

Birkbeck, University of London, UK

Cem Okulmus ✉ 

Paderborn University, Germany

Abstract

We propose an extension of the graph query language SPARQL to support temporal data. The proposal follows the approach of temporal databases, where each tuple is annotated with a finite set of intervals, and joins and other operators implicitly take account of these annotations. We adapt this model to the specifics of SPARQL query evaluation, which is defined in terms of compatibility of partial solution mappings. Furthermore, unlike existing proposals, our model allows predicates over intervals (e.g., Allen's relations) in temporal atoms of filter expressions, while ensuring that query evaluation remains tractable in data complexity. Our semantics comes in three equivalent flavours, with different levels of abstraction. The basic semantics provides a snapshot view of data (it is, however, applicable only to queries without temporal atoms). In the second semantics, induced by an SPM-semiring, solution mappings are annotated with finite sets of intervals. This semantics covers temporal atoms and coincides with the snapshot semantics when the latter is defined. The third, most concrete semantics is equivalent to the second, but based on single intervals, which naturally leads to an implementation over a triple store or a relational database, with tables featuring columns for interval endpoints. As a proof of concept, we evaluate a database storage prototype and show that it addresses previously identified performance limitations of triple stores for temporal data.

2012 ACM Subject Classification Information systems → Temporal data; Information systems → Query languages

Keywords and phrases Query languages, SPARQL, interval-based temporal data

Digital Object Identifier 10.4230/OASICS.TIME.2026.6

1 Introduction

Temporal data has long been of interest to various fields of computer science, from database theory to knowledge representation, and is also seeing growing applications, as evidenced by the continued interest in specialised database systems dealing with time, such as time-series databases and extensions of the SQL standard. Notably, SQL:2011 features time period definitions, temporal integrity constraints and temporal predicates (Allen's relations).

Formal models of databases (in particular, temporal databases) are typically grounded in relational algebra, which makes strong assumptions about the structure of the data being queried. As a result, these models do not easily extend to missing, ill-typed or heterogeneous data. In contrast, knowledge graphs (KGs) and associated query languages provide a robust framework for modelling such data. However, temporal KGs remain largely underdeveloped.

In this work, we focus on the graph query language SPARQL, a widely used W3C standard. At first sight, SPARQL is similar to SQL, but its core algebra operations (union, join and difference) are of a different nature: they operate on (multi)sets of *partial* functions rather than relations, which can be regarded as (multi)sets of *total* functions. This distinction has far-reaching consequences when it comes to equipping SPARQL with temporal features,



© Julien Corman, Roman Kontchakov, and Cem Okulmus;
licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 6; pp. 6:1–6:21

OpenAccess Series in Informatics



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

■ **Listing 1** An example Temporal SPARQL query, adapted from [14].

```
PREFIX : <http://www.example.org#>
PREFIX time: <http://www.w3.org/2006/time#>
SELECT ?e ?dep1 ?t1 {
  << ?e :worksIn ?dep1 >> time:hasTime ?t1 .
  << ?dep1 :locatedIn :barcelona >> time:hasTime ?t1 .
  FILTER NOT EXISTS {
    << ?e :worksIn ?dep2 >> time:hasTime ?t2 .
    << ?dep2 :locatedIn :rome >> time:hasTime ?t2 .
    FILTER ( time:Before(?t2,?t1) ).
  }
}
```

and, to our knowledge, proposals in this direction have largely ignored it. In other words, some foundational work is still missing that provides SPARQL with temporal abilities while respecting its semantics, which is the gap that we propose to fill.

More precisely, we adapt to SPARQL the classical model of temporal relational databases, where data and answers are annotated with intervals, and extend it with the possibility to compare maximal temporal intervals. As a concrete illustration, we reproduce an example by Klarman and Meyer [14].¹

► **Example 1.** Consider the query in Listing 1, given in SPARQL-star syntax, where reified triples are enclosed in `<<...>>`. The query retrieves employees `?e` who worked in department `?dep1` located in Barcelona during some time interval `?t1`, but did not work in any department `?dep2` located in Rome during an interval `?t2` that immediately precedes `?t1` (Allen’s relation `time:Before`). Each answer consists of one such employee, with their department `?dep1` and the relevant interval `?t1`. Observe that, similarly to temporal databases [2], all the joins in the query are (*implicitly*) *temporal*. For instance, the temporal variable `?t1` in the query result represents an interval when the following two statements annotated with `time:hasTime ?t1` hold: (i) `?e` worked in `?dep1` and (ii) `?dep1` was based in Barcelona. However, such an interval may not come directly from the data: it is instead the intersection of two *maximal* intervals where (i) and (ii) respectively hold (for the same employee and department). In other words, `?t1` ranges over maximal periods of time where both statements hold, and similarly for `?t2`. This ensures that the temporal atom `time:Before(?t2,?t1)` can be interpreted unambiguously.

As this example illustrates, preserving maximality of temporal intervals, which is called *temporal coalescing* [3], is crucial for a clear and well-defined semantics of *temporal atoms*, which feature predicates on intervals, for instance, Allen’s relations. On the other hand, as we shall see below, an unrestricted use of temporal variables could lead to either intractability (in the size of the data) or a semantics that is not well-defined. Careful restrictions on the *scope* of temporal variables are thus needed, and our proposal achieves this by means of SPARQL’s EXISTS operator, as illustrated in Listing 1.

Our main contributions are as follows.

¹ Note, however, that Klarman and Meyer employ two-sorted first-order query language and consider background knowledge (ontologies) for enriching query answering, whereas our focus is solely on SPARQL and its algebra, without any ontologies.

1. We identify a suitable fragment of SPARQL-star (an extension of SPARQL, originally proposed by Hartig and Thompson [11] and is standardised by the W3C as part of the upcoming SPARQL 1.2) to easily express temporal join operations as well as comparison of intervals (using temporal atoms with Allen’s relations, for example). We impose suitable restrictions on this fragment to ensure that it admits a well-defined semantics and tractable query evaluation. We dub this language Temporal SPARQL.
2. We provide a formal semantics for Temporal SPARQL that complies with the specifics of the SPARQL algebra. More precisely, we follow the approach of Dignös et al. [7] to temporal databases and show that this semantics is induced by an annotation semiring suitable for SPARQL [9].
3. We conclude the paper by a brief feasibility study, which explores how Temporal SPARQL can be effectively translated into SQL. Among other applications, this makes Temporal SPARQL a good candidate input language for temporal Ontology-Based Data Access (OBDA) [12], which would bring the original proposal of Klarman and Meyer [14] closer to an effective and standard-compliant implementation. Integration with ontologies and query rewriting, however, is left for future work.

Related Work. Chekol et al. [4] present algorithms for joins over interval-based temporal data as well as an efficient algorithm for performing temporal coalescing and provide a proof-of-concept implementation to show how their proposal outperforms direct translations into standard SPARQL running on existing triple stores. Geerts et al. [9] describe the semantics of SPARQL based on algebraic structures called SPM-semirings with the aim of capturing provenance of answers to SPARQL queries on RDF graphs annotated with elements from some domain K . Dignös et al. [7] propose a semantics for snapshot-reducible database queries over relational interval-based temporal data called K -relations. While not directly applicable to our model, which uses SPARQL semantics as a starting point, we follow their presentation whereby the semantics is presented as three distinct “models” of increasing abstraction: the implementation model relies on the structure of the database tables directly, while the logical and abstract models use algebraic structures that enable easier formal reasoning. Kaminski et al. [13] extend the algebraic description of the semantics of core SPARQL [15] to the key features of SPARQL 1.1 including aggregation and nested queries with the EXISTS operator. Our definitions build on the context-based interpretation of the semantics of EXISTS by Kaminski et al. [13]. In the wider literature on temporal query languages, we also note the influential work of Böhlen et al. [3] on temporal coalescing and bespoke algorithms for its implementation. Later extensions of the SQL standard introduced window functions, which are used by Zhou et al. [20] for a more efficient translation of temporal coalescing into SQL. Dignös et al. [6] also provide examples of efficient implementation of temporal inner joins, left joins and temporal coalescing in SQL with window functions.

The rest of the paper is structured as follows. In Section 2, we introduce the syntax of Temporal SPARQL and initially focus on its small fragment (with a single temporal variable and without temporal atoms), which has a simple *snapshot* semantics. In Section 2.3, we switch to the *logical model*, in which query answers are annotated with finite set of intervals and show that the semantics with annotations from a particular SPM-semiring matches the snapshot semantics for this small fragment. In Section 3, we return to Temporal SPARQL with multiple temporal variables and temporal atoms, and introduce syntactic restrictions that ensure tractable query answering in the logical model. In Section 4, we define the so-called *concrete model*, in which answers are assigned single intervals and which is suitable for a translation into the relational setting. In Section 5, we provide a feasibility study that shows how Temporal SPARQL can be effectively and efficiently translated to SQL.

2 Temporal SPARQL: Syntax and Semantics

In Section 2.1, we provide an abstract syntax for Temporal SPARQL, broadly following [18], which is in turn based on the simplified abstract SPARQL algebra from the W3C recommendation². This abstract syntax has a natural SPARQL-star counterpart, where reified triples are annotated with temporal variables, as illustrated in Example 1; note, however, that in Section 3.1, we will need to introduce additional syntactic restrictions to ensure that query evaluation is tractable in data complexity. But first, in Section 2.2, we study an even smaller fragment of Temporal SPARQL that uses a single temporal variable and define its *snapshot* semantics, which coincides with the standard (non-temporal) SPARQL semantics when the timepoint of query evaluation is fixed. In Section 2.3, we equip this snapshot fragment with an equivalent annotation-based semantics based on SPM-semirings, which generalises a wide range of semantics for SPARQL (sets, bags, provenance, etc.).

2.1 Syntax

We assume pairwise disjoint infinitely countable sets of *IRIs* $\mathcal{I}$, *blank nodes* $\mathcal{B}$, *literals* $\mathcal{L}$, *domain variables* $\mathcal{V}$ and *temporal variables* $\mathcal{T}$. The *domain* $\mathcal{D}$ is the union $\mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$. *Graph patterns* P and *filter expressions* F are defined by mutual recursion as follows:

$$\begin{aligned} P &:= (s, p, o, t) \mid \text{FILTER}(P, F) \mid \text{JOIN}(P, P) \mid \text{OPTIONAL}(P, P, F) \mid \text{UNION}(P, P) \\ &\quad \mid \text{MINUS}(P, P) \mid \text{PROJECT}(P, L) \mid \text{EXTEND}(P, v, T), \\ F &:= \text{BOUND}(v) \mid R(T_1, \dots, T_n) \mid A(t_1, \dots, t_n) \mid \text{EXISTS}(P) \mid F \wedge F \mid \neg F, \end{aligned}$$

where $s, p, o \in \mathcal{V} \cup \mathcal{D}$ are (*basic*) *domain terms*, $v \in \mathcal{V}$ and $t, t_1, \dots, t_n \in \mathcal{T}$ are domain and temporal variables, respectively, $L \subseteq \mathcal{V}$ a finite set of domain variables, R and A are n -ary relations over domain elements and time intervals, respectively, and $T, T_1, \dots, T_n$ are *domain terms* built from elements of $\mathcal{V}$ and $\mathcal{D}$ using domain functions such as $+$, CONCAT , etc. The basic building block of graph patterns, (s, p, o, t) , is called a *temporal quad pattern*: s is the *subject*, p is the *predicate*, o is the *object* and t is the temporal variable of the statement. In the concrete SPARQL-star syntax, illustrated in Example 1, (s, p, o) forms a reified triple, used as the subject of a triple pattern with predicate `time:hasTime` and object t . As usual in SPARQL,³ we assume that each variable v in $\text{EXTEND}(P, v, T)$ does not occur in pattern P .

2.2 Abstract Model: Snapshot Semantics

We first consider a fragment of Temporal SPARQL reducible to the snapshot view of temporal data. A graph pattern is called *snapshot* if it uses a single temporal variable, t , and does not contain filter (sub-)expressions of the form $A(t_1, \dots, t_n)$ or $\text{EXISTS}(P)$. Note that the query in Listing 1 is not snapshot, as it contains two temporal variables, `t1` and `t2`. In contrast, the queries in Listing 2 in Section 5.1 and in Listings 4 and 5 in Section 5.2 are all snapshot.

A *temporal domain* $\mathbb{T}$ is a linearly ordered set, say, $\mathbb{Q}$ or $\mathbb{N}$. The strict linear order on $\mathbb{T}$ will be denoted by $<$, with $\leq$ and other relations defined in the usual way. We denote by $\mathcal{RDF}$ the set $(\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times \mathcal{D}$, whose elements are called *RDF triples*.⁴ An (atemporal) *RDF graph* is a finite subset of $\mathcal{RDF}$. A *snapshot RDF graph* $\mathcal{G}$ maps each time instant $\tau \in \mathbb{T}$

² <https://www.w3.org/TR/sparql11-query/#sparqlAlgebra>

³ <https://www.w3.org/TR/sparql11-query/#bind>

⁴ According to the RDF 1.1 W3C recommendation (<https://www.w3.org/TR/rdf11-concepts/#section-triples>), a literal can only occur as an object, and a predicate can only be an IRI, but this is not essential for us.

to an RDF graph $\mathcal{G}(\tau)$. The *snapshot semantics* of snapshot graph patterns closely follows the standard set semantics of SPARQL and boils down to evaluation on a single $\mathcal{G}(\tau)$. We reproduce it here so that it can be compared to our interval-based semantics.

A (non-temporal) *solution mapping* μ is a partial map from $\mathcal{V}$ to $\mathcal{D}$ with a finite domain, which is denoted by $\text{dom}(\mu)$. We say that two solution mappings μ and μ' are *compatible* and write $\mu \sim \mu'$ if $\mu(v) = \mu'(v)$, for each $v \in \text{dom}(\mu) \cap \text{dom}(\mu')$. In case $\mu \sim \mu'$, we write $\mu \dot{\cup} \mu'$ for the solution mapping such that $\text{dom}(\mu \dot{\cup} \mu') = \text{dom}(\mu) \cup \text{dom}(\mu')$ and $(\mu \dot{\cup} \mu')(v) = \mu(v)$ for all $v \in \text{dom}(\mu)$ and $(\mu \dot{\cup} \mu')(v) = \mu'(v)$ for all $v \in \text{dom}(\mu')$. We denote by $\mu|_L$ the restriction of μ on L , that is, the solution mapping with domain $\text{dom}(\mu) \cap L$ that is compatible with μ .

The semantics of a domain term T is defined as usual: given a solution mapping μ , set

$$\begin{aligned} \llbracket c \rrbracket^\mu &= c, & \llbracket v \rrbracket^\mu &= \begin{cases} \mu(v), & \text{if } v \in \text{dom}(\mu), \\ \epsilon, & \text{otherwise,} \end{cases} \\ \llbracket f(T_1, \dots, T_n) \rrbracket^\mu &= \begin{cases} f(\llbracket T_1 \rrbracket^\mu, \dots, \llbracket T_n \rrbracket^\mu), & \text{if } \llbracket T_1 \rrbracket^\mu \neq \epsilon, \dots, \llbracket T_n \rrbracket^\mu \neq \epsilon, \\ \epsilon, & \text{otherwise,} \end{cases} \end{aligned}$$

where ϵ denotes the *error value*, which is used in SPARQL to deal with the fact that solution mappings are partial functions. Next, for a solution mapping μ , the following definitions reflect the standard 3-valued logic of SPARQL filter expressions F (see, e.g., [18, Section 2.1]), where the subscript s stands for “snapshot” semantics:

$$\begin{aligned} \llbracket \text{BOUND}(v) \rrbracket_s^\mu &= \begin{cases} \top & \text{if } v \in \text{dom}(\mu), \\ \perp & \text{otherwise,} \end{cases} & \llbracket R(T_1, \dots, T_n) \rrbracket_s^\mu &= \begin{cases} \epsilon & \text{if } \llbracket T_i \rrbracket^\mu = \epsilon, \text{ for some } i \leq n, \\ \top & \text{if } R(\llbracket T_1 \rrbracket^\mu, \dots, \llbracket T_n \rrbracket^\mu), \\ \perp & \text{otherwise,} \end{cases} \\ \llbracket \neg F \rrbracket_s^\mu &= \begin{cases} \epsilon & \text{if } \llbracket F \rrbracket_s^\mu = \epsilon, \\ \top & \text{if } \llbracket F \rrbracket_s^\mu = \perp, \\ \perp & \text{otherwise,} \end{cases} & \llbracket F \wedge F' \rrbracket_s^\mu &= \begin{cases} \epsilon & \text{if } \llbracket F \rrbracket_s^\mu = \epsilon \text{ or } \llbracket F' \rrbracket_s^\mu = \epsilon, \\ \top & \text{if } \llbracket F \rrbracket_s^\mu = \llbracket F' \rrbracket_s^\mu = \top, \\ \perp & \text{otherwise.} \end{cases} \end{aligned}$$

Next, let G denote the snapshot RDF graph $\mathcal{G}$ at time τ , that is, $\mathcal{G}(\tau)$. The *evaluation* $\llbracket P \rrbracket_s^G$ of a snapshot graph pattern P over G is defined by recursion on the structure of P as usual (see, e.g., [18, Section 2.1]):

$$\begin{aligned} \llbracket (s, p, o, t) \rrbracket_s^G &= \{ \mu \mid \text{dom}(\mu) = \text{var}(s, p, o) \text{ and } (s, p, o)^\mu \in G \}, \\ \llbracket \text{FILTER}(P_1, F) \rrbracket_s^G &= \{ \mu \in \llbracket P_1 \rrbracket_s^G \mid \llbracket F \rrbracket_s^\mu = \top \}, \\ \llbracket \text{JOIN}(P_1, P_2) \rrbracket_s^G &= \{ \mu_1 \dot{\cup} \mu_2 \mid \mu_1 \in \llbracket P_1 \rrbracket_s^G \text{ and } \mu_2 \in \llbracket P_2 \rrbracket_s^G \text{ with } \mu_1 \sim \mu_2 \}, \\ \llbracket \text{UNION}(P_1, P_2) \rrbracket_s^G &= \llbracket P_1 \rrbracket_s^G \cup \llbracket P_2 \rrbracket_s^G, \\ \llbracket \text{OPTIONAL}(P_1, P_2, F) \rrbracket_s^G &= \llbracket \text{FILTER}(\text{JOIN}(P_1, P_2), F) \rrbracket_s^G \\ &\quad \cup \{ \mu_1 \in \llbracket P_1 \rrbracket_s^G \mid \llbracket F \rrbracket_s^{\mu_1 \dot{\cup} \mu_2} \neq \top, \text{ for all } \mu_2 \in \llbracket P_2 \rrbracket_s^G \text{ with } \mu_1 \sim \mu_2 \}, \\ \llbracket \text{MINUS}(P_1, P_2) \rrbracket_s^G &= \{ \mu_1 \in \llbracket P_1 \rrbracket_s^G \mid \text{there is no } \mu_2 \in \llbracket P_2 \rrbracket_s^G \text{ with } \mu_1 \sim \mu_2 \\ &\quad \text{and } \text{dom}(\mu_1) \cap \text{dom}(\mu_2) \neq \emptyset \}, \\ \llbracket \text{PROJECT}(P_1, L) \rrbracket_s^G &= \{ \mu|_L \mid \mu \in \llbracket P_1 \rrbracket_s^G \}, \\ \llbracket \text{EXTEND}(P_1, v, T) \rrbracket_s^G &= \{ \mu \dot{\cup} \{v \mapsto \llbracket T \rrbracket^\mu\} \mid \mu \in \llbracket P_1 \rrbracket_s^G \}, \end{aligned}$$

where $\text{var}(s, p, o)$ denotes the set of variables in a tuple (s, p, o) of basic domain terms, and $(s, p, o)^\mu$, for a solution mapping μ , denotes the image of the tuple under μ , that is, the result of replacing each variable $v \in \text{dom}(\mu)$ in it with $\mu(v)$. Note that this semantics is applicable only to snapshot graph patterns, which have a single temporal variable, t , and therefore this variable plays no role in the definition; the time instant is fixed with the choice of $G = \mathcal{G}(\tau)$.

2.3 Logical Model: Temporal Annotations

Similarly to what was done by Dignös et al. [7] for relational algebra, we introduce an alternative representation of the semantics above, where, for a given graph, a query maps a solution mapping to (the union of) a set of intervals. This allows us to relate Temporal SPARQL to a long line of research in relational databases (notably by Green et al. [10]), and more recently for SPARQL by Geerts et al. [9], where a query maps tuples to annotations in a semiring (possibly extended with other operators [8, 9]). In this view, the semantics of a query language is defined for a whole class of semirings, thus providing a shared abstraction (as well as algorithms, proof techniques, etc.) for seemingly different formalisms, which include standard set and bag semantics, but also data annotated with provenance or probabilities. As an additional benefit, this abstract view will ease the correspondence with the more procedural model introduced in Section 4.

An *interval* over $\mathbb{T}$ is either $\mathbb{T}$ itself or one of the following subsets of $\mathbb{T}$:

- $\{\tau \in \mathbb{T} \mid \tau_1 \leq \tau < \tau_2\}$, denoted by $[\tau_1, \tau_2)$, for $\tau_1, \tau_2 \in \mathbb{T}$,
- $\{\tau \in \mathbb{T} \mid \tau_1 \leq \tau\}$, for $\tau_1 \in \mathbb{T}$, or
- $\{\tau \in \mathbb{T} \mid \tau < \tau_2\}$, for $\tau_2 \in \mathbb{T}$;

note that these intervals include the left endpoint (if any) but not the right endpoint. We use $\mathbb{I}$ to denote the set of all intervals.

We consider the set $S_{\mathbb{T}}$ of all subsets of $\mathbb{T}$ that can be represented as finite sets of intervals. Formally, $S_{\mathbb{T}} = \{\bigcup I \mid I \in \mathcal{P}_{fin}(\mathbb{I})\}$, where $\mathcal{P}_{fin}(\mathbb{I})$ is the set of finite subsets of $\mathbb{I}$. The set $S_{\mathbb{T}}$ is closed under finite set union ($\cup$) by definition. We show that it is also closed under set difference ($\setminus$), and so under finite set intersection ($\cap$), since $A \cap B = A \setminus (A \setminus B)$ for any sets A and B . To this end, we first observe that, for any $I, J \in \mathbb{I}$, the set $I \setminus J$ is the empty set, or a single element of $\mathbb{I}$, or the union of two such intervals, and so, in each case, $I \setminus J \in S_{\mathbb{T}}$. Then we can lift this to elements of $S_{\mathbb{T}}$ observing that $(I_1 \cup I_2) \setminus J = (I_1 \setminus J) \cup (I_2 \setminus J)$ and $I \setminus (J_1 \cup J_2) = (I \setminus J_1) \setminus J_2$. So, for any elements $I = \bigcup_{i=1}^n I_i$ and $J = \bigcup_{j=1}^m J_j$ of $S_{\mathbb{T}}$, with $I_i, J_j \in \mathbb{I}$ for each i and j , by induction on $n + m$, we obtain $I \setminus J \in S_{\mathbb{T}}$.

Interestingly, the algebraic structure $\mathcal{S}_{\mathbb{T}} = \langle S_{\mathbb{T}}, \cup, \cap, \setminus, \emptyset, \mathbb{T} \rangle$ belongs to the equational class (a.k.a. variety) called SPM-semirings, where “SPM” stands for “SPARQL Minus” [9]. An *SPM-semiring* $\langle S, \oplus, \otimes, \ominus, \mathbf{0}, \mathbf{1} \rangle$ is a commutative semiring (i.e., its multiplicative operation $\otimes$ is also commutative) equipped with a difference operation $\ominus$ satisfying the following identities:

$$a \ominus a = \mathbf{0}, \tag{id_1}$$

$$a \ominus (b \oplus c) = (a \ominus b) \ominus c, \tag{id_2}$$

$$a \otimes (b \ominus c) = (a \otimes b) \ominus c, \tag{id_3}$$

$$(a \ominus (a \ominus b)) \oplus (a \ominus b) = a. \tag{id_4}$$

One can easily verify that $\mathcal{S}_{\mathbb{T}}$ is an SPM-semiring. Indeed, both $\langle S_{\mathbb{T}}, \cup, \emptyset \rangle$ and $\langle S_{\mathbb{T}}, \cap, \mathbb{T} \rangle$ are commutative monoids since $\cup$ and $\cap$ are associative and commutative, and both $I \cup \emptyset = I$ and $I \cap \mathbb{T} = I$ hold for any $I \in S_{\mathbb{T}}$. Next, $\cap$ distributes over $\cup$ and $I \cap \emptyset = \emptyset$. Finally, the four identities for $\ominus$ hold for any sets A, B and C (so in particular for the elements of $S_{\mathbb{T}}$): $A \setminus A = \emptyset$, $A \setminus (B \cup C) = (A \setminus B) \setminus C$, $A \cap (B \setminus C) = (A \cap B) \setminus C$ and $(A \setminus (A \setminus B)) \cup (A \setminus B) = A$.

Geerts et al. [9] define a semantics for SPARQL queries induced by an arbitrary SPM-semiring: each triple in an RDF graph and each solution mapping in the evaluation of a query is annotated with an element of the semiring. With an appropriate choice of the semiring, this semantics coincides with the standard set or bag semantics of SPARQL (among others). We reproduce the SPM-semiring semantics of SPARQL [9], slightly extended to accommodate for OPTIONAL with a filter expression (in the original paper [9], only a tautology $\top$ is used as a filter expression in OPTIONAL). Let $\mathcal{S} = \langle S, \oplus, \otimes, \ominus, \mathbf{0}, \mathbf{1} \rangle$ be an SPM-semiring. An

$\mathcal{S}$ -annotated RDF graph is a function $G: \mathcal{RDF} \rightarrow S$ that maps finitely many triples to an element different from $\mathbf{0}$ (intuitively, $G(s, p, o) \neq \mathbf{0}$ if the RDF graph contains (s, p, o)). The evaluation $\llbracket P \rrbracket_a^G$ of a snapshot graph pattern P over an $\mathcal{S}$ -annotated RDF graph G is defined as follows (the subscript a in $\llbracket P \rrbracket_a^G$ stands for “annotation”-based semantics):

$$\begin{aligned} \llbracket (s, p, o, t) \rrbracket_a^G(\mu) &= \begin{cases} G((s, p, o)^\mu), & \text{if } \text{dom}(\mu) = \text{var}(s, p, o), (s, p, o)^\mu \in \mathcal{RDF}, \\ \mathbf{0}, & \text{otherwise,} \end{cases} \\ \llbracket \text{FILTER}(P_1, F) \rrbracket_a^G(\mu) &= \llbracket P_1 \rrbracket_a^G(\mu) \otimes \llbracket F \rrbracket_s^\mu, \\ \llbracket \text{JOIN}(P_1, P_2) \rrbracket_a^G(\mu) &= \bigoplus_{\mu_1 \dot{\cup} \mu_2 = \mu} \left(\llbracket P_1 \rrbracket_a^G(\mu_1) \otimes \llbracket P_2 \rrbracket_a^G(\mu_2) \right), \\ \llbracket \text{UNION}(P_1, P_2) \rrbracket_a^G(\mu) &= \llbracket P_1 \rrbracket_a^G(\mu) \oplus \llbracket P_2 \rrbracket_a^G(\mu), \\ \llbracket \text{OPTIONAL}(P_1, P_2, F) \rrbracket_a^G(\mu) &= \llbracket Q \rrbracket_a^G(\mu) \oplus \left(\llbracket P_1 \rrbracket_a^G(\mu) \ominus \bigoplus_{\mu' \sim \mu} \llbracket Q \rrbracket_a^G(\mu' \dot{\cup} \mu) \right), \\ &\quad \text{where } Q = \text{FILTER}(\text{JOIN}(P_1, P_2), F), \\ \llbracket \text{MINUS}(P_1, P_2) \rrbracket_a^G(\mu) &= \llbracket P_1 \rrbracket_a^G(\mu) \ominus \bigoplus_{\substack{\mu' \sim \mu, \\ \text{dom}(\mu') \cap \text{dom}(\mu) \neq \emptyset}} \llbracket P_2 \rrbracket_a^G(\mu'), \\ \llbracket \text{PROJECT}(P_1, L) \rrbracket_a^G(\mu) &= \bigoplus_{\mu'_L = \mu} \llbracket P_1 \rrbracket_a^G(\mu'), \\ \llbracket \text{EXTEND}(P_1, v, T) \rrbracket_a^G(\mu) &= \begin{cases} \llbracket P_1 \rrbracket_a^G(\mu'), & \text{if } v \notin \text{dom}(\mu') \text{ and } \mu = \mu' \dot{\cup} \{v \mapsto \llbracket T \rrbracket^{\mu'}\}, \\ \mathbf{0}, & \text{otherwise,} \end{cases} \end{aligned}$$

where $\llbracket \cdot \rrbracket: \{\top, \perp, \epsilon\} \rightarrow \{\mathbf{0}, \mathbf{1}\}$ is the function defined by taking $\llbracket \top \rrbracket = \mathbf{1}$ and $\llbracket \perp \rrbracket = \llbracket \epsilon \rrbracket = \mathbf{0}$. Note again that this semantics is applicable only to snapshot graph patterns, which have a single temporal variable, t , and FILTER uses the snapshot semantics for filter expressions F (which is sufficient at this stage, but will be replaced with a more general semantics in Section 3.2).

We now show that the semantics induced by the semiring $\mathcal{S}_{\mathbb{T}}$ captures the snapshot-based semantics of Section 2.2. This property is called *snapshot-reducibility*, and an analogous result holds for temporal relational algebra [7]. A snapshot RDF graph $\mathcal{G}$ is called *representable* if

$$\{\tau \in \mathbb{T} \mid (s, p, o) \in \mathcal{G}(\tau)\} \in S_{\mathbb{T}}, \quad \text{for all } (s, p, o) \in \mathcal{RDF}.$$

If this is the case, then we write $\mathcal{G}_{\mathcal{S}_{\mathbb{T}}}$ for the $\mathcal{S}_{\mathbb{T}}$ -annotated RDF graph of $\mathcal{G}$:

$$\mathcal{G}_{\mathcal{S}_{\mathbb{T}}}(s, p, o) = \{\tau \in \mathbb{T} \mid (s, p, o) \in \mathcal{G}(\tau)\}, \quad \text{for all } (s, p, o) \in \mathcal{RDF}.$$

The following result says that, for representable snapshot graphs, the semantics of SPARQL induced by the semiring $\mathcal{S}_{\mathbb{T}}$ coincides with the snapshot semantics of Section 2.2.

► **Theorem 2** (snapshot-reducibility). *For a representable snapshot graph $\mathcal{G}$, snapshot graph pattern P , time point τ and solution mapping μ , we have $\tau \in \llbracket P \rrbracket_a^{\mathcal{G}_{\mathcal{S}_{\mathbb{T}}}}(\mu)$ iff $\mu \in \llbracket P \rrbracket_s^{\mathcal{G}(\tau)}$.*

Proof. The standard set semantics of SPARQL is induced by the Boolean SPM-semiring $\mathcal{B} = \langle \{\perp, \top\}, \vee, \wedge, \dot{-}, \perp, \top \rangle$, where $a \dot{-} b = a \wedge \neg b$; see, e.g., [9]. More precisely, for a snapshot RDF graph $\mathcal{G}$ and $\tau \in \mathbb{T}$, we define a $\mathcal{B}$ -annotated graph $\mathcal{G}_{\mathcal{B}}^\tau$ by taking $\mathcal{G}_{\mathcal{B}}^\tau(s, p, o) = \top$ iff $(s, p, o) \in \mathcal{G}(\tau)$. Then, by induction on the structure of P , one can show that

$$\llbracket P \rrbracket_a^{\mathcal{G}_{\mathcal{B}}^\tau}(\mu) = \top \quad \text{iff} \quad \mu \in \llbracket P \rrbracket_s^{\mathcal{G}(\tau)}. \quad (1)$$

Next, for $\tau \in \mathbb{T}$, define $h_\tau: S_{\mathbb{T}} \rightarrow \{\perp, \top\}$ by taking $h_\tau(s) = \top$ iff $\tau \in s$. Observe that this function is a homomorphism from $S_{\mathbb{T}}$ to $\mathcal{B}$, because it satisfies $h_\tau(s_1 \cup s_2) = h_\tau(s_1) \vee h_\tau(s_2)$, $h_\tau(s_1 \cap s_2) = h_\tau(s_1) \wedge h_\tau(s_2)$, $h_\tau(s_1 \setminus s_2) = h_\tau(s_1) \dot{-} h_\tau(s_2)$, $h_\tau(\emptyset) = \perp$ and $h_\tau(\mathbb{T}) = \top$. It follows that $h_\tau(\llbracket P \rrbracket_a^{\mathcal{G}_{S_{\mathbb{T}}}}(\mu)) = \llbracket P \rrbracket_a^{\mathcal{G}_{\mathcal{B}}}(\mu)$, and so $\llbracket P \rrbracket_a^{\mathcal{G}_{S_{\mathbb{T}}}}(\mu) = \top$ iff $\tau \in \llbracket P \rrbracket_a^{\mathcal{G}_{S_{\mathbb{T}}}}(\mu)$. The claim is then immediate from (1). $\blacktriangleleft$

The same temporal annotation (i.e., an element of $S_{\mathbb{T}}$) can be represented by different sets of intervals, which may be impractical (for instance, two equivalent queries may produce syntactically different results); see also [7]. This can be easily addressed by means of (temporal) *coalescing* [3]. A set of intervals $I \subseteq \mathbb{I}$ is said to be *coalesced* if each $I \in I$ is maximal, that is, there is no $I' \in \mathbb{I}$ such that $I \subset I' \subseteq \bigcup I$. Clearly, for a temporal annotation $s \in S_{\mathbb{T}}$, there is a unique finite coalesced set of intervals $C(s)$ such that $\bigcup C(s) = s$. Thus, $C(s)$ can serve as a canonical representation for s .

3 Comparing Intervals

We now extend our annotation-based semantics to a fragment of Temporal SPARQL that supports comparisons between maximal intervals, notably by means of Allen's relations. This is achieved via multiple temporal variables and existential quantification induced by the structure of EXISTS operations. For instance, in Example 1, an interval I_1 bound to the variable `?t1` should be retained if it ends before *some* interval I_2 bound to `?t2`. The comparison itself (`time:Before(?t2, ?t1)` in this example) is an atomic Boolean expression $A(t_1, \dots, t_n)$ in the grammar in Section 2.1, which we call a *temporal atom*. Its semantics can only be defined if each variable t_i is associated with a (unique) set I_i of intervals. To ensure that computing I_i remains tractable in the size of the data, we impose syntactic restrictions on Temporal SPARQL, which are described next.

3.1 Projected Temporal Variable

We introduce syntactic restrictions on the query language defined in Section 2.1: intuitively, a query must project exactly one temporal variable. More precisely, we say that a graph pattern P is *well-formed* and *binds* temporal variable t if it is a quad pattern of the form (s, p, o, t) , for some s, p and o , or

- (i) all quad patterns in P outside filter expressions bind t , and
- (ii) in each filter expression of the form $\text{EXISTS}(P')$ that occurs in P , the graph pattern P' is well-formed and binds a temporal variable *other* than t .

Restriction (ii) prevents any “implicit” temporal join induced by a graph pattern P and a subexpression $\text{EXISTS}(P')$ of P binding the same temporal variable t . This restriction is not strictly needed, but has the benefit of simplifying the semantics of the language, while not reducing its expressivity (one can always introduce an “explicit” temporal join instead). Restriction (i) on the other hand is essential: it excludes patterns that bind two or more temporal variables. As a counterexample, consider a query of the form $\text{FILTER}(\text{JOIN}(P_1, P_2), \text{EXISTS}(\text{FILTER}(P_3, A(t_1, t_3))))$, where each pattern P_i binds the temporal variable t_i . Its subpattern $\text{JOIN}(P_1, P_2)$ is not well-formed because it binds two temporal variables, t_1 and t_2 . We shall see below that coalescing a temporal annotation for such a pattern is intractable in the size of the graph (i.e., in data complexity). And apart from the coalesced form, we are not aware of any representation that specifies a unique set of intervals for t_1 (and a given solution mapping). So this rules out an efficient evaluation of $A(t_1, t_3)$.

To complete this argument, we show that coalescing is intractable even for two temporal variables. To this end, we exploit a well-known technique, already used in the context of bi-temporal databases [17], namely, a (linear-time) reduction of the following NP-hard [5] problem: given $k \in \mathbb{N}$ and a finite set $\mathbf{R} = \{R_1, \dots, R_n\}$ of rectangles, whose union is a polygon $p(\mathbf{R})$, decide whether $p(\mathbf{R})$ can be covered with k rectangles. Each rectangle $R_i \in \mathbf{R}$ can be represented as the product $I_i \times J_i$ of two intervals in the Euclidean plane. Then one can construct a graph $\mathbf{G}_{\mathbf{R}}$ that encodes each pair (I_i, J_i) as two triple annotations $\mathbf{G}_{\mathbf{R}}(\mathbf{r}_i, \mathbf{a}, \mathbf{a}) = I_i$ and $\mathbf{G}_{\mathbf{R}}(\mathbf{r}_i, \mathbf{b}, \mathbf{b}) = J_i$. Now, consider the graph pattern $P = \text{PROJECT}(\text{JOIN}((x, \mathbf{a}, \mathbf{a}, t_I), (x, \mathbf{b}, \mathbf{b}, t_J)), \emptyset)$, where $x \in \mathcal{V}$ and $t_I, t_J \in \mathcal{T}$. Observe that P is independent of $\mathbf{R}$ and k . If one is to generalise the semantics of Section 2.3 to patterns that bind two variables, then $\llbracket P \rrbracket_a^{\mathbf{G}_{\mathbf{R}}, \emptyset}$ on the solution mapping with the empty domain should give us a binary relation of the form $B = (I_1 \times J_1) \cup \dots \cup (I_n \times J_n)$; moreover, we would naturally say that a representation $\{(I_1, J_1), \dots, (I_m, J_m)\}$ of B is *coalesced* if each $I_j \times J_j$ is maximal w.r.t. set inclusion for $1 \leq j \leq m$, that is, no interval I satisfies $I_j \subset I$ and $I \times J_j \subseteq B$, and similarly for J_j . It should be clear that m is the least number of rectangles needed to cover $p(\mathbf{R})$, therefore $p(\mathbf{R})$ can be covered with k rectangles iff $m \leq k$.

Tractability can be regained by requiring that rectangles in a cover do not overlap. In this case, a minimal (w.r.t. cardinality) set of rectangles that covers $p(\mathbf{R})$ can be computed in polynomial time. Unfortunately, minimal covers are *not unique* anymore. For instance, an L-shaped polygon can be covered by two non-overlapping rectangles, but there are two alternative ways of doing so. This is why we cannot base our semantics on this weaker form of coalescing (even for two temporal variables).

3.2 Annotation-Based Semantics with Interval Comparisons

We now extend the annotation-based semantics of Section 2.3 to the remaining constructs, namely, EXISTS and temporal atoms, in such a way that it still coincides with the snapshot semantics of Section 2.2 (obviously, for snapshot graph patterns only).

In order to handle the EXISTS operator, we follow Kaminski et al. [13] and use contexts, which allows us to evaluate a pattern of the form $\text{FILTER}(P, \text{EXISTS}(P'))$ by “injecting” a solution mapping from P into P' . Formally, a *context* ξ is a partial finite-domain map from object variables $\mathcal{V}$ to the domain $\mathcal{D}$ and from temporal variables $\mathcal{T}$ to the set $\mathbb{I}$ of intervals. We adjust the semantics $\llbracket P \rrbracket_a^{\mathbf{G}}$ of graph patterns accordingly: we write $\llbracket P \rrbracket_a^{\mathbf{G}, \xi}$, where ξ is a context. We also need to extend the evaluation function to a filter expressions F to cover the cases of EXISTS and temporal atoms:

$$\begin{aligned} \llbracket \text{EXISTS}(P) \rrbracket_a^{\mathbf{G}, \xi} &= \begin{cases} \top, & \text{if } \llbracket P \rrbracket_a^{\mathbf{G}, \xi}(\mu) \neq \mathbf{0}, \text{ for some } \mu, \\ \perp, & \text{otherwise,} \end{cases} \\ \llbracket A(t_1, \dots, t_n) \rrbracket_a^{\mathbf{G}, \xi} &= \begin{cases} \epsilon, & \text{if } t_i \notin \text{dom}(\xi), \text{ for some } 1 \leq i \leq n, \\ \top, & \text{if } (\xi(t_1), \dots, \xi(t_n)) \in A, \\ \perp, & \text{otherwise;} \end{cases} \end{aligned}$$

For all other types of filter expressions, $\llbracket F \rrbracket_a^{\mathbf{G}, \xi}$ is defined analogously to $\llbracket F \rrbracket_s^{\mu}$. Note that $\llbracket F \rrbracket_a^{\mathbf{G}, \xi}$ depends not only on the context ξ (which is a solution mapping, like μ in $\llbracket F \rrbracket_s^{\mu}$), but also on the annotated RDF graph $\mathbf{G}$, as the semantics of filter expressions and graph patterns are now mutually recursive.

The full semantics of a temporal quad pattern is

$$\llbracket (s, p, o, t) \rrbracket_a^{\mathbf{G}, \xi}(\mu) = \begin{cases} \mathbf{G}((s, p, o)^{\mu \dot{\cup} \xi}), & \text{if } \text{dom}(\mu) = \text{var}((s, p, o)^{\xi}) \text{ and } (s, p, o)^{\mu \dot{\cup} \xi} \in \mathcal{RDF}, \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

All other operators but **FILTER** propagate contexts inductively. For instance, the semantics of **UNION** is given by

$$\llbracket \text{UNION}(P_1, P_2) \rrbracket_a^{G, \xi}(\mu) = \llbracket P_1 \rrbracket_a^{G, \xi}(\mu) \oplus \llbracket P_2 \rrbracket_a^{G, \xi}(\mu).$$

To support temporal atoms, the semantics of the **FILTER** operator is modified in the expected way: instead of reducing the whole annotation s for a solution mapping to the empty set, $\mathbf{0}$, we allow a **FILTER** to reduce it in a more granular way, by discarding individual intervals from the coalesced representation $C(s)$ of s (see Section 2.3). Precisely, if P binds temporal variable t , then

$$\llbracket \text{FILTER}(P, F) \rrbracket_a^{G, \xi}(\mu) = \bigoplus_{I \in C(\llbracket P \rrbracket_a^{G, \xi}(\mu))} I \otimes \llbracket F \rrbracket_a^{G, \xi \cup \mu \cup \{t \mapsto I\}}.$$

Finally, the evaluation $\llbracket P \rrbracket_a^G$ of a graph pattern P on an annotated RDF graph G is simply $\llbracket P \rrbracket_a^{G, \emptyset}$, where $\emptyset$ denotes the context with the empty domain. We illustrate these definitions with the following example.

► **Example 3.** Consider graph pattern P_1 , where

$$\begin{aligned} P_1 &= \text{FILTER}(P_2, \neg \text{EXISTS}(P_3)), \\ P_2 &= \text{JOIN}((e, \text{:worksIn}, dep_1, t_1), (dep_1, \text{:locatedIn}, \text{:barcelona}, t_1)), \\ P_3 &= \text{FILTER}(P_4, \text{time:Before}(t_2, t_1)), \\ P_4 &= \text{JOIN}((e, \text{:worksIn}, dep_2, t_2), (dep_2, \text{:locatedIn}, \text{:rome}, t_2)). \end{aligned}$$

The answers to P_1 are found in the following way. First, to obtain $\llbracket P_1 \rrbracket_a^{G, \emptyset}$, we evaluate $\llbracket P_2 \rrbracket_a^{G, \emptyset}$, which produces, say, the temporal solution mappings

$$([1999, 2003], \{e \mapsto \text{:john}, dep_1 \mapsto \text{:fin}\}), ([2000, 2003], \{e \mapsto \text{:mark}, dep_1 \mapsto \text{:hr}\});$$

note that P_2 binds temporal variable t_1 . Next, we evaluate $\llbracket \neg \text{EXISTS}(P_3) \rrbracket_a^{G, \xi_i}$ for contexts

$$\begin{aligned} \xi_1 &= \{t_1 \mapsto [1999, 2003], e \mapsto \text{:john}, dep_1 \mapsto \text{:fin}\}, \\ \xi_2 &= \{t_1 \mapsto [2000, 2003], e \mapsto \text{:mark}, dep_1 \mapsto \text{:hr}\}; \end{aligned}$$

note that these contexts map object variables e and dep_1 along with temporal variable t_1 . In order to evaluate $\llbracket P_3 \rrbracket_a^{G, \xi_i}$, we first evaluate $\llbracket P_4 \rrbracket_a^{G, \xi_i}$, which produces, say, the following sets of temporal solution mapping

$$\{([1997, 1998], \{dep_2 \mapsto \text{:hq}\})\} \text{ for } \xi_1 \quad \text{and} \quad \emptyset \text{ for } \xi_2;$$

observe that e is not in the domain of this temporal solution mapping, because its value is provided by the context, ξ_1 . Next, we can evaluate $\llbracket \text{FILTER}(P_4, \text{time:Before}(t_2, t_1)) \rrbracket_a^{G, \xi_i}$: we check $\llbracket \text{time:Before}(t_2, t_1) \rrbracket_a^{G, \xi_i}$ for the context

$$\xi'_1 = \{t_1 \mapsto [1999, 2003], e \mapsto \text{:john}, dep_1 \mapsto \text{:fin}, t_2 \mapsto [1998, 1999], dep_2 \mapsto \text{:hq}\};$$

observe that both temporal variables, t_1 and t_2 , are given values by this context – the result is positive, so $\llbracket \neg \text{EXISTS}(P_3) \rrbracket_a^{G, \xi_1}$ evaluates to $\perp$; while $\llbracket \neg \text{EXISTS}(P_3) \rrbracket_a^{G, \xi_2} = \top$.

It should be clear that this extended version of the annotation-based semantics coincides with the semantics defined in Section 2.3 for any snapshot graph pattern: such a pattern contains no **EXISTS**, therefore the context remains empty for all its sub-patterns, while the context for a filter expression becomes a solution mapping extended with a mapping for the only temporal variable, t , with the latter having no effect (since filter expressions in this fragment contain no temporal atoms, they cannot refer to t).

4 Concrete Model: Tuple-Based View

The logical model introduced in Section 2.3 and extended in Section 3.2 relies on (unions of) finite sets of intervals as annotations. While convenient to reason about the algebraic properties of query evaluation, this view does not immediately translate to an implementation over a relational database or a triple store, for two reasons:

- (i) the size of annotations (i.e., the number of intervals required to represent them) is unbounded, while a relational database or a triple store naturally manipulates tuples with a bounded or, respectively, fixed arity,
- (ii) the semantics of temporal atoms assumes coalesced annotations, but leaves effective coalescing underspecified.

This is why we follow Dignös et al. [7] and introduce a more concrete model, where each tuple carries a single interval, which addresses (i). As for (ii), below we use an explicit temporal coalescing operator TC , which maps a set of tuples to an equivalent one where intervals are coalesced. Interestingly, we shall see that TC is needed after certain SPARQL operators but is redundant for their “natural counterparts” in temporal relational algebra.

A *temporal solution mapping* (I, μ) consists of a nonempty interval $I \in \mathbb{I}$ and a solution mapping μ . For a set $\mathbf{T}$ of temporal solution mappings, we write $\overline{\mathbf{T}}$ to denote the function that maps solution mappings to their sets of intervals: $\overline{\mathbf{T}}(\mu) = \{I \mid (I, \mu) \in \mathbf{T}\}$, for every μ . The operator TC transforms a set $\mathbf{T}$ of temporal solution mappings to its coalesced form: formally, $\text{TC}(\mathbf{T})$ is the unique set of temporal solution mappings that satisfies

$$\overline{\text{TC}(\mathbf{T})}(\mu) = \mathcal{C}(\bigcup \overline{\mathbf{T}}(\mu)), \quad \text{for any } \mu;$$

in Section 5, we will discuss a possible implementation of TC in SQL.

A *temporal graph* $\mathbf{G}$ is a finite set of temporal triples $(I, (s, p, o))$, where $I \in \mathbb{I}$ is a nonempty interval and $(s, p, o) \in \mathcal{RDF}$ an RDF triple. We denote by $\overline{\mathbf{G}}$ the function that maps each triple $(s, p, o) \in \mathcal{RDF}$ to the set of intervals $\{I \mid (I, (s, p, o)) \in \mathbf{G}\}$. A temporal graph $\mathbf{G}$ is said to be *coalesced* if $\overline{\mathbf{G}}(s, p, o) = \mathcal{C}(\overline{\mathbf{G}}(s, p, o))$, for every $(s, p, o) \in \mathcal{RDF}$.

The evaluation $\llbracket P \rrbracket_i^{\mathbf{G}, \xi}$ of a graph pattern P over a coalesced temporal graph $\mathbf{G}$ with context ξ is a finite set of temporal solution mappings defined inductively in as follows:

$$\begin{aligned} \llbracket (s, p, o, t) \rrbracket_i^{\mathbf{G}, \xi} &= \{ (I, \mu) \mid \text{dom}(\mu) = \text{var}((s, p, o)^\xi), (I, (s, p, o)^{\mu \dot{\cup} \xi}) \in \mathbf{G} \}, \\ \llbracket \text{FILTER}(P_1, F) \rrbracket_i^{\mathbf{G}, \xi} &= \{ (I, \mu) \in \llbracket P_1 \rrbracket_i^{\mathbf{G}, \xi} \mid \llbracket F \rrbracket_i^{\mathbf{G}, \xi \dot{\cup} \mu \dot{\cup} \{t \mapsto I\}} = \top \}, \\ &\quad \text{where } P_1 \text{ binds temporal variable } t, \\ \llbracket \text{JOIN}(P_1, P_2) \rrbracket_i^{\mathbf{G}, \xi} &= \text{TC} \{ (I_1 \cap I_2, \mu_1 \dot{\cup} \mu_2) \mid (I_1, \mu_1) \in \llbracket P_1 \rrbracket_i^{\mathbf{G}, \xi}, (I_2, \mu_2) \in \llbracket P_2 \rrbracket_i^{\mathbf{G}, \xi}, \\ &\quad \mu_1 \sim \mu_2 \text{ and } I_1 \cap I_2 \neq \emptyset \}, \\ \llbracket \text{UNION}(P_1, P_2) \rrbracket_i^{\mathbf{G}, \xi} &= \text{TC}(\llbracket P_1 \rrbracket_i^{\mathbf{G}, \xi} \cup \llbracket P_2 \rrbracket_i^{\mathbf{G}, \xi}), \\ \llbracket \text{OPTIONAL}(P_1, P_2, F) \rrbracket_i^{\mathbf{G}, \xi} &= \text{TC}(\llbracket Q \rrbracket_i^{\mathbf{G}, \xi} \cup \{ (I', \mu) \mid (I, \mu) \in \llbracket P_1 \rrbracket_i^{\mathbf{G}, \xi}, I' \subseteq I \setminus \bigcup \mathbf{J}^\mu \}), \\ &\quad \text{where } \mathbf{J}^\mu = \{ J \mid (J, \mu \dot{\cup} \mu') \in \llbracket Q \rrbracket_i^{\mathbf{G}, \xi} \text{ for some } \mu' \sim \mu \} \\ &\quad \text{and } Q = \text{FILTER}(\text{JOIN}(P_1, P_2), F), \\ \llbracket \text{MINUS} \rrbracket_i^{\mathbf{G}, \xi}(P_1, P_2) &= \text{TC} \{ (I', \mu) \mid (I, \mu) \in \llbracket P_1 \rrbracket_i^{\mathbf{G}, \xi}, I' \subseteq I \setminus \bigcup \mathbf{J}^\mu \}, \\ &\quad \text{where } \mathbf{J}^\mu = \{ J \mid (J, \mu') \in \llbracket P_2 \rrbracket_i^{\mathbf{G}, \xi}, \mu' \sim \mu, \text{dom}(\mu') \cap \text{dom}(\mu) \neq \emptyset \}, \\ \llbracket \text{PROJECT}(P_1, L) \rrbracket_i^{\mathbf{G}, \xi} &= \text{TC} \{ (I, \mu|_L) \mid (I, \mu) \in \llbracket P_1 \rrbracket_i^{\mathbf{G}, \xi} \}, \\ \llbracket \text{EXTEND}(P_1, v, T) \rrbracket_i^{\mathbf{G}, \xi} &= \{ (I, \mu \dot{\cup} \{v \mapsto \llbracket T \rrbracket^\mu\}) \mid (I, \mu) \in \llbracket P_1 \rrbracket_i^{\mathbf{G}, \xi} \}. \end{aligned}$$

As before, the evaluation of a graph pattern P over G is $\llbracket P \rrbracket_i^{G, \emptyset}$. Again, because the definitions of graph patterns and filter expressions (with EXISTS) are mutually recursive, we also need to extend this evaluation function to a filter expression F :

$$\llbracket \text{EXISTS}(P) \rrbracket_i^{G, \xi} = \begin{cases} \top, & \text{if } \llbracket P \rrbracket_i^{G, \xi} \neq \emptyset, \\ \perp, & \text{otherwise.} \end{cases}$$

For all other types of filter expressions, $\llbracket F \rrbracket_i^{G, \xi}$ is defined analogously to $\llbracket F \rrbracket_a^{G, \xi}$.

We first observe that, similarly to temporal relation algebra, some operations, namely UNION and PROJECT, naturally require an additional TC operator in their semantics [3, Theorem 4.1]; on the other hand, in line with temporal relation algebra, FILTER does not require TC [3, Theorem 4.2]. Rather interestingly, the compatibility condition for JOIN in SPARQL leads to a more subtle definition of its semantics, which surprisingly also requires an additional TC operator (which is in stark contrast to its temporal relational algebra counterpart, INNER JOIN, which preserves coalescing without any additional measures [3, Theorem 4.2]). We illustrate the issue on the following example.

► **Example 4.** Consider $\llbracket \text{JOIN}(P_1, P_2) \rrbracket_i^{G, \emptyset}$ with

$$\llbracket P_1 \rrbracket_i^{G, \emptyset} = \left\{ \begin{array}{l} ([0, 1), \{x \mapsto :a\}) \\ ([1, 2), \{y \mapsto :b\}) \end{array} \right\}, \quad \llbracket P_2 \rrbracket_i^{G, \emptyset} = \{([0, 2), \{x \mapsto :a, y \mapsto :b\})\}.$$

Both solution mappings in $\llbracket P_1 \rrbracket_i^{G, \emptyset}$ are compatible with the solution mapping in $\llbracket P_2 \rrbracket_i^{G, \emptyset}$, and their intervals overlap. So the simple expression $(I_1 \cap I_2, \mu_1 \dot{\cup} \mu_2)$ would give us the following two temporal solution mappings:

$$([0, 1), \{x \mapsto :a, y \mapsto :b\}) \quad \text{and} \quad ([1, 2), \{x \mapsto :a, y \mapsto :b\}),$$

which are evidently not coalesced.

The semantics of both OPTIONAL and MINUS obviously requires the TC operator as the $I' \subseteq I \setminus \bigcup J^\mu$ component gives us all possible I' (not coalesced) and is in fact even potentially unbounded. While we could push the TC operator inside the set formation for MINUS:

$$\llbracket \text{MINUS} \rrbracket_i^{G, \xi}(P_1, P_2) = \{(I', \mu) \mid (I, \mu) \in \llbracket P_1 \rrbracket_i^{G, \xi}, I' \in \mathcal{C}(I \setminus \bigcup J^\mu)\},$$

the TC operator in OPTIONAL would still be required due to $\cup$, as our next example shows.

► **Example 5.** Consider $\llbracket \text{OPTIONAL}(P_1, P_2, F) \rrbracket_i^{G, \emptyset}$ with $F = \neg \text{BOUND}(z)$ and

$$\llbracket P_1 \rrbracket_i^{G, \emptyset} = \left\{ \begin{array}{l} ([0, 1), \{x \mapsto :a\}) \\ ([1, 2), \{x \mapsto :a, y \mapsto :b\}) \end{array} \right\}, \quad \llbracket P_2 \rrbracket_i^{G, \emptyset} = \left\{ \begin{array}{l} ([0, 1), \{x \mapsto :a, y \mapsto :b\}) \\ ([1, 2), \{x \mapsto :a, z \mapsto :c\}) \end{array} \right\}.$$

Evaluating $\llbracket Q \rrbracket_i^{G, \emptyset}$, for $Q = \text{FILTER}(\text{JOIN}(P_1, P_2), F)$, gives us

$$([0, 1), \{x \mapsto :a, y \mapsto :b\}).$$

On the other hand, since $([1, 2), \mu) \in \llbracket P_1 \rrbracket_i^{G, \emptyset}$ but $J^\mu = \{[0, 1)\}$, for $\mu = \{x \mapsto :a, y \mapsto :b\}$, the second component of the union produces

$$([1, 2), \{x \mapsto :a, y \mapsto :b\}),$$

which means that the set with the two temporal solution mappings above needs be coalesced.

■ **Listing 2** Married Athletes query in Temporal SPARQL (for prefix declarations, see Listing 1).

```
SELECT DISTINCT ?x ?z ?c ?t
WHERE {
  << ?x :playsFor ?c >> time:hasTime ?t .
  << ?x :marriedTo ?z >> time:hasTime ?t .
}
```

Note, however, that, in some concrete graph patterns, the TC operator could be redundant. For example, if both P_1 and P_2 in $\text{JOIN}(P_1, P_2)$ are temporal basic graph patterns (BGPs), that is, JOINS of temporal quad patterns, then the TC operator in $\llbracket \text{JOIN}(P_1, P_2) \rrbracket_i^{\mathcal{G}, \xi}$ is not needed (this can be shown by using the same argument as for the INNER JOIN in temporal relational algebra [3, Theorem 4.2]). The same applies to combinations of JOIN and FILTER patterns. A more in-depth study of sufficient conditions for optimisations eliminating superfluous TC operators is left for future work.

We now show that the defined semantics is correct, meaning that it coincides with the one defined for the logical model in Section 3.2. Let $\mathcal{G}_{S_T}$ denote the S_T -annotated graph associated with $\mathcal{G}$, defined by taking $\mathcal{G}_{S_T}(s, p, o) = \bigcup \overline{\mathcal{G}}(s, p, o)$, for all $(s, p, o) \in \mathcal{RDF}$. Then we have the following result (shown in Appendix A).

► **Theorem 6.** *For any coalesced temporal graph $\mathcal{G}$, graph pattern P , context ξ and solution mapping μ , we have $\llbracket P \rrbracket_i^{\mathcal{G}, \xi}(\mu) = \mathcal{C}(\llbracket P \rrbracket_a^{\mathcal{G}_{S_T}, \xi}(\mu))$.*

5 Experiments

For our experiments, we take the setting of *virtual* knowledge graphs, where the data source is relational and exposed as an RDF graph via a mapping. Input queries in Temporal SPARQL are rewritten by the system (using the mapping) into equivalent SQL queries to be evaluated over the source, thus leveraging the existing capabilities of the relational world. In addition to its flexibility, this approach is arguably better suited to joins over temporal annotations than a triple store, as we shall see below. Besides, as shown by Zhou et al. [19], coalescing can be efficiently performed in SQL by means of window functions. So, we present here two scenarios, complete with data and input temporal queries, and show in each case how such a system could produce equivalent SQL queries and what evaluation time can be achieved, to highlight the promise of our approach.

The experiments were run on a Lenovo ThinkPad X13 Gen 5 with 32GB RAM of LPDDR5 memory running at 7467 MT/s, an Intel Core Ultra 7 155U CPU, under Ubuntu 25.10, using Linux kernel version 6.11.0-17. The OS and data are running on an SSD. The datasets were loaded on a local Postgres v. 17.9 installation. The Temporal SPARQL queries were translated by hand; the resulting SQL queries are given in this section as well as in Appendix B. We report the evaluation times in Table 1.

5.1 Scenario 1: Married Athletes on Wikidata

Chekol et al. [4] develop bespoke algorithms for computing temporal joins in SPARQL because one of their motivating examples demonstrates a clear need for a better temporal join implementation: existing SPARQL endpoints show poor performance. The query uses Wikidata and asks for all people and their spouses (predicate `wdt:marriedTo/P56`) such that they were married at some point while they represented a team/club (`wdt:playsFor/P54`). In fact, to

■ **Table 1** Evaluation times of Temporal SPARQL queries translated to SQL.

query	evaluation time (ms)
Married Athletes Query (translated)	623.955
Flights Query 1 (translated)	1 054.958
Flights Query 2 (translated)	204 458.840

■ **Table 2** WikiData cardinality info for the join selectivity in the Married Athletes query.

basic graph pattern	cardinality
<code>?x p:playsFor ?r1.</code>	1 762 340
<code>?x p:marriedTo ?r2.</code>	920 713
<code>?x p:playsFor ?r1. ?x p:marriedTo ?r2 .</code>	12 593
<code>?r1 pq:startTime ?s1 .</code>	10 641 734
<code>?r1 pq:endTime ?e1 .</code>	6 457 762
<code>?r1 pq:startTime ?s1 . ?r1 pq:endTime ?e1 .</code>	timeout

refer to temporal annotations (`pq:startTime/P580` and `pq:endTime/P582`) of the statements, one has to access *reified* statements using pairs of predicates `p:marriedTo/ps:marriedTo` and `p:playsFor/ps:playsFor`, so the following basic graph pattern is used:

```
?x p:playsFor ?r1. ?r1 ps:playsFor ?z; pq:startTime ?s1; pq:endTime ?e1.
?x p:marriedTo ?r2. ?r2 ps:marriedTo ?y; pq:startTime ?s2; pq:endTime ?e2.
```

where `;` separates predicate-object pairs that share the same subject. The authors reported that this query timed out over the public Wikidata endpoint. We repeated their experiment on the present-day public Wikidata endpoint: it actually does evaluate, but takes 45.519s to run⁵. Of course, this does not contradict the findings, as the underlying systems and software are likely to have changed, but nevertheless nearly a minute for such a simple query is not very impressive. To understand the challenge, we present some statistics in Table 2. The join size is really dominated by the initial join between relations `p:playsFor` and `p:marriedTo`. Of the $1\,762\,340 \times 920\,713 \approx 1.5 \cdot 10^{12}$ possible combinations, only minuscule 12 593 end up in the join result. In addition, the join over the start (`pq:startTime`) and end (`pq:endTime`) points, i.e., the objects that have both some start and end point, already times out on its own. Looking at the cardinality of the two component triple patterns, we know that the join result has at most $\min\{10\,641\,734, 6\,457\,762\} = 6\,457\,762$ possible pairs, since any given object can only have a single start and a single end point in time.

We can see in Table 1 our SQL version of this query (shown in Listing 3) only needs 624ms to evaluate in Postgres, and is thus multiple orders of magnitude faster than the SPARQL query with reification. For this experiment, we extracted the information on athletes and married couples from Wikidata into two SQL tables, `athletes` and `married`, with 1 212 855 and 404 486 rows, respectively. Note that about 31% of the `wdt:playsFor` relationships have no `pq:startTime` annotation, and so are discarded; on the other hand, we use the current date for the end of an interval if there is a `pq:startTime` but no `pq:endTime`. The same applies to `wdt:marriedTo`, where, in addition, since it is a symmetric property, we use two columns, `spousea` or `spouseb`, in the `married` table to halve the number of rows.

⁵ <https://w.wiki/NBcE>

■ **Listing 3** SQL translation of the Married Athletes query in Listing 2.

```
SELECT a.person, m.spouseb AS spouse, a.club,
       GREATEST(a.s, m.s) AS ts, LEAST(a.e, m.e) AS te
FROM   athletes AS a, married AS m
WHERE  a.person = m.spousea AND GREATEST(a.s, m.s) < LEAST(a.e, m.e)
UNION ALL
SELECT a.person, m.spousea AS spouse, a.club,
       GREATEST(a.s, m.s) AS ts, LEAST(a.e, m.e) AS te
FROM   athletes AS a, married AS m
WHERE  a.person = m.spouseb AND GREATEST(a.s, m.s) < LEAST(a.e, m.e)
```

The query in Listing 3 is obtained by a rather straightforward translation of the Temporal SPARQL query in Listing 2. We encode the temporal join as an inner join over the **person** and **spouse** columns with an additional temporal overlap condition: the filter condition `GREATEST(a.s, m.s) < LEAST(a.e, m.e)` ensures that intervals $[a.s, a.e)$ and $[m.s, m.e)$ overlap, while `GREATEST(a.s, m.s)` and `LEAST(a.e, m.e)` produce the endpoints for the intersection of these intervals; note that SQL functions `GREATEST` and `LEAST` return the maximum and the minimum in a given list of values. Finally, since `wdt:marriedTo` is a symmetric property, the results of two subqueries that obtain spouses from **spousea** and **spouseb** columns are combined with `UNION ALL`.

The SQL version can be so much faster since it can limit itself to a simple two-way join over the two tables of interest, whereas the SPARQL query essentially has to consider a BGP of 8 triple patterns, each of which involves millions triples completely unrelated to the (very small) query output (2 353 matches).

5.2 Scenario 2: Flight Dataset

The second scenario is over an excerpt of public international flight data for November 2014 [1], which has previously been used as a dataset for experiments on interval join algorithms [16]. The dataset consists of 57 585 entries.

Over this flights dataset, we are considering two queries. Listing 4 shows a Temporal SPARQL query that considers all combinations of pairs of flights to **FRA** (Frankfurt Airport) and **MUC** (Munich Airport), which are in the air at the same time, meaning that their intervals between flight departures and arrivals intersect. We refer to these interval as the “flight intervals”. Listing 5 extends the first query by an `OPTIONAL` that adds a third flight to **CDG** (Paris Charles de Gaulle International Airport). In each query, the time interval output is the flight interval that the two (three, respectively) flights have in common.

In Table 1, we can see that the SQL translation of the query in Listing 4 runs particularly fast, needing a mere second. This is no surprise, since, similarly to Listing 3, the temporal join can be translated into a simple self-join over the **flights** table with an additional inequality. In comparison, the SQL translation of the query in Listing 5 involves a temporal `LEFT JOIN` and is much slower. This too comes at no surprise, when one considers the size of the translated query. The encoding of temporal `LEFT JOIN` follows the example translations given by Dignös et al. [6] and requires a number of common table expressions (CTEs) with SQL window functions.

■ **Listing 4** Flights Query 1 (for prefix declarations, see Listing 1).

```
SELECT DISTINCT ?x ?t
WHERE {
  << ?x :fap 'FRA' >> time:hasTime ?t .
  << ?z :fap 'MUC' >> time:hasTime ?t .
}
```

■ **Listing 5** Flights Query 2 (for prefix declarations, see Listing 1).

```
SELECT DISTINCT ?x ?w ?t
WHERE {
  << ?x :fap 'FRA' >>      time:hasTime ?t .
  << ?z :fap 'MUC' >>      time:hasTime ?t .
  OPTIONAL {
    << ?w :fap 'CDG' >> time:hasTime ?t .
  }
}
```

6 Conclusion

We equipped SPARQL with temporal abilities, for a fragment of the language that we call Temporal SPARQL, with suitable restrictions to support interval comparisons (using, e.g., Allen’s relations) while retaining tractable query evaluation. We provided a formal semantics for this language that complies with the specifics of the SPARQL algebra, which is based on the compatibility of partial solution mapping. Analogously to what was done by Dignös et al. [7] for relational algebra, this semantics comes in three flavours, with different levels of abstraction, where time is presented as time points, unions of intervals and single intervals, respectively. The third model operates directly on interval-based temporal data. We concluded the paper with a feasibility study, showing that Temporal SPARQL can be translated to SQL, and that this solution may outperform triple stores.

A natural continuation of this work is extending OBDA systems with the ability to express queries over interval-based temporal data. Our paper has identified Temporal SPARQL as a strong candidate input language of such a system, and the topic is of high interest to the OBDA community, albeit highly non-trivial. As for the language itself, even though this fragment can express a number of interesting temporal queries, it does not handle *aggregation* over time, for instance, getting the maximal measurement within a certain time span, or counting events within a certain period. A careful investigation is needed to understand whether this can be achieved without compromising the desirable properties of this fragment.

References

- 1 Andreas Behrend and Gereon Schüller. A case study in optimizing continuous queries using the magic update technique. In *Proc. of the 26th Int. Conf. on Scientific and Statistical Database Management, SSDBM’14*, pages 31:1–31:4. ACM, 2014. doi:10.1145/2618243.2618285.
- 2 Michael H. Böhlen, Anton Dignös, Johann Gamper, and Christian S. Jensen. Temporal data management – an overview. In *Tutorial Lectures of the 7th European Summer School on Business Intelligence and Big Data (eBISS)*, volume 324 of *Lecture Notes in Business Information Processing*, pages 51–83. Springer-Verlag, 2017. doi:10.1007/978-3-319-96655-7_3.

- 3 Michael H. Böhlen, Richard T. Snodgrass, and Michael D. Soo. Coalescing in temporal databases. In *Proc. of 22nd Int. Conf. on Very Large Data Bases, VLDB'96*, pages 180–191. Morgan Kaufmann, 1996. URL: <http://www.vldb.org/conf/1996/P180.PDF>.
- 4 Melisachew Wudage Chekol, Giuseppe Pirrò, and Heiner Stuckenschmidt. Fast interval joins for temporal SPARQL queries. In *Companion of the 2019 World Wide Web Conference, WWW 2019*, pages 1148–1154. ACM, 2019. doi:10.1145/3308560.3314997.
- 5 Joseph C. Culberson and Robert A. Reckhow. Covering polygons is hard (preliminary abstract). In *Proc. of the 29th Annual Symposium on Foundations of Computer Science, SFCS'88*, pages 601–611. IEEE Computer Society, 1988. doi:10.1109/SFCS.1988.21976.
- 6 Anton Dignös, Boris Glavic, Xing Niu, Michael H. Böhlen, and Johann Gamper. Snapshot semantics for temporal multiset relations (extended version). *CoRR*, abs/1902.04938, 2019. doi:10.48550/arXiv.1902.04938.
- 7 Anton Dignös, Boris Glavic, Xing Niu, Johann Gamper, and Michael H. Böhlen. Snapshot semantics for temporal multiset relations. *Proc. VLDB Endow.*, 12(6):639–652, 2019. doi:10.14778/3311880.3311882.
- 8 Floris Geerts and Antonella Poggi. On database query languages for k-relations. *Journal of Applied Logic*, 8(2):173–185, 2010. doi:10.1016/j.jal.2009.09.001.
- 9 Floris Geerts, Thomas Unger, Grigoris Karvounarakis, Irini Fundulaki, and Vassilis Christophides. Algebraic structures for capturing the provenance of SPARQL queries. *J. ACM*, 63(1):7:1–7:63, 2016. doi:10.1145/2810037.
- 10 Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In *Proc. of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'07*, pages 31–40. ACM, 2007. doi:10.1145/1265530.1265535.
- 11 Olaf Hartig and Bryan Thompson. Foundations of an alternative approach to reification in RDF. *CoRR*, abs/1406.3399, 2014. doi:10.48550/arXiv.1406.3399.
- 12 Elem Güzel Kalayci, Sebastian Brandt, Diego Calvanese, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyashev. 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.
- 13 Mark Kaminski, Egor V. Kostylev, and Bernardo Cuenca Grau. Query nesting, assignment, and aggregation in SPARQL 1.1. *ACM Trans. Database Syst.*, 42(3):17:1–17:46, 2017. doi:10.1145/3083898.
- 14 Szymon Klarman and Thomas Meyer. Querying temporal databases via OWL 2 QL. In *Proc. of the 8th Int. Conf. on Web Reasoning and Rule Systems, RR 2014*, Lecture Notes in Computer Science, pages 92–107. Springer, 2014. doi:10.1007/978-3-319-11113-1_7.
- 15 Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3):16:1–16:45, 2009. doi:10.1145/1567274.1567278.
- 16 Danila Piatov, Sven Helmer, and Anton Dignös. An interval join optimized for modern hardware. In *Proc. of the 32nd IEEE Int. Conf. on Data Engineering, ICDE 2016*, pages 1098–1109. IEEE Computer Society, 2016. doi:10.1109/ICDE.2016.7498316.
- 17 David Toman. Point-based temporal extensions of SQL and their efficient implementation. In *Temporal Databases: Research and Practice*, volume 1399 of *Lecture Notes in Computer Science*, pages 211–237. Springer, 1998. doi:10.1007/BFb0053704.
- 18 Guohui Xiao, Roman Kontchakov, Benjamin Cogrel, Diego Calvanese, and Elena Botoeva. Efficient handling of SPARQL OPTIONAL for OBDA. In *Proc. of the 17th Int. Conf. on The Semantic Web, ISWC 2018, Part I*, volume 11136 of *Lecture Notes in Computer Science*, pages 354–373. Springer, 2018. doi:10.1007/978-3-030-00671-6_21.
- 19 Gang Zhou, Yafeng Wu, Ting Yan, Tian He, Chengdu Huang, John A. Stankovic, and Tarek F. Abdelzaher. A multifrequency mac specially designed for wireless sensor network applications. *ACM Trans. Embed. Comput. Syst.*, 9(4):39:1–39:41, 2010. doi:10.1145/1721695.1721705.
- 20 Xin Zhou, Fusheng Wang, and Carlo 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 *Lecture Notes in Computer Science*, pages 676–686. Springer, 2006. doi:10.1007/11827405_66.

A Proof of Theorem 6

We show that for a coalesced temporal graph G , context ξ and solution mapping μ ,

$$\begin{aligned}\overline{\llbracket P \rrbracket_i^{G,\xi}(\mu)} &= C(\llbracket P \rrbracket_a^{G_{S_T},\xi}(\mu)), \text{ for any graph pattern } P, \\ \llbracket F \rrbracket_i^{G,\mu} &= \llbracket F \rrbracket_a^{G_{S_T},\mu}, \text{ for any filter expression } F,\end{aligned}$$

by mutual induction on the structure of P and F . To avoid notational clutter, we fix G and omit G and G_{S_T} as superscripts, i.e., we write $\llbracket P \rrbracket_i^\xi$, $\llbracket P \rrbracket_a^\xi$, $\llbracket F \rrbracket_i^\mu$ and $\llbracket F \rrbracket_a^\mu$.

■ $P = (s, p, o, t)$.

- If $\text{dom}(\mu) \neq \text{var}((s, p, o)^\xi)$ or $(s, p, o)^{\mu \dot{\cup} \xi} \notin \mathcal{RDF}$, then $\overline{\llbracket P \rrbracket_i^\xi(\mu)} = \emptyset$ and $\llbracket P \rrbracket_a^\xi(\mu) = \emptyset$.
- Otherwise, we have

$$\begin{aligned}C(\llbracket (s, p, o) \rrbracket_a^\xi(\mu)) &= C(G_{S_T}((s, p, o)^{\mu \dot{\cup} \xi})) && (\llbracket \cdot \rrbracket_a^\xi \text{ def.}) \\ &= C(\bigcup \overline{G}((s, p, o)^{\mu \dot{\cup} \xi})) && (G_{S_T} \text{ def.}) \\ &= \bigcup \overline{G}((s, p, o)^{\mu \dot{\cup} \xi}) && (G \text{ is coalesced}) \\ &= \overline{\llbracket (s, p, o) \rrbracket_i^\xi(\mu)}. && (\llbracket \cdot \rrbracket_i^\xi \text{ def.})\end{aligned}$$

■ $P = \text{FILTER}(P_1, F)$, where P_1 binds temporal variable t .

$$\begin{aligned}C(\llbracket \text{FILTER}(P_1, F) \rrbracket_a^\xi(\mu)) &= C(\bigcup \{I \in C(\llbracket P_1 \rrbracket_a^\xi(\mu)) \mid \llbracket F \rrbracket_a^{\xi \dot{\cup} \mu \dot{\cup} \{t \mapsto I\}} = \top\}) && (\llbracket \cdot \rrbracket_a^\xi \text{ def.}) \\ &= \{I \in C(\llbracket P_1 \rrbracket_a^\xi(\mu)) \mid \llbracket F \rrbracket_a^{\xi \dot{\cup} \mu \dot{\cup} \{t \mapsto I\}} = \top\} && (\dagger) \\ &= \{I \in \overline{\llbracket P_1 \rrbracket_i^\xi(\mu)} \mid \llbracket F \rrbracket_i^{\xi \dot{\cup} \mu \dot{\cup} \{t \mapsto I\}} = \top\} && (\text{IH}) \\ &= \overline{\{(I, \mu') \in \llbracket P_1 \rrbracket_i^\xi \mid \llbracket F \rrbracket_i^{\xi \dot{\cup} \mu' \dot{\cup} \{t \mapsto I\}} = \top\}(\mu)} && (\dot{\cup} \text{ def.}) \\ &= \overline{\llbracket \text{FILTER}(P_1, F) \rrbracket_i^\xi(\mu)}, && (\llbracket \cdot \rrbracket_i^\xi \text{ def.})\end{aligned}$$

where $(\dagger)$ holds because a subset of a coalesced set of intervals is also coalesced.

■ $P = \text{UNION}(P_1, P_2)$.

$$\begin{aligned}C(\llbracket \text{UNION}(P_1, P_2) \rrbracket_a^\xi(\mu)) &= C(\llbracket P_1 \rrbracket_a^\xi(\mu) \cup \llbracket P_2 \rrbracket_a^\xi(\mu)) && (\llbracket \cdot \rrbracket_a^\xi \text{ def.}) \\ &= C(\bigcup C(\llbracket P_1 \rrbracket_a^\xi(\mu)) \cup \bigcup C(\llbracket P_2 \rrbracket_a^\xi(\mu))) && (s = \bigcup C(s)) \\ &= C(\bigcup \overline{\llbracket P_1 \rrbracket_i^\xi(\mu)} \cup \bigcup \overline{\llbracket P_2 \rrbracket_i^\xi(\mu)}) && (\text{IH}) \\ &= C(\bigcup (\overline{\llbracket P_1 \rrbracket_i^\xi(\mu)} \cup \overline{\llbracket P_2 \rrbracket_i^\xi(\mu)})) && (\dot{\cup} \text{ def.}) \\ &= \overline{\text{TC}(\llbracket P_1 \rrbracket_i^\xi \cup \llbracket P_2 \rrbracket_i^\xi)(\mu)} && (\text{TC def.}) \\ &= \overline{\llbracket \text{UNION}(P_1, P_2) \rrbracket_i^\xi(\mu)}. && (\llbracket \cdot \rrbracket_i^\xi \text{ def.})\end{aligned}$$

■ $P = \text{JOIN}(P_1, P_2)$.

$$\begin{aligned}C(\llbracket \text{JOIN}(P_1, P_2) \rrbracket_a^\xi(\mu)) &= C\left(\bigcup_{\mu_1 \dot{\cup} \mu_2 = \mu} [\llbracket P_1 \rrbracket_a^\xi(\mu_1) \cap \llbracket P_2 \rrbracket_a^\xi(\mu_2)]\right) && (\llbracket \cdot \rrbracket_a^\xi \text{ def.}) \\ &= C\left(\bigcup_{\mu_1 \dot{\cup} \mu_2 = \mu} \left[\left(\bigcup C(\llbracket P_1 \rrbracket_a^\xi(\mu_1))\right) \cap \left(\bigcup C(\llbracket P_2 \rrbracket_a^\xi(\mu_2))\right)\right]\right) && (s = \bigcup C(s))\end{aligned}$$

$$\begin{aligned}
&= \mathsf{C} \left(\bigcup_{\mu_1 \dot{\cup} \mu_2 = \mu} \left[\left(\bigcup \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu_1) \right) \cap \left(\bigcup \overline{\llbracket P_2 \rrbracket_i^\xi}(\mu_2) \right) \right] \right) \quad (\text{IH}) \\
&= \mathsf{C} \left(\bigcup_{\mu_1 \dot{\cup} \mu_2 = \mu} \left[\left(\bigcup \{I_1 \mid (I_1, \mu_1) \in \llbracket P_1 \rrbracket_i^\xi\} \right) \cap \left(\bigcup \{I_2 \mid (I_2, \mu_2) \in \llbracket P_2 \rrbracket_i^\xi\} \right) \right] \right) \quad (\cdot \text{ def.}) \\
&= \mathsf{C} \left(\bigcup_{\mu_1 \dot{\cup} \mu_2 = \mu} \bigcup \{I_1 \cap I_2 \mid (I_1, \mu_1) \in \llbracket P_1 \rrbracket_i^\xi, (I_2, \mu_2) \in \llbracket P_2 \rrbracket_i^\xi\} \right) \quad (\cap \text{ distr. over } \cup) \\
&= \mathsf{C} \left(\bigcup \{I_1 \cap I_2 \mid \mu_1 \dot{\cup} \mu_2 = \mu, (I_1, \mu_1) \in \llbracket P_1 \rrbracket_i^\xi, (I_2, \mu_2) \in \llbracket P_2 \rrbracket_i^\xi\} \right) \quad (\cup \text{ assoc.}) \\
&= \mathsf{C} \left(\bigcup \{ (I_1 \cap I_2, \mu_1 \dot{\cup} \mu_2) \mid \llbracket P_1 \rrbracket_i^\xi \ni (I_1, \mu_1) \sim (I_2, \mu_2) \in \llbracket P_2 \rrbracket_i^\xi \}(\mu) \right) \quad (\cdot \text{ def.}) \\
&= \overline{\text{TC}\{(I_1 \cap I_2, \mu_1 \dot{\cup} \mu_2) \mid \llbracket P_1 \rrbracket_i^\xi \ni (I_1, \mu_1) \sim (I_2, \mu_2) \in \llbracket P_2 \rrbracket_i^\xi\}(\mu)} \quad (\text{TC def.}) \\
&= \overline{\llbracket \text{JOIN}(P_1, P_2) \rrbracket_i^\xi}(\mu), \quad (\llbracket \cdot \rrbracket_i^\xi \text{ def.})
\end{aligned}$$

where we write $(I_1, \mu_1) \sim (I_2, \mu_2)$ if $\mu_1 \sim \mu_2$ and $I_1 \cap I_2 \neq \emptyset$.

■ $P = \text{MINUS}(P_1, P_2)$.

$$\begin{aligned}
&\mathsf{C}(\llbracket \text{MINUS}(P_1, P_2) \rrbracket_a^\xi(\mu)) \\
&= \mathsf{C} \left(\llbracket P_1 \rrbracket_a^\xi(\mu) \setminus \bigcup_{\substack{\mu_2 \sim \mu \\ \text{dom}(\mu_2) \cap \text{dom}(\mu) \neq \emptyset}} \llbracket P_2 \rrbracket_a^\xi(\mu_2) \right) \quad (\llbracket \cdot \rrbracket_a^\xi \text{ def.}) \\
&= \mathsf{C} \left(\bigcup \mathsf{C}(\llbracket P_1 \rrbracket_a^\xi(\mu)) \setminus \bigcup_{\substack{\mu_2 \sim \mu \\ \text{dom}(\mu_2) \cap \text{dom}(\mu) \neq \emptyset}} \bigcup \mathsf{C}(\llbracket P_2 \rrbracket_a^\xi(\mu_2)) \right) \quad (s = \bigcup \mathsf{C}(s)) \\
&= \mathsf{C} \left(\bigcup \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu) \setminus \bigcup_{\substack{\mu_2 \sim \mu \\ \text{dom}(\mu_2) \cap \text{dom}(\mu) \neq \emptyset}} \bigcup \overline{\llbracket P_2 \rrbracket_i^\xi}(\mu_2) \right) \quad (\text{IH}) \\
&= \mathsf{C} \left(\bigcup \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu) \setminus \bigcup_{\substack{\mu_2 \sim \mu \\ \text{dom}(\mu_2) \cap \text{dom}(\mu) \neq \emptyset}} \bigcup \{J \mid (J, \mu_2) \in \llbracket P_2 \rrbracket_i^\xi\} \right) \quad (\cdot \text{ def.}) \\
&= \mathsf{C} \left(\bigcup \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu) \setminus \bigcup \mathbf{J}^\mu \right) \quad (\mathbf{J}^\mu \text{ def.}) \\
&= \mathsf{C} \left(\bigcup \{I_1 \setminus \bigcup \mathbf{J}^\mu \mid I_1 \in \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu)\} \right) \quad (\setminus \text{ right-distr. over } \cup) \\
&= \mathsf{C} \left(\bigcup \{I \in \mathbb{I} \mid I_1 \in \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu), I \subseteq I_1 \setminus \bigcup \mathbf{J}^\mu\} \right) \quad (\dagger) \\
&= \mathsf{C} \left(\bigcup \{I \in \mathbb{I} \mid (I_1, \mu_1) \in \llbracket P_1 \rrbracket_i^\xi, I \subseteq I_1 \setminus \bigcup \mathbf{J}^{\mu_1}\}(\mu) \right) \quad (\cdot \text{ def.}) \\
&= \overline{\text{TC}\{(I, \mu_1) \mid (I_1, \mu_1) \in \llbracket P_1 \rrbracket_i^\xi, I \subseteq I_1 \setminus \bigcup \mathbf{J}^{\mu_1}\}(\mu)} \quad (\text{TC def.}) \\
&= \overline{\llbracket \text{MINUS}(P_1, P_2) \rrbracket_i^\xi}(\mu). \quad (\llbracket \cdot \rrbracket_i^\xi \text{ def.})
\end{aligned}$$

where $(\dagger)$ holds because $I' = \bigcup (2^{I'} \cap \mathbb{I})$, for any interval I' (in particular for $I' = I_1 \setminus \bigcup \mathbf{J}^\mu$).

■ $P = \text{OPTIONAL}(P_1, P_2, F)$ with $Q = \text{FILTER}(\text{JOIN}(P_1, P_2), F)$.

$$\begin{aligned}
&\mathsf{C}(\llbracket \text{OPTIONAL}(P_1, P_2, F) \rrbracket_a^\xi(\mu)) \\
&= \mathsf{C} \left(\llbracket Q \rrbracket_a^\xi(\mu) \cup \left(\llbracket P_1 \rrbracket_a^\xi(\mu) \setminus \bigcup_{\mu_2 \sim \mu} \llbracket Q \rrbracket_a^\xi(\mu \dot{\cup} \mu_2) \right) \right) \quad (\llbracket \cdot \rrbracket_a^\xi \text{ def.}) \\
&= \mathsf{C} \left(\bigcup \mathsf{C}(\llbracket Q \rrbracket_a^\xi(\mu)) \cup \left(\bigcup \mathsf{C}(\llbracket P_1 \rrbracket_a^\xi(\mu)) \setminus \bigcup_{\mu_2 \sim \mu} \bigcup \mathsf{C}(\llbracket Q \rrbracket_a^\xi(\mu \dot{\cup} \mu_2)) \right) \right) \quad (s = \bigcup \mathsf{C}(s)) \\
&= \mathsf{C} \left(\bigcup \overline{\llbracket Q \rrbracket_i^\xi}(\mu) \cup \left(\bigcup \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu) \setminus \bigcup_{\mu_2 \sim \mu} \bigcup \overline{\llbracket Q \rrbracket_i^\xi}(\mu \dot{\cup} \mu_2) \right) \right) \quad (\text{IH})
\end{aligned}$$

$$\begin{aligned}
&= \mathsf{C} \left(\bigcup \overline{\llbracket Q \rrbracket_i^\xi}(\mu) \cup \left(\bigcup \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu) \setminus \bigcup_{\mu_2 \sim \mu} \bigcup \{J \mid (J, \mu \dot{\cup} \mu_2) \in \llbracket Q \rrbracket_i^\xi\} \right) \right) & (\bar{\cdot} \text{ def.}) \\
&= \mathsf{C} \left(\bigcup \overline{\llbracket Q \rrbracket_i^\xi}(\mu) \cup \left(\bigcup \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu) \setminus \bigcup \mathbf{J}^\mu \right) \right) & (\mathbf{J}^\mu \text{ def.}) \\
&= \mathsf{C} \left(\bigcup \left(\overline{\llbracket Q \rrbracket_i^\xi}(\mu) \cup \{I_1 \setminus \bigcup \mathbf{J}^\mu \mid I_1 \in \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu)\} \right) \right) & (\setminus \text{ right-distr. over } \cup) \\
&= \mathsf{C} \left(\bigcup \left(\overline{\llbracket Q \rrbracket_i^\xi}(\mu) \cup \{I \in \mathbb{I} \mid I_1 \in \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu), I \subseteq I_1 \setminus \bigcup \mathbf{J}^\mu\} \right) \right) & (\dagger) \\
&= \mathsf{C} \left(\bigcup \left(\overline{\llbracket Q \rrbracket_i^\xi} \cup \{(I, \mu_1) \mid (I, \mu_1) \in \llbracket P_1 \rrbracket_i^\xi, I \subseteq I_1 \setminus \bigcup \mathbf{J}^{\mu_1}\} \right) \right) & (\bar{\cdot} \text{ def.}) \\
&= \overline{\mathsf{TC} \left(\llbracket Q \rrbracket_i^\xi \cup \{(I, \mu_1) \mid (I, \mu_1) \in \llbracket P_1 \rrbracket_i^\xi, I \subseteq I_1 \setminus \bigcup \mathbf{J}^{\mu_1}\} \right) (\mu)} & (\mathsf{TC} \text{ def.}) \\
&= \overline{\llbracket \text{OPTIONAL}(P_1, P_2, F) \rrbracket_i^\xi}(\mu). & (\llbracket \cdot \rrbracket_i^\xi \text{ def.})
\end{aligned}$$

where $(\dagger)$ holds because $I' = \bigcup (2^{I'} \cap \mathbb{I})$, for any interval I' (in particular for $I' = I_1 \setminus \bigcup \mathbf{J}^\mu$).

■ $P = \text{PROJECT}(P_1, L)$.

$$\begin{aligned}
\mathsf{C} \left(\llbracket \text{PROJECT}(P_1, L) \rrbracket_a^\xi(\mu) \right) &= \mathsf{C} \left(\bigcup_{\mu'_L = \mu} \llbracket P_1 \rrbracket_a^\xi(\mu') \right) & (\llbracket \cdot \rrbracket_a^\xi \text{ def.}) \\
&= \mathsf{C} \left(\bigcup_{\mu'_L = \mu} \bigcup \mathsf{C} \left(\llbracket P_1 \rrbracket_a^\xi(\mu') \right) \right) & (s = \bigcup \mathsf{C}(s)) \\
&= \mathsf{C} \left(\bigcup_{\mu'_L = \mu} \bigcup \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu') \right) & (\text{IH}) \\
&= \mathsf{C} \left(\bigcup \{I \mid (I, \mu') \in \llbracket P_1 \rrbracket_i^\xi, \mu'_L = \mu\} \right) & (\bar{\cdot} \text{ def.}) \\
&= \overline{\mathsf{TC} \{ (I, \mu'') \mid (I, \mu') \in \llbracket P_1 \rrbracket_i^\xi, \mu'_L = \mu'' \} (\mu)} & (\mathsf{TC} \text{ def.}) \\
&= \overline{\mathsf{TC} \{ (I, \mu'_L) \mid (I, \mu') \in \llbracket P_1 \rrbracket_i^\xi \} (\mu)} & (\text{eliminate } \mu'') \\
&= \overline{\llbracket \text{PROJECT}(P_1, L) \rrbracket_i^\xi}(\mu). & (\llbracket \cdot \rrbracket_i^\xi \text{ def.})
\end{aligned}$$

■ $P = \text{EXTEND}(P_1, v, T)$.

- If $v \notin \text{dom}(\mu)$, then, by definition, $\overline{\llbracket P \rrbracket_i^\xi}(\mu) = \emptyset$ and $\llbracket P \rrbracket_a^\xi(\mu) = \emptyset$.
- Otherwise, let $\mu' = \mu|_{\text{dom}(\mu) \setminus \{v\}}$.

$$\begin{aligned}
\mathsf{C} \left(\llbracket \text{EXTEND}(P_1, v, T) \rrbracket_a^\xi(\mu) \right) &= \mathsf{C} \left(\llbracket P_1 \rrbracket_a^\xi(\mu') \right) & (\llbracket \cdot \rrbracket_a^\xi \text{ def.}) \\
&= \mathsf{C} \left(\bigcup \mathsf{C} \left(\llbracket P_1 \rrbracket_a^\xi(\mu') \right) \right) & (s = \bigcup \mathsf{C}(s)) \\
&= \mathsf{C} \left(\bigcup \overline{\llbracket P_1 \rrbracket_i^\xi}(\mu') \right) & (\text{IH}) \\
&= \mathsf{C} \left(\bigcup \{I \mid (I, \mu') \in \llbracket P_1 \rrbracket_i^\xi\} \right) & (\bar{\cdot} \text{ def.}) \\
&= \mathsf{C} \left(\bigcup \{I \mid (I, \mu') \in \llbracket P_1 \rrbracket_i^\xi, \mu = \mu' \dot{\cup} \{v \mapsto \llbracket T \rrbracket^{\mu'}\}\} \right) & (\mu' \text{ def.}) \\
&= \overline{\mathsf{TC} \{ (I, \mu'') \mid (I, \mu') \in \llbracket P_1 \rrbracket_i^\xi, \mu'' = \mu' \dot{\cup} \{v \mapsto \llbracket T \rrbracket^{\mu'}\}\} (\mu)} & (\mathsf{TC} \text{ def.}) \\
&= \overline{\mathsf{TC} \{ (I, \mu' \dot{\cup} \{v \mapsto \llbracket T \rrbracket^{\mu'}\}) \mid (I, \mu') \in \llbracket P_1 \rrbracket_i^\xi \} (\mu)} & (\text{eliminate } \mu'') \\
&= \overline{\llbracket \text{EXTEND}(P_1, v, T) \rrbracket_i^\xi}(\mu). & (\llbracket \cdot \rrbracket_i^\xi \text{ def.})
\end{aligned}$$

■ $F = \text{EXISTS}(P)$. Since $\llbracket \text{EXISTS}(P) \rrbracket_a^\xi$ is either $\top$ or $\perp$, the following proves the claim:

$$\begin{aligned}
\llbracket \text{EXISTS}(P) \rrbracket_a^\xi &= \top \text{ iff } \llbracket P \rrbracket_a^\xi(\mu) \neq \emptyset, \text{ for some } \mu, & (\llbracket \cdot \rrbracket_a^\xi \text{ def.}) \\
&\text{iff } \bigcup \mathsf{C} \left(\llbracket P \rrbracket_a^\xi(\mu) \right) \neq \emptyset, \text{ for some } \mu, & (s = \bigcup \mathsf{C}(s))
\end{aligned}$$

- iff $\bigcup \overline{\llbracket P \rrbracket_i^\xi}(\mu) \neq \emptyset$, for some μ , (IH)
- iff $\bigcup \{I \mid (I, \mu) \in \llbracket P \rrbracket_i^\xi\} \neq \emptyset$, for some μ , ($\neg$ def.)
- iff $\llbracket P \rrbracket_i^\xi \neq \emptyset$ ($\emptyset \notin \mathbb{I}$)
- iff $\llbracket \text{EXISTS}(P) \rrbracket_i^\xi = \top$. ($\llbracket \cdot \rrbracket_i^\xi$ def.)

- $F \neq \text{EXISTS}(P)$. For all other types of filter expressions, $\llbracket F \rrbracket_i^\xi = \llbracket F \rrbracket_a^\xi$ trivially holds by induction on the structure of F , because the definitions of $\llbracket F \rrbracket_i^\xi$ and $\llbracket F \rrbracket_a^\xi$ are analogous.

B Experiments

- **Listing 6** SQL translation of the Flights Query 1 in Listing 4.

```
SELECT DISTINCT GREATEST(f1.ad,f2.ad) AS ad, LEAST(f1.aa,f2.aa) AS aa, f1.id
FROM flights f1 JOIN flights f2
ON GREATEST(f1.ad,f2.ad) < LEAST(f1.aa,f2.aa) AND f1.fap='FRA' AND f2.fap='MUC'
```

- **Listing 7** SQL translation of the Flights Query 2 in Listing 5.

```
WITH
-- name left- and right-hand-side arguments of the left join
lefty AS (
  SELECT GREATEST(f1.ad,f2.ad) AS ad, LEAST(f1.aa,f2.aa) AS aa,
         f1.id AS id1, f1.fap AS fap1, f2.fap AS fap2, ROW_NUMBER() OVER () AS rownum
  FROM flights f1, flights f2
  WHERE GREATEST(f1.ad,f2.ad) < LEAST(f1.aa,f2.aa) AND f1.fap='FRA' AND f2.fap='MUC'),
righty AS (
  SELECT aa, ad, id, fap, ROW_NUMBER() OVER () AS rownum
  FROM flights
  WHERE fap='CDG'),

-- gather interval split points (and remove duplicates)
split_points AS (SELECT ad AS t FROM lefty UNION SELECT aa AS t FROM lefty
                  UNION SELECT ad AS t FROM righty UNION SELECT aa AS t FROM righty),

-- split the left-hand side with the interval split points
left_split AS (
  -- interval: current ad to next ad (or lastaa if it's the last interval in the group)
  SELECT ad, COALESCE(LEAD(ad) OVER w, lastaa) AS aa, id1, fap1, fap2
  FROM (
    SELECT rownum, ad, aa AS lastaa, id1, fap1, fap2
    FROM lefty
    UNION ALL
    SELECT rownum, t AS ad, aa AS lastaa, id1, fap1, fap2
    FROM lefty JOIN split_points ON (t > ad AND t < aa))
  WINDOW w AS (PARTITION BY rownum ORDER BY ad NULLS FIRST)),

-- split the right-hand side with the interval split points
right_split AS (
  SELECT ad, COALESCE(LEAD(ad) OVER w, lastaa) AS aa, id, fap
  FROM (
    SELECT rownum, ad, aa AS lastaa, id, fap
    FROM righty
    UNION ALL
    SELECT rownum, t AS ad, aa AS lastaa, id, fap
    FROM righty JOIN split_points ON (t > ad AND t < aa))
  WINDOW w AS (PARTITION BY rownum ORDER BY ad NULLS FIRST))

-- now left_split and right_split refer to the same interval endpoints
SELECT DISTINCT ls.id1, rs.id, ls.ad, ls.aa
FROM left_split ls LEFT JOIN right_split rs ON (ls.ad = rs.ad AND ls.aa = rs.aa)
```


A Multi-Scale Process for Mining and Abstracting Environmental Spatio-Temporal Graphs

Assaad Zaghina ✉ 

Université de Strasbourg, CNRS, ENGEES, ICube UMR 7357, Strasbourg, France

Aurélié Leborgne ✉ 

Université de Strasbourg, CNRS, ENGEES, ICube UMR 7357, Strasbourg, France

Florence Le Ber ✉ 

Université de Strasbourg, CNRS, ENGEES, ICube UMR 7357, Strasbourg, France

Antoine Vacavant ✉ 

Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, France

Abstract

Understanding the continuous evolution of environmental systems increasingly relies on tracking large-scale, heterogeneous data across shifting timelines, commonly structured as temporal multigraphs. While such representations are highly expressive, their temporal density and structural heterogeneity make the automated analysis of long-term transitions particularly challenging. In this paper, we propose a unified process for extracting and abstracting recurrent evolutionary patterns within time-varying multigraphs. The proposed pipeline articulates four successive stages: (1) a spatio-temporal modeling stage based on RCC-8 relations to capture both topological structures and the chronological evolution of geographic entities across consecutive time steps; (2) a frequent pattern mining stage designed to discover recurring temporal sequences within multigraph representations; (3) a pattern matching stage that enables efficient, multi-scale localization of these evolutionary signatures; and (4) a hierarchical abstraction stage relying on super-nodes to produce a compact, condensed representation of the graph's historical progression. An interactive visual environment is additionally provided as a complementary exploration tool to inspect the timeline outputs of the pipeline. We report on the results obtained from historical graph analyses of CORINE Land Cover data across multiple European cities, demonstrating the process's capacity to reveal recurrent environmental behaviors and meaningful, time-dependent trajectories of change.

2012 ACM Subject Classification Information systems → Data mining; Computing methodologies → Neural networks; Information systems → Geographic information systems

Keywords and phrases Graph Deep Learning, Temporal Graphs, Pattern Mining, Temporal Analysis, Graph Abstraction, Subgraph Matching

Digital Object Identifier 10.4230/OASICS.TIME.2026.7

1 Introduction

Environmental monitoring has progressively evolved from isolated sensor measurements toward continuous spatio-temporal observation systems that capture dynamic environmental processes over time [14]. With the increasing availability of time-dependent, heterogeneous, and multidimensional data streams, the main challenge is no longer limited to data collection, but rather to understanding temporal evolutions, recurring patterns, and long-term dependencies within these complex datasets. Temporal (or spatio-temporal) graphs provide an expressive modeling framework capable of capturing temporal dynamics alongside multiple interconnected dimensions simultaneously. By representing entities as nodes and encoding temporal, spatial, and semantic relationships as edges, they offer a unified view of evolving environmental processes [30]. In particular, temporal intervals enable the modeling of the evolution of relationships over time, while the use of Region Connection Calculus (RCC-8) [20]



© Assaad Zaghina, Aurélié Leborgne, Florence Le Ber, and Antoine Vacavant;
licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 7; pp. 7:1–7:15

OpenAccess Series in Informatics



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

provides a qualitative description of spatial topology. However, despite their expressive power, temporal graphs often suffer from structural complexity that hinders both algorithmic analysis and human interpretation.

Frequent pattern mining, which aims to identify recurring subgraph structures, has emerged as a powerful approach for uncovering meaningful relational configurations in complex networks [31]. Such patterns often reveal fundamental interaction schemes and provide insights into the underlying organization of the data [11]. However, traditional pattern mining approaches face severe computational limitations, as the number of candidate subgraphs grows exponentially, making subgraph enumeration and matching NP-hard problems. These challenges are further amplified in spatio-temporal graph settings, where multiple edge types and overlapping relations must be considered [18]. Consequently, recent advances increasingly advocate the use of neural-based approaches for frequent pattern mining, as they offer scalable alternatives capable of learning expressive representations of complex graph structures, including spatio-temporal and multigraph settings [26, 27]. In particular, approaches such as SPMiner or Multi-SPMiner exploit learned projection spaces and graph representation learning to guide the discovery of frequent graph patterns and spatio-temporal graphs respectively [29].

In this paper, we propose a complete framework for the analysis of environmental temporal graphs, with a particular focus on the temporal evolution of environmental structures and relationships. The process is articulated around four successive stages that together form a coherent analytical pipeline. The contributions can be summarized as follows:

- **Modeling stage:** environmental data are structured into temporal multigraphs using temporal intervals to capture dynamic evolutions over time and RCC-8 base relations to encode spatial topology.
- **Mining stage:** Multi-SPMiner [29] is employed to extract significant spatio-temporal patterns that characterize recurring temporal behaviors within the multigraph.
- **Matching stage:** the neural representation learned during mining is reused to efficiently localize pattern occurrences and their temporal manifestations within the graphs.
- **Abstraction stage:** super-nodes are introduced to represent frequent spatio-temporal patterns, while the semantic hierarchy of RCC-8 relations is exploited to aggregate edges, yielding a compact hierarchical representation of temporal graph evolution.

Unlike previous work, which is limited to pattern discovery, this paper extends the framework by introducing a neural pattern-matching strategy that reuses the representation learned during mining to localize pattern occurrences, a multi-scale analysis enabled by the RCC-8 semantic hierarchy, and a hierarchical graph formulation that integrates the discovered patterns as super-nodes together with semantic edge aggregation to produce a compact representation of temporal graph evolution. The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 details the four stages of the proposed process, with particular emphasis on the projection space of Multi-SPMiner, the pattern growth procedure, the structural constraints imposed on the subgraph matching module, and the hierarchical abstraction mechanism. Section 4 presents experimental results obtained on Corine Land Cover data, and Section 5 concludes the paper.

2 Related Work

Spatio-temporal Modeling. Extracting knowledge from massive heterogeneous datasets requires a precise understanding of the elements characterizing phenomena, their spatial distribution, and their temporal occurrence [19]. Early efforts in spatio-temporal modeling

largely relied on the Snapshot model, which organizes data into a set of independent layers where each layer describes spatial states at a fixed time [2]. However, such classical models are generally limited to representing sudden changes and often fail to capture continuous evolution effectively.

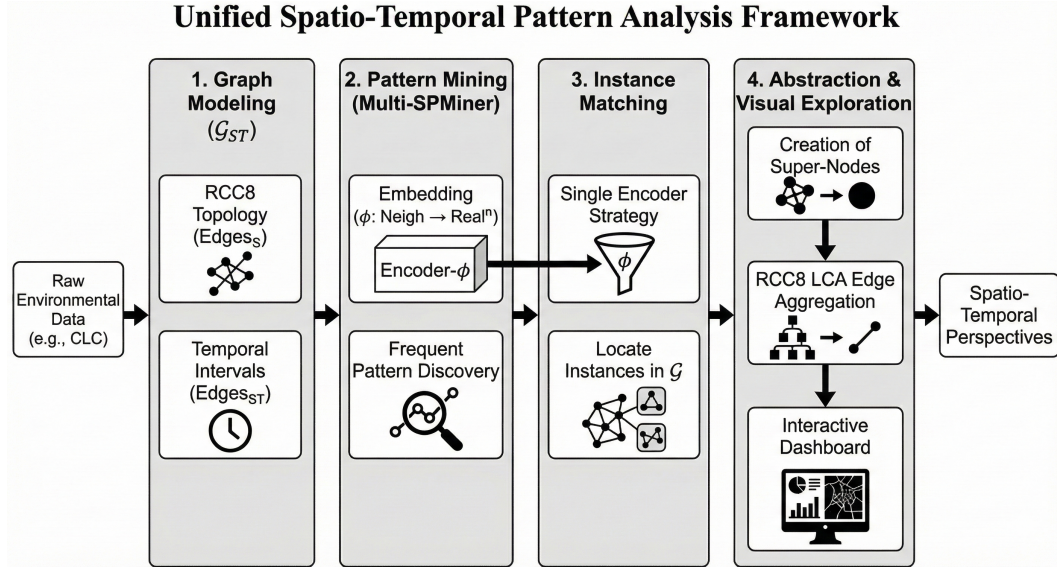
To address the complexities of identity change, graph theory has emerged as a powerful framework for modeling dynamic phenomena, where vertices represent entities and edges represent their relationships. Early graph-based research primarily focused on modeling spatial relations between entities [10]. Subsequent works introduced spatio-temporal graphs where edges link entities at consecutive time steps, thereby representing the connection between spatial footprints across temporal intervals [12]. Other variations include directed graphs that combine spatial relations with spatio-temporal transitions [24], as well as interaction graphs used to simulate landscape processes [4]. Among the existing spatio-temporal graph modeling approaches, Del Mondo et al. [5] proposed a representation framework for evolving geographic entities based on multiple types of relationships. Their model explicitly distinguishes between three categories of relations: spatial, temporal, and filiation. Such a structured representation enables the extraction of meaningful knowledge from spatio-temporal data, including recurrent patterns and interaction patterns.

Pattern Mining in Dynamic Graphs. Frequent pattern mining enables the identification of repetitive structures that reveal underlying interactions within complex systems [11]. However, this process is computationally intensive and known to be NP-hard, particularly when applied to multigraphs, which contain multiple edge types and directions [18]. Spatio-temporal graphs are a specific class of multigraphs that incorporate spatial and temporal dimensions to model dynamic phenomena. While traditional subgraph mining algorithms such as SIGRAM [13], GRAMI [7], and MuGram [9] have been adapted for multigraphs, they often suffer from scalability issues. Similarly, methods targeting dynamic or heterogeneous graphs [21] face limitations regarding genericity and processing speed at scale. To address these constraints, recent approaches in Graph Deep Learning (GDL) have emerged [26, 29], utilizing Graph Neural Networks (GNNs) to identify frequent patterns by offering reliable projections into embedding spaces, from which subgraph pattern frequency can be effectively estimated.

Hierarchical Abstraction of Large Graphs. Processing and representing large-scale temporal graphs raises a fundamental scalability issue known as the 'hairball' phenomenon, where excessive edge density renders the underlying structure illegible [8]. Abstraction techniques are therefore essential to reduce structural complexity while preserving topological information [25]. A standard approach to graph abstraction is the use of compound graphs, where subgraphs are collapsed into single derived vertices, commonly referred to as *super-nodes* [1]. In the context of pattern mining, this strategy enables repetitive structural patterns to be replaced by high-level aggregates, significantly reducing the number of explicit nodes [6].

3 A Multi-Stage Process for Spatio-temporal Graph Analysis

This section presents the proposed process for the analysis of environmental temporal graphs. The pipeline is structured into four successive stages: (1) graph modeling, (2) frequent pattern mining, (3) pattern matching, and (4) hierarchical abstraction, as illustrated in Figure 1. A lightweight visual environment, described at the end of this section, is additionally provided to inspect the outputs of the process.



■ **Figure 1** Overview of the proposed process for analyzing environmental spatio-temporal data.

3.1 Spatio-temporal Graph Modeling

Following the formal model introduced in [5] and its adaptation in [16], we define a spatio-temporal graph as $G = (V, E)$, composed of a spatial subgraph G_S and a spatio-temporal subgraph G_{ST} . Let $O(t)$ denote the set of geographic objects at time t . The vertex set is defined as $V = \bigcup_t O(t)$, and the edge set as $E = E_S \cup E_{ST}$, where E_S encodes spatial relations and E_{ST} encodes temporal relations.

Spatial Relation Modeling (G_S). The spatial subgraph G_S models topological relations between entities coexisting at the same time instant t . Each edge $(u, v) \in E_S$ is labeled with one of the eight exhaustive and mutually exclusive RCC-8 base relations [20]. As proposed in [15], the RCC-8 relation between two objects u and v can be derived from a dimensional intersection matrix evaluating intersections between interiors ($^\circ$), boundaries (∂), and exteriors ($^-$):

$$M(u, v) = \begin{pmatrix} \dim(u^\circ \cap v^\circ) & \dim(u^\circ \cap \partial v) & \dim(u^\circ \cap v^-) \\ \dim(\partial u \cap v^\circ) & \dim(\partial u \cap \partial v) & \dim(\partial u \cap v^-) \\ \dim(u^- \cap v^\circ) & \dim(u^- \cap \partial v) & \dim(u^- \cap v^-) \end{pmatrix}. \quad (1)$$

To limit graph density, the *Disconnected* (DC) relation is treated implicitly; only connectivity relations (EC , PO , TPP , $NTPP$, EQ) generate explicit edges. To handle spatial discontinuities common in environmental polygon data, we additionally model proximity relations following [17]:

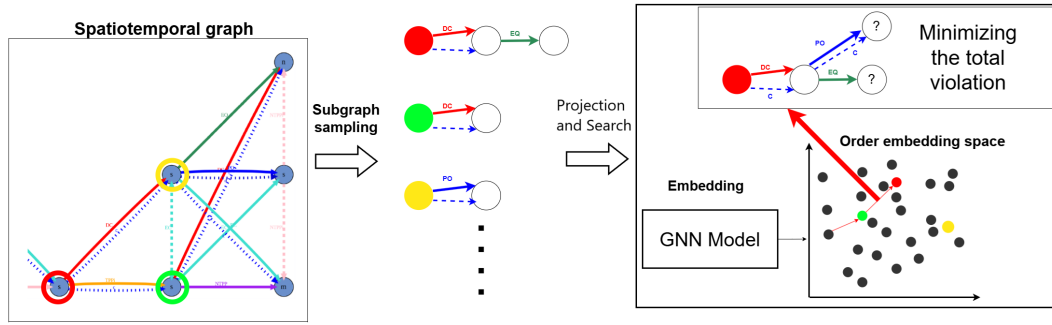
- **Absolute proximity** (p_a), defined using a fixed distance threshold b between object boundaries.
- **Relative proximity** (p_r), defined using a distance threshold proportional to the area of the neighboring object, capturing zones of spatial influence.

Spatio-temporal Relations and Indexing. The spatio-temporal subgraph G_{ST} captures temporal evolution by linking objects between consecutive time steps t and $t+1$. To account for digitization uncertainty in multi-annual datasets, we adopt the thick-boundary model [3]

as suggested by [17]. A temporal edge is created when the buffer zones of two geometries intersect, ensuring topological continuity under minor spatial shifts, using Fuzzy RCC-8 to encode temporal relations. All spatial relation computations are accelerated using an R-tree index, which reduces the number of candidate polygon comparisons through spatial indexing.

3.2 Frequent Pattern Mining with Multi-SPMiner

To identify recurrent structural patterns within the resulting multigraph, we rely on Multi-SPMiner, an extension of the SPMIner architecture [26] adapted for labeled, directed multigraphs [29]. Unlike traditional isomorphism-counting methods, Multi-SPMiner leverages graph deep learning to identify patterns anchored at nodes that exceed a support threshold s . This stage is itself divided into two phases: embedding and pattern search (see Figure 2).



■ **Figure 2** Multi-SPMiner is composed of two main phases: (1) encoding, which focuses on learning graph embeddings, and (2) decoding, which is dedicated to the search and extraction of frequent patterns [29].

3.2.1 Order-Preserving Embedding

The graph G is first decomposed into a set of neighborhoods $\mathcal{N} = \{N_v \mid v \in V\}$, where each N_v represents the k -hop subgraph anchored at node v . We set k to be larger than the maximum expected diameter of a pattern. A Multigraph Convolutional Network (MGCN) is then used to project these neighborhoods into an ordered embedding space $\mathbb{R}^n$. The embedding function $\phi : \mathcal{G} \mapsto \mathbb{R}^n$ is trained to preserve the subgraph isomorphism relation. Formally, for two graphs A and B , we enforce the order constraint

$$A \subseteq B \iff \phi(A) \leq \phi(B), \quad (2)$$

where $\leq$ denotes a coordinate-wise inequality. In the 2D plane, this implies that the embedding of a subgraph A must reside to the bottom-left of the embedding of its supergraph B . The MGCN is optimized by minimizing a max-margin loss function over positive pairs (subgraph relationships) and negative pairs:

$$\mathcal{L} = \sum_{(A,B) \in P} E(A,B) + \sum_{(A',B') \in N} \max(0, \alpha - E(A',B')), \quad (3)$$

where α is a margin hyperparameter and E is the order violation penalty defined as

$$E(A,B) = \|\max(0, \phi(A) - \phi(B))\|^2. \quad (4)$$

3.2.2 Pattern Search

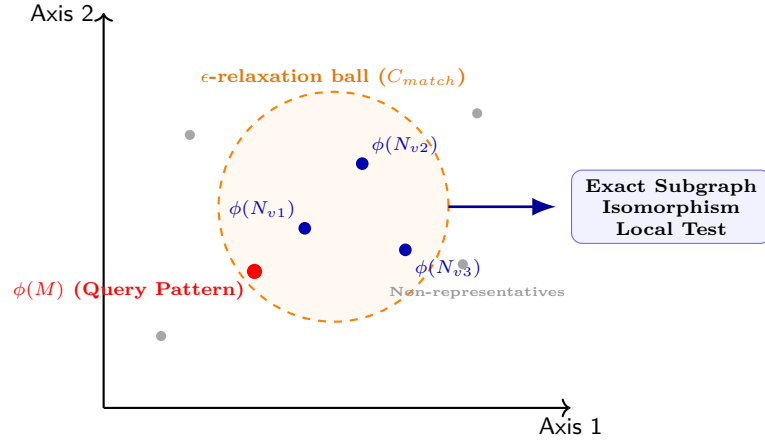
Once the neighborhoods are embedded, a heuristic search is performed to identify frequent patterns. The search is iterative, starting from a randomly selected single-node graph. At each step, the candidate pattern M is grown by adding adjacent nodes and edges found in the underlying multigraph. The growth strategy is guided by the embeddings: we seek to construct a pattern M that is a subgraph of as many neighborhoods $N \in \mathcal{N}$ as possible. This is achieved by minimizing the total order violation margin $m(M)$ across the dataset:

$$m(M) = \sum_{N \in \mathcal{N}} \|\max(0, \phi(M) - \phi(N))\|^2. \quad (5)$$

This optimization ensures that the discovered pattern M effectively fits into the embeddings of a large subset of the graph's neighborhoods.

3.3 Pattern Matching via Same-Encoder Strategy

Once an abstract pattern definition M has been identified, the next step consists in locating all its specific instances $\{I_1, I_2, \dots, I_k\}$ within the global graph G . To this end, we adopt a *Same-Encoder Strategy*. This approach leverages the property that the MGCN encoder ϕ was explicitly trained to preserve subgraph isomorphism relationships in the embedding space. Accordingly, no separate global matching algorithm is required before candidate filtering: the pre-trained encoder ϕ is directly reused as a high-efficiency filter (see Figure 3).



■ **Figure 3** Subgraph matching principle. The model projects local neighborhoods into a latent space using the function ϕ learned during the mining stage, translating topological inclusion into a geometric ordering constraint. By applying the same encoder to the query pattern, occurrences are identified directly through their relative positions in the embedding space. The circle denotes the neighborhood defined by an epsilon margin, used to apply an isomorphism test for more precise pattern matching.

For a target pattern $M \in \mathcal{M}$ treated as a query, a node $v \in V$ is considered a candidate anchor for an instance if the order constraint is satisfied in the embedding space, i.e., $\phi(M) \leq \phi(N_v)$. This condition implies that M is likely a subgraph of the neighborhood anchored at v . This strategy drastically prunes the search space. The candidate retrieval stage effectively amounts to an approximate range query in the embedding space, followed by a local isomorphism verification to confirm the exact RCC-8 and temporal edge configurations.

The encoder ϕ enforces a partial order on the vector space, as expressed in Equation (2). Consequently, for a node v to be a candidate anchor for pattern M , its neighborhood embedding $\phi(N_v)$ must structurally “contain” the pattern embedding $\phi(M)$. We define the *order violation* term $V(M, N_v)$ as the magnitude of the violation of this inequality:

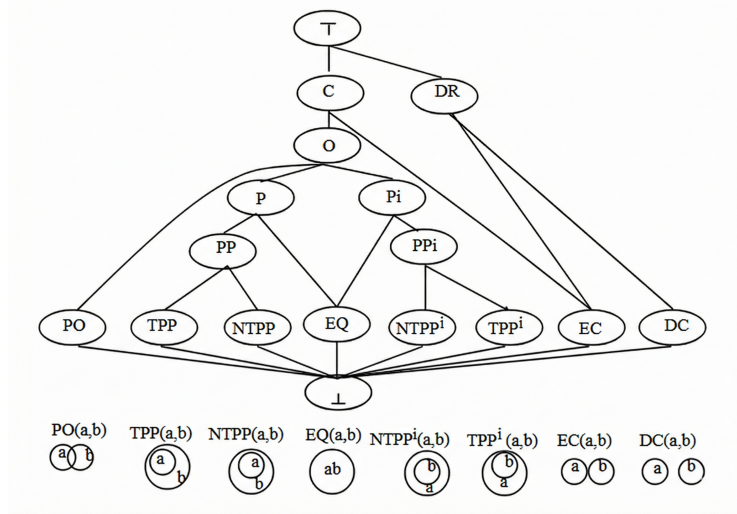
$$V(M, N_v) = \|\max(0, \phi(M) - \phi(N_v))\|^2. \quad (6)$$

Ideally, for a perfect matching, $V(M, N_v) = 0$. However, given the approximate nature of graph neural networks and the continuous nature of the latent space, we introduce a relaxation threshold ϵ . The set of candidate anchor nodes C_{match} for pattern M is defined as

$$C_{match} = \{v \in V \mid V(M, N_v) \leq \epsilon\}. \quad (7)$$

For each candidate anchor $v \in C_{match}$, the corresponding k -hop neighborhood N_v is extracted from the original graph, and an exact subgraph isomorphism test is applied between M and N_v . By restricting exact matching to neighborhoods selected through proximity in the embedding space, the combinatorial search space is significantly reduced while structural correctness is preserved.

3.4 Hierarchical Abstraction via Partitioned LCA



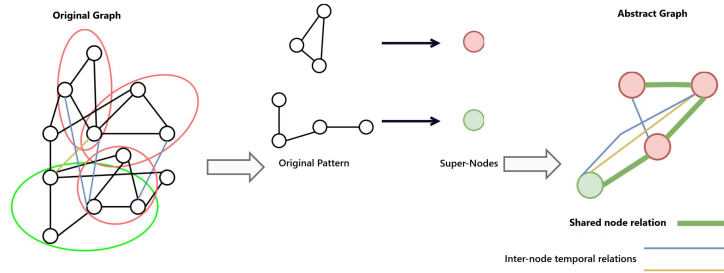
■ **Figure 4** Hierarchical representation of RCC relations [20]. The RCC-8 relations form the lower level of the hierarchy.

The outputs produced by the mining and matching stages are numerous and still interleaved with the original graph structure. To produce a compact representation of the graph that remains faithful to its semantics, we aggregate detected pattern instances into super-nodes. However, collapsing all interactions between two super-nodes S_i and S_j into a single averaged or majority relation hides essential semantic structure. Environmental spatio-temporal graphs frequently exhibit *mixed* RCC-8 relations between the same regions, reflecting genuinely complex dynamics that should not be oversimplified into a single generic link. To faithfully capture this variability, we introduce a flexible multi-scale abstraction mechanism driven by the hierarchical structure of the RCC relations [20] (see Figure 4).

Partitioned LCA with User-Controlled Granularity. The RCC hierarchy $\mathcal{H}$ forms a fixed-depth lattice of four conceptual levels (from the root relation T to the atomic leaves). A discrete granularity parameter $\lambda \in \{0, 1, 2, 3, 4\}$ specifies the desired level of abstraction. Increasing λ by one moves the abstraction one level upward in the RCC hierarchy:

- $\lambda = 0$: maximum precision, where relations remain at the atomic level;
- $\lambda = 1, 2, 3$: intermediate abstraction levels, each merging relations according to the corresponding level of the RCC hierarchy;
- $\lambda = 4$: maximum abstraction, where all relations collapse to the root relation T .

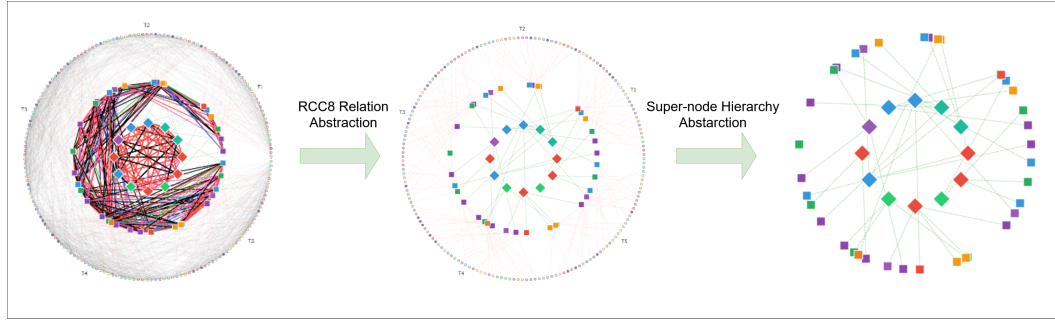
Let Ω_{ij} denote the set of RCC relations observed between the nodes composing super-nodes S_i and S_j . Instead of merging all these relations into a single abstract label, we group them according to their proximity in the RCC-8 hierarchy. Two relations are assigned to the same group if their lowest common ancestor in the hierarchy is sufficiently specific with respect to the chosen granularity level λ . Each group then produces a single abstract edge labeled by this common ancestor. This grouping strategy ensures that semantically similar relations are merged while distinct spatial relations are preserved. As a result, the abstraction between two super-nodes may consist of multiple edges, each representing a different type of spatial interaction. An illustrative example is presented in Figure 5.



■ **Figure 5** Following the pattern search and matching stages on the original graph, identified pattern instances are aggregated into super-nodes. At this higher abstraction level, these super-nodes are retained while independent nodes are filtered out, preserving only the most critical spatio-temporal dynamics.

3.5 A Complementary Visual Environment for Result Exploration

The four-stage process described above produces a hierarchical, compact, and semantically-aware representation of the input temporal graph. To facilitate the inspection of these outputs, we additionally provide a lightweight visual environment. This environment is not a contribution of the process itself; it is a companion tool whose role is to let the analyst browse mining and matching results, examine super-nodes, and compare configurations obtained on different geographic areas. Concretely, the environment offers a coordinated multi-view layout combining an abstract graph representation of the hierarchical output with a geospatial anchor view, so that patterns produced by the pipeline can be related to their real-world geographic counterparts. Selecting a super-node highlights its corresponding geographic footprint, while selecting a geographic region reveals the associated patterns, and semantic and temporal filters are used to reduce visual load when exploring dense results (see Figure 6). While the proposed framework enables the discovery of a variety of spatio-temporal patterns, a comprehensive analysis of these patterns and their relevance from a domain perspective is outside the scope of this work. Such an analysis is presented in [28].



■ **Figure 6** Example of hierarchical abstraction of a spatio-temporal graph. Left: dense original graph. Middle: aggregation of RCC8 relations. Right: super-node representation based on frequent patterns.

4 Experiments and Results

To evaluate the proposed process, we conducted an extensive case study on real-world temporal data. The experiments aim to assess the ability of the pipeline to (i) extract meaningful temporal patterns, (ii) efficiently localize their occurrences with high accuracy, and (iii) produce a compact yet interpretable hierarchical abstraction of large graphs through super-node representations. We additionally report on the implementation details and execution times that characterize the practical behavior of the pipeline.

Data Description. The CORINE Land Cover (CLC)¹ dataset is a large-scale geospatial dataset produced at the European Union level from satellite imagery. The dataset follows a hierarchical nomenclature with three levels of semantic granularity, covering categories such as artificial surfaces, agricultural areas, forests, wetlands, and water bodies. In our experiments, we rely on the second level of the classification, resulting in 15 distinct land-use types. The minimum mapping unit is 25 hectares. The dataset is available as a long-term time series spanning five temporal snapshots: 1990, 2000, 2006, 2012, and 2018. Each land polygon is represented as a node labeled according to its land-use class.

4.1 Impact of the Encoder Architecture on the Mining Stage

To analyze the robustness of the mining stage, we evaluate the impact of different GNN-based encoders on the identification of frequent spatio-temporal patterns. The objective of this experiment is twofold: (i) to assess the stability of the embedding-based pattern representation across architectures, and (ii) to evaluate the generalization capability of the learned encoders across synthetic and real-world spatio-temporal graphs. We define motif detection accuracy (equivalent to precision in this context) as the percentage of discovered motifs that are frequent motifs.

Two complementary training scenarios are considered. In the first setting, models are trained exclusively on synthetically generated spatio-temporal graphs containing patterns of predefined sizes, produced with a random ST-graph generator [16]. These synthetic graphs are designed to closely resemble real-world conditions by incorporating five temporal snapshots and high-frequency pattern insertions. The trained models are then evaluated on

¹ <https://land.copernicus.eu/en/products/corine-land-cover>

real spatio-temporal graphs generated from the CLC dataset, constructed from 20 randomly selected European cities. This scenario evaluates the ability of the model to generalize across domains after a single training phase. In the second setting, both training and evaluation graphs are derived from distinct geographic regions within the CLC dataset. For each scenario, 200 graphs are used for training and an additional set of 20 graphs is reserved for testing. The evaluation focuses on patterns of sizes 6, 8, and 10, which represent a practical compromise between expressiveness and computational complexity.

■ **Table 1** Pattern identification accuracy on the CLC dataset for different GNN-based encoders, evaluated on patterns of sizes 6, 8, and 10, under synthetic and real-data training scenarios. Bold values denote the best result per row.

Target pattern size	Training Data	MGCNN (Baseline)	GAT	R-GCN	GGCN	HAN	MGAT	MMAGAT
6	Synthetic	0.67 ± 0.01	0.69 ± 0.02	0.70 ± 0.02	0.71 ± 0.02	0.72 ± 0.02	0.72 ± 0.01	0.73 ± 0.02
	CLC	0.68 ± 0.01	0.69 ± 0.02	0.71 ± 0.01	0.72 ± 0.02	0.73 ± 0.02	0.73 ± 0.02	0.74 ± 0.01
8	Synthetic	0.66 ± 0.03	0.70 ± 0.02	0.69 ± 0.03	0.69 ± 0.02	0.70 ± 0.02	0.68 ± 0.03	0.71 ± 0.02
	CLC	0.66 ± 0.02	0.70 ± 0.03	0.71 ± 0.01	0.71 ± 0.02	0.72 ± 0.02	0.73 ± 0.02	0.73 ± 0.03
10	Synthetic	0.65 ± 0.01	0.68 ± 0.03	0.70 ± 0.03	0.68 ± 0.02	0.68 ± 0.03	0.68 ± 0.03	0.69 ± 0.02
	CLC	0.64 ± 0.03	0.67 ± 0.02	0.68 ± 0.03	0.68 ± 0.02	0.69 ± 0.03	0.69 ± 0.03	0.69 ± 0.01

Table 1 reports the pattern identification accuracies obtained for the seven encoders considered: the MGCN baseline, Graph Attention Network (GAT), Relational Graph Convolutional Network (R-GCN) [23], Gated GCN (GGCN), Heterogeneous Attention Network (HAN), Multi-view GAT (MGAT), and Multi-Mask Aggregators for GAT (MMAGAT) [22]. Three complementary observations can be drawn from these results. First, all GNN encoders surpass the MGCN baseline by a margin ranging from 2 to 6 percentage points across pattern sizes, confirming that enriched relational representations consistently benefit pattern discovery within the proposed mining stage. Second, the gap between models trained on synthetic data and those trained directly on real CLC graphs remains marginal (less than 1% on average), which suggests that the learned order embeddings capture transferable structural and temporal regularities that are largely independent of the data source. This is a practically valuable property since the cost of generating ground truth on synthetic graphs is significantly lower than on real graphs. Third, attention-based architectures (HAN, MGAT, MMAGAT) exhibit a consistent advantage, particularly for smaller patterns (size 6), with MMAGAT achieving the best overall performance. This behavior can be attributed to their ability to selectively emphasize salient spatial and temporal relations, which is especially beneficial in multigraph settings involving heterogeneous edge semantics such as RCC-8 and temporal links. The relative gap between encoders narrows as pattern size increases, reflecting the intrinsic difficulty of mining larger structures that span multiple temporal snapshots.

■ **Table 2** Average mining-stage execution time per CLC graph (seconds) and number of trainable parameters for each encoder. Times include both pattern search and per-pattern decoding.

Metric	MGCNN	GAT	R-GCN	GGCN	HAN	MGAT	MMAGAT
Avg. mining time (s)	19.4	27.1	25.8	29.2	28.5	27.6	28.7
Trainable params (M)	0.18	0.31	0.42	0.54	0.49	0.38	0.61

Table 2 reports the average mining-stage execution time per CLC graph, alongside the number of trainable parameters for each encoder. All experiments were conducted on a machine equipped with an NVIDIA GeForce RTX 4080 SUPER GPU, 64 GB of RAM, and an AMD Ryzen 7 9800X3D processor. All neural encoders complete the mining stage in under 30 seconds per graph, which contrasts sharply with the ground-truth MuGram

algorithm [9] that requires on average 14 minutes (840 seconds) per CLC graph of comparable size. The MGCN baseline is the lightest both in execution time and in parameter count, while MMAGAT, despite reaching the highest accuracy, is the most parameter-intensive encoder due to its per-neighbor relational masks. R-GCN offers a favorable trade-off between accuracy and footprint and may be preferred in resource-constrained settings.

4.2 Effectiveness of Pattern Matching

The effectiveness of the matching stage relies on the same neural encoder used during the mining stage. The objective of this experiment is twofold: (i) to assess the precision and recall of embedding-based pattern localization using the Same-Encoder strategy alone, and (ii) to quantify the benefits of integrating an exact subgraph isomorphism verification step within the local neighborhoods retrieved in the embedding space. Two matching configurations are considered:

- **Same-Encoder (Neural)**: pattern occurrences are identified solely on the basis of the order-embedding constraint in the latent space, controlled by the relaxation threshold ϵ .
- **Same-Encoder + Subgraph Isomorphism (Neural + Iso.)**: candidate occurrences retrieved in the embedding space are further validated using an exact subgraph isomorphism test within the corresponding local graph neighborhoods.

To rigorously characterize the matching stage, we report both precision and recall against the exhaustive ground truth produced by MuGram. Recall measures the fraction of true pattern occurrences that are recovered, whereas precision measures the fraction of returned occurrences that correspond to genuine matches. This distinction is essential because the embedding-based retrieval, governed by the relaxation threshold ϵ , is intentionally tuned to avoid missing true positives, which inflates the candidate set and inevitably introduces false positives. The threshold ϵ is estimated at the end of the training phase by analyzing the average distance between different embedding representations of identical patterns and averaging these distances across the training set. More specifically, ϵ is defined as a vector having the same dimensionality as the embedding space, where each component captures the average admissible variation observed for the corresponding embedding dimension.

■ **Table 3** Pattern matching precision and recall (%) on the CLC dataset for the Same-Encoder (Neural) configuration. Recall is by design near-saturated, while precision is limited by false positives introduced by the approximate nature of continuous embeddings.

Metric	MGCNN (Baseline)	GAT	R-GCN	MGAT	MMAGAT
Recall	95.62 $\pm$ 0.39	95.78 $\pm$ 0.36	95.81 $\pm$ 0.34	95.93 $\pm$ 0.28	96.04 $\pm$ 0.26
Precision	86.74 $\pm$ 0.62	87.41 $\pm$ 0.58	87.86 $\pm$ 0.55	88.92 $\pm$ 0.49	89.37 $\pm$ 0.46

■ **Table 4** Average matching execution time per CLC graph for each encoder.

Matching Strategy	MGCNN (Baseline)	GAT	R-GCN	MGAT	MMAGAT
Neural	1m18s	1m21s	1m19s	1m23s	1m24s
Neural + Iso.	3m33s	3m35s	3m36s	3m41s	3m43s

Tables 3 and 4 jointly summarize the matching performance of the pipeline, from which three observations can be drawn. First, recall remains consistently high and stable around 95.9% across all encoders, and is essentially unaffected by the verification step since the latter only filters previously retrieved candidates. The relaxation threshold ϵ is intentionally

calibrated to exhaustively cover the candidate set, ensuring that the embedding-based retrieval rarely misses a true occurrence. The small inter-encoder differences reflect the fidelity with which each encoder preserves subgraph containment in the latent space: more expressive encoders such as MGAT and MMAGAT produce embeddings that leave fewer true matches outside the ϵ -ball. Precision differs markedly between the two configurations. The Same-Encoder strategy alone yields precision values ranging from 86.7% (MGCN baseline) to 89.4% (MMAGAT). Once the exact subgraph isomorphism verification is applied within the retrieved neighborhoods, precision is perfect for all encoders, which justifies the omission of a dedicated table for the post-verification values, as the step operates as a high-precision verification step that removes false positives almost entirely without affecting true positives. The additional verification cost remains modest. Pure neural matching is completed in approximately 1m20s per CLC graph, whereas the Neural + Iso. configuration takes around 3m40s. This overhead is contained because exact matching is restricted to the small set of candidates retrieved in the embedding space rather than applied globally over the full graph. By comparison, a purely algorithmic subgraph isomorphism approach applied directly to the entire graph requires up to 22 minutes per graph. Overall, these results validate the design of the matching stage: the neural retrieval component ensures high recall and fast candidate generation, while the isomorphism step delivers high precision. Decoupling the two roles allows each component to be tuned independently and yields a matching stage whose final precision-recall profile is significantly more favorable than what either component could achieve in isolation.

4.3 Hierarchical Abstraction

The abstraction stage transforms the rich but dense output of the matching stage into a compact, semantically-aware hierarchical representation of the graph. Frequent patterns capture recurrent structural and semantic configurations that characterize the dominant spatio-temporal behaviors observed in the data, while rare or noisy interactions are naturally filtered out. By relying on frequent patterns as building blocks, the abstraction stage produces a synthetic summary of the underlying graph structure: instead of manipulating thousands of individual nodes and heterogeneous relations, the graph is summarized through a reduced set of pattern instances and their interactions. We now evaluate this abstraction quantitatively, by measuring the compression achieved at successive hierarchical levels, and qualitatively, by inspecting the structural dependencies that emerge between super-nodes.

We apply the abstraction to spatio-temporal graphs of approximately 2000 nodes extracted from different European regions, and measure the structural compression achieved at three successive levels of the hierarchy: (L0) the original raw graph; (L1) the first abstraction layer obtained by replacing detected pattern instances with first-order super-nodes; (L2) the second abstraction layer obtained by re-applying the mining and matching stages on the L1 graph and aggregating the resulting pattern co-occurrences into second-order super-nodes.

Table 5 shows that the first abstraction layer reduces the size of the graph by approximately 71.5% in nodes and 74.8% in edges on average, while the second layer drives this compression to an average reduction of 91.4% in nodes and 93.6% in edges with respect to the original graph. Although the absolute graph sizes are comparable across the five cities considered, the compression rates exhibit meaningful city-level variability: more structurally homogeneous cities such as Helsinki and Madrid yield stronger compression (around 73% at L1 and 92% at L2), reflecting the presence of more redundant structural patterns that the abstraction can collapse, whereas more heterogeneous land-use mosaics such as Berlin and Vienna yield slightly weaker compression (around 70% at L1 and 90% at L2). This pattern suggests that

■ **Table 5** Quantitative evaluation of the hierarchical abstraction on five representative European cities. All graphs use the second-level CLC nomenclature (15 land-use classes). Mining and matching are recursively applied at each level using the same MMAGAT encoder, with an estimated frequency threshold of approximately 25 occurrences per pattern. $|V|$ and $|E|$ denote the number of nodes and edges at each level. Reduction percentages are computed with respect to the original graph (L0).

City	L0 (Raw)		L1 (pattern super-nodes)				L2 (Co-occurrence super-nodes)			
	$ V $	$ E $	$ V $	$ E $	$\Delta V $	$\Delta E $	$ V $	$ E $	$\Delta V $	$\Delta E $
Paris	2148	8316	612	2104	−71.5%	−74.7%	187	542	−91.3%	−93.5%
Berlin	2103	8194	638	2189	−69.7%	−73.3%	204	591	−90.3%	−92.8%
Madrid	2167	8358	581	2002	−73.2%	−76.0%	168	487	−92.2%	−94.2%
Vienna	2089	8087	624	2126	−70.1%	−73.7%	195	562	−90.7%	−93.0%
Helsinki	2118	8205	568	1949	−73.2%	−76.2%	162	471	−92.4%	−94.3%
Mean	2125	8232	605	2074	−71.5%	−74.8%	183	531	−91.4%	−93.6%

the abstraction mechanism captures regularities that are intrinsic to the underlying spatio-temporal process while remaining sensitive to the specific structural diversity of each region. The relative reduction in edges is consistently higher than the reduction in nodes across all cities because the partitioned LCA mechanism merges semantically similar RCC-8 relations between the same pair of super-nodes into a single abstract edge, while still preserving genuinely distinct relations. By contrast, applying a higher granularity λ on the original graph aggressively merges RCC-8 relations along their hierarchy, which mechanically increases the number of detectable structural patterns but yields patterns of limited significance, since they collapse semantically distinct configurations into a single coarse label.

4.4 Discussion

Taken together, the experiments characterize the proposed process along three complementary axes. The mining stage identifies frequent patterns with an accuracy of approximately 72% on real CLC data using the best encoder (MMAGAT), while running roughly an order of magnitude faster than the exhaustive MuGram baseline. The matching stage achieves a stable recall of approximately 95.9% and a perfect precision once the isomorphism verification step is added, with an overall execution time that remains within a few minutes per graph. Finally, the abstraction stage compresses the graph by a factor of three to ten across two hierarchical levels while preserving multi-relational structure through a tunable granularity parameter. The clear separation of roles between stages, the use of a single encoder for both mining and matching, and the recursive applicability of the pipeline to its own outputs together yield a process that is interpretable, scalable, and faithful to the multi-relational nature of environmental spatio-temporal graphs.

5 Conclusion and Perspectives

This paper has presented a complete process for mining, matching, and abstracting frequent spatio-temporal patterns in large environmental graphs. The pipeline addresses both the scalability and the interpretability challenges inherent to real-world spatio-temporal data. Experiments on the Corine Land Cover dataset have shown that the learned embeddings robustly capture recurrent spatio-temporal structures and generalize across synthetic and real data. The Same-Encoder matching strategy enables efficient pattern localization, while the integration of a constrained subgraph isomorphism verification step improves accuracy

with a controlled computational overhead, offering a favorable trade-off compared to purely algorithmic approaches. Finally, the hierarchical abstraction based on frequent patterns and partitioned aggregation produces a compact and interpretable summary of complex graphs, and enables the identification of higher-order pattern co-occurrences.

Future work will focus on extending the process to dynamic pattern evolution analysis and to other classes of spatio-temporal graphs, as well as on refining the companion visual environment to better support the exploratory analysis of pipeline outputs.

References

- 1 James Abello, Frank Van Ham, and Neeraj Krishnan. Ask-graphview: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006. doi:10.1109/TVCG.2006.120.
- 2 J. Chen, H. Wu, S. Li, A. Liao, C. He, and S. Peng. Temporal logic and operation relations based knowledge representation for land cover change web services. *ISPRS Journal of Photogrammetry and Remote Sensing*, 83:140–150, 2013.
- 3 Eliseo Clementini and P. Di Felice. Approximate topological relations. *International Journal of Approximate Reasoning*, 16(2):173–204, 1997. doi:10.1016/S0888-613X(96)00127-2.
- 4 P. Degenne and D. Lo Seen. Ocelet: Simulating processes of landscape changes using interaction graphs. *SoftwareX*, 5:89–95, 2016. doi:10.1016/J.SOFTX.2016.05.002.
- 5 G. Del Mondo, J. G. Stell, C. Claramunt, and R. Thibaud. A graph model for spatio-temporal evolution. *Journal of Universal Computer Science*, 16:1452–1477, 2010. doi:10.3217/JUCS-016-11-1452.
- 6 Cody Dunne and Ben Shneiderman. Motif simplification: improving network visualization readability with pattern replacement. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1923–1932, 2013.
- 7 M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis. Grami: Frequent subgraph and pattern mining in a single large graph. *Proceedings of the VLDB Endowment*, 7(7):517–528, 2014. doi:10.14778/2732286.2732289.
- 8 Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000. doi:10.1109/2945.841119.
- 9 V. Ingalalli, D. Ienco, and P. Poncelet. Mining frequent subgraphs in multigraphs. *Information Sciences*, 451:50–66, 2018.
- 10 Bin Jiang and Christophe Claramunt. Topological analysis of urban street networks. *Environment and Planning B: Planning and design*, 31(1):151–162, 2004.
- 11 C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(1):75–105, 2013. doi:10.1017/S0269888912000331.
- 12 L. Khiali, D. Ienco, and M. Teisseire. Object-oriented satellite image time series analysis using a graph-based representation. *Ecological Informatics*, 43:52–64, 2018. doi:10.1016/J.ECOINF.2017.11.003.
- 13 M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph. *Data mining and knowledge discovery*, 11(3):243–271, 2005. doi:10.1007/S10618-005-0003-9.
- 14 Suprava Ranjan Laha, Binod Kumar Pattanayak, and Saumendra Pattnaik. Advancement of environmental monitoring system using IoT and sensor: A comprehensive analysis. *AIMS Environmental Science*, 9(6):771–800, 2022.
- 15 F. Le Ber and A. Napoli. Design and comparison of lattices of topological relations for spatial representation and reasoning. *Journal of Experimental & Theoretical Artificial Intelligence*, 15(3):331–371, 2003. doi:10.1080/0952813031000119700.
- 16 Aurélie Leborgne, Marija Kirandjiska, and Florence Le Ber. Random generation of a locally consistent spatio-temporal graph. In *International Conference on Conceptual Structures*, pages 155–169. Springer, 2021. doi:10.1007/978-3-030-86982-3_12.

- 17 Aurélie Leborgne, Adrien Meyer, Henri Giraud, Florence Le Ber, and Stella Marc-Zwecker. Un graphe spatio-temporel pour modéliser l'évolution de parcelles agricoles. In *SAGEO 2019*, Clermont-Ferrand, France, 2019.
- 18 Pei Li, Jianyong Yu, Jianxun Liu, Dong Zhou, and Buqing Cao. Generating weighted social networks using multigraph. *Physica A: Statistical Mechanics and its Applications*, 539:122894, 2020.
- 19 Donna J. Peuquet and Elizabeth Wentz. An approach for time-based analysis of spatiotemporal data. In *Advances in GIS research. Proc. 6th symposium*, pages 489–504, 1994.
- 20 David A. Randell, Zhan Cui, and Anthony G. 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)*, pages 165–176, 1992.
- 21 Abhik Ray, Lawrence B Holder, and Albert Bifet. Efficient frequent subgraph mining on large streaming graphs. *Intelligent Data Analysis*, 23(1):103–132, 2019. doi:10.3233/IDA-173705.
- 22 Ahmet Sarigün and Ahmet Sureyya Rifaioglu. Multi-mask aggregators for graph neural networks. In *The First Learning on Graphs Conference*, 2022. URL: <https://openreview.net/forum?id=hZ3b8CskgC>.
- 23 Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks, 2017. arXiv:1703.06103.
- 24 M. Sriti, R. Thibaud, and C. Claramunt. A fuzzy identity-based temporal gis for the analysis of geomorphometry changes. In *Journal on Data Semantics III*, pages 81–99, 2005.
- 25 Tatiana Von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke J. Van Wijk, Jean-Daniel Fekete, and Dieter W. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011. doi:10.1111/J.1467-8659.2011.01898.X.
- 26 R. Ying, A. Wang, J. You, and J. Leskovec. Frequent subgraph mining by walking in order embedding space. *The International Conference on Machine Learning (ICML)*, 2020.
- 27 Assaad Zeghina, Aurélie Leborgne, Florence Le Ber, and Antoine Vacavant. Deep qminer: Towards a generalized deep q-learning approach for graph pattern mining. In Luc Brun, Vincenzo Carletti, Sébastien Bogleux, and Benoît Gaüzère, editors, *Graph-Based Representations in Pattern Recognition*, pages 134–144, Cham, 2025. Springer Nature Switzerland.
- 28 Assaad Zeghina, Aurélie Leborgne, Florence Le Ber, and Antoine Vacavant. Neural pattern mining in spatiotemporal multigraphs: A pan-european land-use study. In *International Conference on Pattern Recognition Workshops GREEN-PR*, 2026.
- 29 Assaad Zeghina, Aurélie Leborgne, Florence Le Ber, and Antoine Vacavant. Multi-SPMiner: A Deep Learning Framework for Multi-Graph Frequent Pattern Mining with Application to Spatiotemporal Graphs. *Procedia Computer Science*, 225:1094–1103, 2023. doi:10.1016/J.PROCS.2023.10.097.
- 30 Assaad Zeghina, Aurélie Leborgne, Florence Le Ber, and Antoine Vacavant. Deep learning on spatiotemporal graphs: A systematic review, methodological landscape, and research opportunities. *Neurocomputing*, 594:127861, 2024. doi:10.1016/j.neucom.2024.127861.
- 31 Z. Zhang, P. Cui, and W. Zhu. Deep learning on graphs: A survey. *IEEE Trans. on Knowledge and Data Engineering*, 34(1):249–270, 2020.

The Satisfiability Problem of Temporal-Spatial Logics over Quasi-Temporal Graphs

Eric Alsmann 

Theoretical Computer Science / Formal Methods, University of Kassel, Germany

Martin Lange 

Theoretical Computer Science / Formal Methods, University of Kassel, Germany

Igor Semezie 

Theoretical Computer Science / Formal Methods, University of Kassel, Germany
Laboratoire Méthodes Formelles, ENS Paris-Saclay, France

Abstract

Temporal Graph Neural Networks (TGNN) are used to detect patterns in so-called temporal graphs (TG): graphs in which edges may be added or removed over time. Sälzer et al. suggested to study the expressive power of TGNNs through temporal-spatial logics, specifically the combination of the linear-time temporal logic LTL and modal logic K. In this paper we investigate the computational complexity of the satisfiability problem for this logic and its natural extension in which the LTL part is replaced by a linear-time μ -calculus, reflecting TGNNs' ability to recognise not just star-free (word) languages. We consider their interpretation over a more natural class of models which we call quasi-temporal graphs (QTG), relaxing certain conditions on the temporal evolutions of nodes that are indifferent to the logic. We formalise this by giving an adjusted notion of bisimulation which preserves satisfaction in these logics. It turns out that satisfiability over the class of QTGs is PSPACE-complete for the temporal-spatial logic based on LTL but becomes EXPTIME-complete when based on the μ -calculus. This is in contrast to the situation on words where both logics are PSPACE-complete. We also discuss consequences for the special satisfiability problems over the class of TGs.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Modal and temporal logics; Theory of computation $\rightarrow$ Machine learning theory

Keywords and phrases linear-time temporal logic, modal logic, temporal graphs, computational complexity, automated reasoning

Digital Object Identifier 10.4230/OASICS.TIME.2026.8

1 Introduction

Machine learning models are computational devices that have enabled revolutionary developments and success in artificial intelligence tasks with numerous applications in all kinds of industrial, societal, scientific etc. domains. One such model which has gained recent interest because of its use in the analysis of social networks for example, is that of a *Temporal Graph Neural Network* (TGNN) [25, 33]. It builds upon the better known model of *Graph Neural Network* (GNN) [18] which can be used to recognise patterns in static graphs, to the recognition of properties of graphs that are dynamically evolving in time.

Sälzer et al. [30] suggested to use product logics as yardsticks for the expressiveness of TGNNs, specifically two-dimensional products where the operators of one dimension are used to navigate through the sequence – called the *temporal* dimension – and the operators of the other – called the *spatial* dimension – are used to navigate through a single graph. This corresponds closely to the functionality of TGNNs. Formally, a TGNN recognises *temporal graphs* (TG) which are finite sequences of graphs which share the same node set. A TGNN processes such a sequence by iteratively updating node representations, combining



© Eric Alsmann, Martin Lange, and Igor Semezie;
licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 8; pp. 8:1–8:16

OpenAccess Series in Informatics



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

spatial information propagated through graph edges within each graph and with temporal information carried forward from preceding time steps. Crucially, spatial information is aggregated only from nodes within a bounded neighbourhood, while temporal information is passed on recursively in most TGNNs, so that the representation of a node at step $i + 1$ depends on its own representation at step i .

A natural candidate for a two-dimensional logic to capture the expressiveness of TGNNs is $\text{LTL}_f \times K$, using Linear-Time Temporal Logic on finite traces (LTL_f) [17] in the temporal dimension and the standard modal logic K in the spatial dimension. This provides some limited recursive mechanism temporally and no recursive mechanism spatially, i.e. expressible properties can address moments arbitrarily far in the future, but within a graph they can only speak about a neighbourhood of a node of fixed radius. This corresponds exactly to the way that TGNN collect information about a temporal graph, at least when abstracting away from the difference between real-valued and Boolean-valued information retrieval.

A natural extension of $\text{LTL}_f \times K$ is obtained when LTL is replaced by the linear-time μ -calculus μTL [2, 35], known to exceed the expressiveness of LTL by capturing all regular properties [11]. This is also motivated by the question after the formal expressiveness of TGNNs, as the use of non-linear functions in them permits them to do modulo counting, i.e. to recognise temporal graphs of even length for example. Hence, $\text{LTL}_f \times K$ may be a natural logic to consider with respect to TGNN expressiveness from a pragmatic point of view due to the popularity and simplicity of LTL ; but a logic based on μTL instead seems to be better suited from a technical perspective in order to fully capture the expressiveness of TGNNs.

As with LTL , μTL can of course be interpreted purely on finite words only, leading to the logic μTL_f [24] which comes with less combinatorial complexity. In particular, there is no need to distinguish least and greatest fixpoint operators when formulas are guarded, using the same kind of reasoning as is possible for the modal μ -calculus on acyclic structures [28]. Asymptotically, all four logics – LTL and μTL , interpreted on finite or infinite words – share the same complexity: their satisfiability problems are PSPACE-complete [32, 35].

Structure and contribution. We study the computational complexity of logics suggested for temporal graphs, in particular the aforementioned $\text{LTL}_f \times K$ and $\mu\text{TL}_f \times K$. In Sect. 2, we formally introduce (quasi-)temporal graphs and the logics $\text{LTL}_f \times K$ and $\mu\text{TL}_f \times K$. In Sect. 3 we show that their properties are invariant under bisimulations, adapted suitably to the temporal-spatial setting, while the class of TGs is not, thus justifying the use of the relaxed model of *quasi*-temporal graphs (QTG). In Sect. 4 we study the complexity of satisfiability for $\text{LTL}_f \times K$ and $\mu\text{TL}_f \times K$ over QTGs, obtaining the perhaps surprising result that – whilst all the ingredients LTL_f , μTL_f and K are PSPACE-complete, $\text{LTL}_f \times K$ remains PSPACE-complete but $\mu\text{TL}_f \times K$ becomes EXPTIME-complete. In Sect. 5 we conclude with remarks on the complexity of satisfiability over the class of TGs and the expressiveness of TGNNs, as well as further work for the future.

Related work. Logical characterisations of static GNNs via modal logics [3, 29] have inspired analogous questions for temporal architectures. For TGNNs, no connection to logics had been established prior to [30], where three classes of TGNNs are characterised via two-dimensional product logics of temporal and modal logics. Questions regarding the exact complexity of their satisfiability problems as well as their ability to distinguished temporal graphs from non-temporal graphs were not investigated there.

Many-dimensional logics [16, 27] offer a general framework for combining modal logics, each capturing a distinct aspect of reasoning such as time, space, or knowledge. The standard theory distinguishes *fusion* logics, in which the component logics interact only through

shared propositional variables, from *product* logics, in which models carry a Cartesian product structure. Satisfiability and axiomatisation of two-dimensional product logics have been studied in depth by Gabbay et al. [16]. There, product logics are, however, interpreted on structures with *static* edge sets, in fact *products* of structures, which enforces the commutativity axioms $\Diamond \bigcirc p \leftrightarrow \bigcirc \Diamond p$ and $\bigcirc \Box p \rightarrow \Box \bigcirc p$. These are not axioms anymore when edges may change over time.

The work that is most closely related to the present study is on temporal description logics, cf. [26]. The structures known as temporal graphs in the machine learning community are known as “temporal interpretations over a constant domain” in the description logic setting, although they are fairly simple special cases with a single role (binary relation) only. The logic $\text{LTL} \times \mathbf{K}$, as considered here, could be seen as a simple description logic over such structures. A minor difference lies in the treatment of time: while the time line there is considered to be infinite, it is finite in the study here, hence we consider $\text{LTL}_f \times \mathbf{K}$. The main difference, though, is that we also study an extension in the temporal dimension, namely the logic $\mu\text{TL}_f \times \mathbf{K}$, which is not done in the description logic work, and there the focus lies on different reasoning problems where complexity is introduced via TBox axioms which is not present here.

2 Preliminaries

Let AP be a countable set of atomic propositions and $\mathcal{V}$ be a countable set of (monadic second-order) variables. A *graph* is a triple $(S, \rightarrow, L)$, where S is a finite set, $\rightarrow \subseteq S \times S$ is the *transition relation*, and $L : S \rightarrow 2^{\text{AP}}$ is the *labeling of the nodes* such that $|L(v)| < \infty$ for all $v \in S$.

(Quasi-)Temporal graphs. A *quasi-temporal graph* (QTG) is a tuple $\mathfrak{G} = (G_0, \sigma_0, G_1, \sigma_1, \dots, \sigma_{n-2}, G_{n-1})$, such that for all $0 \leq i < n$, $G_i = (S_i, \rightarrow_i, L_i)$ is a directed, 2^{AP} -labeled graph, and for all $0 \leq i < n-1$, $\sigma_i : S_i \rightarrow S_{i+1}$ is the *i-th time successor function* which is partial in general. A *pointed QTG* is a triple $(\mathfrak{G}, v, i)$ where $\mathfrak{G}$ is a QTG as above, $i \in \{0, \dots, n-1\}$ and $v \in S_i$. Thus, it simply singles out a particular node at a particular moment in time in the QTG. Intuitively, $\sigma_i(v)$ is what v becomes in the *i-th* time step. Nodes need not have temporal successors, but if they do, then these are unique. This is why each σ_i is a partial function. A node at moment $i > 0$ can be the temporal successor of more than one node at moment $i-1$, though. We introduce some useful notation. Let $\mathfrak{G}$ be as above. For i, j with $0 \leq i < j < n-1$ we write $\sigma_{i,j}$ for the composition $\sigma_i \circ \dots \circ \sigma_{j-1}$. Here, composition of partial functions can be seen as relation composition. In particular, nodes in G_i need not have a $\sigma_{i,j}$ -successor for $j > i$ but if they do then it is the unique node that is obtained by following the functions $\sigma_i, \dots, \sigma_{j-1}$ successively starting from $v \in S_i$. Moreover, in this case we have $\sigma_{i,j}(v) \in S_j$ unless it does not exist.

A *temporal graph* (TG) is a quasi-temporal graph $\mathfrak{G} = (G_0, \sigma_0, G_1, \sigma_1, \dots, \sigma_{n-2}, G_{n-1})$ in which each σ_i is total and bijective for $i = 0, \dots, n-2$. This is clearly equivalent to the formulation found in the literature which defines a temporal graph as a sequence of edge relations over the same nodes set.

Linear-time temporal logic for temporal graphs. Let AP be as above. Formulas of $\text{LTL}_f \times \mathbf{K}$ over AP are defined by the grammar

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi \mid \varphi \cup \varphi \mid \Diamond \varphi$$

where $p \in \text{AP}$. *Next* ($\bigcirc$) and *until* (U) are the *temporal* operators, and *Diamond* ($\Diamond$) is the spatial operator. The set $\text{sub}(\varphi)$ of subformulas of an $\text{LTL}_f \times \text{K}$ formula φ is defined straightforwardly. The *size* of φ is $|\varphi| := |\text{sub}(\varphi)|$.

We define the usual Boolean, temporal and modal abbreviations as follows.

$$\begin{array}{lll} \text{ff} := p \wedge \neg p \quad \text{for some } p \in \text{AP} & \text{tt} := \neg \text{ff} & \Box \varphi := \neg \Diamond \neg \varphi \\ \varphi \vee \psi := \neg(\neg \varphi \wedge \neg \psi) & & \odot \varphi := \neg \bigcirc \neg \varphi \\ \varphi \text{ R } \psi := \neg(\neg \varphi \text{ U } \neg \psi) & \text{F } \varphi := \text{tt U } \varphi & \text{G } \varphi := \neg \text{F } \neg \varphi \end{array}$$

An $\text{LTL}_f \times \text{K}$ formula is in *negation normal form* (NNF) if it is built from literals p and $\neg p$ using the operators $\wedge, \vee, \Diamond, \Box, \bigcirc, \odot, \text{U}$ and R only. It is standard to show that every formula can equivalently be transformed into NNF with an at most linear blow-up in size. Equivalence needs a formal notion of semantics of course, which we define next.

Formulas of $\text{LTL}_f \times \text{K}$ are interpreted on pointed QTG $(\mathfrak{G}, v, i)$ with $\mathfrak{G} = (G_0, \sigma_0, \dots, \sigma_{n-2}, G_{n-1})$ and $G_i = (S_i, \rightarrow_i, L_i)$ for $i = 0, \dots, n-1$ as follows.

$$\begin{array}{ll} \mathfrak{G}, v, i \models p & \iff p \in L_i(v) \\ \mathfrak{G}, v, i \models \varphi \wedge \psi & \iff \mathfrak{G}, v, i \models \varphi \text{ and } \mathfrak{G}, v, i \models \psi \\ \mathfrak{G}, v, i \models \neg \varphi & \iff \mathfrak{G}, v, i \not\models \varphi \\ \mathfrak{G}, v, i \models \bigcirc \varphi & \iff i < n-1 \text{ and } \mathfrak{G}, \sigma_i(v), i+1 \models \varphi \\ \mathfrak{G}, v, i \models \varphi \text{ U } \psi & \iff \text{there is } k \text{ s.t. } i \leq k < n \text{ and } \mathfrak{G}, \sigma_{i,k}(v), k \models \psi \\ & \quad \text{and for all } j \text{ with } i \leq j < k : \mathfrak{G}, \sigma_{i,j}(v), j \models \varphi \\ \mathfrak{G}, v, i \models \Diamond \varphi & \iff \text{there is } u \in S_i \text{ s.t. } v \rightarrow_i u \text{ and } \mathfrak{G}, u, i \models \varphi \end{array}$$

Two formulas φ, ψ of $\text{LTL}_f \times \text{K}$ are equivalent, written $\varphi \equiv \psi$, if for all pointed QTGs $\mathfrak{G}, v, i$ we have $\mathfrak{G}, v, i \models \varphi$ iff $\mathfrak{G}, v, i \models \psi$.

The following is standard knowledge about LTL, resp. LTL_f , and holds in $\text{LTL}_f \times \text{K}$ for exactly the same reasons.

► **Lemma 1.** *For all $\varphi, \psi \in \text{LTL}_f \times \text{K}$ we have*

$$\varphi \text{ U } \psi \equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \text{ U } \psi)) \quad , \quad \varphi \text{ R } \psi \equiv \psi \wedge (\varphi \vee \odot(\varphi \text{ R } \psi)) \quad .$$

Fixpoint logic for temporal graphs. The two-dimensional logic $\mu\text{TL}_f \times \text{K}$ is obtained from $\text{LTL}_f \times \text{K}$ by replacing the temporal Until operator with a general fixpoint operator rec . Traditionally, μ -calculi are introduced with two different fixpoint operators μ and ν to denote the generally different *least* and *greatest* fixpoint of a predicate transformer $\varphi(X)$ making use of the Knaster-Tarski Theorem [21, 34]. However, the difference only shows up in limits, i.e. in infinite unfoldings. This is prevented here by a combination of the finiteness in the temporal dimension of models and the impossibility to explore paths of infinite length inside a single graph, ensured by the syntactic restriction of guardedness.

Let AP be given as above. Moreover, let $\text{Var} = \{X, Y, \dots\}$ be a countable set of monadic second-order variables. Formulas of $\mu\text{TL}_f \times \text{K}$ over AP (and, technically, also over Var) are defined by the grammar

$$\varphi ::= p \mid X \mid \varphi \wedge \varphi \mid \neg \varphi \mid \bigcirc \varphi \mid \text{rec } X. \varphi \mid \Diamond \varphi$$

where $p \in \text{AP}$ and $X \in \text{Var}$.

The set $\text{sub}(\varphi)$ of subformulas of φ is defined as usual. Again, the size of φ is $|\varphi| := |\text{sub}(\varphi)|$. As with $\text{LTL}_f \times K$, we introduce $\mathbf{ff}, \mathbf{tt}, \vee, \odot$ and $\square$ as abbreviations. A $\mu\text{TL}_f \times K$ formula is said to be in NNF, if it is built from literals p and $\neg p$ for any $p \in \text{AP}$, as well as variables $X \in \text{Var}$, using only the operators $\wedge, \vee, \odot, \odot, \text{rec}, \diamond$ and $\square$. Again, it is standard to show that every formula can equivalently be transformed into one in NNF with an at most linear blow-up, with equivalence formally defined below.

The operator rec acts as a *binder* for the variable X . We only consider sentences, i.e. formulas without unbound occurrences of variables, as top-level formulas. Formulas with free variables can of course occur as subformulas of a formula in $\mu\text{TL}_f \times K$.

We employ further syntactic restrictions on formulas. They are all standard in the literature on μ -calculi, cf. [6, 7, 12]. (I) Formulas must be *well-named*: each variable X gets bound at most once in any subformula of the form $\text{rec } X.\psi$. Then such a φ comes with an identifier function fp_φ mapping a variable X that is bound in φ to the unique subformula $\text{rec } X.\psi$ in which it is bound. (II) There must be an even number of negation symbols on any path in the syntax tree of a formula φ , starting in $fp_\varphi(X)$ for any $X \in \text{Var}$ and ending in a leaf node X . This ensures that fixpoints are well-defined because the underlying predicate transformers $\psi(X)$ are indeed monotone. (III) Formulas must be *temporally guarded*. In the syntax tree there must be at least one occurrence of an operator $\odot$, on any path starting in $fp_\varphi(X)$ and ending in a leaf node X , for any variable X .

Unlike (I) and (II), restriction (III) is in fact a semantic restriction. One cannot simply use the well-known fact that formulas of the μ -calculus can equivalently be transformed into guarded form [22, 37, 8], because here we require variables in fixpoint formulas not just to be guarded by *some* modal or temporal operator, but specifically by a temporal one. Hence, formulas like $\text{rec } X.p \vee \diamond X$ are guarded from a modal μ -calculus perspective but temporally unguarded regarding condition (III). Such a formula expresses reachability of a p -node in the current graph at an arbitrary distance, and this is a property that TGNN most likely cannot check which justifies the restriction to temporally guarded formulas. We leave it as an open question whether equivalence of an unrestricted formula to a temporally guarded one can be decided and such a temporally guarded equivalent can be constructed automatically. In any case, such considerations rely on a formal semantics for $\mu\text{TL}_f \times K$ which we explain next.

Formulas of $\mu\text{TL}_f \times K$ are interpreted in pointed QTGs $(\mathfrak{G}, v, i)$. It is more convenient to fix a QTG $\mathfrak{G} = (G_0, \sigma_0, \dots, \sigma_{n-2}, G_{n-1})$ with $G_i = (S_i, \rightarrow_i, L_i)$ for $i = 0, \dots, n-1$ over the same set AP of atomic propositions, and then to define, by induction on the structure of a given formula φ , its *extension* $\llbracket \varphi \rrbracket_\rho^\mathfrak{G}$ in $\mathfrak{G}$, consisting of all points (v, i) such that $\mathfrak{G}, v, i \models \varphi$ in the same sense as for $\text{LTL}_f \times K$. Induction on the formula structure bears the burden of dealing with free variables in subformulas. This is done by extending the interpretation to a set $\llbracket \varphi \rrbracket_\rho^\mathfrak{G}$ of points (v, i) in time that satisfy φ under the assumption that any free variable X is satisfied by the points in $\rho(X)$. Thus, $\rho : \text{Var} \rightarrow \mathcal{P}(\bigcup_{i=0}^{n-1} S_i \times \{i\})$. We write $\rho[X \mapsto T]$ to denote the function which maps X to T and behaves like ρ otherwise.

$$\begin{aligned}
\llbracket p \rrbracket_\rho^\mathfrak{G} &= \{(v, i) \mid p \in L_i(v)\} \\
\llbracket X \rrbracket_\rho^\mathfrak{G} &= \rho(X) \\
\llbracket \neg \varphi \rrbracket_\rho^\mathfrak{G} &= \bigcup_{i=0}^{n-1} (S_i \times \{i\}) \setminus \llbracket \varphi \rrbracket_\rho^\mathfrak{G} \\
\llbracket \varphi \wedge \psi \rrbracket_\rho^\mathfrak{G} &= \llbracket \varphi \rrbracket_\rho^\mathfrak{G} \cap \llbracket \psi \rrbracket_\rho^\mathfrak{G} \\
\llbracket \odot \varphi \rrbracket_\rho^\mathfrak{G} &= \bigcup_{i=0}^{n-2} \{(v, i) \mid v \in S_i, \sigma_i(v) \text{ exists and } (\sigma_i(v), i+1) \in \llbracket \varphi \rrbracket_\rho^\mathfrak{G}\}
\end{aligned}$$

$$\begin{aligned} \llbracket \text{rec } X.\varphi \rrbracket_\rho^\mathfrak{G} &= \bigcap \left\{ T \subseteq \bigcup_{i=0}^{n-1} S_i \times \{i\} \mid \llbracket \varphi \rrbracket_{\rho[X \rightarrow T]}^\mathfrak{G} \subseteq T \right\} \\ \llbracket \Diamond \varphi \rrbracket_\rho^\mathfrak{G} &= \bigcup_{i=0}^{n-1} \{(v, i) \in S_i \times \{i\} \mid \text{there is } u \in S_i \text{ s.t. } v \rightarrow_i u \text{ and } (u, i) \in \llbracket \varphi \rrbracket_\rho^\mathfrak{G}\} \end{aligned}$$

It is standard to show that sentences φ are invariant under variable interpretations in the sense that $\llbracket \varphi \rrbracket_\rho^\mathfrak{G} = \llbracket \varphi \rrbracket_{\rho'}^\mathfrak{G}$ for any $\mathfrak{G}$, closed φ and ρ, ρ' . Hence, in such cases we can simply write $\llbracket \varphi \rrbracket^\mathfrak{G}$ or, likewise, $\mathfrak{G}, v, i \models \varphi$ for some v .

The notion of equivalence between two sentences, $\varphi \equiv \psi$, carries over from $\text{LTL}_f \times K$ to $\mu\text{TL}_f \times K$. Since sentences of the latter and formulas of the former are both interpreted in pointed QTGs, equivalence between these is also well-defined.

We get an embedding of $\text{LTL}_f \times K$ into $\mu\text{TL}_f \times K$ by a straight-forward extension of the known translation from LTL into μTL where $\varphi \cup \psi$ becomes $\text{rec } X.\psi \vee (\varphi \wedge \bigcirc X)$.

► **Lemma 2.** *For every $\varphi \in \text{LTL} \times K$ there is $\varphi' \in \mu\text{TL}_f \times K$ s.t. $|\varphi'| = \mathcal{O}(|\varphi|)$ and $\varphi' \equiv \varphi$.*

For reasoning about fixpoint formulas it is useful to introduce approximants. Since we are only interested in finite models, we do not need trans-finite ordinals in their construction. For a fixpoint formula $\text{rec } X.\varphi$ we define its approximants for all $i \in \mathbb{N}$ via $\text{rec}^0 X.\varphi := \text{ff}$ and $\text{rec}^{i+1} X.\varphi := \varphi[\text{rec}^i X.\varphi/X]$. The following enables reasoning about formulas with free variables by interpreting them as approximants. Its proof is standard, relies on temporal guardedness and is omitted.

► **Lemma 3.** *Let $\mathfrak{G}$ be a QTG of some length n . For every $\text{rec } X.\varphi \in \mu\text{TL}_f \times K$ and every variable assignment ρ we have $\llbracket \text{rec } X.\varphi \rrbracket_\rho^\mathfrak{G} = \llbracket \text{rec}^n X.\varphi \rrbracket_\rho^\mathfrak{G}$.*

3 Temporal Bisimulations

We adjust the well-known notion of bisimulation to the setting of QTGs.

► **Definition 4.** *Let $\mathfrak{G} = (G_0, \sigma_0, \dots, \sigma_{n-2}, G_{n-1})$ and $\mathfrak{G}' = (G'_0, \sigma'_0, \dots, \sigma'_{n'-2}, G'_{n'-1})$ be two QTGs, not necessarily of the same length, with $G_i = (S_i, \rightarrow_i, L_i)$ and $G'_j = (S'_j, \rightarrow'_j, L'_j)$ for $i = 0, \dots, n-1$ and $j = 0, \dots, n'-1$.*

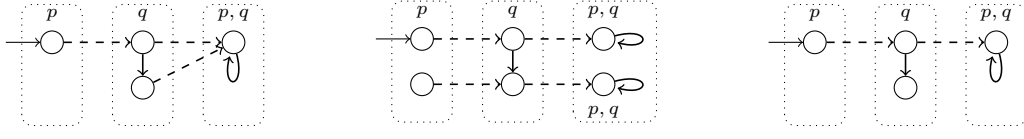
A relation β between points of $\mathfrak{G}$ and $\mathfrak{G}'$ is a temporal bisimulation if for all $(v, i) \beta (u, j)$ we have

- (1) $L_i(v) = L'_j(u)$,
 - (2) *for all $v' \in S_i$ with $v \rightarrow_i v'$ there is $u' \in S'_j$ with $u \rightarrow'_j u'$ and $(v', i) \beta (u', j)$,*
 - (3) *for all $u' \in S'_j$ with $u \rightarrow'_j u'$ there is $v' \in S_i$ with $v \rightarrow_i v'$ and $(v', i) \beta (u', j)$,*
 - (4) $\sigma_i(v)$ exists iff $\sigma'_j(u)$ exists, and in case they do, $(\sigma_i(v), i+1) \beta (\sigma'_j(u), j+1)$.
- As usual, two points (v, i) and (u, j) are temporally bisimilar, written $(\mathfrak{G}, v, i) \rightleftharpoons (\mathfrak{G}', u, j)$ if there is a bisimulation β between $\mathfrak{G}$ and $\mathfrak{G}'$ such that $(v, i) \beta (u, j)$.*

In condition (4), it should be understood that $\sigma_i(v)$ does not exist whenever $i = n-1$, because the function σ_{n-1} itself does not exist (and similarly for when $j = n'-1$).

In the following, we will simply speak of bisimulations etc. when in fact we mean temporal bisimulations. No confusion should arise, as ordinary bisimulations play no technical role here.

The following theorem states that $\mu\text{TL}_f \times K$ is invariant under temporal bisimulations. The proof is the same as for one-dimensional μ -calculi (see, for instance, [12]). Finiteness of QTGs in length simplifies the proof because fixpoints are reached after a finite amount of



■ **Figure 1** Examples of QTGs, resp. a TG (in the middle), concerning temporal bisimilarity.

iterations, cf. Lemma 3. We introduce elementary equivalence between two pointed QTGs in the usual way: we have $(\mathfrak{G}, v, i) \equiv_{\mu\text{TL}_f \times K} (\mathfrak{G}', u, j)$ if for all $\varphi \in \mu\text{TL}_f \times K$: $\mathfrak{G}, v, i \models \varphi$ iff $\mathfrak{G}', u, j \models \varphi$.

► **Theorem 5.** *For two pointed QTGs $(\mathfrak{G}, v, i)$ and $(\mathfrak{G}', u, j)$ we have $\mathfrak{G}, v, i \rightleftharpoons \mathfrak{G}', u, j$ iff $(\mathfrak{G}, v, i) \equiv_{\mu\text{TL}_f \times K} (\mathfrak{G}', u, j)$.*

The proof is in Appendix A. Whilst Thm. 5 is not surprising, it is significant for a study of the expressiveness of TGNNs that are interpreted over TGs. Note that, unlike the class of QTGs, the class of TGs is not closed under temporal bisimulations. Specifically, the two pointed QTGs shown on the left and in the middle of Fig. 1, with spatial neighbourhood relation shown in solid arrows and temporal successors in dashed arrows, are bisimilar but only one of them is a TG. These two suggest that QTGs can be “unwound” and given extra dummy nodes in early graphs of the sequence in order to turn them into a bisimilar TG. However, the QTG on the right show that this is not possible, specifically when there are maximal paths from the initial point that do not have the same number of temporal transitions in them.

One may argue that this calls for a stronger notion of quasi-temporal graph, namely sequences $(G_0, \sigma_0, \dots, \sigma_{n-2}, G_{n-1})$ such that all temporal successor functions σ are *total*. We call such QTGs (temporally) *synchronised*. The aforementioned technique of unwinding temporal paths and potentially completing their pasts yields the following result.

► **Theorem 6.** *For every pointed and temporally synchronised QTG $(\mathfrak{G}, v, 0)$ there is a pointed TG $(\mathfrak{G}', v, 0)$ with the same temporal length and at most polynomial blow-up in nodes such that $(\mathfrak{G}, v, 0) \rightleftharpoons (\mathfrak{G}', v, 0)$.*

Thm. 6 does not hold for temporally unsynchronised QTGs. This is also witnessed by the formula $p \wedge \bigcirc(q \wedge \Diamond \bigcirc \text{ff} \wedge \bigcirc \text{tt})$ which is satisfied by the QTG on the right in Fig. 1 but cannot be satisfied by a synchronised QTG.

4 The Satisfiability Problems

We study the computational complexity of the satisfiability problems for $\text{LTL}_f \times K$ and $\mu\text{TL}_f \times K$ over the class of QTGs. We begin with lower bounds.

4.1 Lower Bounds

The satisfiability problem for LTL and also LTL_f is well-known to be PSPACE-complete [32]. The lower bound trivially carries over to $\text{LTL}_f \times K$ since every formula of LTL_f is also one of $\text{LTL}_f \times K$ and a QTG model $(\mathfrak{G}, v, 0)$ of the latter is easily turned into a pure linear-time model of the form $v, \sigma_{0,1}(v), \sigma_{0,2}(v), \dots, \sigma_{0,n-1}(v)$ for the respective temporal successor functions $\sigma_0, \dots, \sigma_{n-2}$.

► **Theorem 7.** *The satisfiability problem for $\text{LTL}_f \times K$ is PSPACE-hard.*

In the pure linear-time setting, μTL has higher expressiveness than LTL but its satisfiability problem is not more difficult, at least not more than by a polynomial overhead: it is also PSPACE-complete [20]. Interestingly, $\mu\text{TL}_f \times \text{K}$ appears to be more difficult than $\text{LTL}_f \times \text{K}$. We show that its satisfiability problem is EXPTIME-hard. This is at least curious in that $\mu\text{TL}_f \times \text{K}$ appears to be a linear-time *temporal* logic whose satisfiability problems are typically complete for space-complexity classes. It is typically the satisfiability problems for branching-time temporal logics that tend to be complete for time-complexity classes [13, 15, 14, 36, 19] with the exception of K which is arguably not really a *temporal* logic as its formulas cannot “look” arbitrarily deep into a structure. The satisfiability problem for K is also known to be PSPACE-complete [23].

So the combination of the PSPACE-complete logics LTL_f and K into $\text{LTL}_f \times \text{K}$ turns out not to be more difficult than PSPACE – a matching upper bound is given below – whilst the combination of the PSPACE-complete logics μTL and K becomes EXPTIME-hard.

We show EXPTIME-hardness by a reduction from a tiling game problem, known to be EXPTIME-complete [10] as it captures the essence of the word problem for alternating, space bounded Turing machines and $\text{APSPACE} = \text{EXPTIME}$ [9].

► **Definition 8.** A tiling system is a $\mathcal{T} = (T, H, V, t_I, t_F)$ where T is a finite set of tiles with two designated initial and final tiles $t_I, t_F \in T$ such that $t_I \neq t_F$. $H, V \subseteq T \times T$ are the horizontal and vertical matching relation.

The n -corridor tiling game is played between players $\mathbf{S}$ and $\mathbf{C}$ who successively places tiles on the grid $\{0, \dots, n-1\} \times \mathbb{N}$ row-by-row, starting with row 0, i.e. placing tiles on the coordinates $(0, 0), \dots, (n-1, 0)$, then filling row 1, etc. The filling of the i -th row is done as follows.

1. Player $\mathbf{S}$ chooses a tile $t \in T$ and places it on the coordinate $(0, i)$.
 2. Player $\mathbf{C}$ chooses tiles $t_1, \dots, t_{n-1} \in T$ places them on the coordinates $(1, i), \dots, (n-1, i)$.
- This can also be seen as the successive construction of a mapping $\tau : [n] \times \mathbb{N} \rightarrow T$. Thus, we will write $\tau(j, i)$ for the tile placed in the j -th column in row i .

Player $\mathbf{C}$ wins a play constructing ρ if one of the following conditions holds.

- $\rho(0, 0) \neq t_I$, i.e. $\mathbf{S}$ did not choose the designated initial tile at the very beginning.
- There is $i \in \mathbb{N}$ such that $(\tau(0, i), \tau(0, i+1)) \notin V$, i.e. player $\mathbf{S}$ has violated the vertical matching condition when placing a tile.
- There are $j \in [n]$, $i \in \mathbb{N}$ such that $\tau(j, i) = t_F$, i.e. some player placed the final tile.

Likewise, $\mathbf{S}$ wins the play if there is $j \in \{0, \dots, n-2\}$ and $i \in \mathbb{N}$ such that $(\tau(j, i), \tau(j+1, i)) \notin H$, i.e. $\mathbf{C}$ placed a tile does not match its neighbour horizontally.

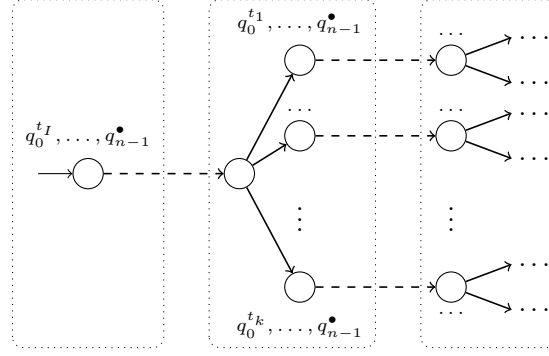
The n -corridor tiling game problem is the following. Given a tiling system $\mathcal{T}$ and an $n \in \mathbb{N}$ encoded unarily, decide whether player $\mathbf{C}$ has a winning strategy for the n -corridor tiling game.

► **Proposition 9** ([10]). The n -corridor tiling game problem is EXPTIME-complete.

► **Theorem 10.** Satisfiability for $\mu\text{TL}_f \times \text{K}$ over quasi-temporal graphs is EXPTIME-hard.

Proof. By a reduction from the n -corridor tiling game problem. Let $\mathcal{T} = (T, H, V, t_I, t_F)$ and $n \in \mathbb{N}$ be given, with n encoded unarily. This means that the construction only needs to be polynomial in the value of n , not necessarily the size needed to write down n succinctly in binary encoding for instance.

We construct a formula $\Phi_{\mathcal{T}, n} \in \mu\text{TL}_f \times \text{K}$ that is satisfied in a QTG iff $\mathbf{C}$ has a winning strategy for the game on $\mathcal{T}$ and n . Intuitively such a winning strategy can be seen as a tree with nodes representing tiled rows (and separating nodes in between) as shown in Fig. 2.



■ **Figure 2** Representation of winning strategies in n -corridor tiling games as quasi-temporal graphs.

We use $\text{AP} = T \times [n]$ and write q_i^t for the proposition (t, i) . Note that an entirely tiled row $t_0, \dots, t_{n-1}$ can be represented in the label of a single graph node v as $L(v) = \{q_0^{t_0}, \dots, q_{n-1}^{t_{n-1}}\}$. Fig. 2 also shows how the tree-like winning strategy can be seen as a quasi-temporal graph in which every graph has a designated node that is the temporal successor of a node in the previous graph, and has multiple spatial neighbours. The branching represents the choices that **S** makes at the start of each row.

We construct $\Phi_{\mathcal{T},n}$ as the conjunction of the following five formulas, resp. properties.

(I) The labeling on the first node and every second one after that needs to represent the tiling of a row in the sense that for every $j \in [n]$ the label contains a unique proposition q_j^t .

$$\varphi_{\text{rows}} := \text{rec } X. \left(\bigwedge_{j=0}^{n-1} \bigvee_{t \in T} q_j^t \wedge \bigwedge_{\substack{t' \in T \\ t' \neq t}} \neg q_j^{t'} \right) \wedge \odot \Box X$$

(II) The first tile is t_I . This is expressed simply as $\varphi_{\text{init}} := q_0^{t_I}$. (III) The horizontal matching relation is respected everywhere.

$$\varphi_{\text{horiz}} := \text{rec } X. \left(\bigwedge_{j=0}^{n-2} \bigwedge_{(t,t') \notin H} \neg (q_j^t \wedge q_{j+1}^{t'}) \right) \wedge \odot \Box X$$

(IV) The vertical matching relation is respected everywhere that **C** makes a move.

$$\varphi_{\text{vert}} := \text{rec } X. \left(\bigwedge_{j=1}^{n-1} \bigwedge_{(t,t') \notin V} \neg (q_j^t \wedge \odot \Diamond q_j^{t'}) \right) \wedge \odot \Box X$$

(v) The tree representing a winning strategy for **C** contains all the choices that are possible for **S**, unless the final tile has been placed already. We use the abbreviation tV for $\{t' \mid (t, t') \in V\}$.

$$\varphi_{\text{play}} := \text{rec } X. \left(\left(\bigwedge_{j=0}^{n-1} \neg q_j^{t_F} \right) \rightarrow \left(\bigwedge_{t \in T} q_0^t \rightarrow \bigwedge_{t' \in tV} \odot \Diamond q_0^{t'} \right) \right) \wedge \odot \Box X$$

Note that, if $tV = \emptyset$ then the corresponding conjunction becomes true, and no diamonds need to be satisfied. This guarantees the existence of finite models representing winning strategies in which **C** wins plays because **S** has no legal move to play.

$$\begin{array}{ll}
(\text{Or}) \quad \frac{\varphi_i, \Phi}{\varphi_1 \vee \varphi_2, \Phi} & (\text{Mod}) \quad \frac{\Phi, \Psi \quad \varphi_1, \Gamma \quad \dots \quad \varphi_k, \Gamma}{\bigcirc \Phi, \odot \Psi, \diamond \varphi_1, \dots, \diamond \varphi_k, \square \Gamma, \Lambda} \quad \Phi \neq \emptyset \\
(\text{And}) \quad \frac{\varphi_1, \varphi_2, \Phi}{\varphi_1 \wedge \varphi_2, \Phi} & (\text{Mod}_0) \quad \frac{\varphi_1, \Gamma \quad \dots \quad \varphi_k, \Gamma}{\odot \Psi, \diamond \varphi_1, \dots, \diamond \varphi_k, \square \Gamma, \Lambda} \\
(\text{FP}) \quad \frac{\varphi, \Phi}{X, \Phi} \quad fp(X) = \text{rec } X.\varphi & (\text{Var}) \quad \frac{X, \Phi}{\text{rec } X.\varphi, \Phi}
\end{array}$$

■ **Figure 3** Rules for the satisfiability tableaux for $\mu\text{TL}_f \times \text{K}$.

Clearly, $\Phi_{\mathcal{T},n} := \varphi_{\text{rows}} \wedge \varphi_{\text{init}} \wedge \varphi_{\text{horiz}} \wedge \varphi_{\text{vert}} \wedge \varphi_{\text{play}}$ is of polynomial size in $|\mathcal{T}|$ and the value of n . Moreover, a quasi-temporal graph satisfying it represents a winning strategy in the way that is sketched above, and vice-versa. This construction therefore facilitates a polynomial reduction from the n -corridor tiling game problem to the satisfiability problem of $\mu\text{TL}_f \times \text{K}$. ◀

4.2 Upper Bounds

We begin with an upper bound for the more general logic $\mu\text{TL}_f \times \text{K}$. We characterise its satisfiability problem in terms of tableaux as witnesses for satisfiability and then show that their existence can be decided in exponential time.

We use Greek capital letters to denote sets of formulas. We avoid braces and simply write φ, Φ for the set $\{\varphi\} \cup \Phi$, additionally implying $\varphi \notin \Phi$. For the modal and temporal next-operators $\otimes \in \{\diamond, \square, \bigcirc, \odot\}$ we write $\otimes\Phi$ to denote a set of formulas in which everyone starts with the operator $\otimes$. In the context of such an $\otimes\Phi$, Φ then denotes the set of all formulas that are obtained from $\otimes\Phi$ by stripping of the top-level temporal, resp. modal operator $\otimes$. The symbol Λ is solely used for a propositionally consistent set of literals, i.e. a set of formulas of the form q or $\neg q$ that does not contain both the positive and negative version of any proposition.

► **Definition 11.** A tableau for an $\text{LTL}_f \times \text{K}$, resp. $\mu\text{TL}_f \times \text{K}$ formula φ in negation normal form is a finite tree whose nodes are labeled with sets of subformulas of φ , built according to the rules presented in Fig. 3. The tableau is closed if all its branches end in an axiom, i.e. a set of the form $\odot \Psi, \square \Gamma, \Lambda$ with Ψ and Γ possibly empty, and Λ being a consistent set of literals.

Remember that $\mu\text{TL}_f \times \text{K}$ formulas are required to be temporally guarded. Hence, any tableau for such a formula φ_0 is structured as follows. Starting with the root node there are a linearly (in $|\varphi_0|$) bounded number of applications of the rules (Or), (And), (FP) and (Var), leading to a non-branching growth of the tableau tree until a formula set is reached that matches the premise of either of the rules (Mod) or (Mod₀). We call such a sequence of formula sets a *block*. The application of either of the latter rules creates a branching of degree given by the number of $\diamond$ -formulas, i.e. also linear in $|\varphi_0|$. From the premises obtained after the application of either of the modal rules, new blocks can be obtained just like they are from the root node. Hence, any such tableau can be compressed into a tree whose nodes are the blocks and whose branching is given by the modal rules.

This tableau calculus is sound and complete w.r.t. satisfiability of $\mu\text{TL}_f \times \text{K}$ formulas over the class of QTGs.

$$\begin{array}{c}
(\mathbf{U}^-) \frac{\psi, \Phi}{\varphi \mathbf{U} \psi, \Phi} \quad (\mathbf{U}^+) \frac{\varphi, \bigcirc(\varphi \mathbf{U} \psi), \Phi}{\varphi \mathbf{U} \psi, \Phi} \quad (\mathbf{R}^-) \frac{\psi, \varphi, \Phi}{\varphi \mathbf{R} \psi, \Phi} \\
(\mathbf{R}^+) \frac{\psi, \bigcirc(\varphi \mathbf{R} \psi), \Phi}{\varphi \mathbf{R} \psi, \Phi}
\end{array}$$

■ **Figure 4** Rules for the satisfiability tableaux for $\text{LTL}_f \times \mathbf{K}$ to replace **(FP)** and **(Var)**.

► **Theorem 12.** *Let φ_0 be a formula of $\mu\text{TL}_f \times \mathbf{K}$. It is satisfiable over the class of quasi-temporal graphs iff there is a closed tableau for φ_0 .*

The proof is in Appendix A and follows standard arguments: a closed tableau can be collapsed into a tree-like QTG model of the formula, and a given model can be used to guide the construction of a closed tableau.

An immediate consequence of Thm. 12 is the fact that satisfiability in $\mu\text{TL}_f \times \mathbf{K}$ over QTGs can be decided in exponential time.

► **Theorem 13.** *The satisfiability problem for $\mu\text{TL}_f \times \mathbf{K}$ over QTGs is in EXPTIME.*

Proof. We sketch an alternating, polynomially space-bounded algorithm [9]. Note that every branch of length more than 2^{2^n} for formulas of size n must repeat some formula set. Hence, tableaux can be assumed to be of height at most 2^{2^n} . Then an alternating algorithm can check for the existence of such a “small” tableau by nondeterministically guessing applications of rule **(Or)** and deterministically applying **(And)**, **(FP)** and **(Var)** whenever needed. When eventually rule **(Mod)** or **(Mod₀)** need to be applied, the algorithm universally branches to follow all premises. Additionally, it maintains a counter that is started at 2^{2^n} and decreased in every step of a rule application. If the counter reaches 0, it rejects. It accepts when a branch can be closed.

The space needed for this is polynomial: it only needs to store a formula set and a counter of maximal value 2^{2^n} . The latter only requires polynomially many bits. The former is also only of polynomial size, as it only requires the storing of a single bit per subformula. ◀

So this raises the question after the exact complexity of $\text{LTL}_f \times \mathbf{K}$. The crucial use of unbounded alternations between modal and temporal operators in the fixpoint formulas constructed to simulate the tiling game problem would suggest that satisfiability for $\text{LTL}_f \times \mathbf{K}$ is not as hard as that for $\mu\text{TL}_f \times \mathbf{K}$. The fact that LTL_f and μTL_f have the same asymptotic complexity may suggest that so should $\text{LTL}_f \times \mathbf{K}$ and $\mu\text{TL}_f \times \mathbf{K}$. This is not true, though. The lack of a general fixpoint operator that allows formulas to look arbitrarily deep into graphs (that are sufficiently far away in a QTG) makes for a simpler complexity, at least up to common complexity-theoretic assumptions.

To formally reason about satisfiability in $\text{LTL}_f \times \mathbf{K}$ we introduce a similar tableau calculus: simply replace rules **(FP)** and **(Var)** by the four rules shown in Fig. 4. Correctness of these tableau is proved in the same way as in Thm. 12. The only difference here is that the formulas occurring in a tableau for φ_0 are not just subformulas of φ_0 . Instead, for each subformula of the form $\varphi \mathbf{U} \psi$ or $\varphi \mathbf{R} \psi$, the extended subformulas $\bigcirc(\varphi \mathbf{U} \psi)$ and $\bigcirc(\varphi \mathbf{R} \psi)$ may also occur. This is asymptotically harmless. Moreover, no additional reasoning for free variables is needed in the correctness proof.

► **Corollary 14.** *Let φ_0 be a formula of $\text{LTL}_f \times \mathbf{K}$. It is satisfiable over the class of quasi-temporal graphs iff there is a closed tableau for φ_0 .*

The crucial insight leading to an improved upper complexity bound is the observation that for $\text{LTL}_f \times K$ a nondeterministic, rather than alternating polynomial-space algorithm suffices. Then by Savitch's Theorem [31] we get a PSPACE upper bound. This may seem strange, though, because rule (Mod) and (Mod₀) still induce genuine branching in the tableaux, and nondeterministic choices also need to be made in rule (0r) and the rules for temporal operators. However, in an $\text{LTL}_f \times K$ formula, the argument of a modal operator is genuinely smaller in the sense that in an application of rule (Mod) or (Mod₀) to a premise containing φ one cannot “go back” to $\Diamond \varphi$ or $\Box \varphi$, at least not through φ .

► **Theorem 15.** *The satisfiability problem for $\text{LTL}_f \times K$ over QTGs is in PSPACE.*

Proof. An algorithm can nondeterministically check for satisfiability by successively applying rules whilst performing guesses for (0r), (U^-), (U^+), (R^-) and (R^+). No guess is needed for rule (And). When a formula set is reached that forms the end of a block, i.e. that requires an application of rule (Mod) or (Mod₀), we proceed as follows. Consider the case of (Mod), as the one for (Mod₀) is just a special case of that one. Suppose the number of premises for this rule application is k . Note that necessarily $k \leq n + 1$. The algorithm stores the $k - 1$ left premises and continues searching for a tableau for the rightmost premise. Upon successful termination it backtracks and continues with the next rightmost premise that has not been explored yet, re-using the space that was needed for the exploration of the previous premise.

Since the input formula is from $\text{LTL}_f \times K$, the recursion depth of this backtracking procedure is bounded: all but the leftmost premise of rule (Mod) have a modal depth that is strictly smaller than the modal depth of the conclusion. Hence, the call stack for the recursion in this algorithm has at most linear depth, and each stack frame has at most quadratic size, yielding an at most cubic space consumption. ◀

5 Conclusion

This paper studied the computational complexity of satisfiability for the two temporal-spatial logics $\text{LTL}_f \times K$ and $\mu\text{TL}_f \times K$, interpreted over quasi-temporal graphs. The results draw a slightly unexpected picture: while satisfiability for the underlying one-dimensional logics LTL_f and μTL_f is PSPACE-complete in either case, the two-dimensional combinations behave differently. $\text{LTL}_f \times K$ remains PSPACE-complete, whereas $\mu\text{TL}_f \times K$ becomes EXPTIME-complete, a complexity that is more reminiscent of branching-time temporal logics than of linear-time ones.

The use of QTGs rather than the more restrictive class of TGs is justified by the temporal-spatial bisimulation invariance of $\mu\text{TL}_f \times K$ established in Thm. 5. The class of TGs is not closed under bisimulation, which from a logical perspective makes QTGs the more natural class to consider. Nevertheless, satisfiability over the class of TGs is a meaningful problem in its own right, not least because TGs are the objects TGNNs reason about. TG-satisfiability is a strictly stronger condition than QTG-satisfiability as seen with a formula like $\Diamond \odot \text{ff} \wedge \bigcirc \text{tt}$ which is QTG-satisfiable, but neither satisfiable over synchronised QTGs nor TGs. Hence, the decision procedures do not directly transfer to the TG-setting. However, the lower bounds do. PSPACE-hardness for $\text{LTL}_f \times K$ and EXPTIME-hardness for $\mu\text{TL}_f \times K$ remain valid over the class of TGs. For matching upper bounds, the situation is more delicate. The tableau construction in Thm. 12 yields a QTG whose temporal successor functions are neither total nor injective. Thm. 6 provides a partial bridge by showing that every synchronised QTG can be unfolded into a bisimilar TG, but the formula above witnesses that this bridge cannot be extended to unsynchronised QTGs. We conjecture that TG-satisfiability is no harder than

QTG-satisfiability for either logic, with the upper bound obtained by an adaptation of the tableau calculus that enforces totality and injectivity of the temporal successor functions on closed branches. A precise proof of this, and the question of whether the formulation can be carried out without further blow-up in complexity, is left for future work.

The original motivation for studying these logics stems from their role as yardsticks for the expressive power of Temporal Graph Neural Networks. A potential next step is to replace the spatial logic K by stronger modal logics, such as $K^\#$ [1] or logics with Presburger-style counting [4], which are often considered in the expressiveness research of TGNNs [29, 5].

References

- 1 C. Areces, G. Hoffmann, and A. Denis. Modal logics with counting. In *Proc. 17th Workshop on Logic, Language, Information and Computation, WoLLIC'10*, volume 6188 of *LNCS*, pages 98–109. Springer, 2010. doi:10.1007/978-3-642-13824-9_9.
- 2 B. Banieqbal and H. Barringer. Temporal logic with fixed points. In *Proc. Coll. on Temporal Logic in Specification*, volume 398 of *LNCS*, pages 62–73. Springer, 1989. doi:10.1007/3-540-51803-7_22.
- 3 P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. L. Reutter, and J. P. Silva. The logical expressiveness of graph neural networks. In *Proc. 8th Int. Conf. on Learning Representations, ICLR'20*. OpenReview.net, 2020. URL: <https://openreview.net/forum?id=r1lZ7AEKvB>.
- 4 B. Bednarczyk, M. Orlowska, A. Pacanowska, and T. Tan. On classical decidable logics extended with percentage quantifiers and arithmetics. In *Proc. 41st IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'21*, volume 213 of *LIPIcs*, pages 36:1–36:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.36.
- 5 M. Benedikt, C.-H. Lu, B. Motik, and T. Tan. Decidability of graph neural networks via logical characterizations. In *Proc. 51st Int. Coll. on Automata, Languages, and Programming, ICALP'24*, volume 297 of *LIPIcs*, pages 127:1–127:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ICALP.2024.127.
- 6 J. Bradfield and C. Stirling. Modal mu-calculi. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic: Studies in Logic and Practical Reasoning Volume 3*, volume 3 of *Studies in logic and practical reasoning*, pages 721–756. North-Holland, 2007. doi:10.1016/S1570-2464(07)80015-2.
- 7 J. Bradfield and I. Walukiewicz. The mu-calculus and model checking. In *Handbook of Model Checking*, pages 871–919. Springer, 2018. doi:10.1007/978-3-319-10575-8_26.
- 8 F. Bruse, O. Friedmann, and M. Lange. On guarded transformation in the μ -calculus. *Logic Journal of the IGPL*, 23(2):194–216, 2015. doi:10.1093/jigpal/jzu030.
- 9 A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981. doi:10.1145/322234.322243.
- 10 B. S. Chlebus. Domino-tiling games. *Journal of Computer and System Sciences*, 32(3):374–392, 1986. doi:10.1016/0022-0000(86)90036-X.
- 11 M. Dam. Fixpoints of Büchi automata. In *Proc. 12th Conf. on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'92*, volume 652 of *LNCS*, pages 39–50. Springer, 1992. doi:10.1007/3-540-56287-7_93.
- 12 S. Demri, V. Goranko, and M. Lange. *Temporal Logics in Computer Science: Finite-State Systems*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016. doi:10.1017/CB09781139236119.
- 13 E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. Syst. Sci.*, 30(1):1–24, 1985. doi:10.1016/0022-0000(85)90001-7.
- 14 E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Inf. Control.*, 61(3):175–201, 1984. doi:10.1016/S0019-9958(84)80047-9.
- 15 M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979. doi:10.1016/0022-0000(79)90046-1.

- 16 D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyashev. *Many-dimensional modal logics: theory and applications*. Studies in Logic and the Foundations of Mathematics. Elsevier, 2003.
- 17 G. De Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proc. 23rd Int. Joint Conf. on A.I., IJCAI'13*, pages 854–860. IJCAI/AAAI, 2013. URL: <http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997>.
- 18 J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proc. 34th Int. Conf. on Machine Learning, ICML'17*, volume 70 of *Proc. Machine Learning Research*, pages 1263–1272. PMLR, 2017. URL: <http://proceedings.mlr.press/v70/gilmer17a.html>.
- 19 J. Johannsen and M. Lange. CTL⁺ is complete for double exponential time. In *Proc. 30th Int. Coll. on Automata, Logics and Programming, ICALP'03*, volume 2719 of *LNCS*, pages 767–775. Springer, 2003. doi:10.1007/3-540-45061-0_60.
- 20 R. Kaivola. A simple decision method for the linear time μ -calculus. In *Proc. Int. Workshop on Structures in Concurrency Theory, STRICT'95*, pages 190–204, 1995. doi:10.1007/978-1-4471-3078-9_13.
- 21 B. Knaster. Un théorème sur les fonctions d'ensembles. *Annals Soc. Pol. Math.*, 6:133–134, 1928.
- 22 D. Kozen. Results on the propositional μ -calculus. *Theor. Comput. Sci.*, 27:333–354, 1983. doi:10.1016/0304-3975(82)90125-6.
- 23 R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977. doi:10.1137/0206033.
- 24 Y. Liu, Z. Duan, C. Tian, and B. Cui. Satisfiability of linear time μ -calculus on finite traces. In *Proc. 22nd Int. Conf. on Computing and Combinatorics, COCOON'16*, volume 9797 of *LNCS*, pages 611–622. Springer, 2016. doi:10.1007/978-3-319-42634-1_49.
- 25 A. Longa, V. Lachi, G. Santin, M. Bianchini, B. Lepri, P. Lio, F. Scarselli, and A. Passerini. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities. *Trans. Mach. Learn. Res.*, 2023. doi:10.48550/arXiv.2302.01018.
- 26 C. Lutz, F. Wolter, and M. Zakharyashev. Temporal description logics: A survey. In *Proc. 15th Int. Symp. on Temporal Representation and Reasoning, TIME'08*, pages 3–14. IEEE, 2008. doi:10.1109/TIME.2008.14.
- 27 M. Marx and Y. Venema. *Multi-Dimensional Modal Logic*, pages 1–9. Springer, 1997. doi:10.1007/978-94-011-5694-3_1.
- 28 R. Mateescu. Local model-checking of modal μ -calculus on acyclic labeled transition systems. In *Proc. 8th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'02*, volume 2280 of *LNCS*, pages 281–295. Springer, 2002. doi:10.1007/3-540-46002-0_20.
- 29 P. Nunn, M. Sälzer, F. Schwarzentruher, and N. Troquard. A logic for reasoning about aggregate-combine graph neural networks. In *Proc. 33rd Int. Joint Conf. on A.I., IJCAI'24*, pages 3532–3540. ijcai.org, 2024. URL: <https://www.ijcai.org/proceedings/2024/391>.
- 30 M. Sälzer, P. A. Wałęga, and M. Lange. The logical expressiveness of temporal gnns via two-dimensional product logics. In *Proc. 39th Annual Conf. on Neural Information Processing Systems, NeurIPS'25*. Openreview.net, 2025. doi:10.48550/arXiv.2505.11930.
- 31 W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.
- 32 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.
- 33 J. Skarding, B. Gabrys, and K. Musiał. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*, 9:79143–79168, 2021. doi:10.1109/ACCESS.2021.3082932.
- 34 A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pacific Journal of Mathematics*, 5:285–309, 1955. doi:10.2140/pjm.1955.5.285.

- 35 M. Y. Vardi. A temporal fixpoint calculus. In *Proc. Conf. on Principles of Programming Languages, POPL'88*, pages 250–259. ACM, 1988. doi:10.1145/73560.73582.
- 36 M. Y. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *Proc. 17th Symp. on Theory of Computing, STOC'85*, pages 240–251. ACM, 1985. doi:10.1145/22145.22173.
- 37 I. Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ -calculus. *Inf. and Comput.*, 157(1-2):142–182, 2000. doi:10.1006/inco.1999.2836.

A Proof Details

Proof of Thm. 5. Let $\mathfrak{G} = (G_0, \sigma_0, \dots, \sigma_{n-2}, G_{n-1})$ and $\mathfrak{G}' = (G'_0, \sigma'_0, \dots, \sigma'_{n'-2}, G'_{n'-1})$ with $G_i = (S_i, \rightarrow_i, L_i)$ and $G'_j = (S'_j, \rightarrow'_j, L'_j)$.

“ $\Rightarrow$ ” By a slight abuse of notation, we write $\rightleftharpoons$ for the restriction of the bisimilarity relation to points in $\mathfrak{G}$ and $\mathfrak{G}'$. It is standard to argue that $\rightleftharpoons$ is the union of all bisimulation relations between $\mathfrak{G}$ and $\mathfrak{G}'$, and that it is itself a bisimulation between $\mathfrak{G}$ and $\mathfrak{G}'$.

Let $T \subseteq \bigcup_{i=0}^{n-1} S_i \times \{i\}$ and $T' \subseteq \bigcup_{j=0}^{n'-1} S'_j \times \{j\}$. We say that the pair (T, T') respects bisimilarity if for all $(v, i) \rightleftharpoons (u, j)$ we have $(v, i) \in T$ iff $(u, j) \in T'$. Hence, our goal is to show that for all $\varphi \in \mu\text{TL}_f \times \mathbf{K}$, the pair $(\llbracket \varphi \rrbracket^{\mathfrak{G}}, \llbracket \varphi \rrbracket^{\mathfrak{G}'})$ respects bisimilarity. In order to enable a proof by induction on the structure of formulas, we need to strengthen the claim to cover free variables. We say that a pair (ρ, ρ') of variable assignments for $\mathfrak{G}$ and $\mathfrak{G}'$ respectively respects bisimilarity if $(\rho(X), \rho'(X))$ respects bisimilarity for all $X \in \text{Var}$. We show that for all $\varphi \in \mu\text{TL}_f \times \mathbf{K}$ and all (ρ, ρ') that respect bisimilarity, we have that $(\llbracket \varphi \rrbracket_{\rho}^{\mathfrak{G}}, \llbracket \varphi \rrbracket_{\rho'}^{\mathfrak{G}'})$ respects bisimilarity.

The case $\varphi = p \in \text{AP}$ follows from condition (1) in Def. 4, and the case of $\varphi = X \in \text{Var}$ from the assumption about ρ and ρ' . The cases of the boolean operators $\wedge$ and $\neg$ follow directly from the hypothesis.

Let $\varphi = \bigcirc \psi$. Suppose that $(v, i) \rightleftharpoons (u, j)$ and $(v, i) \in \llbracket \bigcirc \psi \rrbracket_{\rho}^{\mathfrak{G}}$. Then $i < n - 1$, $\sigma_i(v)$ exists and $(\sigma_i(v), i + 1) \in \llbracket \psi \rrbracket_{\rho}^{\mathfrak{G}}$. By condition (4) in Def. 4 we have $j < n' - 1$, $\sigma'_j(u)$ exists and $(\sigma_i(v), i + 1) \rightleftharpoons (\sigma'_j(u), j + 1)$. By the induction hypothesis, $(\sigma'_j(u), j + 1) \in \llbracket \psi \rrbracket_{\rho'}^{\mathfrak{G}'}$ and thus $(u, j) \in \llbracket \bigcirc \psi \rrbracket_{\rho'}^{\mathfrak{G}'}$. The converse implication is treated symmetrically. Hence, $(\llbracket \varphi \rrbracket_{\rho}^{\mathfrak{G}}, \llbracket \varphi \rrbracket_{\rho'}^{\mathfrak{G}'})$ respects bisimilarity.

The case $\varphi = \Diamond \psi$ is treated similarly, using conditions (2) and (3) of Def. 4.

Finally, let $\varphi = \text{rec } X.\psi$. Suppose that $(v, i) \rightleftharpoons (u, j)$ and $(v, i) \in \llbracket \text{rec } X.\psi \rrbracket_{\rho}^{\mathfrak{G}}$. According to Lemma 3, we have $(v, i) \in \llbracket \text{rec}^n X.\psi \rrbracket_{\rho}^{\mathfrak{G}}$. The crucial insight here is that, while $\text{rec}^n X.\psi$ is generally larger than $\text{rec } X.\psi$ and not a subformula of it, $\text{rec}^n X.\psi$ does not contain X anymore. By a separate induction on the approximant level n we can infer that $(u, j) \in \llbracket \text{rec}^{n'} X.\psi \rrbracket_{\rho'}^{\mathfrak{G}'}$, making use of the main induction hypothesis for formulas not containing X anymore. A second application of Lemma 3 then yields $(u, j) \in \llbracket \text{rec } X.\psi \rrbracket_{\rho'}^{\mathfrak{G}'}$.

“ $\Leftarrow$ ” It suffices to show that $\equiv_{\mu\text{TL}_f \times \mathbf{K}}$ is a temporal bisimulation. Condition (1) is satisfied because propositions are formulas. Hence, suppose that the pair of points (v, i) and (u, j) could not be included because $L_i(v) \neq L'_j(u)$ then there is some $p \in \text{AP}$ such that $\mathfrak{G}, v, i \models p$ and $\mathfrak{G}', u, j \not\models p$ or vice-versa.

The reasoning for conditions (2) and (3) is the same as that for pure modal logic. Suppose that, say, (v, i) has a spatial successor (v', i) that cannot be matched by any successor (u', j) of (u, j) . Due to the co-inductive nature of bisimilarity, we can assume that there is a formula φ satisfied by (v', i) but not by any successor (u', j) of (u, j) . Then $\mathfrak{G}, v, i \models \Diamond \varphi$ but $\mathfrak{G}', u, j \not\models \Diamond \varphi$.

The reasoning for condition (4) is also done along the same line. Here the separating formula is of the form $\bigcirc \varphi$. ◀

Proof of Thm 12. “ $\Rightarrow$ ” Suppose that $(\mathfrak{G}, v_0, 0) \models \varphi_0$ for some QTG $\mathfrak{G} = (G_0, \sigma_0, \dots, \sigma_{n-2}, G_{n-1})$ with $G_i = (S_i, \rightarrow_i, L_i)$ and node $v \in S_0$.

We build a tableau for φ_0 , guided by nodes in $\mathfrak{G}$, starting with v_0 and the root node $\{\varphi_0\}$ of the tableau. Note that, by assumption, we have $(\mathfrak{G}, v_0, 0) \models \varphi_0$. The construction of the tableau will maintain the invariant that $(\mathfrak{G}, v, i) \models \varphi$ for all $\varphi \in \Phi$, provided that v is the current guide node, $v \in S_i$ and Φ is the current formula set at the end of a non-closed tableau branch. We call i the *height* of the tableau node.

Suppose that v is the current node, and the current node set Φ is not of the form $\bigcirc \Phi, \odot \Psi, \diamond \Delta, \square \Gamma, \Lambda$. Then it must contain a conjunction $\psi_1 \wedge \psi_2$, a fixpoint formula $\text{rec } X.\psi$, a variable X or a disjunction $\psi_1 \vee \psi_2$. In the first three cases we simply apply rule (And), (FP) or (Var) to deterministically obtain a new formula set that is the premise of the rule application. The current node remains to be v . It is easy to see that the invariant is maintained under rule (And).

For rules (FP) and (Var) this is less obvious because now the formula set may contain free variables. We refine the invariant as follows, thus also maintaining it with these two rules: if the current guide node is $v \in S_i$ and the current formula set is Φ then we have $(\mathfrak{G}, v, i) \models \varphi$ for all $\varphi \in \Phi$ where we interpret any variable $X \in \text{Var}$ as $\text{rec}^{n-i+1} X.\psi$ if $\text{fp}_{\varphi_0}(X) = \text{rec } X.\psi$.

At last, if Φ contains a disjunction $\psi_1 \vee \psi_2$ then by the invariant we have $(\mathfrak{G}, v, i) \models \psi_1 \vee \psi_2$. Hence, it is possible to choose one of the two disjuncts with rule (Or) to maintain the invariant.

Now suppose that the current node set is of the form $\bigcirc \Phi, \odot \Psi, \diamond \Delta, \square \Gamma, \Lambda$. Since $(\mathfrak{G}, v, i) \models \varphi$ for all φ in it, the following must be true.

- Λ must indeed be propositionally consistent because $q \in L_i(v)$ and $q \notin L_i(v)$ is impossible.
- If $\Phi \neq \emptyset$ then $i < n-1$ and $\sigma_i(v) \in S_{i+1}$, and for all $\varphi \in \Phi \cup \Psi$ we have $(\mathfrak{G}, \sigma_i(v), i+1) \models \varphi$.
- If $\Delta \neq \emptyset$ then for every $\varphi \in \Delta$ there is $u_\varphi \in S_i$ with $v \rightarrow_i u_\varphi$ and $(\mathfrak{G}, u_\varphi, i) \models \psi$ for all $\psi \in \{\varphi\} \cup \Gamma$.

This shows that rule (Mod) can be applied maintaining the invariant for new guide nodes $\sigma_i(v)$ and u_φ for $\varphi \in \Delta$. The reasoning for rule (Mod₀) is the same; just the second condition of the three becomes void.

This shows that, for as long as a tableau branch is not closed, the construction can be continued, guided by nodes in $\mathfrak{G}$. Temporal guardedness ensures that no branch of the tableau can have two applications of rule (Var) for the same variable without going through the leftmost premise of rule (Mod). On the other hand, (Var) is the only rule that increases the current formula set. Hence, the heights of nodes on every tableau branch must eventually be strictly increasing, but no node can have height $\geq n$. Thus, each branch must eventually be guided by a node with no temporal successor, closing the branch.

“ $\Leftarrow$ ” Suppose that φ_0 has a closed tableau. According to the observation made above, it can be collapsed into a tree of blocks. This tree can be rearranged as a quasi-temporal graph $\mathfrak{G} = (G_0, \sigma_0, G_1, \dots, G_n)$ by assigning each block as a node to graph G_i if the path from that block to the root block of the tableau contains i many steps through the leftmost premise of (Mod). The other connections between blocks obtained through non-leftmost premises of (Mod) or applications of (Mod₀) are translated into edges in the respective graphs. The temporal successors functions σ_i map any block to the block (necessarily in the next graph) that is started by the leftmost premise of the application of (Mod) at the end of the block.

One can then show, for $i = n, \dots, 0$ and every node v from G_i that is obtained as a collapsed block $\Phi_1, \dots, \Phi_k$ that $(\mathfrak{G}, v, i) \models \varphi$ for all $\varphi \in \Phi_1 \cup \dots \cup \Phi_k$. Again, we interpret a variable X occurring in a block forming a node in G_i as $\text{rec}^{n-i+1} X.\varphi$ when $\text{fp}_{\varphi_0}(X) = \text{rec } X.\varphi$. Consequently, since φ_0 is contained in the root block of the tableau, it has a model. ◀

Dynamic Quantitative Skill Distribution in Multi-Agent Temporal Planning as SMT

Josselin Guéron 

Université Caen Normandie, ENSICAEN, CNRS, Normandie Univ, GREYC UMR6072,
F-14000 Caen, France

Frédéric Maris 

Université de Toulouse, IRIT, France

Abstract

We introduce a new framework for multi-agent temporal planning, based on event-based temporal planning, dynamic team formation and Multi-Robot Task Allocation (MRTA). Unlike existing approaches, team formation is directly integrated into the planning and scheduling process. Then, both the teams and agents' skills may evolve over time. In particular, our framework allows us to address complex and dynamic multi-agent problems, such as cooperative task execution, where heterogeneous agents with qualitative and quantitative skills must cooperate dynamically. We propose an SMT encoding of these team temporal planning problems that utilizes the linear arithmetic theory over the rational numbers, QF-LRA. Next, to demonstrate that our approach is suitable for real-world applications, we present experimental results on disaster response problems.

2012 ACM Subject Classification Computing methodologies → Planning and scheduling

Keywords and phrases Multi-Agent Temporal Planning, Multi-Agent Task Allocation, SMT-based Planning, Dynamic Team Formation

Digital Object Identifier 10.4230/OASICS.TIME.2026.9

1 Introduction

The deployment of autonomous agents in complex, real-world environments, such as search-and-rescue operations, automated manufacturing, and space exploration, relies increasingly on Multi-Agent Systems (MAS). In these domains, tasks are often too complex, heavy, or specialized to be executed by a single agent. Consequently, heterogeneous agents must cooperate by forming temporary groups to fulfill specific mission requirements. This challenge falls at the intersection of Multi-Agent Task Allocation (MRTA), dynamic team formation and temporal planning.

A critical aspect of modeling heterogeneous agents lies in accurately capturing their capabilities. In this paper, we distinguish between two fundamental types of skills. Qualitative skills represent static, innate capabilities (e.g. possessing a specific sensor) that an agent either masters or does not. Conversely, quantitative skills represent measurable, bounded capacities (e.g. lifting capacity or battery level) that can be aggregated among agents and consumed or produced over time.

Furthermore, real-world tasks are rarely instantaneous. They are dynamic processes that unfold over time, requiring different skills at different stages and inducing specific effects at precise moments. While existing literature on task-driven team formation extensively covers the assignment of agents to tasks based on skill matching, it often abstracts away the fine-grained temporal constraints of task execution. On the other hand, expressive temporal planning frameworks handle complex temporal and numeric constraints but traditionally struggle with the combinatorial explosion of dynamically forming teams. Encoding such heavily numeric and temporal problems into classical boolean Satisfiability (SAT) is notoriously inefficient due to the discretization of time and resources.



© Josselin Guéron and Frédéric Maris;

licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 9; pp. 9:1–9:20

OpenAccess Series in Informatics



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

To bridge this gap, this paper introduces a novel framework for Multi-Agent Temporal Planning Problems. We propose an event-based temporal task model that seamlessly integrates dynamic team formation with fine-grained temporal and numeric reasoning. By leveraging Satisfiability Modulo Theories (SMT), we map the discrete logic of qualitative skill alignment to the boolean engine, while delegating the continuous temporal constraints and quantitative resource aggregations to specialized arithmetic theory solvers.

Specifically, the main contributions of this paper are the following:

- We formalize a temporal multi-agent planning domain encompassing heterogeneous agent classes with qualitative and quantitative skills.
- We extend classical event-based temporal semantics to support task-driven team formation, defining strict applicability conditions based on skill requirements and temporal effects.
- We present an efficient SMT encoding that natively resolves the complex interaction between discrete team assignments and the numeric constraints of temporal execution, validated by experimental results on disaster response problems.

This paper is organized as follows: Section 2 covers related work across SMT, temporal planning, MRTA, and dynamic team formation. Section 3 formalizes the multi-agent temporal planning problem, while Section 4 describes its SMT encoding. Section 5 evaluates the framework through experiments in a disaster response domain. Finally, Section 6 concludes this work and outlines future research directions.

2 Related Work

2.1 Satisfiability Modulo Theory (SMT)

Satisfiability Modulo Theories (SMT) extends the classic boolean Satisfiability (SAT) problem by interpreting symbols with respect to background theories, such as Linear Integer Arithmetic (LIA) or Linear Real Arithmetic (LRA) [1]. Traditional SAT solvers strictly operate on propositional formulas, while SMT solvers can natively evaluate logical formulas containing complex numeric and arithmetic constraints (e.g. $x + y \leq z$). Although SAT is widely used in classical automated planning, SAT-based approaches inherently struggle with two major aspects of real-world dynamic domains: temporal flexibility (continuous or dense time) and numeric resources [14]. This limitation arises from the severe combinatorial explosion caused by discretizing large numeric domains, quantitative capacities, or real-valued timepoints into purely boolean variables.

SMT overcomes these limitations by delegating arithmetic reasoning to specialized theory solvers, allowing for scalable reasoning over continuous time and quantitative state variables. In the context of temporal and numeric planning, SMT has been successfully used to encode highly expressive formalisms such as PDDL2.1. Approaches like SMTPlan+ [4] and works in temporal planning [13] demonstrate that SMT is particularly well-suited for event-based planning. In these encodings, real-valued or integer variables represent the exact occurrence times of events (e.g. τ_i) and the levels of numeric resources. The temporal relationships between tasks are efficiently encoded using Difference Logic, while resource consumptions and productions are modeled using Linear Arithmetic.

Our multi-agent temporal planning problem aligns perfectly with the capabilities of SMT. The event-based framework introduced in the previous sections relies simultaneously on strict logical prerequisites and cumulative numeric constraints. Encoding this model into SMT allows us to jointly and efficiently solve two interdependent problems:

- The discrete logic of task assignments and team formation based on qualitative skill matching, natively handled by the boolean SAT engine.
- The arithmetic aggregation of quantitative skills (e.g. ensuring team resources meet task requirements) and the resolution of the relative temporal constraints over event times, natively handled by the LRA theory solver.

2.2 Event-based Temporal Planning

To model durative actions and continuous time, we rely on an event-based temporal planning framework similar to that given in [6]. Rather than treating actions as atomic units with fixed durations, this framework decomposes each action into a set of instantaneous *events* (e.g. starting an action, establishing a fact, or ending a condition) scheduled over a temporal horizon.

State variables representing properties of the environment that evolve over time are modeled as boolean *fluents*. An action (or action-instance) a is defined by its constituent events, denoted $Events(a)$, which interact with fluents in two ways:

- Effects: The event $a \rightarrow f$ establishes fluent f (making it true), while $a \rightarrow \neg f$ destroys f (making it false).
- Conditions: The notations $f \mid \rightarrow a$ and $f \rightarrow \mid a$ represent the exact start and end timepoints, respectively, of the continuous interval over which fluent f must remain true for action a to be valid.

A real-valued function $\tau(E) \in \mathbb{R}$ assigns the absolute execution time to any event E . The internal temporal structure of an action a is governed by relative temporal constraints $Constr(a)$. For any pair of events $e_1, e_2 \in Events(a)$, their relative time difference is bounded by an interval $[\alpha_a(e_1, e_2), \beta_a(e_1, e_2)]$, imposing that $\tau(e_1) - \tau(e_2) \in [\alpha_a(e_1, e_2), \beta_a(e_1, e_2)]$. This mechanism naturally models variable action durations and internal event orderings.

A valid temporal plan must satisfy two fundamental categories of constraints:

1. Inherent Constraints: Every action instance $a \in A$ must respect its internal timing rules defined by $Constr(a)$.
2. Contradictory-Effects Constraints: To prevent physical inconsistencies, no two events occurring at the exact same instant can establish and destroy the same fluent. Formally, for any pair of action instances $a_i, a_j \in A$ and any fluent $f \in Del(a_i) \cap Add(a_j)$, we require that $\tau(a_i \rightarrow \neg f) \neq \tau(a_j \rightarrow f)$.

► **Definition 1** (Temporal Planning Problem). A temporal planning problem is a tuple $\langle I, A, G \rangle$ where I is the initial set of true fluents, G is the set of goal fluents, and A is the set of actions.

► **Definition 2** (Temporal Plan). A temporal plan for a problem $\langle I, A, G \rangle$ is a pair $P = \langle A, \tau \rangle$, where A is a finite set of action-instances $\{a_1, \dots, a_n\}$ and τ is a real-valued function on $Events(A)$, is a temporal plan for the problem $\langle I, A', G \rangle$ if (1) $A \subseteq A'$, and (2) P satisfies the inherent and contradictory-effect constraints on A ; and when P is executed (i.e. fluents are established or destroyed at the times given by τ) starting from the initial state I : (3) for all $a_i \in A$, each $f \in Cond(a_i)$ is true when it is required, and (4) all goal fluents $g \in G$ are true at the end of the execution of P .

Events are instantaneous, whereas actions are not only durative but may also be of variable length. Thus a temporal plan P does not schedule its action-instances directly but schedules all the events in its action-instances.

2.3 Multi-Robot Task Allocation

Multi-Robot Task Allocation (MRTA) is a coordination problem where a group of agents must efficiently assign themselves to tasks in order to perform them.

► **Definition 3** (MRTA framework). *Formally, the MRTA framework is defined in the literature as a tuple $\langle N, T, u \rangle$, such that:*

- $N = \{r_1, r_2, \dots, r_n\}$, a set of n robots.
- $T = \{t_1, t_2, \dots, t_m\}$, a set of m tasks.
- $u : N \times T \rightarrow \mathbb{R}$, a utility function that assigns the value of performing a task to a robot. The resulting value u_{ij} represents the utility gained (or cost incurred, as negative utility) from robot r_i completing task t_j . In the extended case where tasks require multiple robots, this utility function is typically generalized over subsets of agents: $u : 2^N \times T \rightarrow \mathbb{R}$.

The more standard formalism of MRTA defines this problem in a “one-to-one” way: one agent can be assigned to one task, and one task can be assigned by one agent. However, as defined in [9], there is a comprehensive taxonomy allowing us to describe more complex types of problems. This taxonomy is defined following three axes:

- Single-task (ST) vs Multi-task (MT) robots: ST means that one robot can only perform one task simultaneously, while MT means that one robot can perform one or more task simultaneously.
- Single-robot (SR) vs Multi-robot (MR) tasks: SR means that a task can be performed by a single agent, while MR means that multiple robots can be required to perform the task.
- Instantaneous (IA) vs Time-extended (TA) assignment: IA means that we can only create an assignment for the next execution phase, without future planning. TA means that we can schedule assignments in advance, before sequential and/or asynchronous executions.

According to this taxonomy, the challenge addressed in this paper, i.e. of coordinating a fleet of heterogeneous agents to execute complex tasks through an automated planning approach, falls into the ST-MR-TA category.

2.4 Dynamic Team Formation

While classic MRTA algorithms excel at optimizing utility and task scheduling, they often treat the required groups as black boxes. To address the inner mechanics of group composition, the Multi-Agent Systems (MAS) literature proposed the field of *Dynamic Team Formation* [16, 8]. Dynamic team formation focuses on cooperative, short-lived and task-driven alliances between agents. This framework is highly relevant for critical and high-uncertainty domains such as Search and Rescue (SAR), where teams of heterogeneous robots must be formed on-the-fly to combine complementary capabilities against emerging disaster events [11]. In these scenarios, teams dissolve immediately upon task completion so that individual assets can be fluidly redistributed to other pressing sub-goals [8].

To ensure the feasibility of these dynamic teams, the allocation framework must explicitly model the capabilities of heterogeneous agents. In the literature, this is achieved by matching agent profiles against task requirements using two distinct paradigms:

- Qualitative Skill Alignment: Tasks specify a set of mandatory operational roles or technical skills (e.g. search capability, specific sensor). A formed team is valid if and only if the joint capabilities of its assigned members satisfy the qualitative prerequisites of the task, ensuring that all required roles are covered [5].

- **Quantitative Resource Aggregation:** Beyond binary skills or roles, agents possess bounded, numeric resources (e.g. payload capacity, battery power). The quantitative capability of a team is modeled as the cumulative sum of the individual resources contributed by its members, which must satisfy or exceed the task's dynamic requirement threshold [15].

Despite the maturity of both MRTA and dynamic team formation, a significant gap remains at their intersection with formal temporal planning. Most existing team formation algorithms operate under large-scale temporal abstractions, treating tasks as blocks with static durations and fixed resource costs. Conversely, expressive temporal planners can schedule complex timelines but struggle with the combinatorial explosion induced by the dynamism of heterogeneous teams. Our work bridges this gap by unifying dynamic, skill-driven team formation with an event-based temporal planning framework, leveraging the efficiency of SMT solvers to handle the underlying numeric and logical constraints.

3 Multi-Agent Temporal Planning

We address a Multi-Agent Temporal Planning Problem. Briefly, this involves multi-agent planning within a dynamic and temporal environment, where the cooperation between heterogeneous agents is required to successfully execute a plan.

3.1 Agents and Mastered Skills over Time

► **Definition 4 (Environment).** *Let us define the multi-agent environment of our problem as a tuple $\mathcal{E} = \langle N, \mathbb{T} \rangle$, where $N = \{n_1, \dots, n_m\}$ is a set of m agents and $\mathbb{T} \in \{\mathbb{Z}, \mathbb{Q}\}$ represents the temporal domain of the environment (discrete or continuous).*

Agents and tasks are characterized by a set of skills K , partitioned into qualitative and quantitative types based on their temporal evolution and aggregation properties.

► **Definition 5 (Skills).** *Let $K = K_{ql} \cup K_{qt}$ be the set of all available skills, defined as the disjoint union of qualitative (boolean) skills $K_{ql} = \{k_1, \dots, k_l\}$ and quantitative (measurable) skills $K_{qt} = \{k_{l+1}, \dots, k_{l+l'}\}$.*

► **Example 6 (Running domain).** Throughout this section, we illustrate the definitions on a simplified Disaster Response domain with $K_{ql} = \{\text{Fire extinguishing}\}$ and $K_{qt} = \{\text{Water Capacity}\}$.

To model agent heterogeneity, we group agents into stationary *classes* (e.g. operational roles) with fixed profiles of mastered skills that cannot be acquired or lost over time.

► **Definition 7 (Agent Class).** *An agent class $w \in \mathcal{W}$ is defined as a tuple $w = \langle N^w, K_{ql}^w, K_{qt}^w, P_{qt}^w \rangle$ where:*

- $N^w \subseteq N$ is a subset of agents that belongs to the class w ,
- $K_{ql}^w \subseteq K_{ql}$ is a subset of qualitative skills that agents in this class have mastered,
- $K_{qt}^w \subseteq K_{qt}$ is a subset of quantitative skills that agents in this class have mastered,
- $P_{qt}^w : K_{qt}^w \rightarrow \mathbb{Q}_+$ is a capacity function mapping each quantitative skill $k \in K_{qt}^w$ to its maximum capacity for an agent of class w .

► **Example 8 (Agent Profile).** In a Disaster Response scenario, an agent class $w = \text{Firefighting Drone}$ could be defined with $K_{ql}^w = \{\text{Fire extinguishing}\}$, and a bounded quantitative skill $K_{qt}^w = \{\text{Water Capacity}\}$ with $P_{qt}^w(\text{Water Capacity}) = 100L$.

Qualitative skills represent static capabilities required to participate in a task. They are evaluated via a class-dependent indicator function.

► **Definition 9** (Qualitative skills function). *Let $m^{01} : N \times K_{qt} \rightarrow \{0, 1\}$ be the indicator function determining whether an agent $n \in N^w$ of class $w \in \mathcal{W}$ masters a skill $k \in K_{qt}$:*

$$m^{01}(n, k) = \begin{cases} 1 & \text{if } k \in K_{qt}^w \\ 0 & \text{otherwise} \end{cases}$$

► **Example 10** (Qualitative instantiation). For n a *Firefighting Drone*, $m^{01}(n, \text{Fire extinguishing}) = 1$, while $m^{01}(n, k) = 0$ for any other qualitative skill $k \in K_{qt}$.

Accordingly, $m_n^{01} = \{k \in K_{qt} \mid m^{01}(n, k) = 1\}$ denotes the skills of agent n , and $m_C^{01} = \bigcup_{n \in C} m_n^{01}$ defines the qualitative skills aggregated by a team of agents $C \subseteq N$ via set union. Conversely, quantitative skills describe bounded capacities (e.g. resource) whose operational values can dynamically evolve but are bounded by the agent's class capacity.

► **Definition 11** (Quantitative skills function). *Let $m^Q : N \times K_{qt} \times \mathbb{T} \rightarrow \mathbb{Q}_+$ be the dynamic function determining the level of a quantitative skill for agent $n \in N_w$ of class $w \in \mathcal{W}$ at time $\tau \in \mathbb{T}$:*

$$\forall k \in K_{qt}, \quad 0 \leq m^Q(n, k, \tau) \leq \begin{cases} P_{qt}^w(k) & \text{if } k \in K_{qt}^w \\ 0 & \text{otherwise} \end{cases}$$

► **Example 12** (Quantitative instantiation). At time τ , a *Firefighting Drone* n with a half-full tank has $m^Q(n, \text{Water Capacity}, \tau) = 50 \leq P_{qt}^w(\text{Water Capacity}) = 100$.

The current profile of agent n is denoted $m_n^Q(\tau) = \{(k, m^Q(n, k, \tau)) \mid k \in K_{qt}^w \wedge m^Q(n, k, \tau) > 0\}$, and the aggregated quantitative skills of a team $C \subseteq N$ are modeled via arithmetic sum: $m_C^Q(\tau) = \{(k, v) \mid k \in K_{qt}, v = \sum_{n \in C} m^Q(n, k, \tau)\}$.

3.2 Event-Based Tasks

We extend the event-based temporal planning framework in [6], representing plans as networks of primitive events occurring at specific timepoints. We use a placeholder dot ($\cdot$) to denote any entity (e.g. a task t or Nature) interacting with the environment state.

► **Definition 13** (Primitive Events). *Let $f \in \mathcal{F}$ be a fluent, and $k_{qt} \in K_{qt}$, and $k_{qt} \in K_{qt}$ with a value $v \in \mathbb{Q}_+$ respectively a qualitative and a quantitative skill. We define the following primitive events:*

- *Fluent events:* $\cdot \rightarrow f$ (establish f), $\cdot \rightarrow \neg f$ (destroy f), and $f \rightarrow \cdot$ (require f).
- *Skill events:* $k_{qt} \rightarrow \cdot$ (require k_{qt}), $(k_{qt} \geq v) \rightarrow \cdot$ (require capacity v), $\cdot \rightarrow k_{qt}(+v)$ (produce capacity v), and $\cdot \rightarrow k_{qt}(-v)$ (consume capacity v).

► **Example 14** (Primitive events). The task $t = \text{Extinguish Fire}$ requires the qualitative skill via $\text{Fire extinguishing} \rightarrow t$, requires an aggregated capacity of 150L via $(\text{Water Capacity} \geq 150) \rightarrow t$, and later consumes it via $t \rightarrow \text{Water Capacity}(-150)$.

When a specific task $t \in T$ triggers an event, the placeholder $\cdot$ is replaced by t . Tasks are modeled as collections of temporal requirements and effects over an inherent timeline.

► **Definition 15** (Task Events). *For a task $t \in T$ and a relative timepoint $\tau \in \mathbb{T}$ at which the event occurs, we define the following profile functions:*

- *Fluent requirement:* $\text{cond}_f : T \times \mathcal{F} \times \mathbb{T} \rightarrow \{0, 1\}$ indicates if $f \in \mathcal{F}$ is necessary (1) or not (0).

- *Qualitative requirement:* $r^{01} : T \times K_{ql} \times \mathbb{T} \rightarrow \{0, 1\}$ indicates if $k \in K_{ql}$ is necessary (1) or not (0).
- *Quantitative requirement:* $r^Q : T \times K_{qt} \times \mathbb{T} \rightarrow \mathbb{Q}_+$ specifies the minimum capacity needed for $k \in K_{qt}$.
- *Fluent effect:* $\text{eff}_f : T \times \mathcal{F} \times \mathbb{T} \rightarrow \{0, 1\}$ indicates whether fluent f is established (1) or destroyed (0).
- *Quantitative production:* $\text{add}_{qt} : T \times K_{qt} \times \mathbb{T} \rightarrow \mathbb{Q}_+$ represents the capacity produced.
- *Quantitative consumption:* $\text{del}_{qt} : T \times K_{qt} \times \mathbb{T} \rightarrow \mathbb{Q}_+$ represents the capacity consumed.

► **Example 16** (Task profile instantiation). For $t = \text{Extinguish Fire}$: at $\tau = 0$, $r^{01}(t, \text{Fire extinguishing}, 0) = 1$ and $r^Q(t, \text{Water Capacity}, 0) = 150$; at $\tau = 2$, $\text{del}_{qt}(t, \text{Water Capacity}, 2) = 150$ (the capacity is consumed once the fire is extinguished).

We map the occurrence times of these structural conditions and effects directly to the execution plan. Let $\tau_i(e)$ denote the relative timepoint of the i -th instance of event e (simplified to $\tau(e)$ if it occurs exactly once).

► **Definition 17** (Event Mapping). Let $t \in T$ be a temporal task. The relationship between the syntactic events of t and its temporal profile functions is defined as follows:

$$\begin{aligned}
 \tau(f \rightarrow t) = \tau_i &\iff \text{cond}_f(t, f, \tau_i) = 1 \\
 \tau(k_{ql} \rightarrow t) = \tau_i &\iff r^{01}(t, k_{ql}, \tau_i) = 1 \\
 \tau((k_{qt} \geq v) \rightarrow t) = \tau_i &\iff r^Q(t, k_{qt}, \tau_i) \geq v \\
 \tau(t \rightarrow f) = \tau_i &\iff \text{eff}_f(t, f, \tau_i) = 1 \\
 \tau(t \rightarrow \neg f) = \tau_i &\iff \text{eff}_f(t, f, \tau_i) = 0 \\
 \tau(t \rightarrow k_{qt}(+v)) = \tau_i &\iff \text{add}_{qt}(t, k_{qt}, \tau_i) = v \\
 \tau(t \rightarrow k_{qt}(-v)) = \tau_i &\iff \text{del}_{qt}(t, k_{qt}, \tau_i) = v
 \end{aligned}$$

Internal temporal constraints Constr_t are expressed as relations over these event times (e.g. $\tau((k_{qt} \geq v) \rightarrow t) \leq \tau(t \rightarrow k_{qt}(-v))$ ensures that capacity verification precedes consumption). Tasks can only be executed if the assigned team jointly satisfies these prerequisites.

► **Definition 18** (Executable Task Condition). A task $t \in T$ is executable by a team $C \subseteq N$ at time $\tau \in \mathbb{T}$ if and only if it satisfies both:

- *Qualitative validity:* $\forall k \in K_{ql}, r^{01}(t, k, \tau) = 1 \implies k \in m_C^{01}$
- *Quantitative validity:* $\forall k \in K_{qt}, r^Q(t, k, \tau) \leq \sum_{n \in C} m^Q(n, k, \tau)$

We denote $T_C(\tau) \subseteq T$ as the subset of tasks that are executable by the team C at time τ .

► **Example 19** (Task Execution). Consider a task $t = \text{Extinguish Fire}$. It requires the qualitative skill $r^{01}(t, \text{Fire Extinguishing}, \tau) = 1$ to be able spread water, and a quantitative aggregate capacity $r^Q(t, \text{Water Capacity}, \tau) \geq 150L$. A single *Firefighting Drone* (max 100L) cannot execute this task alone. The Executable Task Condition forces the dynamic formation of a team C containing at least two drones to satisfy the 150L threshold, alongside an agent providing the *Fire Extinguishing* capability.

► **Definition 20** (Temporal State). The dynamic state of the environment at time $\tau \in \mathbb{T}$ is a tuple $S(\tau) = \langle \mathcal{C}(\tau), m^Q(\tau), F(\tau) \rangle$, where $\mathcal{C}(\tau)$ is the active team structure (a partition of N), $m^Q(\tau)$ tracks individual quantitative skill levels, and $F(\tau) \subseteq \mathcal{F}$ is the set of true fluents.

3.3 Temporal Task Model and Team Formation Process

To formalize the problem, we first define the overall model, which encapsulates agents, skills, and tasks.

► **Definition 21** (Temporal Task Model). A Temporal Task Model (*TTM*) is a tuple $\mathcal{M} = \langle \mathcal{W}, N, T, K, \mathcal{F}, \mathbb{T} \rangle$ where $\mathcal{W}$ is the set of agent classes, N is the set of m agents partitioned into $\mathcal{W}$, T is the set of temporal tasks, $K = K_{ql} \cup K_{qt}$ is the partitioned set of skills, $\mathcal{F}$ is the set of propositional fluents, and $\mathbb{T}$ is the temporal domain.

As previously mentioned, in our environment, a task is not instantaneous. Its conditions and effects are tied to relative timepoints τ representing the progression of the task.

► **Definition 22** (Temporal Task). A temporal task $t \in T$ is a tuple $t = \langle E_t, Constr_t \rangle$, where $E_t = \langle Cond_t, Eff_t \rangle$ encapsulates the task events, and $Constr_t$ represents the internal synchronization and duration constraints over its relative timepoints τ :

- Task Conditions ($Cond_t$):
 - $cond_f(t, f, \tau) \in \{0, 1\}$ indicates whether fluent $f \in \mathcal{F}$ must be true (1) at timepoint τ .
 - $r^{01}(t, k, \tau) \in \{0, 1\}$ indicates if qualitative skill $k \in K_{ql}$ is required.
 - $r^Q(t, k, \tau) \in \mathbb{Q}_+$ specifies the minimum required aggregate level of quantitative skill $k \in K_{qt}$.
- Task Effects (Eff_t):
 - $eff_f(t, f, \tau) \in \{0, 1\}$ dictates whether fluent f is established (1) or destroyed (0) at timepoint τ .
 - $add_{qt}(t, k, \tau) \in \mathbb{Q}_+$ and $del_{qt}(t, k, \tau) \in \mathbb{Q}_+$ respectively define the quantitative skill amounts produced or consumed at timepoint τ .

► **Definition 23** (Team Formation Process). Let $\mathcal{M}$ be a *TTM*. A Team Formation Process (*TFP*) is a tuple $p = \langle C, X, \mathcal{T}_b, \mathcal{T}_d, \mathcal{T}_e \rangle$ where:

- $C \subseteq N$ is the subset of agents allocated to the process.
- $X \subseteq T$ is the subset of tasks to be executed by the team.
- $\mathcal{T}_b \leq \mathcal{T}_d \leq \mathcal{T}_e$ are three ordered global timepoints representing the assembly phase ($\mathcal{T}_b$), the effective deployment ($\mathcal{T}_d$), and the dissolution of the team ($\mathcal{T}_e$).

For any task $t \in X$, $\mathcal{T}_b^t$ and $\mathcal{T}_e^t$ denote its specific start and end execution times, satisfying $\mathcal{T}_d \leq \mathcal{T}_b^t \leq \mathcal{T}_e^t \leq \mathcal{T}_e$.

► **Example 24** (Team Formation Process). Continuing the running example, a TFP $p = \langle \{n_1, n_2\}, \{Extinguish\ Fire\}, 0, 1, 3 \rangle$ assembles two *Firefighting Drones* n_1, n_2 at $\mathcal{T}_b = 0$, deploys them as a unified team from $\mathcal{T}_d = 1$, during which they jointly satisfy the 150L requirement of *Extinguish Fire* ($100L + 100L \geq 150L$), and dissolves the team at $\mathcal{T}_e = 3$.

A TFP is applicable in a state $S(\mathcal{T}_b)$ if all participating agents are initially free, i.e. they belong to singleton sets in the active team structure: $\forall n_i \in C, \{n_i\} \in \mathcal{C}(\mathcal{T}_b)$. Once triggered, the process enforces the following structural state transitions:

- $\forall n_i \in C, \{n_i\} \in \mathcal{C}(\tau)$ is maintained for all states during the assembly interval $\tau \in [\mathcal{T}_b, \mathcal{T}_d]$;
- $C \in \mathcal{C}(\tau)$ represents the unified operational team for all states during execution $\tau \in [\mathcal{T}_d, \mathcal{T}_e]$;
- $\forall n_i \in C, \{n_i\} \in \mathcal{C}(\mathcal{T}_e)$ models the instant dissolution of the team back into individual assets.

3.4 Multi-Agent Temporal Planning Problem

► **Definition 25** (Team Temporal Planning Problem). A Team Temporal Planning Problem (TTPP) is a tuple $\langle \mathcal{M}, I, G \rangle$ where $\mathcal{M}$ is a Temporal Task Model (TTM), $I = \langle C_0, m_0^Q, F_0 \rangle$ is the initial temporal state, and $G \subseteq \mathcal{F}$ the set of goal fluents.

To ensure physical feasibility, the scheduling of tasks and TFPs must strictly satisfy three *Exclusivity Constraints*:

- Agent exclusivity: An agent can participate in at most one active TFP at any given time. For any two concurrent TFPs sharing an agent, their global temporal intervals $[\mathcal{T}_b, \mathcal{T}_e]$ must not overlap.
- Task exclusivity: A specific task instance can be executed by at most one TFP simultaneously. For any two instances of the same task t_1, t_2 executed by two different TFPs, their respective execution intervals $[\mathcal{T}_b^{t_1}, \mathcal{T}_e^{t_1}]$ and $[\mathcal{T}_b^{t_2}, \mathcal{T}_e^{t_2}]$ must not overlap.
- Team exclusivity: A formed team executes exactly one task at a time. For any two tasks t_1, t_2 executed by the same TFP, their respective execution intervals $[\mathcal{T}_b^{t_1}, \mathcal{T}_e^{t_1}]$ and $[\mathcal{T}_b^{t_2}, \mathcal{T}_e^{t_2}]$ must not overlap.

Consequently, agent's skills are inherently dedicated to a single task at any given moment, naturally preventing the simultaneous use of its skills for two tasks in parallel without requiring an explicit skill exclusivity constraint.

Furthermore, any valid plan must respect *Inherent Constraints* (the internal timing bounds Constr_t within each task) and *Contradictory-Effects Constraints*. Since qualitative skills are static in our model, contradictory effects only apply to fluents and quantitative capacities: two primitive events occurring at the exact same instant τ cannot assign conflicting boolean values to the same fluent $f \in \mathcal{F}$, nor assign diverging absolute values to the same quantitative skill level of an agent.

► **Definition 26** (Team Temporal Plan). Let $\langle \mathcal{M}, I, G \rangle$ be a TTPP. A Team Temporal Plan (TTP) is a tuple $\Pi = \langle \mathcal{P}, \tau \rangle$, where $\mathcal{P} = \{p_1, \dots, p_n\}$ is a finite set of TFPs and τ is a rational-valued scheduling function mapping every primitive event to a specific timepoint. Π is a valid temporal plan for the TTPP if the following conditions are met:

1. *Consistency*: $\mathcal{P}$ strictly satisfies all inherent, contradictory-effect and exclusivity constraints over $\mathcal{M}$.
2. *Execution Validity*: When executing Π from the initial state I (i.e. fluents and quantitative skills are updated at the precise times given by τ):
 - a. For all executed tasks, required conditions $f \in \mathcal{F}$ hold true over their specified requirement intervals.
 - b. For every task t executed by a team C at time τ_t , the team satisfies the Executable Task Conditions (i.e. C covers all required qualitative skills r^{01} and continuously provides sufficient aggregated quantitative capacities r^Q).
 - c. All goal fluents $g \in G$ are true at the end of the execution of Π .

4 Multi-Agent Temporal Planning as SMT

To encode our temporal multi-agent problem into SMT, we must first define the variables representing the timeline of events. Unlike standard SAT encodings that discretize time into fixed steps, our SMT approach uses rational-valued variables to represent exact timepoints. First, we encode the chronological validity of the TFPs. A TFP cannot occur instantaneously, it follows a strict sequence of stages. For any given TFP, its initialization must precede its execution phase (but can not precede the initial timepoint), and finally conclude at its end

time. We also define the *virtual event* $\tau(n \dashrightarrow p)$ corresponding to a past timepoint τ where we establish that agent n is tied to p and can not be used for another TFP. To enforce this structure, we impose the following constraint for all TFPs:

$$\bigwedge_{p \in \mathcal{P}} \left((\mathcal{T}_{Init} - \mathcal{T}_b^p \leq 0) \wedge (\mathcal{T}_b^p - \mathcal{T}_d^p \leq 0) \wedge (\mathcal{T}_d^p - \mathcal{T}_e^p \leq 0) \right)$$

By formulating these temporal relations as difference logic inequalities (e.g. $x - y \leq 0$), we ensure that the SMT solver's arithmetic theory engine can efficiently resolve the temporal bounds without combinatorial explosion.

$$\bigwedge_{\substack{p_1 \in \mathcal{P} \\ n \in N}} \left(\left((n \in C^{p_1}) \Rightarrow \left(\tau(n \dashrightarrow p_1) - \mathcal{T}_b^{p_1} \leq 0 \right) \wedge \left(\bigvee_{\substack{p_2 \in \mathcal{P} \\ p_1 \neq p_2}} \left((n \in C^{p_2}) \wedge (\tau(n \dashrightarrow p_1) - \mathcal{T}_e^{p_2} = 0) \right) \vee \left(\tau(n \dashrightarrow p_1) - \mathcal{T}_{Init} = 0 \right) \right) \right) \right)$$

$$\bigwedge_{\substack{n \in N \\ p_1, p_2 \in \mathcal{P} \\ p_1 \neq p_2}} \left(\left((n \in C^{p_1}) \wedge (n \in C^{p_2}) \Rightarrow \left(\tau(n \dashrightarrow p_1) - \mathcal{T}_e^{p_2} \geq 0 \right) \vee \left(\tau(n \dashrightarrow p_2) - \mathcal{T}_e^{p_1} \geq 0 \right) \right) \right)$$

To track the continuous evolution of quantitative skills, our encoding relies on two mechanisms: local updates via a pass-through logic, and global causal links protected by formulas φ_1 and φ_2 below. On the one hand, the pass-through mechanism acts as a frame axiom. For any task event, if an agent is not part of the assigned team ($\neg(n \in C^p)$), the solver enforces a strict equality between their before and after skill values ($val_{AddB}^{p,t,k,n} = val_{AddA}^{p,t,k,n}$). The event is mathematically transparent to them, preventing unintended resource modifications.

$$\bigwedge_{\substack{p \in \mathcal{P} \\ t \in T}} \left((t \in X^p) \Rightarrow \left(\bigwedge_{(t \rightarrow k_{qt}(+v)) \in E_t} \left(\left((u_{|N|}^{p,t,k} = 0) \wedge (u_0^{p,t,k} = v) \right) \wedge \bigwedge_{i \in \{1, \dots, |N|\}} \left(u_{i-1}^{p,t,k} - u_i^{p,t,k} - val_{AddA}^{p,t,k,n_{i-1}} + val_{AddB}^{p,t,k,n_{i-1}} = 0 \right) \right) \wedge \left(\neg(n \in C^p) \Rightarrow \left(val_{AddA}^{p,t,k,n} - val_{AddB}^{p,t,k,n} = 0 \right) \right) \wedge \left(val_{AddB}^{p,t,k,n} - val_{AddA}^{p,t,k,n} \leq 0 \right) \wedge \bigwedge_{\substack{n \in N \\ w \in \mathcal{W} \\ n \in N^w \\ k \in K_{qt}^w}} \left(val_{AddA}^{p,t,k,n} - P_{qt}^w(k) \leq 0 \right) \right) \right)$$

On the other hand, for the participating agents, we must link their current before state to a valid past event. This is achieved through virtual timepoints and the two sub-formulas: φ_1 (Causal Support) that forces the solver to find a preceding event (an effect of a past task, or the initial state) whose after value exactly matches the current before requirement, effectively establishing a causal link. φ_2 (Causal Link Protection) that resolves potential threats by ensuring that no other event modifying this agent's specific skill can be scheduled inside the interval between the supporting past event and the current task event. Any interfering task must be constrained to occur before the support or after the current event. As before, we use again a virtual event for each agent n , for example $\tau^{p,n}(t \dashrightarrow k_{qt}(+v))$, to reserve in the past the amount of contribution $val_{AddB}^{p,t,k,n}$ of skill k from agent n for the task t timepoint $\tau^p(t \rightarrow k_{qt}(+v))$.

$$\begin{aligned}
& \bigwedge_{\substack{p \in \mathcal{P} \\ t \in T}} \left((t \in X^p) \Rightarrow \bigwedge_{\substack{n \in N \\ (t \rightarrow k_{qt}(+v)) \in E_t}} (\tau^{p,n}(t \dashrightarrow k_{qt}(+v)) - \tau^p(t \rightarrow k_{qt}(+v)) \leq 0 \wedge \varphi_1 \wedge \varphi_2) \right) \\
\varphi_1 = & \left(\bigvee_{p_2 \in \mathcal{P}} \left(\bigvee_{\substack{t_2 \in T \\ (t_2 \rightarrow k_{qt}(+v_2)) \in E_{t_2} \\ (p \neq p_2 \vee t \neq t_2)}} \left(\begin{aligned} & (t_2 \in X^{p_2}) \\ & \wedge (val_{AddB}^{p,t,k,n} - val_{AddA}^{p_2,t_2,k,n} = 0) \\ & \wedge (\tau^{p,n}(t \dashrightarrow k_{qt}(+v)) - \tau^{p_2}(t_2 \rightarrow k_{qt}(+v_2)) = 0) \end{aligned} \right) \right) \right. \\
& \left. \vee \bigvee_{\substack{t_2 \in T \\ (t_2 \rightarrow k_{qt}(-v_2)) \in E_{t_2}}} \left(\begin{aligned} & (t_2 \in X^{p_2}) \\ & \wedge (val_{AddB}^{p,t,k,n} - val_{DelA}^{p_2,t_2,k,n} = 0) \\ & \wedge (\tau^{p,n}(t \dashrightarrow k_{qt}(+v)) - \tau^{p_2}(t_2 \rightarrow k_{qt}(-v_2)) = 0) \end{aligned} \right) \right) \right. \\
& \left. \vee ((val_{AddB}^{p,t,k,n} = val_{Init}^{k,n}) \wedge (\tau^{p,n}(t \dashrightarrow k_{qt}(+v)) - \mathcal{T}_{Init} = 0)) \right) \\
\varphi_2 = & \bigwedge_{p_3 \in \mathcal{P}} \left((n \in C^{p_3}) \Rightarrow \left(\begin{aligned} & \bigwedge_{\substack{t_3 \in T \\ (t_3 \rightarrow k_{qt}(-v_3)) \in E_{t_3} \\ (p \neq p_3 \vee t \neq t_3)}} \left(\begin{aligned} & (t_3 \in X^{p_3}) \Rightarrow \\ & \tau^{p_3}(t_3 \rightarrow k_{qt}(-v_3)) - \tau^{p,n}(t \dashrightarrow k_{qt}(+v)) \leq 0 \\ & \vee \tau^p(t \rightarrow k_{qt}(+v)) - \tau^{p_3}(t_3 \rightarrow k_{qt}(-v_3)) \leq 0 \end{aligned} \right) \\ & \wedge \bigwedge_{\substack{t_3 \in T \\ (t_3 \rightarrow k_{qt}(+v_3)) \in E_{t_3} \\ (p \neq p_3 \vee t \neq t_3)}} \left(\begin{aligned} & (t_3 \in X^{p_3}) \Rightarrow \\ & \tau^{p_3}(t_3 \rightarrow k_{qt}(+v_3)) - \tau^{p,n}(t \dashrightarrow k_{qt}(+v)) \leq 0 \\ & \vee \tau^p(t \rightarrow k_{qt}(+v)) - \tau^{p_3}(t_3 \rightarrow k_{qt}(+v_3)) \leq 0 \end{aligned} \right) \end{aligned} \right) \right)
\end{aligned}$$

This causal linking mechanism is not limited to quantitative skills, but serves as a universal foundation for our encoding. We apply an identical virtual event formulation to manage the logical consistency of fluents and qualitative skills. For any required propositional condition, the solver instantiates a virtual event that binds the requirement to a valid past establishing event (such as a previous task's effect or the initial state). Simultaneously, the encoding secures this causal chain by ensuring that any conflicting event (e.g. a deletion effect) is strictly isolated outside the protected temporal interval. Consequently, both the numeric and propositional states of the environment remain strictly consistent throughout the timeline. All additional encoding rules are given in Appendix A.

In terms of expressivity, our model captures the numeric and temporal semantics of PDDL2.1, naturally mapping quantitative skills to **increase**, **decrease**, and **assign** effects. While encoding such expressiveness in SAT causes combinatorial explosion through time and resource discretization, our SMT formulation over Quantifier-Free Linear Real Arithmetic (QF-LRA) natively handles continuous variables without artificial discretization. Theoretically, concurrent planning is EXPSpace-complete over discrete time ($\mathbb{T} = \mathbb{Z}$) [12] and generally undecidable over dense time ($\mathbb{T} = \mathbb{Q}$) [10]. However, by bounding the maximum number of TFPs, our encoding restricts the continuous state space to a finite, polynomially bounded set of events. This reduces our specific planning task to a bounded scheduling problem. Since the satisfiability of QF-LRA is NP-complete, our bounded formulation fundamentally lowers the theoretical complexity. To guarantee that our SMT formulation accurately captures the semantics of the Multi-Agent Temporal Planning Problem, we state the following theorem.

► **Theorem 27** (Soundness and Completeness). *Let $\langle \mathcal{M}, I, G \rangle$ be a Team Temporal Planning Problem (TTPP). Any satisfying assignment produced by the SMT encoding corresponds to a valid Team Temporal Plan Π for this TTPP (Soundness). Conversely, if a valid plan Π exists for the TTPP within the bounded number of TFPs, the SMT encoding is satisfiable (Completeness).*

The complete formal proof of this theorem is given in Appendix B.

5 Experimental Evaluation

To evaluate our approach, we model a disaster response scenario, which describes a highly uncertain environment requiring heterogeneous agents to dynamically form teams to execute complex, time-critical tasks (e.g. firefighting, rescuing victims).

5.1 Domain Description and Setup

Our domain defines five innate *qualitative skills* (Search, Rescue, Fire extinguishing, Clear, Mark) and two *quantitative skills* (Water supply and Strength, bounded in $[0, 100]$). To solve the tasks, we define five heterogeneous agent classes, summarized in Table 1.

■ **Table 1** Agent classes and their respective qualitative and quantitative skill profiles.

Class	Type	Qualitative Skills	Quantitative Skills
Firefighter	Human	Fire ext., Rescue, Mark	Water supply, Strength
Reconnaissance Drone	UAV	Search, Mark	-
Firefighting Drone	UAV	Fire ext.	Water supply
Ground Support Drone	UGV	Rescue	Water supply
High-risk Inspection Drone	UGV	Search	Strength

To conduct a robust evaluation, temporal tasks are generated using randomized structural profiles. Tasks can be either *production tasks* (20% probability), establishing 1 to 3 new fluents with no fluent prerequisites, or as *transition tasks* (80%), requiring 1 to $|F|/2$ fluents and establishing 1 to $|F|/2$ new ones, with a 50% chance of consuming each initial condition. Qualitative skills have a 30% probability of being required. Quantitative skills have a 30% chance of being required (sampled uniformly in $[10, 100]$). If a capacity is required, it has a 50% chance of being consumed; otherwise, it has a 5% chance of being restored (sampled in $[10, 100]$). Additionally, dedicated refueling tasks are explicitly generated to replenish specific quantitative capacities. Finally, every task condition and effect is assigned a randomized relative timepoint to accurately simulate complex internal task synchronizations.

5.2 Protocol and metrics

We generated benchmarks varying along three axes: the number of agents $|N| \in [3, 10]$ (uniformly distributed among classes), tasks $|T| \in \{10, 15, 20\}$, and fluents $|F| \in \{6, 12\}$. Since our event-based SMT formulation requires an upper bound on the number of Team Formation Processes (TFPs) to unfold the timeline, we applied a dichotomic search strategy to find the minimal feasible plan. For $|N| \leq 7$, the search explored up to 8 TFPs with a 300s timeout per instance. For $|N| \geq 8$, the bound was extended to 10 TFPs with a 600s timeout.

Instances were encoded into SMT using TouIST [7] and solved with Yices on an Intel Core i7-1370P (32GB RAM). We recorded the translation time (T_{trans}), solving time (T_{solve}), total execution time (T_{exec}), number of teams formed ($|\mathcal{TF}|$), number of executed tasks ($|X|$), and makespan.

It is crucial to note that the **Success Rate** reported in our results strictly isolates non-trivial SAT instances. The remaining percentage encompasses not only classical solver timeouts and proven UNSAT instances, but also trivially solvable scenarios where the goal state was immediately satisfied by the initial state or by uncontrolled environment events (Nature) without requiring any active team formation.

It is important to emphasize why direct empirical comparisons with existing temporal-numeric planners (e.g. OPTIC [2], SMTPlan+ [4] or ARIES [3]) are omitted. Standard PDDL-based and temporal-numeric planners do not natively support dynamic team formation. Attempting to encode our TTP into classical temporal-numeric languages would require explicitly grounding all possible $2^{|N|}$ team combinations as individual actions. This artificial translation induces a combinatorial explosion before the temporal scheduling and resource aggregation begins.

5.3 Results

Tables 2 and 3 summarize the average metrics for successful runs. Preliminary observations indicated that the number of logical fluents exerts a marginal impact compared to the combinatorial explosion of team assignments, thus, our analysis focuses on $|N|$ and $|T|$.

■ **Table 2** Average performance metrics grouped by number of agents ($|N|$). Times in seconds.

$ N $	Success Rate	T_{trans}	T_{solve}	T_{exec}	$ \mathcal{TF} $	$ X $	Makespan
3	29.0%	0.17	2.34	2.51	1.45	4.00	27.64
4	30.7%	0.18	2.75	2.93	1.45	4.28	30.37
5	31.7%	0.20	3.91	4.11	1.20	3.74	25.32
6	32.0%	0.20	5.03	5.23	1.48	4.83	33.99
7	36.0%	0.28	1.20	1.48	1.30	3.57	27.55
8	33.3%	0.49	2.06	2.55	1.30	3.31	21.94
9	26.7%	0.48	6.35	6.84	1.41	3.85	30.07
10	33.0%	0.59	2.64	3.24	1.31	3.60	28.26

■ **Table 3** Average performance metrics grouped by number of tasks ($|T|$). Times in seconds.

$ T $	Success Rate	T_{trans}	T_{solve}	T_{exec}	$ \mathcal{TF} $	$ X $	Makespan
10	27.4%	0.18	2.24	2.43	1.38	3.25	21.05
15	31.0%	0.34	2.96	3.30	1.35	3.95	29.13
20	36.3%	0.42	4.11	4.53	1.35	4.31	32.43

As observed, the SMT encoding demonstrates remarkable efficiency. The translation time (T_{trans}) remains negligible, consistently staying well below one second, which highlights the structural compactness of our TouIST encoding rules.

The solving time (T_{solve}) is highly competitive. Across all configurations, the solver successfully navigates the complex interplay between dynamic capability matching and dense temporal constraints in an average of 2 to 6 seconds. While scaling the problem naturally induces combinatorial growth, the framework comfortably manages up to 10 heterogeneous agents and 20 temporal tasks within acceptable planning windows.

Furthermore, analyzing the generated plans reveals an interesting structural behavior: the average number of formed teams ($|\mathcal{TF}| \approx 1.3$) is above 1 which means that, on average, several TFPs are created, but the fact that the average number is between 1 and 2 also indicates that the solver actively leverages the agents' combined skills by packing multiple sequential tasks ($|X| \approx 4$) into a single, long-lived team rather than frequently dissolving and reassembling different groups. This behavior is fundamentally tied to the nature of standard

SMT solvers, which are designed to find the first logically valid assignment (satisfiability) rather than the globally optimal one. Without an explicit objective function to minimize the makespan, the solver lacks the mathematical incentive to parallelize tasks across multiple simultaneous teams. Extending this framework to support optimization metrics to actively minimize the makespan represents a highly promising avenue for future work.

5.4 Discussion on Expressive Capabilities

To clarify the contribution of our model independently of the underlying solver, it is essential to position its expressive capabilities against existing temporal planning frameworks such as PDDL2.1. While standard action-based languages are highly expressive and could theoretically model dynamic team formation, doing so requires severe and inefficient structural workarounds. To represent a cooperative task, a modeler would have to either explicitly ground a distinct action for every possible subset of agents (causing an immediate $2^{|N|}$ combinatorial explosion in the domain definition itself), or introduce intricate sequences of intermediate “join-team” and “leave-team” actions. The latter approach artificially inflates the plan length and severely complicates the temporal synchronization of concurrent effects.

Our framework offers a fundamentally different representation that avoids these pitfalls by treating dynamic team formation and capability aggregation as first-order concepts. The expressive advantage of our model does not lie in bypassing classical decidability limits, but in its ability to natively and compactly represent cooperative task execution: a task dynamically dictates its qualitative and quantitative requirements over time, and a valid team emerges to fulfill them simultaneously. This provides a significantly more natural representation for complex Multi-Robot Task Allocation (MRTA) scenarios, by clearly decoupling the temporal logic of the tasks from the combinatorial complexity of agent assignments.

6 Conclusion and Perspectives

In this paper, we have introduced a new framework for multi-agent temporal planning that effectively integrates dynamic team formation with event-based temporal reasoning. Our approach overcomes the traditional limitations of SAT encoding for complex timelines and also unifies the dynamics of team formation with the arithmetic complexity of quantitative skill allocation through SMT encoding. Our experiments on disaster response scenarios demonstrate that our unified encoding is both expressive and computationally scalable, successfully coordinating heterogeneous agents to execute complex and distributed tasks. This work establishes a robust foundation for multi-agent temporal planning where cooperative team structures are not predefined, but emerge dynamically as a direct result of the temporal scheduling process.

In this article, we focus on a framework describing only two classes of skills, with fixed domains and aggregation functions. Future work may include the generalization of classes, with a higher-level definition of their domains and aggregation functions. Thus, this would enable us to use our model to define a wider range of problems in a flexible manner.

The experiments presented in this work focused on finding valid plans, while optimization was performed manually by adjusting the number of TFPs in order to identify the minimum achievable makespan for a given problem instance. However, some TFPs may be created and certain tasks executed even though they are not strictly necessary for the plan to be valid. A promising direction for future work would be to investigate multi-objective optimization approaches that jointly consider the makespan, the number of TFPs, and the number of executed tasks. Such optimization could notably be addressed using another solver such as Z3 through the SMT-LIB formalism.

References

- 1 Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of model checking*, pages 305–343. Springer, 2018. doi:10.1007/978-3-319-10575-8_11.
- 2 J. Benton, Amanda Jane Coles, and Andrew Coles. Temporal planning with preferences and time-dependent continuous costs. In Lee McCluskey, Brian Charles Williams, José Reinaldo Silva, and Blai Bonet, editors, *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*. AAAI, 2012. URL: <http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4699>.
- 3 Arthur Bit-Monnot. Experimenting with Lifted Plan-Space Planning as Scheduling: Aries in the 2023 IPC. In *Proceedings of the 2023 International Planning Competition*, Prague, Czech Republic, July 2023.
- 4 Michael Cashmore, Maria Fox, Derek Long, and Daniele Magazzeni. A compilation of the full PDDL+ language into SMT. In *Proceedings of the international conference on automated planning and scheduling*, volume 26, pages 79–87, 2016. URL: <http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13101>.
- 5 Luiz Chaimowicz, Mario F.M. Campos, and Vijay Kumar. Dynamic role assignment for cooperative robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 293–298. IEEE, 2002.
- 6 Martin C. Cooper, Frederic Maris, and Pierre Régnier. Monotone temporal planning: Tractability, extensions and applications. *J. Artif. Intell. Res. (JAIR)*, 50:447–485, 2014. doi:10.1613/JAIR.4358.
- 7 Jorge H. Fernandez, Olivier Gasquet, Andreas Herzig, Dominique Longin, Emiliano Lorini, Frédéric Maris, and Pierre Régnier. Touist: a friendly language for propositional logic and more. In *29th International Joint Conference on Artificial Intelligence (IJCAI 2020)*, pages 5240–5242. International Joint Conferences on Artificial Intelligence Organization, 2020. doi:10.24963/IJCAI.2020/756.
- 8 Matthew E. Gaston and Marie DesJardins. Agent-organized networks for dynamic team formation. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 230–237, 2005. doi:10.1145/1082473.1082508.
- 9 Brian P. Gerkey and Maja J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International journal of robotics research*, 23(9):939–954, 2004. doi:10.1177/0278364904045564.
- 10 Nicola Gigante, Andrea Micheli, Angelo Montanari, and Enrico Scala. Decidability and complexity of action-based temporal planning over dense time. *Artif. Intell.*, 307:103686, 2022. doi:10.1016/J.ARTINT.2022.103686.
- 11 Tyler Gunn and John Anderson. Dynamic heterogeneous team formation for robotic urban search and rescue. *Journal of Computer and System Sciences*, 81(3):553–567, 2015. doi:10.1016/J.JCSS.2014.11.009.
- 12 Jussi Rintanen. Complexity of concurrent temporal planning. In Mark S. Boddy, Maria Fox, and Sylvie Thiébaux, editors, *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, pages 280–287. AAAI, 2007. URL: <http://www.aaai.org/Library/ICAPS/2007/icaps07-036.php>.
- 13 Jussi Rintanen. Discretization of temporal models with application to planning with SMT. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.
- 14 Jussi Rintanen. Planning and SAT. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 765–789. IOS Press, 2021. doi:10.3233/FAIA201003.
- 15 Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial intelligence*, 101(1-2):165–200, 1998. doi:10.1016/S0004-3702(98)00045-9.
- 16 Milind Tambe. Towards flexible teamwork. *Journal of artificial intelligence research*, 7:83–124, 1997. doi:10.1613/JAIR.433.

A Additional Encoding Rules (Complete Encoding)

This section details the exhaustive SMT encoding rules applied to generate the valid Team Temporal Plans. To ensure readability, formulas use the mathematical notations introduced in Section 3 and Section 4. Let $\tau^p(t)$ denote the base reference timepoint of task t within the TFP p . $\mathcal{T}_b^{p,t}$ and $\mathcal{T}_e^{p,t}$ represent the specific start and end temporal execution boundaries of task t within p . For quantitative aggregations, $u_i^{p,t,k}$ represents the cumulative sum variable over the agents.

Structural Constraints

Agents cannot belong to a Team Formation Process (TFP) if it does not execute any task:

$$\bigwedge_{p \in \mathcal{P}} \bigwedge_{n \in N} \left((n \in C^p) \Rightarrow \bigvee_{t \in T} (t \in X^p) \right)$$

All TFPs must strictly start after the initial chronological timepoint $\mathcal{T}_{Init}$:

$$\bigwedge_{p \in \mathcal{P}} \left(\mathcal{T}_{Init} - \mathcal{T}_b^p \leq 0 \right)$$

If a task t is executed within a TFP p , its precise temporal bounds $\mathcal{T}_b^{p,t}$ and $\mathcal{T}_e^{p,t}$ are restrained within the TFP execution phase $[\mathcal{T}_d^p, \mathcal{T}_e^p]$. Furthermore, every primitive event $e \in E_t$ must strictly occur within the task boundaries:

$$\bigwedge_{p \in \mathcal{P}} \bigwedge_{t \in T} \left((t \in X^p) \Rightarrow \left((\mathcal{T}_d^p - \mathcal{T}_b^{p,t} \leq 0) \wedge (\mathcal{T}_e^p - \mathcal{T}_e^{p,t} \geq 0) \right) \wedge \bigwedge_{e \in E_t} \left((\mathcal{T}_b^{p,t} - \tau^p(e) \leq 0) \wedge (\mathcal{T}_e^{p,t} - \tau^p(e) \geq 0) \right) \right)$$

Inherent Internal Task Constraints ($Constr_t$)

For every executed task t in a TFP p , the relative occurrence of its condition and effect events must respect the structural offset bounds Δ_t defined in its domain profile:

Fluents Constraints:

$$\bigwedge_{p \in \mathcal{P}} \bigwedge_{t \in T} \left(\begin{array}{l} \bigwedge_{(f \rightarrow t) \in E_t} \left(\tau^p(f \rightarrow t) - \tau^p(t) = \Delta_t(f \rightarrow t) \right) \\ \wedge \bigwedge_{(t \rightarrow f) \in E_t} \left(\tau^p(t \rightarrow f) - \tau^p(t) = \Delta_t(t \rightarrow f) \right) \\ \wedge \bigwedge_{(t \rightarrow \neg f) \in E_t} \left(\tau^p(t \rightarrow \neg f) - \tau^p(t) = \Delta_t(t \rightarrow \neg f) \right) \end{array} \right)$$

Quantitative Skills Constraints:

$$\bigwedge_{p \in \mathcal{P}} \bigwedge_{t \in T} \left(\begin{array}{l} \bigwedge_{((k_{qt} \geq v) \rightarrow t) \in E_t} \left(\tau^p((k_{qt} \geq v) \rightarrow t) - \tau^p(t) = \Delta_t((k_{qt} \geq v) \rightarrow t) \right) \\ \wedge \bigwedge_{(t \rightarrow k_{qt}(+v)) \in E_t} \left(\tau^p(t \rightarrow k_{qt}(+v)) - \tau^p(t) = \Delta_t(t \rightarrow k_{qt}(+v)) \right) \\ \wedge \bigwedge_{(t \rightarrow k_{qt}(-v)) \in E_t} \left(\tau^p(t \rightarrow k_{qt}(-v)) - \tau^p(t) = \Delta_t(t \rightarrow k_{qt}(-v)) \right) \end{array} \right)$$

Qualitative Skills Constraints:

$$\bigwedge_{p \in \mathcal{P}} \bigwedge_{t \in T} \bigwedge_{(k_{ql} \rightarrow t) \in E_t} \left(\tau^p(k_{ql} \rightarrow t) - \tau^p(t) = \Delta_t(k_{ql} \rightarrow t) \right)$$

Qualitative Capability Checking

Tasks requiring qualitative skills k_{ql} must be executed by a team where at least one assigned agent natively masters the specific skill:

$$\bigwedge_{p \in \mathcal{P}} \bigwedge_{t \in T} \left((t \in X^p) \Rightarrow \bigwedge_{(k_{ql} \rightarrow t) \in E_t} \bigvee_{\substack{n \in N \\ k_{ql} \in m_n^{01}}} (n \in C^p) \right)$$

Causality and Protection over Propositional Fluents (F)

Goal Fluents (G): For each goal $f \in G$, its virtual requirement event $\tau^{req}(G, f)$ must occur at or before $\mathcal{T}_{Goal}$. It must be supported by a preceding production effect or the initial state I . Any deletion of f must explicitly occur strictly before the protection interval starts.

$$\bigwedge_{f \in G} \left(\begin{array}{l} \left(\tau^{req}(G, f) - \mathcal{T}_{Goal} \leq 0 \right) \\ \wedge \left(\bigvee_{p_1 \in \mathcal{P}} \bigvee_{\substack{t_1 \in T \\ (t_1 \rightarrow f) \in E_{t_1}}} \left((t_1 \in X^{p_1}) \wedge \left(\tau^{req}(G, f) - \tau^{p_1}(t_1 \rightarrow f) = 0 \right) \right) \right) \\ \vee \left\{ \begin{array}{ll} \left(\tau^{req}(G, f) - \mathcal{T}_{Init} = 0 \right) & \text{if } f \in I \\ \perp & \text{otherwise} \end{array} \right. \\ \wedge \bigwedge_{p_2 \in \mathcal{P}} \bigwedge_{\substack{t_2 \in T \\ (t_2 \rightarrow \neg f) \in E_{t_2}}} \left((t_2 \in X^{p_2}) \Rightarrow \tau^{p_2}(t_2 \rightarrow \neg f) - \tau^{req}(G, f) < 0 \right) \end{array} \right)$$

Task Fluents Conditions ($CondF$): Any task condition f must similarly be supported by a valid past event. Any conflicting deletion must be scheduled entirely outside the resulting causal protection link.

$$\bigwedge_{p_2 \in \mathcal{P}} \bigwedge_{t_2 \in T} \left((t_2 \in X^{p_2}) \Rightarrow \begin{array}{l} \left(\tau^{p_2, req}(f \dashrightarrow t_2) - \tau^{p_2}(f \rightarrow t_2) \leq 0 \right) \\ \wedge \left(\bigvee_{p_1 \in \mathcal{P}} \bigvee_{\substack{t_1 \in T \\ (t_1 \rightarrow f) \in E_{t_1}}} \left((t_1 \in X^{p_1}) \wedge \left(\tau^{p_2, req}(f \dashrightarrow t_2) - \tau^{p_1}(t_1 \rightarrow f) = 0 \right) \right) \right) \\ \vee \left\{ \begin{array}{ll} \left(\tau^{p_2, req}(f \dashrightarrow t_2) - \mathcal{T}_{Init} = 0 \right) & \text{if } f \in I \\ \perp & \text{otherwise} \end{array} \right. \\ \wedge \bigwedge_{p_3 \in \mathcal{P}} \bigwedge_{\substack{t_3 \in T \\ (t_3 \rightarrow \neg f) \in E_{t_3}}} \left((t_3 \in X^{p_3}) \Rightarrow \left(\tau^{p_3}(t_3 \rightarrow \neg f) - \tau^{p_2, req}(f \dashrightarrow t_2) < 0 \right) \vee \left(\tau^{p_2}(f \rightarrow t_2) - \tau^{p_3}(t_3 \rightarrow \neg f) < 0 \right) \right) \end{array} \right)$$

Quantitative Constraints and Causal Link Logic

The encoding of quantitative skills relies on two symmetrical mechanisms for each event type (Addition, Consumption, and Condition): the intra-task capacity aggregation, and the inter-task causal link protection. As the complete formulas for capacity production ($AddQt$)

are already detailed in Section 4, we define the consumption (*DelQt*) and requirement (*CondQt*) constraints by highlighting their specific deviations from the *AddQt* base formulas to maintain brevity.

1. Task Quantitative Aggregation: The aggregation logic iterates over all agents to sum their individual contributions $u_i^{p,t,k}$.
 - For Requirements (*CondQt*), the exact matching $u_0^{p,t,k} = v$ is relaxed to a lower bound $u_0^{p,t,k} \geq v$. Furthermore, since a condition does not consume the resource, the before/after state variables (val_{AddA}, val_{AddB}) are replaced by a single contribution variable $val_{Cond}^{p,t,k,n}$. For non-participating agents, we enforce $val_{Cond} = 0$.
 - For Consumptions (*DelQt*), the logic mirrors *AddQt* but reverses the before/after dynamics. The sequence enforces $u_{i-1}^{p,t,k} - u_i^{p,t,k} - val_{DelB}^{p,t,k,n} + val_{DelA}^{p,t,k,n} = 0$. For participating agents, we ensure non-negativity and strict value tracking: $val_{DelB} - val_{DelA} \geq 0$ and $val_{DelA} \geq 0$. Non-participating agents pass their values through unchanged ($val_{DelB} = val_{DelA}$).
2. Virtual Causal Events and Protection: To guarantee causal consistency, every individual contribution ($val_{Cond}^{p,t,k,n}$ or $val_{DelB}^{p,t,k,n}$) must be anchored to a past establishing event without any contradictory events occurring in between. The formulas are strictly identical to the φ_1 (Causal Support) and φ_2 (Causal Protection) formulas provided for *AddQt* in Section 4, subject to the following variable substitutions:
 - For *CondQt*: The virtual timepoint $\tau^{p,n}(t \dashrightarrow k_{qt}(+v))$ is replaced by a condition virtual timepoint (e.g. $\tau^{p,n}((k_{qt} \geq v) \dashrightarrow t)$), and the target variable $val_{AddB}^{p,t,k,n}$ is replaced by $val_{Cond}^{p,t,k,n}$.
 - For *DelQt*: The virtual timepoint $\tau^{p,n}(t \dashrightarrow k_{qt}(+v))$ is replaced by $\tau^{p,n}(t \dashrightarrow k_{qt}(-v))$, and the target variable $val_{AddB}^{p,t,k,n}$ is replaced by $val_{DelB}^{p,t,k,n}$.

Exclusivity Constraints (Mutex)

Agent Exclusivity: An agent n assigned to two distinct TFPs ensures strict sequentiality between the deployment intervals.

$$\bigwedge_{n \in N} \bigwedge_{\substack{p_1 \in \mathcal{P} \\ p_2 \in \mathcal{P} \\ p_1 \neq p_2}} \bigwedge_{p_2 \in \mathcal{P}} \left((n \in C^{p_1}) \wedge (n \in C^{p_2}) \Rightarrow (\mathcal{T}_b^{p_1} - \mathcal{T}_e^{p_2} \geq 0) \vee (\mathcal{T}_b^{p_2} - \mathcal{T}_e^{p_1} \geq 0) \right)$$

Task Exclusivity: Instances of the same task cannot be concurrently executed across distinct TFPs.

$$\bigwedge_{p_1 \in \mathcal{P}} \bigwedge_{\substack{p_2 \in \mathcal{P} \\ p_1 \neq p_2}} \bigwedge_{t \in T} \left((t \in X^{p_1}) \wedge (t \in X^{p_2}) \Rightarrow (\mathcal{T}_b^{p_1,t} - \mathcal{T}_e^{p_2,t} \geq 0) \vee (\mathcal{T}_b^{p_2,t} - \mathcal{T}_e^{p_1,t} \geq 0) \right)$$

Team Exclusivity: Tasks executed consecutively within the same TFP must not internally overlap.

$$\bigwedge_{p \in \mathcal{P}} \bigwedge_{\substack{t_1 \in T \\ t_2 \in T \\ t_1 \neq t_2}} \left((t_1 \in X^p) \wedge (t_2 \in X^p) \Rightarrow (\mathcal{T}_b^{p,t_1} - \mathcal{T}_e^{p,t_2} \geq 0) \vee (\mathcal{T}_b^{p,t_2} - \mathcal{T}_e^{p,t_1} \geq 0) \right)$$

B Proof of the Soundness and Completeness Theorem

In this section, we provide the formal proof of Theorem 27 establishing the equivalence between the satisfiability of our SMT encoding and the existence of a valid Team Temporal Plan for a given Team Temporal Planning Problem (TTPP).

► **Theorem 27** (Soundness and Completeness). *Let $\langle \mathcal{M}, I, G \rangle$ be a Team Temporal Planning Problem (TTPP). Any satisfying assignment produced by the SMT encoding corresponds to a valid Team Temporal Plan Π for this TTPP (Soundness). Conversely, if a valid plan Π exists for the TTPP within the bounded number of TFPs, the SMT encoding is satisfiable (Completeness).*

Proof. The proof is divided into two parts: establishing soundness by formally extracting a valid plan from a satisfying SMT assignment, and establishing completeness by mapping a valid plan back to a satisfying assignment.

Soundness. Let $\mathcal{I}$ be a satisfying assignment (a valid model) returned by the SMT solver for the encoded formula over the QF-LRA theory, generated for the specific TTPP $\langle \mathcal{M}, I, G \rangle$. We construct a Team Temporal Plan $\Pi = \langle \mathcal{P}, \tau \rangle$ using the extraction mapping defined in Section A. We must prove that Π satisfies all conditions of a valid temporal plan for this TTPP (Definition 26):

- *Structural Consistency and Exclusivity:* Because $\mathcal{I}$ strictly satisfies the structural and mutex difference-logic constraints, all task bounds $[\mathcal{T}_b^{p,t}, \mathcal{T}_e^{p,t}]$ are appropriately ordered and enclosed within their respective TFP deployment phases. Furthermore, for any two active TFPs sharing an agent or executing the same task, the temporal intervals are strictly disjoint. Thus, Π is physically plausible and respects all exclusivity constraints.
- *Inherent Constraints:* The SMT formulation enforces precise arithmetic offsets $\tau^p(e) - \tau^p(t) = \Delta_t(e)$. Consequently, the relative timing of all primitive events within every scheduled task in Π exactly matches the domain definition $Constr_t$.
- *Execution Validity and Causal Consistency:* The pass-through logic (e.g., $u_{i-1}^{p,t,k} - u_i^{p,t,k} - val_{Cond} = 0$) mathematically guarantees that the total capacity required by a task is exactly the sum of the individual contributions of the assigned agents in Π . Crucially, for any temporal requirement (whether a propositional fluent or a quantitative resource) scheduled at τ_{req} , the causal support formula φ_1 forces $\mathcal{I}$ to anchor a virtual support event to a strictly preceding establishing event (or to the initial state I). Simultaneously, the protection formula φ_2 logically isolates the temporal interval $[\tau_{sup}, \tau_{req}]$. It guarantees that any potentially contradictory event τ_{conf} is scheduled such that either $\tau_{conf} < \tau_{sup}$ or $\tau_{conf} > \tau_{req}$. Thus, no Contradictory-Effects occur, and Π accurately satisfies the Executable Task Conditions.
- *Initial and Goal States Consistency:* The SMT encoding explicitly roots the causal chains in the initial state I , meaning the execution of Π originates from a valid configuration. Finally, the goal constraints enforce that all $g \in G$ are supported by a valid establishing event prior to $\mathcal{T}_{Goal}$ and protected from subsequent deletions, ensuring the final state reached by Π correctly satisfies G .

Since all criteria (consistency, execution validity, initial state consistency, and goal reachability) are met, Π is a valid Team Temporal Plan for the given TTPP.

Completeness. Let $\Pi = \langle \mathcal{P}, \tau \rangle$ be a valid Team Temporal Plan for the TTPP $\langle \mathcal{M}, I, G \rangle$, operating within the established bound n_{cfp} . We construct an assignment $\mathcal{I}$ for the SMT formula as follows:

1. Assign the boolean variables ($t \in X^p$) and ($n \in C^p$) to true according to the exact sets X and C defined in $\mathcal{P}$. Assign all unused TFP variables to false.
2. Assign the temporal variables ($\mathcal{T}_b^p, \tau^p(t), \tau^p(e)$) to the exact rational values dictated by the scheduling function τ mapping the events in Π .

3. Assign the quantitative capability variables (val_{AddA} , val_{DelB} , etc.) to the numeric values representing the agents' true resource levels at those specific timepoints during the valid execution of Π .

We now demonstrate that $\mathcal{I}$ satisfies every clause of the encoded formula:

- Because Π is a valid TTP, it inherently respects all chronological boundaries and internal task durations. Thus, all corresponding structural difference-logic equations ($\leq, =$) naturally evaluate to true in $\mathcal{I}$.
- Because Π rigorously prevents agent and task overlapping, the continuous real-time intervals in τ are fully disjoint. Therefore, the logical disjunctions within the Mutex constraints evaluate to true.
- Because Π is validly executable from the initial state I , all required conditions and resources are correctly provided when needed. Consequently, every condition possesses a valid establishing event in the past timeline of Π . By assigning the virtual event variables ($\tau^{p,n}(t \dashrightarrow k_{qt}(+v))$, τ^{req} , etc.) to the exact timepoints of these past supportive events, the equality constraints within the causal support formulas (φ_1) evaluate to true.
- Furthermore, because Π contains zero unresolvable contradictory effects, strict temporal separation exists between any support event and any interfering effect. Thus, the temporal boundaries inherently satisfy the strict disjunctions required by the protection intervals (φ_2).
- Finally, since Π successfully reaches the goal state G , the goal-specific causal link formulas accurately map to a final sequence of valid establishing events.

Since the assignment $\mathcal{I}$ flawlessly maps the valid properties of the continuous temporal plan into the QF-LRA domains without discretization artifacts, the entire logical formula evaluates to true, thus proving the completeness of the encoding. ◀

A Formal Model for Forensically Sound Timestamp Normalization

Lorenz Hornung 

Open Universiteit, Heerlen, The Netherlands

Carly Bakker

Netherlands Forensic Institute, The Hague, The Netherlands

Bouke Timbermont

Open Universiteit, Heerlen, The Netherlands

Netherlands Forensic Institute, The Hague, The Netherlands

Roel van Dijk

Netherlands Forensic Institute, The Hague, The Netherlands

Harm van Beek 

Open Universiteit, Heerlen, The Netherlands

Netherlands Forensic Institute, The Hague, The Netherlands

Natasha Alechina 

Open Universiteit, Heerlen, The Netherlands

Utrecht University, The Netherlands

Abstract

Digital forensic analysts interpret timestamps from digital systems to assess what happened in the past. Since these systems encode time according to different underlying domains, timestamps cannot be interpreted uniformly. To enable cross-system analysis, timestamps must therefore be normalized. In this paper, we propose a formal model for forensically sound timestamp normalization and examine its properties. We show that normalization generally does not guarantee the comparability of normalized timestamps. This means that the temporal order of digitally tracked events cannot always be determined. Furthermore, we illustrate that existing forensic tools do not adhere to the proposed normalization, resulting in violations of forensic soundness in the example discussed.

2012 ACM Subject Classification Applied computing → Evidence collection, storage and analysis

Keywords and phrases Digital Forensics, Formal Model, Timestamp, Normalization

Digital Object Identifier 10.4230/OASICS.TIME.2026.10

1 Introduction

Law enforcement increasingly relies on digital investigations that analyze devices such as smartphones, computers, smartwatches, or servers. For these investigations, as in traditional forensics, traces¹ are relevant. Conceptually, digital traces are evidential artifacts derived from investigated digital sources that support the assessment of hypotheses (e.g., whether a suspect committed fraud or possessed illegal material) [11].

Important artifacts of this nature are so-called *timestamps*. Digital systems (such as file systems, operating systems, and many others) track when events occur and store this information as timestamps in databases, logs or file metadata [10, 11]. These timestamps are then used for event reconstruction, the investigative task of constructing and assessing hypotheses about past actions. Importantly, event reconstruction normally involves examining timestamps stored across devices and digital systems [9, 23].

¹ In forensic science, traces refer to signs of prior activity. This differs from traces in temporal logic and system verification, where it is defined as sequence of states or transitions (see, for example, [5]).



© Lorenz Hornung, Carly Bakker, Bouke Timbermont, Roel van Dijk, Harm van Beek, and Natasha Alechina;

licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 10; pp. 10:1–10:21

OpenAccess Series in Informatics



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

To enable cross-device and cross-system analysis, timestamps must be represented uniformly [9]. We refer to this task as *timestamp normalization* (informally defined in Section 2.2.3). Although forensic software, such as Plaso [35], Sleuth Kit [36], and Hansken [33], performs this operation and it is recognized as a challenge by digital forensic investigators [28], timestamp normalization for digital forensics has received no theoretical attention.

Contribution. This paper addresses this gap by providing a formal model of timestamp normalization that aligns with the principles of forensic soundness. This formalization establishes a standard that documents the essential properties to which timestamp normalization must adhere in digital forensic analysis. Upon examining the formalization, we show that forensically sound normalization does not guarantee the comparability of normalized timestamps. In other words, for certain timestamps, it is impossible to determine the order of tracked events. Additionally, we demonstrate that state-of-the-art forensic software inadequately normalizes timestamps, which might lead to forensically unsupported conclusions.

Outline. In Section 2, we introduce time tracking of civil time, time tracking in digital systems and how this results in distinct underlying time domains highlighting the need for normalization. Additionally, we informally define timestamp normalization and outline the forensic soundness requirements for it. In Section 3, we introduce the formal framework leading to the definition of normalization of timestamps (Section 3.6) building upon the definition of normalization of time domains (Section 3.3). We show how the formalization of normalization leads to incomparable timestamps in Section 3.5. In Section 4, we illustrate how Sleuth Kit, Plaso, and Hansken do not soundly normalize timestamps and elaborate on the resulting negative consequences using a real-world example. We discuss related work in Section 5 and outline directions for future research in Section 6. Proofs omitted throughout this paper are available in the appendix (Appendix A).

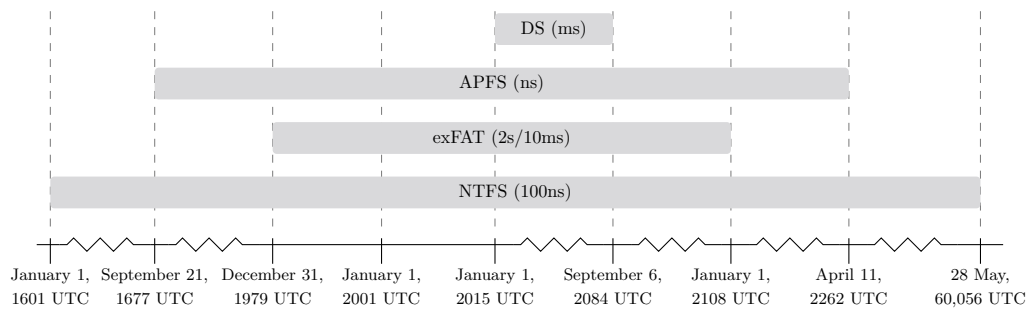
2 Defining Time in Digital Systems

Digital systems depend on defined time points to assign timestamps. We refer to the set of time points that a system can use to assign timestamps as its *time domain*. No single time domain is used by all digital systems; rather, the time domains of digital systems vary.

To illustrate this aspect, we will characterize the time domains of the following digital systems: the extensible File Allocation Table (exFAT) [29], the Apple File System (APFS) [3], the New Technology File System (NTFS) [30], and Discord Snowflake (DS) [18]. Figure 1 provides an overview over their time domains. ExFAT, APFS, and NTFS are file systems frequently found on portable storage devices (e.g., USB flash drives), Apple devices, and Windows devices, respectively. A file system organizes the storage and retrieval of files and directories on a storage device. In contrast, DS is a unique identifier containing a timestamp, used for messages, users, and servers on the social network Discord. We chose exFAT, APFS, and NTFS because they are significant in digital investigations, while DS serves as an example of a less common digital system that is nonetheless interesting in terms of timestamping.

2.1 Time Tracking of Civil Time

Civil life depends on a shared understanding of time. To ensure this, time standards are used which function as unified foundation to refer to points in time. For our purpose, the time standards Universal Time (UT1), International Atomic Time (TAI), and Coordinated Universal Time (UTC) are of interest. For references on UT1, TAI and UTC, see [24, 25]. Simplified, but sufficient for our purpose, these time standards relate as follows.



■ **Figure 1** Comparison of time domains of DS, APFS, exFAT, and NTFS.

TAI is the weighted average of hundreds of cesium atomic clocks spread worldwide and is based on fixed duration of a second (which is tracked by these atomic clocks). Based on this duration of a second, other units of time, like milliseconds, minutes or hours, are defined as fixed multiples of the second. For instance, 1/1000th of a second is a millisecond and 60 seconds are a minute. In contrast, Universal Time (UT1) tracks time by assessing the earth's rotation relative to the sun. Due to irregularities of earth's rotation, UT1 and TAI drift. UTC is the compromise of UT1 and TAI, as it keeps the same duration for seconds as TAI, but adds leap seconds when needed (at unpredictable times) to keep UTC within 0.9 seconds of UT1. Leap seconds are extra seconds added (or subtracted) at March 31, June 30, September 30, or December 31. So far, 27 leap seconds have been added and no leap second has been subtracted. As a consequence, some minutes of UTC consist of 61 instead of 60 seconds. For instance, the last leap second is December 31, 2016 23:59:60 UTC.

UTC is the primary time standard upon which all timezones are based. Local timezones are defined by an offset relative to UTC which can change throughout the year. For example Nepal uses Nepal Standard Time (NPT), defined as UTC+05:45, for the entire year. The Netherlands uses the Central European Time (CET) during winter, defined as UTC+01:00, and Central European Summer Time (CEST) during summer, defined as UTC+02:00.

2.2 Time Tracking in Digital Systems

Digital systems frequently, but not necessarily, use UTC as the foundation of their time domain. However, due to practical limitations UTC does not define their time domain; i.e., the finite set of time points that a system can represent as a timestamp.

2.2.1 Defining Time Points by Date and Time

Some digital systems base the encoding of timestamps on common human calendar systems. For instance, exFAT's time domain is based on the Gregorian calendar. It provides 16 bits for the date (7 bits for the year, 4 bits for the month, and 5 bits for the day), 16 bits for the time with a resolution² of 2 seconds (5 bits for hours, 6 bits for minutes, and 5 bits for seconds with a resolution of two seconds) and, only for file creation and last modification timestamps, 8 additional bits for milliseconds with a resolution of 10 milliseconds. For each group of bits, a range of values is defined. As an example, for years the range value is from 0

² In digital forensics, the notions "resolution", "granularity", and "precision" are used to refer, in the context of timestamps, to the smallest unit of time that a digital system can distinguish. To avoid confusion, we use only "resolution" to refer to this unit.

(year 1980) to 127 (year 2107), while for months the range value is from 1 (month January) to 12 (month December). The valid range for seconds is 0 to 29 and for milliseconds 0 to 199, illustrating that exFAT does not account for leap seconds.

Since exFAT provides these values relative to the local time (sensitive to timezone), it keeps track of the relative offset to UTC. This offset is given by 8 bits (6 bits signed decimal for offset with a resolution of 15 minutes, 1 bit to define whether offset is known). The valid range for the offset is -64 (UTC - 16:00) to 63 (UTC + 15:45) with 0 being UTC. Due to the offset of UTC, exFAT's time domains relative to UTC are defined as follows with D_{exFAT}^1 denoting the time domain for file creation and last modification timestamps, whereas D_{exFAT}^2 refers to the time domain for the last access timestamps.

- D_{exFAT}^1 : Every 10 milliseconds between December 31, 1979 00:08:00.00 UTC and January 1, 2108 15:45:59.99 UTC excluding leap seconds
- D_{exFAT}^2 : Every 2 seconds between December 31, 1979 00:08:00 UTC and January 1, 2108 15:45:58 UTC excluding leap seconds

2.2.2 Defining Time Points by Elapsed Time since Epoch

More frequently, time points are defined by counting forwards and, when applicable, backwards from a fixed point in time, called the epoch. The time domains of DS, APFS, and NTFS are defined by this method. As a foundation, they define the following epochs: January 1, 1970 00:00:00 UTC for APFS, January 1, 1601 00:00:00 UTC for NTFS, January 1, 2015 00:00:00 UTC for DS.

Counting forward and backward depends on a system's time resolution. Each step corresponds to one unit of the chosen resolution. APFS uses nanosecond resolution, NTFS uses 100-nanosecond resolution, and DS uses millisecond resolution.

Since these systems define a finite space allocated for storing the numeric value representing the counted elapsed (41 bits for DS and 64 bits for APFS and NTFS), we can determine the earliest and the latest time point defined of the time domain. Since DS and NTFS do not allow backward counting, DS can theoretically store elapsed times in the range from 0 to $2^{41} - 1 = 2,199,023,255,551$ milliseconds and, NTFS from 0 to $2^{64} - 1 = 18,446,744,073,709,551,615$ hundreds of nanoseconds. In contrast, APFS allows value from the range from $-2^{63} = -9,223,372,036,854,775,808$ to $2^{63} - 1 = 9,223,372,036,854,775,807$ nanoseconds. Picking the epoch as the time point corresponding to the value 0 of the valid range, we can then compute the earliest and latest time point defined. Without taking leap seconds into account due to their unpredictability, their time domains, denoted by D_{APFS} , D_{NTFS} , and D_{DS} , are defined as follows.

- D_{APFS} : Every nanosecond between September 21, 1677 00:12:43.145224192 UTC and April 11, 2262 23:47:16.854775807 UTC excluding leap seconds
- D_{NTFS} : Every 100 nanoseconds between January 1, 1601 00:00:00.0000000 UTC and 28 May, 60056 05:36:10.9551615 excluding leap seconds
- D_{DS} : Every millisecond between January 1, 2015 00:00:00.000 UTC and September 6, 2084 15:47:35.551 UTC excluding leap seconds

2.2.3 Forensic Analysis of Timestamps

Digital investigations require to interpret obtained traces, including timestamps, from different sources. For instance, a case might require to analyze a USB-stick with exFAT, a phone with APFS and a laptop with NTFS. To enable consistent handling, comparison, and

searching across these sources, one needs to establish links between timestamps across distinct underlying time domains. Drawing these links refers to the task of *timestamp normalization*, informally defined as follows.

► **Definition 1.** *Timestamp normalization refers to the process of representing timestamps from different time domains in a single time domain.*

These normalizations cannot be performed arbitrarily, but need to adhere to principle of *forensic soundness* which is essential to obtain admissible evidence for criminal proceedings. This concept refers to the ideals of *integrity* and *chain of custody*. Integrity demands that the preservation of evidence and its meaning be maintained. However, this does not imply that digital evidence has to be preserved in its entire form, as acquiring data can already lead to modification of data. This is why the ideal of chain of custody demands that any action performed on digital evidence is documented [4, 11, 40]. For the normalization of timestamps, we recognize the following implications of forensic soundness:

- To ensure the integrity of timestamps, the temporal relationships within the same domain must be preserved. However, integrity does not require that a time point may only be normalized with the same time point. As we will discuss later, this flexibility is necessary to normalize time points with different resolutions.
- In light of the ideal of chain of custody, any normalization procedure and the resulting links between timestamps across distinct domains must be grounded in a definition to guarantee explainability, accounting for these potential differences in representation of time points.

The second aspect emphasizes the importance of a formal definition of timestamp normalization for ensuring compliance with these requirements.

3 A Formal Framework for Timestamp Normalizations

In this section, we formalize the notion of forensically sound timestamp normalization. To do this, we first define normalizations of time domains (Section 3.3), which builds on core notions like time domains and their intervals (Sections 3.1 and 3.2). We then formally define timestamp normalization (Section 3.6) and reveal a critical insight: normalized time points may not be comparable, which is related to the resolution of the time domain (Sections 3.4 and 3.5) and affects the comparability of normalized timestamps.

3.1 Introducing the Core Notions

Time Domain. As discussed informally in Section 2, time domains are sets of time points which refer to specific moments in time. We denote time domains by capital letters like D and T . Formally, any time domain D is a non-empty (possibly countably infinite) set of time points. The time domains D_{exFAT}^1 , D_{exFAT}^2 , D_{APFS} , D_{NTFS} , and D_{DS} described in the previous section are examples of time domains. We denote any time point of some time domain by t . Examples of time points are t_0 and t_1 .

$$t_0 = \text{January 1, 2020 00:00:00 UTC} \quad t_1 = \text{January 1, 2020 00:00:00.000 UTC}$$

Importantly, t_0 and t_1 do *not* refer to the same moment in time because the latter contains more temporal information than the former one due to the different resolution.

Time Domain Relation. We assume that for any time domain in digital systems, its time points are comparable. Given any t_1 and t_2 of the same time domain, one knows whether t_1 follows t_2 or vice versa. This clearly holds for D_{exFAT}^1 , D_{exFAT}^2 , D_{APFS} , D_{NTFS} , and D_{DS} .

Formally, for any time domain D , we fix a time domain relation $\preceq_D$ on D so that $\preceq_D$ is a total order. That is, $\preceq_D \subseteq D \times D$ is reflexive, antisymmetric, transitive and total. A time domain relation expresses the temporal order within the domain considered.

We fix that for any time domains D and T , if $t_1, t_2 \in D \cap T$, then $t_1 \preceq_D t_2$ if and only if $t_1 \preceq_T t_2$. As a consequence, if $D = T$, then $\preceq_D = \preceq_T$. This condition ensures that any time points shared between different domains preserve the same ordering across all domains.

We further introduce the follow notation: $t_1 \not\preceq_D t_2$ denotes that it does *not* hold that $t_1 \preceq_D t_2$, while $t_1 \prec_D t_2$ denotes that $t_1 \preceq_D t_2$ and $t_2 \not\preceq_D t_1$. Further, $t_1 =_D t_2$ denotes that $t_1 \preceq_D t_2$ and $t_2 \preceq_D t_1$. If the context is clear, we omit the reference to the respective time domain D for $\preceq_D$, $\prec_D$, and $=_D$. That is, $\preceq$, $\prec$, and $=$ denote the time domain relation, its strict part and its equivalence part of D .

Timestamps. Digital system do not provide timestamps for arbitrary events, but for a specific list of events. For example, a file system timestamps the creation and last access time of files, among other things. An event set provides such events and we denote them by E . Formally, an event set is a non-empty (possibly countably infinite) set and we denote an event of E by e . The information that an event occurred is stored as an artifact in form of a timestamp. Given a set of events E and a time domain D , a timestamp relative to E and D is a tuple $\langle e, t \rangle$ with $e \in E$ and $t \in D$. A timestamp is therefore a pair consisting of a trace and a time point specifying one time point of D where e was registered.

Given two timestamps, $\langle e_1, t_1 \rangle$ and $\langle e_2, t_2 \rangle$, relative to the same time domain $t_1, t_2 \in D$, we can then express that event e_1 tracked by $\langle e_1, t_1 \rangle$ (*weakly*) *precedes* e_2 tracked by $\langle e_2, t_2 \rangle$ using the time domain order $\preceq_D$ on D . That is, e_1 tracked by $\langle e_1, t_1 \rangle$ precedes e_2 tracked by $\langle e_2, t_2 \rangle$ if $t_1 \preceq_D t_2$.

3.2 Intervals of Time Domains

As we will discuss in Section 3.3, normalizations of time domains require intervals of time points. Intuitively, an interval of a time domain is a set of time points that lie between an upper and a lower bound.

► **Definition 2.** Let D be a time domain and $\preceq$ be its time domain relation. For any $t_1, t_2 \in D$, $[t_1, t_2]$ is defined as $[t_1, t_2] = \{t \in D \mid t_1 \preceq t \preceq t_2\}$. The set $\mathcal{I}_D^{\preceq}$ of (non-empty) intervals of D relative $\preceq$ is defined as $\mathcal{I}_D^{\preceq} = \{[t_1, t_2] \mid t_1, t_2 \in D \wedge t_1 \preceq t_2\}$.

We say that $[t_1, t_2]$ is an interval of D and $\preceq$, if $[t_1, t_2] \in \mathcal{I}_D^{\preceq}$. For any time domain D and any $t \in D$, $[t, t] = \{t\}$ is an interval of D , because any time domain relation $\preceq$ is reflexive. We refer to $[t, t]$ as the *time point interval* of t . Further, the definition of the set of (non-empty) intervals enforces that no $\mathcal{I}_D^{\preceq}$ contains any empty sets or is empty.

► **Proposition 3.** For any time domain D and its time domain relation $\preceq$, (i) $\mathcal{I}_D^{\preceq} \neq \emptyset$, and (ii) for any $[t_i, t_j] \in \mathcal{I}_D^{\preceq}$ it holds that $[t_i, t_j] \neq \emptyset$.

Building on the fixed time domain relation of a time domain, we define a relation on the set of time intervals of that domain as follows.

► **Definition 4.** Given a time domain D and its time domain relation $\preceq$, the interval relation $\preceq_D^{\mathcal{I}} \subseteq \mathcal{I}_D^{\preceq} \times \mathcal{I}_D^{\preceq}$ is defined as for any $[t_1, t_2], [t_3, t_4] \in \mathcal{I}_D^{\preceq}$, $[t_1, t_2] \preceq_D^{\mathcal{I}} [t_3, t_4] \iff t_2 \preceq t_3$.

Since $\preceq$ is reflexive, the interval relation $\preceq_D^{\mathcal{I}}$ allows ordered intervals to overlap. That is, if $t_2 = t_3$, then $[t_1, t_2] \preceq_D^{\mathcal{I}} [t_3, t_4]$. This is why, we say that $[t_1, t_2]$ *weakly precedes* $[t_3, t_4]$.³ Further, unlike the underlying time domain relation, the interval relation is neither reflexive nor total if the time domain contains more than one time point.

► **Proposition 5.** *Let D be any time domain, $\preceq$ be its time domain relation, and $\preceq_D^{\mathcal{I}} \subseteq \mathcal{I}_D^{\preceq} \times \mathcal{I}_D^{\preceq}$ be its interval relation.*

- (1) $\preceq_D^{\mathcal{I}}$ is transitive and antisymmetric.
- (2) If $|D| = 1$, then $\preceq_D^{\mathcal{I}}$ is reflexive and total.
- (3) If $|D| > 1$, then $\preceq_D^{\mathcal{I}}$ is neither reflexive nor total.

3.3 Time Domain Normalization

As informally specified in Definition 1, normalizing a time domain requires characterizing its time points in terms of time points from another time domain. We formally define normalization of a time domain as follows.

► **Definition 6.** *Let D and T be time domains, and $\preceq_D$ and $\preceq_T$ be their time domain relations. A normalization of D relative to T is a partial function $n : D \rightarrow \mathcal{I}_T^{\preceq}$ such that*

- (1) *if $n(t_1)$ and $n(t_2)$ are defined for $t_1, t_2 \in D$, then $n(t)$ is defined for all $t \in D$ with $t_1 \preceq_D t \preceq_D t_2$;*
- (2) *for any $t_1, t_2 \in D$ with $n(t_1)$ and $n(t_2)$ defined, if $t_1 \prec_D t_2$, then $n(t_1) \preceq_T^{\mathcal{I}} n(t_2)$;*
- (3) *for any $t \in D \cap T$, $n(t) = [t, t]$.*

For any $t \in D$, if $n(t)$ is defined, we call $n(t)$ the normalization of t relative to n and T .

Accordingly, a normalization is a partial function whose domain is a time domain (i.e., D) and whose codomain is a set of intervals (i.e., $\mathcal{I}_T^{\preceq}$). A set of intervals as the codomain is necessary to accommodate time domains of different resolution. Consider, for instance, D_{DS} and D_{APFS} . A time point t in D_{DS} with millisecond precision corresponds to 1,000,000 time points in D_{APFS} , each representing a time point with nanosecond precision. Due to the hierarchical composition of tracked time, t is not equivalent to any single one of these 1,000,000 time points in D_{APFS} , but to the set of all of them.

We can express the fact that not all time points in the domain are necessarily normalized by looking at the partiality of normalizations. This property provides sufficient flexibility when the time domain to be normalized and the time domain used for normalization do not cover the same period of time. For example, D_{exFAT}^1 contains time points starting from December 31, 1979 UTC, whereas the earliest defined time points in D_{DS} begin on January 1, 2015 UTC. If normalizations were not partial, attempting to normalize D_{exFAT}^1 relative to D_{DS} would require assigning time points from the 20th century to intervals occurring at least 35 years later. The forensic soundness would be violated by such a drastic redating.

Constraint (1) limits where a normalization may be undefined. It requires the set of time points for which the normalization is defined must form a contiguous segment of the time domain. As a consequence, the normalization of time points at the boundaries can be undefined (assuming that the normalization of some time points is defined). In the previous example of normalizing D_{exFAT}^1 relative to D_{DS} , this allows all time points before

³ Analogously, strict precedence on intervals is definable by demanding $t_2 \prec t_3$. However, only weak precedence is required to express order preservation during normalization across different resolutions. Nevertheless, future research should address strict precedence when constructing timelines of events based on normalized timestamps.

January 1, 2015 UTC and after September 6, 2084 UTC to remain undefined, while requiring the normalization to be defined for all intermediate time points of D_{exFAT}^1 , if time points of D_{exFAT}^1 are normalized by their counterparts in D_{DS} .

Constraint (2) demands that a normalization does not alter the ordering of time points during a normalization. To ensure this property, if for $t_1, t_2 \in D$ it holds that $t_1 \prec_D t_2$, then for their normalized values $n(t_1)$ and $n(t_2)$, it must hold that interval $n(t_1)$ weakly precedes $n(t_2)$. Note that the inverse does not hold, as illustrated in Example 7.

► **Example 7.** Let us consider the normalization of $D = D_{\text{exFAT}}^1$ relative to $T = D_{\text{exFAT}}^2$ by focusing on the following time points $t_1, \dots, t_{199} \in D$ and $t \in T$.

$$\begin{aligned} t_0 &= \text{January 1, 2000 00:00:00.00 UTC} & t &= \text{January 1, 2000 00:00:00 UTC} \\ &\vdots & & \\ t_{199} &= \text{January 1, 2000 00:00:01.99 UTC} \end{aligned}$$

We define the normalization n of D relative to T as $n(t_0) = \dots = n(t_{199}) = [t, t]$. Since D_{exFAT}^1 is hence normalized by a time domain with a more coarse resolution, time points distinguishable by the time domain relation of D like t_0 and t_{199} (i.e., $t_0 \prec_D t_{199}$), are not distinguishable after the normalization because $n(t_0) = n(t_{199})$. Further, because they are both normalized by the point interval of t , $n(t_{199}) \preceq_T^I n(t_0)$, while $t_{199} \not\prec_D t_0$.

Lastly, let us discuss constraint (3) that considers normalizations where the time domain to be normalized contains time points also contained by the time domain relative to which the former is normalized; i.e., $D \cap T \neq \emptyset$. For the time points occurring in both time domains, it is required that they are normalized by their time point interval. For these values, the interval relation and the time domain relation encode the same ordering.

► **Proposition 8.** Let D and T be time domains, $\preceq_D$ and $\preceq_T$ be their time domain order, and $n : D \rightarrow \mathcal{I}_T^{\preceq_T}$ be any normalization of D relative to T . If $D \cap T \neq \emptyset$, then for any $t_1, t_2 \in D \cap T$, $t_1 \preceq_D t_2 \iff n(t_1) \preceq_T^I n(t_2)$.

Besides these constraints, one can define further properties of normalizations.

► **Definition 9.** Let D and T be time domains, and $\preceq_D$ and $\preceq_T$ be their time domain relations. A normalization $n : D \rightarrow \mathcal{I}_T^{\preceq_T}$ is

- (1) *non-empty* if $n(t)$ is defined for some $t \in D$;
- (2) *gap-free* if for any $t_1, t_2 \in T$ with $t_1 \in n(t'_1)$ and $t_2 \in n(t'_2)$ for some $t'_1, t'_2 \in D$, then for any t with $t_1 \preceq_T t \preceq_T t_2$, there exist $t' \in D$ such that $t \in n(t')$;
- (3) *even* if for any $t_1, t_2 \in D$ with $n(t_1)$ and $n(t_2)$ defined, $|n(t_1)| = |n(t_2)|$.

Non-emptiness indicates that, under the given normalization, the time domains overlap, ensuring that at least one point of D is normalized by an interval of T . Gap-freeness and evenness guarantee that the normalized time points (and the intervals that normalize them) are uniformly distributed over the target time domain. This uniformity is crucial when the domain itself is evenly spaced, such as all time domains based on UTC.

3.4 Comparing Time Domain Resolutions relative to Normalizations

Normalizations enable us to compare the resolution of time domains. That is, given a normalization of a time domain relative to another time domain, we can evaluate whether the former is more coarse or fine-grained than the latter. As we will discuss in Section 3.5, differences in resolution affect the comparability of normalized time points originating from different time domains.

► **Definition 10.** Let D and T be time domains and let $n : D \rightarrow \mathcal{I}_T^{\leq T}$ be a normalization of D relative to T with $\text{Dom}(n) = \{t \in D \mid n(t) \text{ defined}\}$. We say that

- D is at $t \in \text{Dom}(n)$ strictly more fine-grained than T relative to n (denoted $T \triangleleft_n^t D$), if there exist $t' \in D$ such that $t \neq t'$ and $n(t) = n(t')$. Further, D is globally strictly more fine-grained than T relative to n (denoted $T \triangleleft_n^\top D$), if for any $t \in \text{Dom}(n)$, $T \triangleleft_n^t D$.
- D is at $t \in \text{Dom}(n)$ at least as fine-grained as T relative to n (denoted $T \preceq_n^t D$), if $|n(t)| = 1$. Further, D is globally at least as fine-grained as T relative to n (denoted $T \preceq_n^\top D$), if for any $t \in \text{Dom}(n)$, $T \preceq_n^t D$.
- D is at $t \in \text{Dom}(n)$ strictly less fine-grained than T relative to n (denoted $D \triangleleft_n^t T$), if $|n(t)| > 1$. Further, D is globally strictly less fine-grained than T relative to n (denoted $D \triangleleft_n^\top T$), if for any $t \in \text{Dom}(n)$, $D \triangleleft_n^t T$.

Strictly-more-fine-grainedness requires comparing the normalization of multiple time points. Intuitively, a time domain is strictly more fine-grained than another when multiple distinct points in the former normalize to the same value in the latter. For example, seconds are strictly more fine-grained than minutes because 60 distinct seconds normalize to the same minute. At-least-as-fine-grainedness does not demand the comparison of two time-points. In Proposition 11, it will be shown that strictly-more-fine-grainedness implies at-least-as-fine-grainedness. Strictly-less-fine-grainedness occurs when a time point in the domain corresponds to multiple normalized values. For instance, an hour is strictly less fine-grained than minutes because one hour normalizes to 60 minutes. These distinctions lead to the definitions above. Independent of the different definitions, these notions intuitively relate to each other. For instance, if a domain is locally (i.e., at a time point) or globally strictly more fine-grained than the domain used to normalize, it implies that it is also locally or globally at least as fine-grained as the other domain, but not locally or globally strictly less fine-grained. This intuitive property, among others, is given by Proposition 11.

► **Proposition 11.** Let D and T be time domains, while $n : D \rightarrow \mathcal{I}_T^{\leq T}$ be a non-empty normalization of D relative to T with $\text{Dom}(n) = \{t \in D \mid n(t) \text{ defined}\}$. For any $t \in \text{Dom}(n)$,

- (1) $D \triangleleft_n^t T$ if and only if $T \not\preceq_n^t D$;
- (2) if $T \triangleleft_n^t D$ then $T \preceq_n^t D$;
- (3) if $T \triangleleft_n^t D$ then $D \not\triangleleft_n^t T$;
- (4) if $D \triangleleft_n^\top T$ then $T \not\preceq_n^\top D$;
- (5) if $T \triangleleft_n^\top D$ then $T \preceq_n^\top D$;
- (6) if $T \triangleleft_n^\top D$ then $D \not\triangleleft_n^\top T$.

3.5 Comparing Normalized Time Domains

Normalizing multiple time domains relative to the same time domain provides the foundation to compare these time points. However, normalization relative to the same time domain does not ensure that the normalized time points of the distinct time domains are comparable. This means that we are not able to state whether the one normalized time point happened before or after the other normalized time point (in the sense of weak precedence). Such incomparability is formally due to the fact that the interval time domain order $\preceq_D^{\mathcal{I}}$ is not total for a time domain D containing more than one time point, as illustrated by Proposition 5.

► **Example 12.** As illustrated in Section 2.2, exFAT file system provides last access timestamps with a 2 seconds resolution, while NTFS provides file timestamps, including last access timestamps, with a resolution of 100 nanoseconds. An example of time points of these domains are $t \in D_{\text{NTFS}}$ and $t' \in D_{\text{exFAT}}^2$ which are defined as follows.

$$t = \text{January 1, 2026 10:00:00.5000000 UTC} \quad t' = \text{January 1, 2026 10:00:00 UTC}$$

10:10 A Formal Model for Forensically Sound Timestamp Normalization

If t and t' are normalized relative to a time domain T of resolution milliseconds (e.g., $T = D_{\text{DS}}$), it is impossible to determine which time point was registered first. This is due to the coarser resolution of D_{exFAT}^2 . To make this circumstance explicit, we normalize both time points relative to D_{DS} . We define two normalizations n_1 and n_2 such that $n_1(t) = [t_{500}, t_{500}]$ and $n_2(t') = [t_0, t_{1999}]$ where $t_0, \dots, t_{1999}$ are defined as follows.

$$\begin{array}{ll} t_0 = \text{January 1, 2026 10:00:00.000 UTC} & t_{1000} = \text{January 1, 2026 10:00:01.000 UTC} \\ \vdots & \vdots \\ t_{500} = \text{January 1, 2026 10:00:00.500 UTC} & t_{1500} = \text{January 1, 2026 10:00:01.500 UTC} \\ \vdots & \vdots \\ t_{999} = \text{January 1, 2026 10:00:00.999 UTC} & t_{1999} = \text{January 1, 2026 10:00:01.999 UTC} \end{array}$$

The fact that $t_0 \prec_T t_{500} \prec_T t_{1999}$ highlights that we have no information about the relative temporal ordering of these normalized time points.

Generally speaking, incomparable normalized time points occur if the intervals constituting their normalizations overlap so that, analogously to the discussed example, one is not capable to state which normalized time point weakly precedes the other normalized time point.

► **Proposition 13.** *Let D_1 , D_2 , and T be time domains, and n_1 and n_2 be normalizations of D_1 and D_2 relative to T . For any $t \in D_1$ and $t' \in D_2$ with $n_1(t) = [t_1, t_2]$ and $n_2(t') = [t_3, t_4]$, $n_1(t) \not\preceq_T^{\mathcal{I}} n_2(t')$ and $n_2(t') \not\preceq_T^{\mathcal{I}} n_1(t)$ if and only if $t_3 \prec_T t_2$ and $t_1 \prec_T t_4$.*

Incomparability of distinct time points occurs only when normalized time points from different domains are considered and these time domains, including the domain used for the normalizations, are not identical. This result is captured by the following proposition.

► **Proposition 14.** *Given the time domains D_1 , D_2 , and T such that $D_1 = D_2 = T$, then for any normalizations n_1 and n_2 of D_1 and D_2 relative to T respectively, $n_1(t_1) \preceq_T^{\mathcal{I}} n_2(t_2)$ or $n_2(t_2) \preceq_T^{\mathcal{I}} n_1(t_1)$ for any $t_1 \in D_1$ and $t_2 \in D_2$.*

As previously mentioned, the (in)comparability of normalized time points from different domains is related to the differences in resolution between the normalized time domains and the domain used for normalization. Proposition 15 illustrates one aspect of this connection.

► **Proposition 15.** *Let D_1 , D_2 , and T be time domains, and n_1 and n_2 be normalization of D_1 and D_2 relative to T , respectively.*

- (1) *If $T \trianglelefteq_{n_1}^{\top} D_1$ and $T \trianglelefteq_{n_2}^{\top} D_2$, then $n_1(t_1) \preceq_T^{\mathcal{I}} n_2(t_2)$ or $n_2(t_2) \preceq_T^{\mathcal{I}} n_1(t_1)$ for any $t_1 \in D_1$ and $t_2 \in D_2$ with $n_1(t_1)$ and $n_2(t_2)$ defined.*
- (2) *If $T \trianglelefteq_{n_1}^{\top} D_1$ and $D_2 \triangleleft_{n_2}^{\top} T$ where $|n_2(t)| = 2$ for any $t \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$, then $n_1(t_1) \preceq_T^{\mathcal{I}} n_2(t_2)$ or $n_2(t_2) \preceq_T^{\mathcal{I}} n_1(t_1)$ for any $t_1 \in D_1$ and $t_2 \in D_2$ with $n_1(t_1)$ and $n_2(t_2)$ defined.*

Note that (1) also holds if D_1 or D_2 are globally strictly more fine-grained than T , because, as stated in Proposition 11, strictly-more-fine-grainedness implies at-least-as-fine-grainedness. Analogously, (2) also holds if D_1 is globally strictly more fine-grained than T .

Further, proposition (1) suggests that a less fine-grained time domain used for normalization can ensure comparability between time points from different time domains. However, such a less fine-grained time domain implies a loss of resolution in distinguishing time points which means that information is lost. Proposition (2) sheds lights on a relevant fact for

the digital forensic domain. Although digital systems like exFAT use time domains with a resolution of two seconds, normalizing it to a time domain with second resolution does not lead to incomparable time points if another time domain with finer-grained resolution is normalized to the same time domain.

Nevertheless, comparability of time points of different time domains after normalization relative to one time domain cannot be guaranteed. Proposition 16 investigates some of these situations that are forensically interesting. It articulates the inevitability of incomparable normalized time points when two time domains are normalized relative to a reference time domain that is at least as fine-grained as at least one of the time domains being normalized, and when the overlap of the normalized time points contains a sufficiently large number of time points. A natural example is the normalization of time domains, such as D_{DS} and D_{AFPS} , with millisecond and nanosecond resolution, respectively, relative to a time domain with microsecond resolution. Clearly, the overlap of the normalized values should exceed the required three microseconds.

► **Proposition 16.** *Let D_1 , D_2 and T be time domains, and n_1 and n_2 be gap-free normalizations of D_1 and D_2 relative to T , respectively.*

- (1) *If $T \triangleleft_{n_1}^\top D_1$, $D_2 \triangleleft_{n_2}^\top T$ where $|n_2(t)| \geq 3$ for any $t \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$, and $|\{t \in T \mid t \in n_1(t'_1) \text{ for some } t'_1 \in D_1\} \cap \{t \in T \mid t \in n_2(t'_2) \text{ for some } t'_2 \in D_2\}| \geq 3$, then there exist $t_1 \in D_1$ and $t_2 \in D_2$ such that $n_1(t_1) \not\preceq_T^\mathcal{I} n_2(t_2)$ and $n_2(t_2) \not\preceq_T^\mathcal{I} n_1(t_1)$.*
- (2) *If $D_1 \triangleleft_{n_1}^\top T$ where $|n_1(t)| \geq 3$ for any $t \in \{t \in D_1 \mid n_1(t) \text{ defined}\}$, $D_2 \triangleleft_{n_2}^\top T$ where $|n_2(t)| \geq 3$ for any $t \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$, and $|\{t \in T \mid t \in n_1(t'_1) \text{ for some } t'_1 \in D_1\} \cap \{t \in T \mid t \in n_2(t'_2) \text{ for some } t'_2 \in D_2\}| \geq 2$, then there exist $t_1 \in D_1$ and $t_2 \in D_2$ such that $n_1(t_1) \not\preceq_T^\mathcal{I} n_2(t_2)$ and $n_2(t_2) \not\preceq_T^\mathcal{I} n_1(t_1)$.*

3.6 Timestamp Normalization

In the next step, we want to normalize timestamps. Recall that, given a set of events E and a time domain D , a timestamp relative to E and D is tuple $\langle e, t \rangle$ with $e \in E$ and $t \in D$ denoting that e was registered at t . To normalize this timestamp relative to time domain, a normalization of the time domain underlying the timestamp relative to this other time domain is needed. If such a normalization is given, we normalize the timestamp by normalizing its time point. We define this process as follows.

► **Definition 17.** *Let D and T be time domains, E be a set of events, and n be a normalization of D relative to T . Any timestamp $\langle e, t \rangle$ with $e \in E$ and $t \in D$ is normalizable by n relative to T if $n(t)$ is defined. For any normalizable timestamp $\langle e, t \rangle$ with $e \in E$ and $t \in D$, we define the the normalization of $\langle e, t \rangle$ relative to n and T , denoted $n(\langle e, t \rangle)$, as $\langle e, n(t) \rangle$.*

As a consequence of this definition, timestamps and normalized timestamps conceptually differ. Timestamps are pairs containing one event and one time point, while normalized timestamps are pairs containing one event and a set of time points (i.e., the interval that normalizes the time point). This conceptual difference is an inescapable consequences of accounting for different time domains in the process of normalizations.

Timestamp normalization allows us to infer the order of events which are tracked using different time domains. Given two timestamps $\langle e_1, t_1 \rangle$ and $\langle e_2, t_2 \rangle$ based on time domains D_1 and D_2 , and their normalizations $n_1(\langle e_1, t_1 \rangle)$ and $n_2(\langle e_2, t_2 \rangle)$ relative to T , we can establish an order between the tracked events e_1 and e_2 using the interval time domain order $\preceq_T^\mathcal{I}$. That is, e_1 tracked by normalized timestamp $n_1(\langle e_1, t_1 \rangle)$ weakly precedes e_2 tracked by normalized timestamp $n_2(\langle e_2, t_2 \rangle)$ if $n_1(t_1) \preceq_T^\mathcal{I} n_2(t_2)$ and vice versa.

The findings discussed in Section 3.5 regarding incomparable normalized time points affect the comparability of normalized timestamps. If the underlying time domains of two timestamps contain incomparable normalized time points (as, for instance, discussed in Proposition 16), the timestamps may become incomparable. This means the order of events cannot be determined. This occurs when for $n_1(\langle e_1, t_1 \rangle)$ and $n_2(\langle e_2, t_2 \rangle)$ it holds that $n_1(t_1) \not\leq_T^{\mathcal{I}} n_2(t_2)$ and $n_2(t_2) \not\leq_T^{\mathcal{I}} n_1(t_1)$. For a forensically sound analysis, it must therefore always be taken into account that timestamps, despite an underlying time domain where every point in time is comparable, become incomparable through normalization. In Section 4, we will see that state-of-the-art forensic software does not fulfill this demand.

4 Normalization in Forensic Software

In this section, we evaluate the timestamp normalization implementations in Plaso version 20260119, Sleuth Kit version 4.14.0., and Hansken version 47.42.0. We show how they violate our proposed model of forensically sound normalization, illustrating the negative impact with a real-world example.

4.1 Timestamp Normalization in Plaso, Sleuth Kit, and Hansken

Plaso. Plaso [35] represents timestamps from various sources on a unified timeline. For this purpose, it parses artifacts from multiple sources, such as file systems, logs, and cache data. Given this parsed information, it represents the timestamp events on one timeline. Plaso normalizes timestamps into a standardized string representation of the form `YYYY-MM-DD hh:mm:ss.#####[±]HH:MM`. Only the date component (`YYYY-MM-DD`) is mandatory, while the time of day (`hh:mm:ss`), fractional seconds with milli- or microsecond resolution (`#####`), and timezone offset with minute resolution relative to UTC (`[±]HH:MM`) are optional. If no timezone is specified, UTC is assumed.⁴

Sleuth Kit. The Sleuth Kit [36] is frequently used in forensic analysis for volume and file system analysis. The aim of this analysis is to make metadata of files within a file system, like the timestamps of a file, accessible for forensic investigations. Sleuth Kit normalizes file timestamps by converting any timestamp in a UNIX timestamp. For the considered file systems, any exFAT timestamps is converted into the a single positive number of seconds since January 1, 1970 00:00:00 UTC, while any NTFS or APFS timestamp is converted into a single positive number of nanoseconds since this epoch.

Hansken. Hansken [33, 41] extracts traces from digital data, systemizes them, and provides tools to efficiently search this data. It extracts timestamps from various sources and normalizes them into a standardized string of the form `YYYY-MM-DD hh:mm:ss.sss Z`, where `Z` is the standard notation for UTC timezone. The date (`YYYY-MM-DD`) and time (`hh:mm:ss.sss Z`) are both mandatory and are always given relative to UTC. Further, Hansken provides the resolution of the original timestamp, but not the timestamp itself.

⁴ Assigning a default timezone to timestamps lacking timezone information refers to the process of inferring missing temporal context. We briefly outline this challenge as future work in Section 6.

4.2 Assessing Alignment with the Proposed Model

Unlike our model of forensically sound timestamp normalization (Sections 3.3 and 3.6), Plaso, Sleuth Kit, and Hansken normalize timestamps by converting time points into a different time domain, which may have a different resolution. Consequently, their method does not align with ours as it does not utilize intervals. This discrepancy leads to misleading normalized timestamps, as the following example illustrates.

► **Example 18.** Using a script, we created multiple `.txt` files on exFAT and NTFS file systems on Windows 11. Filenames include a sequential creation index and a timestamp with millisecond resolution (UTC+02:00) to track the order and time of file creation. ExFAT and NTFS store a file’s creation time and last access time in its metadata. Since no file was accessed after creation, these timestamps refer to the same time in the corresponding time domain. As discussed in Section 2.2, exFAT tracks the former with a resolution of 10 milliseconds and the latter with a resolution of 2 seconds, while NTFS timestamps both with a resolution of 100 nanoseconds. The following table shows the extracted normalized timestamps of two created files.

■ **Table 1** Timestamps extracted and normalized by Plaso, Sleuth Kit and Hansken for two files.

Software	Create Timestamp	Last Access Timestamp
File 02.2026-04-10T09-07-04.484.txt stored in NTFS file system:		
Plaso	2026-04-10T07:07:04.484325+00:00	2026-04-10T07:07:04.484325+00:00
Sleuth Kit	2026-04-10 09:07:04.484325000 (CEST)	2026-04-10 09:07:04.484325000 (CEST)
Hansken	2026-04-10T07:07:04.484Z	2026-04-10T07:07:04.484Z
File 05.2026-04-10T09-07-05.992.txt stored in exFAT file system:		
Plaso ^{a)}	2026-04-10T09:07:05.000000+00:00	2026-04-10T09:07:04.000000+00:00
Sleuth Kit	2026-04-10 09:07:05 (CEST)	2026-04-10 09:07:04 (CEST)
Hansken	2026-04-10T07:07:05.990Z	2026-04-10T07:07:04.000Z

a) Plaso incorrectly uses UTC instead of UTC+02:00 for both timestamps.

None of the software informs users when a timestamp has been normalized relative to a domain with a different resolution than the original. However, due to the resolution of the original timestamps provided, Hansken provides clues to such differences. Further, none of the tools provide information indicating that the time points given as the normalized value should be interpreted as intervals. Consequently, the timestamps appear to be time points, as shown in Table 1. This can lead to the following undesired conclusions.

Although the create and the last access timestamp within exFAT track the same moment in time when a file is created, the normalized timestamps of the forensic tools incorrectly suggest that 02...txt was created after it was last accessed. Furthermore, while 02...txt was created and hence accessed prior to 05...txt, a comparison limited to 02...txt’s last access time and 05...txt’s creation time reveals a temporal inversion. The normalized last access time of 02...txt appears after the normalized creation time of 05...txt. This data incorrectly implies that 05...txt was accessed before 02...txt was created. In a forensic investigation aimed at reconstructing the sequence of events, such discrepancies could lead to incorrect conclusions regarding the chronology of created and accessed files.

Importantly, forensically unsound conclusions based on timestamps normalized by these tools can be avoided by representing normalized timestamps as time intervals rather than single points. For example, normalizing 02...txt’s create timestamp and 05...txt’s last

access timestamp with a time interval (e.g., a single point interval of 2026-04-10 09:07:04.484 CEST and a range interval from 2026-04-10 09:07:04.000 CEST to 2026-04-10 09:07:06.000 CEST, respectively) would lead to the forensically sound conclusion that one cannot determine whether `05...txt` was accessed before or after `02...txt`.

5 Related Work

To our knowledge, timestamp normalization in digital forensics has not been formally studied as a distinct problem, yet our work relates to research in forensic science, temporal logic including models of different temporal granularities, and temporal expression processing.

Within forensic science, connecting artifacts across domains has received some attention. One approach [19] connects time domains, including ones underlying implicit and explicit timing information (e.g., file IDs and file timestamps), but focuses on making implicit timing information available for forensic investigations rather than normalizing timestamps. Another [12] provides a formal framework for reasoning about timelines from multiple trace sources, but presupposes normalized timestamps. Our work establishes a formal foundation for performing timestamp normalization in a forensically sound manner.

Our proposed representation of tracked time, specified by time domains, time domain relations and intervals, is based on instant-based and interval-based formal models of time [1, 21, 42], which have been applied in the legal domain to, for instance, encode transportation regulations [17]. The time domain normalization and resolution definitions (Definitions 6 and 10) relate to time-domain based models of temporal granularity⁵. Prior work, surveyed in [16, 20], proposes the following domain-based approaches. Approach (i) [15] defines a sequence of time domains, ordered from finest to coarsest granularity, where each domain (except the first) maps to a finite interval of the previous domain, covering all its elements. Approach (ii) [14, 31] relaxes this constraint, modeling levels of granularity as disjoint time domains and stipulating a total order on them to express granularity comparisons. Both (i) and (ii) define a mapping from time points of a coarser domain to intervals of time points in a more fine-grained domain, while (ii) includes a reverse mapping from time points of a more fine-grained domain to a single time point in a more coarse domain. Our approach to time domain normalization relates to these mappings (like them, it maps time points to intervals), but it and our notion of granularity (i.e., resolution) differ. In our approach, non-normalized (i.e., unmapped) time points are permitted; during normalization, time points are always mapped to intervals; and differences in granularity are defined based on normalizations. Approach (iii) [6, 7] defines levels of granularity as mappings from an infinite time domain $\mathbb{R}$ to sets of non-overlapping intervals on $\mathbb{R}$, deriving fine-grainedness comparisons from set-theoretic relations and providing interval-to-interval mappings to relate temporal information across different levels of granularity. Normalizing multiple time domains relative to the same time domain results in distinct sets of intervals on the latter that resemble levels of granularity according to (iii). However, their mappings and granularity relations diverge from ours, as they allow reasoning about levels of granularity (i.e., resolution) after the normalization (interval-to-interval level) and not during the normalization (time-point-to-interval level).

Lastly, handling temporal data is not unique to digital forensics. It is addressed in many fields, including natural language processing [13, 39], medicine [27], archaeology [8], ontology languages [2], and operating systems [22]. They aim to provide a uniform format

⁵ In the literature on formal models of temporal granularity, the notion of “granularity”, similar to the notion of “resolution” used in this paper, refers to different units of time (e.g., seconds, minutes, hours).

for dates and temporal expressions, such as ISO 8601 [24]. Our work shares this goal, but differs in scope. We provide a model of forensically sound timestamp normalization across heterogeneous digital systems rather than parsing temporal expressions and dates.

6 Conclusion and Discussion

The analysis of digital devices such as smartphones and laptops is currently indispensable in criminal investigations. Given the critical role of digital investigations, these investigations also carry a significant responsibility to uphold justice, accuracy, and accountability. In order to live up to these values, digital forensic processes must align with the principles of forensic soundness. In particular, it is crucial to ensure that automated forensic software processes are well understood and adhere to these principles.

Summary. This paper contributes to addressing this fundamental challenge by providing and investigating a formalization of forensically sound timestamp normalization (Sections 3.3 and 3.6), which is a key process that enables cross-device and cross-system timestamp analysis. This formalization not only serves as a novel foundation for timestamp normalization and thus a potential basis for future implementations, but also reveals that normalization might lead to incomparable normalized timestamps due to different resolutions (Sections 3.4 and 3.5). Critically, this second aspect is not adhered to by state-of-the-art forensic software, resulting in forensically unsound results (Section 4).

Limitations and Future Work. From a practical standpoint, although our work establishes a theoretical basis for forensically sound timestamp normalization, an implementation is lacking. This should be the subject of future research. Furthermore, timestamp normalization is only one process among many that play a role in timestamp analysis. Therefore, this paper does not address other processes that influence digital investigations in general, or timestamp analysis in particular, which require further formal investigation.

First, the time tracking of digital systems, such as exFAT, can rely on the local time of the device on which it is running. If the local time differs from the actual time, the timestamps must be adjusted to account for this discrepancy [38, 44]. Ideally, future work should specify normalizations that account for discrepancies between local time and actual time by extending proposed methods like using so-called time anchors [43]. Second, obtained timestamps might be ambiguous or incomplete. For example, so-called naive timestamps miss timezone information [26, 37] or unknown date formats may cause dates like 03-02-2026 to be interpreted as either February 3, 2026 or March 2, 2026. To correctly analyze such ambiguous or incomplete timestamps, they must be interpreted in the context of other artifacts such as location data or system data. Future work should address this aspect in order to lay the groundwork for such implementations through a formal model of timestamp contexts. Third, anti-forensics measures like the manipulation of timestamps is a non-negligible possibility that casts doubt on the reliability of digital investigations involving timestamps [32]. Although several methods for detecting tampered timestamps have already been proposed [34], there is no formal framework for verifying obtained timestamps, which ideally aligns the challenge of completing incomplete timestamps. Therefore, future work should aim to establish such a framework for systematically evaluating the trustworthiness of timestamps.

References

- 1 James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983. doi:10.1145/182.358434.
- 2 Eleftherios Anagnostopoulos, Sotiris Batsakis, and Euripides G.M. Petrakis. CHRONOS: A reasoning engine for qualitative temporal information in OWL. *Procedia Computer Science*, 22:70–77, 2013. doi:10.1016/j.procs.2013.09.082.
- 3 Apple. Apple file system reference. Accessed: March 4, 2026. URL: <https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf>.
- 4 André Årnes, editor. *Digital Forensics*. Wiley, 1 edition, 2017. doi:10.1002/9781119262442.
- 5 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. The MIT Press, 2008.
- 6 Claudio Bettini, Sushil Jajodia, and X. Sean Wang. *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Springer, Berlin, Heidelberg, 2000. doi:10.1007/978-3-662-04228-1.
- 7 Claudio Bettini, X. Sean Wang, and Sushil Jajodia. A general framework for time granularity and its application to temporal reasoning. *Annals of Mathematics and Artificial Intelligence*, 22(1-2):29–58, 1998. doi:10.1023/A:1018938007511.
- 8 Ceri Binding and Douglas Tudhope. Automatic normalization of temporal expressions. *Journal of Computer applications in archaeology*, 6:24–39, 2023. doi:10.5334/jcaa.105.
- 9 Frank Breiting, Hudan Studiawan, and Chris Hargreaves. SoK: Timeline based event reconstruction for digital forensics: Terminology, methodology, and current challenges. *Forensic Science International: Digital Investigation*, 53:301932, 2025. doi:10.1016/j.fsidi.2025.301932.
- 10 Brian Carrier. *File system forensic analysis*. Addison-Wesley Professional, 2005.
- 11 Eoghan Casey. *Digital evidence and computer crime: forensic science, computers and the Internet*. Academic Press, 2011.
- 12 Yoan Chabot, Aurélie Bertaux, Christophe Nicolle, and M-Tahar Kechadi. A complete formalized knowledge representation model for advanced digital forensics timeline analysis. *Digital Investigation*, 11:S95–S105, 2014. doi:10.1016/j.diin.2014.05.009.
- 13 Angel X. Chang and Christopher D. Manning. SUTime: A library for recognizing and normalizing time expressions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC’12)*, pages 3735–3740. European Language Resources Association (ELRA), 2012. URL: <https://nlp.stanford.edu/pubs/lrec2012-sutime.pdf>.
- 14 E. Ciapessoni, E. Corsetti, A. Montanari, and P. San Pietro. Embedding time granularity in a logical specification language for synchronous real-time systems. *Science of Computer Programming*, 20(1-2):141–171, 1993. doi:10.1016/0167-6423(93)90025-K.
- 15 James Clifford and Ahobala Rao. A Simple General Structure for Temporal Domains. *Temporal Aspects of Information Systems*, pages 17–28, 1988.
- 16 Carlo Combi, Massimo Franceschet, and Adriano Peron. Representing and Reasoning about Temporal Granularities. *Journal of Logic and Computation*, 14(1):51–77, 2004. doi:10.1093/logcom/14.1.51.
- 17 Ana De Almeida Borges, Juan José Conejero Rodríguez, David Fernández-Duque, Mireia González Bedmar, and Joost J. Joosten. To drive or not to drive: A logical and computational analysis of European transport regulations. *Information and Computation*, 280:104636, 2021. doi:10.1016/j.ic.2020.104636.
- 18 Discord. API references. Accessed: March 4, 2026. URL: <https://docs.discord.com/developers/reference>.
- 19 Lisa Marie Dreier, Céline Vanini, Christopher J. Hargreaves, Frank Breiting, and Felix Freiling. Beyond timestamps: Integrating implicit timing information into digital forensic timelines. *Forensic Science International: Digital Investigation*, 49:301755, 2024. doi:10.1016/j.fsidi.2024.301755.
- 20 Jérôme Euzenat and Angelo Montanari. Time Granularity. In *Foundations of Artificial Intelligence*, volume 1, pages 59–118. Elsevier, 2005. doi:10.1016/S1574-6526(05)80005-7.

- 21 Valentin Goranko and Antje Rumberg. Temporal Logic. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2025 edition, 2025.
- 22 The Open Group. Posix.1-2024. Accessed: March 4, 2026. URL: <https://pubs.opengroup.org/onlinepubs/9799919799/>.
- 23 Kristinn Guðjónsson. Mastering the Super Timeline With log2timeline, 2010. URL: <https://www.sans.org/white-papers/33438>.
- 24 International Organization for Standardization (ISO). ISO 8601: Date and time — Representations for information interchange — Part 1: Basic rules, 2019. Accessed: March 4, 2026. URL: <https://www.iso.org/standard/70907.html>.
- 25 International Telecommunication Union (ITU). Recommendation ITU-R TF.460-6: Standard-frequency and time-signal emissions, 2002. Accessed: March 4, 2026. URL: https://www.itu.int/dms_pubrec/itu-r/rec/tf/R-REC-TF.460-6-200202-I!!PDF-E.pdf.
- 26 M. Kaart and S. Laraghy. Android forensics: Interpretation of timestamps. *Digital Investigation*, 11(3):234–248, 2014. doi:10.1016/j.diin.2014.05.001.
- 27 Mohcine Madkour, Driss Benhaddou, and Cui Tao. Temporal data representation, normalization, extraction, and reasoning: A review from clinical domain. *Computer Methods and Programs in Biomedicine*, 128:52–68, 2016. doi:10.1016/J.CMPB.2016.02.007.
- 28 Joachim Metz. Pearls and pitfalls of timeline analysis, 2021. Accessed: March 4, 2026. URL: <https://osdfir.blogspot.com/2021/10/pearls-and-pitfalls-of-timeline-analysis.html>.
- 29 Microsoft. exFAT file system specification. Accessed: March 4, 2026. URL: <https://learn.microsoft.com/en-us/windows/win32/fileio/exfat-specification>.
- 30 Microsoft. File times. Accessed: March 4, 2026. URL: <https://learn.microsoft.com/en-us/windows/win32/sysinfo/file-times>.
- 31 Angelo Montanari. *Metric and Layered Temporal Logic for Time Granularity*. PhD thesis, Universiteit van Amsterdam, 1996. URL: <https://eprints.illc.uva.nl/id/eprint/1995/4/DS-1996-02.text.pdf>.
- 32 Christopher Neale. Fool me once: A systematic review of techniques to authenticate digital artefacts. *Forensic Science International: Digital Investigation*, 45:301516, 2023. doi:10.1016/j.fsidi.2023.301516.
- 33 Netherlands Forensic Institute. Hansken Digital Evidence Platform. Accessed: April 15, 2026. URL: <https://www.hansken.nl/>.
- 34 Junghoon Oh, Sangjin Lee, and Hyunuk Hwang. Forensic Detection of Timestamp Manipulation for Digital Forensic Investigation. *IEEE Access*, 12:72544–72565, 2024. doi:10.1109/ACCESS.2024.3395644.
- 35 Log2Timeline Project. Plaso: Python Logfile Analysis and SuperTimeline Analysis. Accessed: April 15, 2026. URL: <https://github.com/log2timeline/plaso>.
- 36 The Sleuth Kit Project. The Sleuth Kit. Accessed: April 15, 2026. URL: <https://sleuthkit.org/sleuthkit/>.
- 37 Sriram Raghavan and Huzur Saran. UniTIME: Timestamp interpretation engine for developing unified timelines. In *2013 8th International Workshop on Systematic Approaches to Digital Forensics Engineering (SADFE)*, pages 1–7. IEEE, 2013. doi:10.1109/SADFE.2013.6911546.
- 38 Bhupendra Singh and Gaurav Gupta. Analyzing Windows Subsystem for Linux Metadata to Detect Timestamp Forgery. In Gilbert Peterson and Sujeet Shenoi, editors, *Advances in Digital Forensics XV*, volume 569, pages 159–182. Springer International Publishing, 2019. doi:10.1007/978-3-030-28752-8_9.
- 39 Jannik Strötgen and Michael Gertz. Heidelberg: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324. Association for Computational Linguistics, 2010. URL: <https://aclanthology.org/S10-1071/>.

- 40 Celine Taylor Parkins-Ozephius. Syntax error! Een verkenning van betrouwbaarheidsconcepten bij digitaal bewijs in strafzaken. *Expertise & Recht*, 2025(4), 2025. URL: <https://www.uitgeverijparis.nl/nl/reader/2000000122/2000001179>.
- 41 H.M.A. van Beek, E.J. van Eijk, R.B. van Baar, M. Ugen, J.N.C. Bodde, and A.J. Siemelink. Digital forensics as a service: Game on. *Digital Investigation*, 15:20–38, 2015. doi:10.1016/j.diin.2015.07.004.
- 42 Johan van Benthem. *The Logic of Time*. Kluwer Academic Publishers, 2 edition, 1991.
- 43 Céline Vanini, Christopher J. Hargreaves, Harm Van Beek, and Frank Breiteringer. Was the clock correct? Exploring timestamp interpretation through time anchors for digital forensic event reconstruction. *Forensic Science International: Digital Investigation*, 49:301759, 2024. doi:10.1016/j.fsidi.2024.301759.
- 44 Svein Willassen. Hypothesis-Based Investigation of Digital Timestamps. In Indrajit Ray and Sujeet Sheno, editors, *Advances in Digital Forensics IV*, volume 285, pages 75–86. Springer US, 2008. doi:10.1007/978-0-387-84927-0_7.

A Proofs of Propositions

► **Proposition 5.** *Let D be any time domain, $\preceq$ be its time domain relation, and $\preceq_D^{\mathcal{I}} \subseteq \mathcal{I}_D^{\prec} \times \mathcal{I}_D^{\prec}$ be its interval relation.*

- (1) $\preceq_D^{\mathcal{I}}$ is transitive and antisymmetric.
- (2) If $|D| = 1$, then $\preceq_D^{\mathcal{I}}$ is reflexive and total.
- (3) If $|D| > 1$, then $\preceq_D^{\mathcal{I}}$ is neither reflexive nor total.

Proof. For (1), we need to show that $\preceq_D^{\mathcal{I}}$ is transitive and antisymmetric. For the former, let $[t_1, t_2], [t_3, t_4], [t_5, t_6] \in \mathcal{I}_D^{\prec}$ be arbitrary. Suppose that $[t_1, t_2] \preceq_D^{\mathcal{I}} [t_3, t_4]$ and $[t_3, t_4] \preceq_D^{\mathcal{I}} [t_5, t_6]$. This implies by Definition 4 that $t_2 \preceq t_3$ and $t_4 \preceq t_5$. According to Definition 2, from $[t_3, t_4] \in \mathcal{I}_D^{\prec}$ it follows that $t_3 \preceq t_4$. Consequently, $t_2 \preceq t_3 \preceq t_4 \preceq t_5$. This means according to Definition 4, $[t_1, t_2] \preceq_D^{\mathcal{I}} [t_5, t_6]$. Further, to show that $\preceq_D^{\mathcal{I}}$ is anti-symmetric, let $[t_1, t_2], [t_3, t_4] \in \mathcal{I}_D^{\prec}$ be arbitrary. Suppose that $[t_1, t_2] \preceq_D^{\mathcal{I}} [t_3, t_4]$ and $[t_3, t_4] \preceq_D^{\mathcal{I}} [t_1, t_2]$. From Definitions 2 and 4, the former implies that $t_1 \preceq t_2 \preceq t_3 \preceq t_4$, while the latter implies that $t_3 \preceq t_4 \preceq t_1 \preceq t_2$. Since $\preceq$ is a time domain relation and hence transitive and antisymmetric (see Section 3.1), it follows that $t_1 = t_2 = t_3 = t_4$. This implies that $[t_1, t_2] = [t_3, t_4]$.

For (2), let $|D| = 1$. This means that $\mathcal{I}_D^{\prec} = \{[t, t]\}$ with $t \in D$ and $\preceq_D^{\mathcal{I}} = \{([t, t], [t, t])\}$. It clearly holds that $\preceq_D^{\mathcal{I}}$ is reflexive and total.

For (3), let $|D| > 1$. We need to prove that $\preceq_D^{\mathcal{I}}$ on $\mathcal{I}_D^{\prec}$ is neither reflexive nor total. For the former, it suffices to show that $[t_1, t_2] \in \mathcal{I}_D^{\prec}$ with $[t_1, t_2] \not\preceq_D^{\mathcal{I}} [t_1, t_2]$, while for the latter it suffices to show that $[t_1, t_2], [t'_1, t'_2] \in \mathcal{I}_D^{\prec}$ with $[t_1, t_2] \not\preceq_D^{\mathcal{I}} [t'_1, t'_2]$ and $[t'_1, t'_2] \not\preceq_D^{\mathcal{I}} [t_1, t_2]$. Since $|D| > 1$ and $\preceq$ is a total order, there exist $t_3, t_4 \in D$ with $t_3 \prec t_4$. By Definition 2, $[t_3, t_4] \in \mathcal{I}_D^{\prec}$. Since $t_3 \prec t_4$ and hence $t_4 \not\preceq t_3$, Definition 4 implies that $[t_3, t_4] \not\preceq_D^{\mathcal{I}} [t_3, t_4]$. This means that $\preceq_D^{\mathcal{I}}$ is not reflexive. However, then there exist $[t_1, t_2], [t'_1, t'_2] \in \mathcal{I}_D^{\prec}$ (by picking $[t_1, t_2] = [t'_1, t'_2] = [t_3, t_4]$) such that $[t_1, t_2] \not\preceq_D^{\mathcal{I}} [t'_1, t'_2]$ and $[t'_1, t'_2] \not\preceq_D^{\mathcal{I}} [t_1, t_2]$. Therefore, $\preceq_D^{\mathcal{I}}$ is not total. ◀

► **Proposition 8.** *Let D and T be time domains, $\preceq_D$ and $\preceq_T$ be their time domain order, and $n : D \rightarrow \mathcal{I}_T^{\prec}$ be any normalization of D relative to T . If $D \cap T \neq \emptyset$, then for any $t_1, t_2 \in D \cap T$, $t_1 \preceq_D t_2 \iff n(t_1) \preceq_T n(t_2)$.*

Proof. Suppose that $t_1, t_2 \in D \cap T$. By constraint (iv) of Definition 6, $n(t_1) = [t_1, t_1]$ and $n(t_2) = [t_2, t_2]$. Further, note that according to the specification of time domain orders (Section 3.1), $t_1, t_2 \in D \cap T$ implies that $t_1 \preceq_D t_2$ if and only if $t_1 \preceq_T t_2$.

For the left-to-right direction, suppose that $t_1 \preceq_D t_2$. Then by the specification of time domain orders, $t_1 \preceq_T t_2$. Then by Definition 4 it follows that $[t_1, t_1] \preceq_T^{\mathcal{I}} [t_2, t_2]$. Consequently, $n(t_1) \preceq_T^{\mathcal{I}} n(t_2)$. For the right-to-left direction, suppose that $n(t_1) \preceq_T^{\mathcal{I}} n(t_2)$ and hence $[t_1, t_1] \preceq_T^{\mathcal{I}} [t_2, t_2]$. Definition 4 implies that $t_1 \preceq_T t_2$. By the specification of time domain orders it follows that $t_1 \preceq_D t_2$. $\blacktriangleleft$

► **Proposition 11.** *Let D and T be time domains, while $n : D \rightarrow \mathcal{I}_T^{\mathcal{I}}$ be a non-empty normalization of D relative to T with $\text{Dom}(n) = \{t \in D \mid n(t) \text{ defined}\}$. For any $t \in \text{Dom}(n)$,*

- (1) $D \triangleleft_n^t T$ if and only if $T \not\triangleleft_n^t D$;
- (2) if $T \triangleleft_n^t D$ then $T \preceq_n^t D$;
- (3) if $T \triangleleft_n^t D$ then $D \not\triangleleft_n^t T$;
- (4) if $D \triangleleft_n^{\top} T$ then $T \not\triangleleft_n^{\top} D$;
- (5) if $T \triangleleft_n^{\top} D$ then $T \preceq_n^{\top} D$;
- (6) if $T \triangleleft_n^{\top} D$ then $D \not\triangleleft_n^{\top} T$.

Proof. For proposition (1), note that from Definition 6 it follows that for $t \in \text{Dom}(n)$, $|n(t)| \geq 1$. Consequently, $|n(t)| = 1$ if and only if $|n(t)| \not\geq 1$. By Definition 10 this means that $D \triangleleft_n^t T$ if and only if $T \not\triangleleft_n^t D$.

For proposition (2), suppose that $T \triangleleft_n^t D$ for $t_1 \in \text{Dom}(n)$. According to Definition 10, $T \triangleleft_n^t D$ means that there exists $t_2 \in \text{Dom}(n)$ such that $t_1 \neq t_2$ and $n(t_1) = n(t_2)$. Note that it needs to be shown that $T \not\triangleleft_n^t D$ and hence that $|n(t_1)| = 1$. For the sake of contradiction, assume that $|n(t_1)| > 1$. Therefore, $n(t_1) = [t_3, t_4]$ with $t_3, t_4 \in T$ and $t_3 \neq t_4$. Since $\preceq_T$ is a total order and $t_3 \neq t_4$, $t_3 \prec_T t_4$ or $t_4 \prec_T t_3$. Thus, by Definition 4, $[t_3, t_4] \not\preceq_T^{\mathcal{I}} [t_3, t_4]$. Further, $n(t_2) = [t_3, t_4]$ because $n(t_1) = n(t_2)$. By constraint (ii) of Definition 6, $n(t_1) \preceq_T^{\mathcal{I}} n(t_2)$ or $n(t_2) \preceq_T^{\mathcal{I}} n(t_1)$ because $t_1 \neq t_2$. This implies that in any of the two case, $[t_3, t_4] \preceq_T^{\mathcal{I}} [t_3, t_4]$. This is a contradiction. Therefore, $|n(t_1)| = 1$ as wanted.

To prove proposition (3), suppose that $T \triangleleft_n^t D$. By proposition (2), $T \preceq_n^t D$. Then by proposition (1), $D \not\triangleleft_n^t T$, as wanted.

To prove (4) - (6), note that n is non-empty. This implies that $|\text{Dom}(n)| \geq 1$. As this holds, (4) follows from (1), (5) follows from (2), and (6) follows from (3). $\blacktriangleleft$

► **Proposition 14.** *Given the time domains D_1 , D_2 , and T such that $D_1 = D_2 = T$, then for any normalizations n_1 and n_2 of D_1 and D_2 relative to T respectively, $n_1(t_1) \preceq_T^{\mathcal{I}} n_2(t_2)$ or $n_2(t_2) \preceq_T^{\mathcal{I}} n_1(t_1)$ for any $t_1 \in D_1$ and $t_2 \in D_2$.*

Proof. Let n_1 and n_2 be arbitrary normalization of D_1 and D_2 relative to T , respectively. Let $t_1, t_2 \in D_1 = D_2 = D$ be arbitrary. It suffices to show that $n_1(t_1) \preceq_T^{\mathcal{I}} n_2(t_2)$ or $n_2(t_2) \preceq_T^{\mathcal{I}} n_1(t_1)$. Since $\preceq_T$ on T is a total order, $t_1 \preceq_T t_2$ or $t_2 \preceq_T t_1$. In any of the two cases, Definition 4 implies that $[t_1, t_1] \preceq_T^{\mathcal{I}} [t_2, t_2]$ or $[t_2, t_2] \preceq_T^{\mathcal{I}} [t_1, t_1]$. Since $D_1 = D_2 = T$, constraint (4) of Definition 6 implies that $n_1(t_1) = [t_1, t_1]$ and $n_2(t_2) = [t_2, t_2]$. This means that $n_1(t_1) \preceq_T^{\mathcal{I}} n_2(t_2)$ or $n_2(t_2) \preceq_T^{\mathcal{I}} n_1(t_1)$. $\blacktriangleleft$

► **Proposition 15.** *Let D_1 , D_2 , and T be time domains, and n_1 and n_2 be normalization of D_1 and D_2 relative to T , respectively.*

- (1) *If $T \triangleleft_{n_1}^{\top} D_1$ and $T \triangleleft_{n_2}^{\top} D_2$, then $n_1(t_1) \preceq_T^{\mathcal{I}} n_2(t_2)$ or $n_2(t_2) \preceq_T^{\mathcal{I}} n_1(t_1)$ for any $t_1 \in D_1$ and $t_2 \in D_2$ with $n_1(t_1)$ and $n_2(t_2)$ defined.*

- (2) If $T \leq_{n_1}^\top D_1$ and $D_2 \triangleleft_{n_2}^\top T$ where $|n_2(t)| = 2$ for any $t \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$, then $n_1(t_1) \preceq_T^\mathcal{I} n_2(t_2)$ or $n_2(t_2) \preceq_T^\mathcal{I} n_1(t_1)$ for any $t_1 \in D_1$ and $t_2 \in D_2$ with $n_1(t_1)$ and $n_2(t_2)$ defined.

Proof. To prove statement (1), let $t_1 \in \{t \in D_1 \mid n_1(t) \text{ defined}\}$ and $t_2 \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$ be arbitrary. The fact that $T \leq_{n_1}^\top D_1$ and $T \leq_{n_2}^\top D_2$ imply by Definition 10 that $|n_1(t_1)| = 1$ and $|n_2(t_2)| = 1$. This means that $n_1(t_1) = [t_3, t_3]$ and $n_2(t_2) = [t_4, t_4]$ for some $t_3, t_4 \in T$. Since $\preceq_T$ on T is a total order, $t_3 \preceq t_4$ or $t_4 \preceq t_3$. By Definition 4, $[t_3, t_3] \preceq_T^\mathcal{I} [t_4, t_4]$ or $[t_4, t_4] \preceq_T^\mathcal{I} [t_3, t_3]$. It follows that $n_1(t_1) \preceq_T^\mathcal{I} n_2(t_2)$ or $n_2(t_2) \preceq_T^\mathcal{I} n_1(t_1)$.

To prove statement (2), let $t_1 \in \{t \in D_1 \mid n_1(t) \text{ defined}\}$ and $t_2 \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$ be arbitrary. Since $T \leq_{n_1}^\top D_1$, $|n_1(t_1)| = 1$ by Definition 10. This means that $n_1(t_1) = [t_3, t_3]$ for some $t_3 \in T$. In contrast, $|n_2(t)| = 2$ for any $t \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$. Therefore, $n_2(t_2) = [t_4, t_5] = \{t_4, t_5\}$ for some $t_4, t_5 \in D$ with $t_4 \prec_T t_5$ and for any $t \in T$ with $t \neq t_4$ and $t \neq t_5$, $t \prec_T t_4$ or $t_5 \prec_T t$. Since $\preceq_T$ on T is a total order, one of the following cases holds: $t_3 \preceq_T t_4 \prec_T t_5$ or $t_4 \prec_T t_5 \preceq_T t_3$. By Definition 4, it follows that in the first case $n_1(t_1) \preceq_T^\mathcal{I} n_2(t_2)$, while in the second case $n_2(t_2) \preceq_T^\mathcal{I} n_1(t_1)$. ◀

► **Proposition 16.** Let D_1, D_2 and T be time domains, and n_1 and n_2 be gap-free normalizations of D_1 and D_2 relative to T , respectively.

- (1) If $T \leq_{n_1}^\top D_1$, $D_2 \triangleleft_{n_2}^\top T$ where $|n_2(t)| \geq 3$ for any $t \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$, and $|\{t \in T \mid t \in n_1(t'_1) \text{ for some } t'_1 \in D_1\} \cap \{t \in T \mid t \in n_2(t'_2) \text{ for some } t'_2 \in D_2\}| \geq 3$, then there exist $t_1 \in D_1$ and $t_2 \in D_2$ such that $n_1(t_1) \not\preceq_T^\mathcal{I} n_2(t_2)$ and $n_2(t_2) \not\preceq_T^\mathcal{I} n_1(t_1)$.
- (2) If $D_1 \triangleleft_{n_1}^\top T$ where $|n_1(t)| \geq 3$ for any $t \in \{t \in D_1 \mid n_1(t) \text{ defined}\}$, $D_2 \triangleleft_{n_2}^\top T$ where $|n_2(t)| \geq 3$ for any $t \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$, and $|\{t \in T \mid t \in n_1(t'_1) \text{ for some } t'_1 \in D_1\} \cap \{t \in T \mid t \in n_2(t'_2) \text{ for some } t'_2 \in D_2\}| \geq 2$, then there exist $t_1 \in D_1$ and $t_2 \in D_2$ such that $n_1(t_1) \not\preceq_T^\mathcal{I} n_2(t_2)$ and $n_2(t_2) \not\preceq_T^\mathcal{I} n_1(t_1)$.

Proof. For statement (1) note that since $|\{t \in T \mid t \in n_1(t'_1) \text{ for some } t'_1 \in D_1\} \cap \{t \in T \mid t \in n_2(t'_2) \text{ for some } t'_2 \in D_2\}| \geq 3$ and n_1 and n_2 are gap-free, there exist $t_4, t_5, t_6 \in T$ such that (i) $t_4 \prec_T t_5 \prec_T t_6$, (ii) for $t \in \{t_4, t_5, t_6\}$, there exist $t'_1 \in D_1$ and $t'_2 \in D_2$ such that $t \in n_1(t'_1)$ and $t \in n_2(t'_2)$, and (iii) for any $t \in T$ with $t \neq t_4$, $t \neq t_5$, and $t \neq t_6$, $t \prec_T t_4$ or $t_6 \prec_T t$. Further, since $D \leq_{n_1}^\top D_1$, it follows by Definition 10 that $|n_1(t)| = 1$ for any $t \in \{t \in D_1 \mid n_1(t) \text{ defined}\}$. Consequently, there exist $t'_4, t'_5, t'_6 \in D_1$ such that $n_1(t'_4) = [t_4, t_4]$, $n_1(t'_5) = [t_5, t_5]$, and $n_1(t'_6) = [t_6, t_6]$. Further, due to fact that $|n_2(t)| \geq 3$ for any $t \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$, (ii) and (iii), one of following cases holds:

- There exists $t_7 \in D_2$ such that $n_2(t_7) = [t_8, t_9]$ with $t_4, t_5, t_6 \in [t_8, t_9]$. This means that $t_8 \preceq_T t_4$ and $t_6 \preceq_T t_9$. By (i) it follows that $t_8 \prec_T t_5 \prec_T t_9$. Proposition 13 implies that then $n_1(t'_5) \not\preceq_T^\mathcal{I} n_2(t_7)$ and $n_2(t_7) \not\preceq_T^\mathcal{I} n_1(t'_5)$.
- There exist $t_7, t'_7 \in D_2$ such that $n_2(t_7) = [t_8, t_4]$ and $n_2(t'_7) = [t_5, t_9]$ with $t_6 \in [t_5, t_9]$. Since $|n_2(t)| \geq 3$ for any $t \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$, $|n_2(t'_7)| = |[t_5, t_9]| \geq 3$. By (iii), $t_6 \prec_T t_9$. It follows that $t_5 \prec_T t_6 \prec_T t_9$. Proposition 13 implies that then $n_1(t'_6) \not\preceq_T^\mathcal{I} n_2(t'_7)$ and $n_2(t'_7) \not\preceq_T^\mathcal{I} n_1(t'_6)$.
- There exist $t_7, t'_7 \in D_2$ such that $n_2(t_7) = [t_8, t_5]$ and $n_2(t'_7) = [t_6, t_9]$ with $t_4 \in [t_8, t_5]$. By analogous reasoning as in the previous case, $n_1(t'_4) \not\preceq_T^\mathcal{I} n_2(t_7)$ and $n_2(t_7) \not\preceq_T^\mathcal{I} n_1(t'_4)$.

In any of the cases it holds that there exist $t_1 \in D_1$ and $t_2 \in D_2$ with $n_1(t_1) \not\preceq_T^\mathcal{I} n_2(t_2)$ and $n_2(t_2) \not\preceq_T^\mathcal{I} n_1(t_1)$ as wanted.

To prove (2), let us first prove the following statement.

► **Statement 17.** *Let D be a time domain with $\preceq$ being the time domain relation of D . For any $[t_1, t_2], [t_3, t_4] \in \mathcal{I}_D^\preceq$ with $|[t_1, t_2]| \geq 2$, $|[t_3, t_4]| \geq 2$ and $[t_1, t_2] \cap [t_3, t_4] \geq 2$, $[t_1, t_2] \not\preceq_D^{\mathcal{I}} [t_3, t_4]$ and $[t_3, t_4] \not\preceq_D^{\mathcal{I}} [t_1, t_2]$.*

Proof. To prove this, let $[t_1, t_2], [t_3, t_4] \in \mathcal{I}_D^\preceq$ satisfy the aforementioned conditions. Since $|[t_1, t_2] \cap [t_3, t_4]| \geq 2$, there exist $t_5, t_6 \in D$ such that $t_5 \prec t_6$ with $t_5, t_6 \in [t_1, t_2]$ and $t_5, t_6 \in [t_3, t_4]$. Further, since $|[t_1, t_2]| \geq 2$ and $|[t_3, t_4]| \geq 2$, $t_1 \preceq t_5 \prec t_6 \preceq t_2$ and $t_3 \preceq t_5 \prec t_6 \preceq t_4$. Suppose towards contradiction that $[t_1, t_2] \preceq_D^{\mathcal{I}} [t_3, t_4]$. This implies that $t_5 \prec t_6 \preceq t_2 \preceq t_3 \preceq t_5 \prec t_6$. So that $t_6 \preceq t_5$ and $t_6 \not\preceq t_5$ (because $t_5 \prec t_6$) which is a contradiction. Therefore, $[t_1, t_2] \not\preceq_D^{\mathcal{I}} [t_3, t_4]$. By analogous reasoning, one obtains that $[t_3, t_4] \not\preceq_D^{\mathcal{I}} [t_1, t_2]$. ◀

Given this result, we can then prove (2). Note that since $|\{t \in T \mid t \in n_1(t'_1) \text{ for some } t'_1 \in D_1\} \cap \{t \in T \mid t \in n_2(t'_2) \text{ for some } t'_2 \in D_2\}| \geq 2$ and n_1 and n_2 are gap-free, there exist $t_4, t_5 \in T$, such that (i) $t_4 \prec_T t_5$, (ii) for $t \in \{t_4, t_5\}$, there exist $t'_1 \in D_1$ and $t'_2 \in D_2$ such that $t \in n_1(t'_1)$ and $t \in n_2(t'_2)$, and (iii) for any $t \in T$ with $t \neq t_4$ and $t \neq t_5$, $t \prec_T t_4$ or $t_5 \prec_T t$. This implies that one of the following cases holds:

- There exist $t_7 \in D_1$ and $t_8 \in D_2$ such that $t_4, t_5 \in n_1(t_7)$ and $t_4, t_5 \in n_2(t_8)$. By Statement 17, it then follows that $n_1(t_7) \not\preceq_T^{\mathcal{I}} n_2(t_8)$ and $n_2(t_8) \not\preceq_T^{\mathcal{I}} n_1(t_7)$.
- There exist $t_7, t'_7 \in D_1$ and $t_8 \in D_2$ such that $n_1(t_7) = [t_9, t_4]$, $n_1(t'_7) = [t_5, t_{10}]$ and $t_4, t_5 \in n_2(t_8)$. Since n_1 and n_2 are gap-free, $|n_1(t)| \geq 3$ for any $t \in \{t \in D_1 \mid n_1(t) \text{ defined}\}$, and $|n_2(t)| \geq 3$ for any $t \in \{t \in D_2 \mid n_2(t) \text{ defined}\}$, $|n_1(t_7) \cap n_2(t_8)| \geq 2$ or $|n_1(t'_7) \cap n_2(t_8)| \geq 2$. It follows by Statement 17 that $n_1(t_7) \not\preceq_T^{\mathcal{I}} n_2(t_8)$ and $n_2(t_8) \not\preceq_T^{\mathcal{I}} n_1(t_7)$, or $n_1(t'_7) \not\preceq_T^{\mathcal{I}} n_2(t_8)$ and $n_2(t_8) \not\preceq_T^{\mathcal{I}} n_1(t'_7)$.
- There exist $t_7 \in D_1$ and $t_8, t'_8 \in D_2$ such that $t_4, t_5 \in n_1(t_7)$, $n_2(t_8) = [t_{11}, t_4]$, and $n_2(t'_8) = [t_5, t_{12}]$. By analogous reasoning as in the previous case, $n_1(t_7) \not\preceq_T^{\mathcal{I}} n_2(t_8)$ and $n_2(t_8) \not\preceq_T^{\mathcal{I}} n_1(t_7)$, or $n_1(t_7) \not\preceq_T^{\mathcal{I}} n_2(t'_8)$ and $n_2(t'_8) \not\preceq_T^{\mathcal{I}} n_1(t_7)$.
- There exist $t_7, t'_7 \in D_1$ and $t_8, t'_8 \in D_2$ such that $n_1(t_7) = [t_9, t_4]$, $n_1(t'_7) = [t_5, t_{10}]$, $n_2(t_8) = [t_{11}, t_4]$, and $n_2(t'_8) = [t_5, t_{12}]$. By analogous reasoning as in the previous cases, $n_1(t_7) \not\preceq_T^{\mathcal{I}} n_2(t_8)$ and $n_2(t_8) \not\preceq_T^{\mathcal{I}} n_1(t_7)$, and $n_1(t'_7) \not\preceq_T^{\mathcal{I}} n_2(t'_8)$ and $n_2(t'_8) \not\preceq_T^{\mathcal{I}} n_1(t'_7)$.

Therefore, in any of the cases it holds that there exist $t_1 \in D_1$ and $t_2 \in D_2$ with $n_1(t_1) \not\preceq_D^{\mathcal{I}} n_2(t_2)$ and $n_2(t_2) \not\preceq_T^{\mathcal{I}} n_1(t_1)$ as wanted. ◀

Decidable Reasoning About Time in Finite-Domain Situation Calculus Theories

Till Hofmann 

RWTH Aachen University, Germany

University of Oxford, UK

Stefan Schupp 

TU Wien, Austria

Gerhard Lakemeyer 

RWTH Aachen University, Germany

Abstract

Representing time is crucial for cyber-physical systems and has been studied extensively in the situation calculus. The most commonly used approach represents time by adding a real-valued function $\text{time}(a)$ that attaches a time point to each action and consequently to each situation. We show that in this approach, checking whether there is a reachable situation that satisfies a given formula is undecidable, even when the domain contains only finitely many objects. We present an alternative approach based on well-established results from timed automata theory by introducing clocks as real-valued fluents with restricted successor state axioms and comparison operators. With this restriction, we can show that the reachability problem for finite-domain basic action theories is decidable. Finally, we apply our results to Golog program realization by presenting a decidable procedure for determining an action sequence that is a successful execution of a given program.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Logic and verification

Keywords and phrases Situation Calculus, Golog, timed automata, clocks, dense time, reachability, decidability, region abstraction, program realization

Digital Object Identifier 10.4230/OASICS.TIME.2026.11

Related Version *Extended Version*: <http://arxiv.org/abs/2402.03164> [19]

Funding This research was supported by the EU ICT-48 2020 project TAILOR (No 952215). Travel funding to present this paper was provided by the EU project euroFMX (No 101299128).

1 Introduction

Time is a ubiquitous concept in various applications, especially in cases where digital systems interact with the real world such as applications in robotics or generally in cyber-physical systems. Early approaches model discrete time, which allows modeling and reasoning about system states at discrete points in time. This approach gives rise to the problem of choosing an appropriate granularity (resolution) of time to facilitate useful reasoning. The introduction of models which incorporate continuous and dense notions of time circumvents this problem and allows for more expressive modeling and analysis approaches. Dense time has been under extensive investigation in the formal methods community, e.g., in the analysis of hybrid systems which started with the introduction of timed automata (TAs) [3].

The situation calculus (SC) is a well-known first-order formalism for reasoning about dynamically changing worlds, where each change in the world is caused by an *action*. Among others, it permits a representation of continuous time by attaching a time of occurrence to each action and hence also each situation [26, 28]. In this paper, we focus on *reachability*, i.e., checking whether there is a reachable situation where some formula ϕ holds. We show that the standard representation of time already makes the reachability problem undecidable,



© Till Hofmann, Stefan Schupp, and Gerhard Lakemeyer;
licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 11; pp. 11:1–11:22

OpenAccess Series in Informatics



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

even when the domain contains only finitely many objects. To recover decidability, we adapt the core techniques of timed automata theory to the situation calculus: we introduce clocks as real-valued fluents that may only be reset or advanced uniformly and compared against natural numbers, and we show that region abstraction, adapted to situations as a time-abstract bisimulation, yields a finite abstraction of the otherwise infinite tree of situations. This transfer lets us decide reachability for clocked basic action theories over finite domains. We then extend this result to GOLOG programs and investigate *program realization*. Given a GOLOG program, we present a decidable method for determining a corresponding action sequence that results in a final program configuration.

Reachability and program realization in this restricted setting are closely related to recent work on synthesis for GOLOG programs. Reactive synthesis has been studied for the propositional situation calculus [10] and for first-order agent programs against uncontrollable environments with LTL_f goals [18]. Our realization procedure complements these: rather than synthesizing a strategy against an adversarial environment, we focus on the fully controllable setting, but extend the formalism with continuous time.

This paper is organized as follows: We discuss related work in Section 2 and present the foundations of the SC in Section 3. We discuss reachability in Section 4, where we show undecidability for the existing approaches and then show a decidable approach in Section 5. In Section 6, we apply these results to program realization and conclude in Section 7.

2 Related Work

Already McCarthy [21] proposed a fluent function *time* that gives the value of time in a situation. Elaborating on this idea, Gelfond et al. [16] described an extension where time is a fluent with values from the integers or real numbers and where each action has a duration. Miller [22] attached time to each action and allowed overlapping actions with durations, along with a partial ordering of actions that allows describing two concurrent sequences of actions. Pinto and Reiter [26] modeled durative actions with instantaneous *start* and *end* actions and continuous time to describe actions and events. By distinguishing actual and possible situations, they define a total ordering on situations that allows expressing linear-time temporal properties within the SC. Finzi and Pirri [15] combine the SC with temporal constraint reasoning, among others supporting *processes* that have start and end actions and explicit start times. Several approaches [22, 28] allow modeling continuous processes, where a fluent continuously changes its value between two situations. The *hybrid situation calculus* [5] combines the SC with hybrid systems by embedding *hybrid automata* [2].

The *event calculus* [20, 24] uses an explicit representation of discrete or continuous time, where *events* occur at certain points on the timeline. In contrast to the SC, formulas do not refer to situations, but to points in time, and it uses circumscription to solve the frame problem. A comparison of the two formalisms can be found in [30]. As there is no action programming language such as GOLOG in the event calculus, we focus on the SC.

Timed automata (TAs) [3] extend finite automata with continuous time. While reachability is decidable for the original model, several extensions render it undecidable [4]. For TAs with *diagonal* constraints of the form $x - y \bowtie c$ (in contrast to $x \bowtie c$), reachability remains decidable [3], while it is undecidable with *additive constraints* ($x + y \bowtie c$) [7]. Allowing clock resets to non-zero values also affects decidability: with diagonal-free constraints, resets of clocks such as $x := x + 1$ preserve decidability, while diagonal constraints render reachability undecidable. Updates of the form $x := x - 1$ always yield undecidability [6]. As decidability for timed automata is well-studied, it is reasonable to adopt this model of time for the SC.

3 The Situation Calculus

The situation calculus (SC) [21, 29] is a sorted predicate logic for reasoning about dynamically changing worlds. All changes in the world are caused by *actions*, which are terms in the language. A *situation* is a possible history of the world, where the distinguished function symbol $\text{do}(\alpha, \sigma)$ describes the resulting situation of doing action α in situation σ and the constant S_0 denotes the initial situation where no actions have yet occurred. Besides actions and situations, the sort *objects* is used for terms describing objects of the world. *Fluents* are relational or functional symbols that describe possibly changing properties of the world and that have a situation as their last argument. Relational and functional symbols that remain unchanged between situations are called *rigid* and do not have a situation argument. We denote relational symbols with upper-case letters, e.g., $\text{IsFull}(p, s)$, and functional symbols with lower-case letters, e.g., $\text{counter}(s)$. Apart from the situation argument, all arguments of a fluent are of type object. We assume that there is a finite set of fluent symbols $\mathcal{F}$ and a finite number of action types. In contrast to the usual definition, we also assume a *finite set of objects* $\mathcal{O}$, for which we require the unique name assumption and domain closure. For an action type A , we write $|A|$ for the number of arguments of A . In the following, s denotes a situation variable, σ a ground situation term, a an action variable, α a ground action term, A an action type, p and o object variables, ρ a ground object term, and $\vec{x}$ a tuple of variables $x_1, \dots, x_k$ of the corresponding sort (similarly for terms). We also write $\phi_1 \supset \phi_2$ for $\neg\phi_1 \vee \phi_2$ and $\phi_1 \equiv \phi_2$ for $\phi_1 \supset \phi_2 \wedge \phi_2 \supset \phi_1$, where $\equiv$ has lowest operator precedence and free variables are assumed to be universally quantified from the outside, e.g., $F(\vec{o}, s) \equiv \phi_1 \vee \phi_2$ stands for $\forall \vec{o}, s. (F(\vec{o}, s) \equiv (\phi_1 \vee \phi_2))$. Finally, we write $\text{do}([\alpha_1; \dots; \alpha_k], S_0)$ for $\text{do}(\alpha_k, \text{do}(\dots, \text{do}(\alpha_1, S_0)))$. A formula is *uniform in s* if it does not mention any situation terms other than s and does not mention Poss (see [29] for a formal definition). A *situation-suppressed formula* is a formula where the situation argument of each fluent is omitted. For a formula ϕ and a situation term σ , we write $\phi[\sigma]$ for the formula obtained by replacing each occurrence of s with σ . Similarly, for a situation-suppressed formula ϕ , $\phi[\sigma]$ is like ϕ with every fluent augmented by the situation argument σ .

The SC has been extended to represent time by adding an additional real-valued sort *time* with the standard interpretation of the real numbers and their operations (addition, subtraction, etc.) and relations ($<$, $\leq$, etc.) and augmenting each action term $A(\vec{o}, t)$ with an additional time argument t , which specifies the time point when the action occurs [26, 29]. An additional function symbol $\text{time}(\alpha)$ is used to describe the starting time of the action situation, i.e., $\text{time}(A(\vec{o}, t)) := t$. This allows axiomatizing the starting time of a situation with $\text{start}(\text{do}(a, s)) = \text{time}(a)$ and *processes* with a duration that is defined by its start and end action [15]. With real numbers as an additional sort and independent of whether actions have an execution timepoint, we may also allow *real-valued fluent functions*. As we will see in Section 4, having either durative processes or real-valued fluent functions without any constraints renders the reachability problem undecidable.

Basic Action Theories

A basic action theory (BAT) [27] is an axiomatization of a domain and describes all available actions with precondition axioms and successor state axioms. In addition to the domain-specific axioms, a BAT also contains the foundational axioms of the SC. Similar to [13], we include domain closure axioms for action types and objects. Here, we axiomatize a finite set of objects.

► **Definition 1.** A basic action theory (BAT) is a set of sentences $\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_{\text{poss}} \cup \mathcal{D}_{\text{ssa}} \cup \mathcal{D}_{\text{ca}} \cup \mathcal{D}_{\text{co}} \cup \Sigma$ where:

- $\mathcal{D}_0$ is a description of the initial situation with a set of sentences uniform in S_0 . Here, we assume complete information, i.e., for every atomic (fluent or rigid) formula ϕ uniform in S_0 , either $\mathcal{D} \models \phi$ or $\mathcal{D} \models \neg\phi$.
- $\mathcal{D}_{\text{poss}}$ are precondition axioms of the form $\text{Poss}(A(\vec{o}), s) \equiv \phi_A(\vec{o}, s)$, one per action type $A \in \mathcal{A}$ and where $\phi_A(\vec{o}, s)$ is a formula uniform in s that describes when the action $A(\vec{o})$ can be performed.
- $\mathcal{D}_{\text{ssa}}$ are successor state axioms (SSAs) describing action effects. For each relational fluent $F \in \mathcal{F}$, $\mathcal{D}_{\text{ssa}}$ contains an axiom of the form $F(\vec{o}, \text{do}(a, s)) \equiv \phi_F(\vec{o}, a, s)$, where $\phi_F(\vec{o}, a, s)$ is a formula uniform in s that describes how the fluent $F(\vec{o})$ changes if some action a is performed. For each functional fluent $f \in \mathcal{F}$, $\mathcal{D}_{\text{ssa}}$ contains an axiom of the form $f(\vec{o}, \text{do}(a, s)) = y \equiv \phi_f(\vec{o}, y, a, s)$, where $\phi_f(\vec{o}, y, a, s)$ is a formula uniform in s describing how the fluent $f(\vec{o})$ changes if an action a is performed.
- $\mathcal{D}_{\text{ca}}$ are domain closure and unique name axioms (UNAs) on actions $\mathcal{A}$, e.g.,

$$\begin{aligned} & \forall a (\exists \vec{o}_1 (a = A_1(\vec{o}_1)) \vee \dots \vee \exists \vec{o}_n (a = A_n(\vec{o}_n))) \\ & \forall \vec{o}_i \vec{o}_j (A_i(\vec{o}_i) \neq A_j(\vec{o}_j)) \text{ for all } i \neq j \\ & \forall \vec{o}_1 \vec{o}'_1 (A_1(\vec{o}_1) = A_1(\vec{o}'_1) \supset \vec{o}_1 = \vec{o}'_1) \wedge \dots \wedge \forall \vec{o}_n \vec{o}'_n (A_n(\vec{o}_n) = A_n(\vec{o}'_n) \supset \vec{o}_n = \vec{o}'_n) \end{aligned}$$

- $\mathcal{D}_{\text{co}}$ are domain closure and UNAs on object constants $\mathcal{O}$, e.g., $\forall o (o = \rho_1 \vee o = \rho_2 \vee \dots) \wedge \rho_1 \neq \rho_2 \wedge \dots$.
- Σ : foundational, domain-independent axioms, possibly including time and start from above and including the relation $\sqsubset$ that provides an ordering relation on situations.

We call $\mathcal{D}$ a *timed BAT* if it includes axioms for time, start, etc. from above and we say $\mathcal{D}$ is *real-valued* if it includes a functional fluent of sort *time*. Note that these are two independent properties: We may have occurrence times for actions without having any real-valued fluents and we may use real-valued fluents without assigning any time values to actions. In the following, we assume that situation terms only mention action types from $\mathcal{A}$ and object terms from $\mathcal{O}$. Following [29], a situation s is *executable*, written $\text{Exec}(s)$, if every action performed to reach s is possible in the situation where it occurred: $\text{Exec}(s) := \text{do}(a, s') \sqsubseteq s \supset \text{Poss}(a, s')$.

4 Reachability

Generally speaking, reachability asks whether there is a reachable situation where some property holds. Formally, given a BAT $\mathcal{D}$ and a formula ϕ , ϕ is *reachable* if there is an executable situation where ϕ holds, i.e., if $\mathcal{D} \models \exists s \text{Exec}(s) \wedge \phi[s]$. Many problems such as planning or verification of safety properties can be formulated as a reachability problem. It is therefore desirable to have a sound and complete method to decide reachability. However, depending on the expressivity of the underlying framework, the reachability problem may become undecidable. It is easy to see that for a given BAT over a finite set of objects that does not make use of time, the reachability problem is decidable. On the other hand, if we include fluents of sort *time* and allow arbitrary arithmetic operations on those fluents, then the problem is no longer decidable:¹

¹ Full proofs can be found in Appendix A.

► **Theorem 2.** *The reachability problem for a real-valued BAT is undecidable.*

Proof Idea. By reduction of the halting problem for 2-counter machines (2CMs), which is undecidable [23]. Given a 2CM $\mathcal{M}$, we can construct a BAT $\mathcal{D}_{\mathcal{M}}$ such that $\mathcal{D}_{\mathcal{M}} \models \exists s. \text{Exec}(s) \wedge \text{Halt}(s)$ iff $\mathcal{M}$ halts. ◀

Intuitively, the problem is that we may use time fluents as counters for 2CMs. These counters may be unbounded because time fluents are not bounded. A natural restriction is thus adding an upper bound b to all fluents of sort *time*: Given a bound $b \in \mathbb{Q}$ and a BAT $\mathcal{D}$ with $\mathcal{F}_{\text{time}}$ denoting all functional fluents of sort time, we express that all fluents are bounded by b with $\text{Bounded}_b(s) := \forall s'. s' \sqsubseteq s \supset \bigwedge_{f \in \mathcal{F}_{\text{time}}} \forall \vec{o}. f(\vec{o}, s') \leq b$. We can now define the *time-bounded reachability problem*: Given a property ϕ and a bound $b \in \mathbb{Q}$, does $\mathcal{D} \models \exists s \text{Exec}(s) \wedge \text{Bounded}_b(s) \wedge \phi[s]$ hold? Unfortunately, this restriction is not sufficient to accomplish decidability:

► **Theorem 3.** *The time-bounded reachability problem for a real-valued BAT and a time bound $b \geq 2$ is undecidable.*

Proof Idea. Again by reduction of the halting problem for 2CMs. A counter value u is encoded by a fluent value 2^{1-u} . A counter is decremented by doubling and incremented by halving the corresponding fluent and tested for zero by comparing the corresponding fluent to 2. ◀

These results suggest that we need to restrict the arithmetic operators on time fluents such that time may not change its value arbitrarily, but is monotonically increasing. However, this is not sufficient; even if we do not allow real-valued fluents but instead only consider actions with time arguments and processes with start times [15], the reachability problem remains undecidable:

► **Theorem 4.** *The reachability problem for timed BATs without real-valued functional fluents is undecidable.*

Proof Idea. By reduction of the language emptiness problem for TAs with 4 clocks $c_1, \dots, c_4$ extended with additive clock constraints of the form $c_1 + c_2 = 1$, which is undecidable [7]. For a TA $\mathcal{A}$, we can construct a BAT and a formula ϕ such that $\mathcal{D} \models \exists s. \text{Exec}(s) \wedge \phi[s]$ iff the language of $\mathcal{A}$ is non-empty. The key idea is to model each clock as a separate process, where a clock reset starts the process, defining the clock value of every clock in every situation. ◀

Together, Theorems 2 to 4 rule out the natural ways of modeling dense time in the SC while retaining decidability. Reachability is undecidable if fluents range over the reals and may be manipulated arithmetically (Theorem 2); bounding the range of these fluents does not help (Theorem 3); and removing real-valued functional fluents altogether, such that time only occurs through timed actions, does not retain decidability either (Theorem 4). In each case, the problem is not continuous time itself, but allowing time values to be combined arithmetically. This motivates the model of Section 5, which keeps dense time but restricts the admissible operations by only allowing resets, comparisons against natural numbers, and uniform progression of time.

5 Clocks in the Situation Calculus

In this section, we propose an alternative approach for modeling time in the SC that maintains decidability. We adopt methods from timed automata theory by defining *clocked BATs* and show that the reachability problem for clocked BATs is decidable. In particular, we introduce

clocks as real-valued functional fluents with restricted SSAs that guarantee uniform passing of time and that allow an action to reset a clock to zero. Furthermore, we require that the BAT uses a distinguished action *wait* that causes time to pass. Apart from *wait*, all actions are instantaneous, i.e., they do not have a time argument and there is no delay between the previous and the current action. This allows using clocks both in action preconditions and in successor state axioms. We restrict formulas mentioning clocks to comparisons to natural numbers:

► **Definition 5.** We distinguish the following types of formulas:

1. A formula ϕ is a clock comparison if it is of the form $f(\vec{o}, s) \bowtie v$ or $v \bowtie v'$, where $f \in \mathcal{F}$ is some functional fluent, $v, v' \in \mathbb{N}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$.
2. A formula ϕ is called clocked if every atomic sub-formula mentioning a term of sort *time* is a clock comparison.
3. A formula ϕ is called time-independent if it does not mention any term of sort *time*.

We call a clock comparison $\mathcal{K}$ -bounded if it only compares against constants $v, v' \leq \mathcal{K}$. A clocked formula is $\mathcal{K}$ -bounded if each occurring clock comparison is $\mathcal{K}$ -bounded. As we will see, these restrictions allow us to apply *regionalization*, which defines an equivalence relation on clock values. Based on regionalization, we will define a *time-abstract bisimulation*, resulting in a *finite* abstraction of the otherwise infinite tree of situations, which permits a decidable procedure for the reachability problem. We start by introducing clocks to BATs:

► **Definition 6.** A BAT $\mathcal{D}$ is called a clocked BAT if:

1. Every functional fluent f is real-valued, i.e., of sort *time*;
2. For each functional fluent f , $\mathcal{D}_0$ contains an axiom $f(\vec{o}, S_0) = 0$;
3. For each functional fluent f , $\mathcal{D}_{ssa}$ contains an axiom of the form

$$f(\vec{o}, \text{do}(a, s)) = y \equiv \exists t. a = \text{wait}(t) \wedge y = f(\vec{o}, s) + t \\ \vee \neg \exists t. a = \text{wait}(t) \wedge (\phi_f(\vec{o}, a, s) \wedge y = 0 \vee \neg \phi_f(\vec{o}, a, s) \wedge y = f(\vec{o}, s))$$

where $\phi_f(\vec{o}, a, s)$ is a time-independent formula uniform in s describing if the action a resets the value of $f(\vec{o})$;

4. For every relational fluent R , $\mathcal{D}_{ssa}$ contains an axiom of the form $R(\vec{o}, \text{do}(a, s)) \equiv \phi_R(\vec{o}, a, s)$ where $\phi_R(\vec{o}, a, s)$ is a clocked formula uniform in s .
5. Every precondition axiom in $\mathcal{D}_{poss}$ is of the form $\text{Poss}(A(\vec{o}), s) \equiv \phi_A(\vec{o}, s)$ such that $\phi_A(\vec{o}, s)$ is a clocked formula uniform in s ;
6. Waiting for some non-negative amount of time is always possible: $\text{Poss}(\text{wait}(t), s) \equiv t \geq 0$;

We call a clocked BAT $\mathcal{D}$ $\mathcal{K}$ -bounded if each clocked formula occurring in $\mathcal{D}$ is $\mathcal{K}$ -bounded. Note that $\mathcal{D}$ is always $\mathcal{K}$ -bounded for some $\mathcal{K} \in \mathbb{N}$, because $\mathcal{D}$ consists of finitely many axioms. For the sake of simplicity, we assume that all functional fluents are of domain *time* and are initialized to zero. While this is not strictly necessary, it simplifies the treatment of functional fluents later on. The restricted form of the SSAs of functional fluents enforces a behavior similar to clocks in timed automata: An action *wait*(t) increments each clock by the value $t \in \mathbb{R}_{\geq 0}$; all other actions either leave the value unchanged or reset it to zero.² Hence, we will also call these fluents *clocks*. We write $\mathcal{C}$ for the set of all ground situation-suppressed clock terms, we denote clock variables with $c, c_1, \dots$ and (possibly situation-suppressed) ground

² This separation allows us to use clock comparisons in precondition axioms and SSAs. If we allowed time to pass with every action, then the precondition axiom would refer to the time of the *previous* situation, as time is only incremented after the action has occurred.

clock terms with $\omega, \omega_1, \dots$. Finally, we also restrict the usage of clocks in all axioms of a clocked BAT: Both precondition and successor state axioms may only use clocked formulas, which restrict any operations on clocks to clock comparisons. As we will see later on, this is an essential restriction for allowing regionalization. Nevertheless, it is easy to show that clocked BATs are expressive enough to represent TAs. We use a variant of the coffee delivery robot [29] as a running example:

► **Example 7.** We model a coffee delivery service that has the durative actions *brew* and *pour* and objects *Pot*, *Mug₁*, and *Mug₂*. The robot needs to brew coffee and then pour it into the mugs. The fluents *Brew* and *Pour* indicate that the robot is currently brewing (respectively pouring) coffee. After brewing, the pot *IsFull* and after pouring into a mug, the mug *IsFull*. The clock c_{brew} tracks the duration of *brew*, while c_{glob} tracks global time. Brewing takes at least 1 time unit. If the robot brews coffee for at least 2 time units, the coffee is *Strong*.

- Initially, the robot is doing nothing, nothing is filled, the coffee is not strong, and all clocks are initialized to zero. Also, only *Mug₁* should be filled with strong coffee:

$$\neg Brew(p, S_0), \neg Pour(p, m, S_0), \neg IsFull(o, S_0), \neg Strong(o, S_0), \\ c_{brew}(p, S_0) = 0, c_{glob}(S_0) = 0, WantStrong(o) \equiv o = Mug_1$$

- The robot may start a durative action if it is not doing anything and end it if it has started it before. Brewing takes at least 1 time unit. In order to pour the coffee, there must be coffee in the pot:

$$\begin{aligned} Poss(sBrew(p), s) &\equiv \neg Brew(p, s) \wedge \neg \exists m. Pour(p, m, s) \\ Poss(eBrew(p), s) &\equiv Brew(p, s) \wedge c_{brew}(p, s) \geq 1 \\ Poss(sPour(p, m), s) &\equiv IsFull(p, s) \wedge \neg \exists p', m'. Pour(p', m', s) \wedge \neg Brew(p, s) \\ Poss(ePour(p, m), s) &\equiv Pour(p, m, s) \\ Poss(wait(t), s) &\equiv t \geq 0 \end{aligned}$$

- The SSAs state that *brew* fills a pot with coffee and makes it strong if it took at least 2 time units:

$$\begin{aligned} IsFull(o, do(a, s)) &\equiv a = eBrew(o) \vee \exists p. a = ePour(p, o) \vee IsFull(o, s) \\ Strong(o, do(a, s)) &\equiv a = eBrew(o) \wedge c_{brew}(o, s) \geq 2 \\ &\quad \vee \exists p. a = ePour(p, o) \wedge Strong(p, s) \vee Strong(o, s) \\ Brew(p, do(a, s)) &\equiv a = sBrew(p) \vee Brew(p, s) \wedge a \neq eBrew(p) \\ Pour(p, m, do(a, s)) &\equiv a = sPour(p, m) \vee Pour(p, m, s) \wedge a \neq ePour(p, m) \\ c_{brew}(p, do(a, s)) = y &\equiv \exists t(a = wait(t) \wedge y = c_{brew}(p, s) + t) \vee \left(\neg \exists t. a = wait(t) \right. \\ &\quad \left. \wedge (a = sBrew(p) \wedge y = 0 \vee a \neq sBrew(p) \wedge y = c_{brew}(p, s)) \right) \\ c_{glob}(do(a, s)) = y &\equiv \exists t. a = wait(t) \wedge y = c_{glob}(s) + t \\ &\quad \vee \neg \exists t. a = wait(t) \wedge y = c_{glob}(s) \end{aligned}$$

- There are no other objects or actions:

$$\begin{aligned} \forall o(o = Pot \vee o = Mug_1 \vee \dots) \wedge Mug_1 \neq Pot \wedge \dots \\ \forall a(\exists o. a = sBrew(o) \vee \dots \vee \exists t. a = wait(t)) \\ \wedge \forall o_1, o_2. sBrew(o_1) \neq eBrew(o_2) \wedge \dots \\ \wedge \forall o_1, o_2. sBrew(o_1) = sBrew(o_2) \supset o_1 = o_2 \wedge \dots \end{aligned}$$

5.1 Regression

A fundamental reasoning task in the SC is *projection*, i.e., determining whether a sentence ϕ is true in a future situation. A common method to solve projection is *regression* [27], which reduces the formula ϕ to an equivalent formula $\mathcal{R}[\phi]$ which only refers to the initial situation. Given a BAT, regression substitutes each fluent mentioned in ϕ by the right-hand side of the corresponding SSA until every fluent only mentions S_0 . Regression is sound and complete and thus may be used for reducing queries about a future situation to the initial situation. Here, we use a simplified variant of regression, where we require that all action terms are ground:

► **Definition 8.** A formula ϕ is *regressable* iff

1. ϕ is first order (i.e., the second-order axioms Σ are not regressable).
2. Every term of sort *situation* mentioned in ϕ has the form $\text{do}([\alpha_1, \dots, \alpha_n], S_0)$, where $\alpha_1, \dots, \alpha_n$ are ground terms of sort *action*.
3. Every atom mentioning a clock f is of the form $f(\vec{o}, s) \bowtie \tau$, where $\vec{o}$ are terms of sort *object* and $\tau \in \mathbb{R}$.³
4. For every atom of the form $\text{Poss}(a, s)$ mentioned in ϕ , a has the form $A(\vec{o})$ for some action type A .
5. ϕ does not quantify over situations.
6. ϕ does not mention the symbol $\sqsubset$, nor does it mention any equality atom $s = s'$ for situation terms s, s' .

We extend the standard regression operator $\mathcal{R}$ with a regression rule for clock comparisons:

► **Definition 9.** Let $\phi = f(\vec{\rho}, \text{do}(\alpha, \sigma)) \bowtie r$ where $f \in \mathcal{F}$ is a clock and $r \in \mathbb{R}$. By Definition 6, there is an SSA of the following form:

$$\begin{aligned} f(\vec{o}, \text{do}(a, s)) = y &\equiv \exists t. a = \text{wait}(t) \wedge y = f(\vec{o}, s) + t \\ &\vee \neg \exists t. a = \text{wait}(t) \wedge (\phi_f(\vec{o}, a, s) \wedge y = 0 \vee \neg \phi_f(\vec{o}, a, s) \wedge y = f(\vec{o}, s)) \end{aligned}$$

Then $\mathcal{R}[\phi]$ is defined as follows:

1. If $\alpha = \text{wait}(t)$ for some $t \in \mathbb{R}_{\geq 0}$, then $\mathcal{R}[\phi] := \mathcal{R}[f(\vec{\rho}, \sigma) \bowtie r']$ with $r' = r - t$.
2. Otherwise: $\mathcal{R}[\phi] := \mathcal{R}[\phi_f(\vec{\rho}, \alpha, \sigma) \wedge 0 \bowtie r \vee \neg \phi_f(\vec{\rho}, \alpha, \sigma) \wedge f(\vec{\rho}, \sigma) \bowtie r]$

We obtain a variant of the regression theorem:

► **Theorem 10.** Let ϕ be a regressable formula. Then $\mathcal{R}[\phi]$ is uniform in S_0 and $\mathcal{D} \models \phi$ iff $\mathcal{D}_0 \cup \mathcal{D}_c \models \mathcal{R}[\phi]$.

With this variant of regression, deciding whether a BAT entails a regressable clocked formula is decidable:

► **Theorem 11.** Let ϕ be a regressable clocked formula and $\mathcal{D}$ a clocked BAT (over finite objects and with complete information). Then $\mathcal{D} \models \phi$ is decidable.

Proof. As regression is sound and complete, it is sufficient to consider $\mathcal{D}_0 \cup \mathcal{D}_c \models \mathcal{R}[\phi]$, which can be reduced to entailment in propositional logic: First, as $\mathcal{O}$ is finite, we can eliminate each quantifier in $\mathcal{D}$ and ϕ by substitution, e.g., by substituting $\exists o. \phi'$ with $\bigvee_{o \in \mathcal{O}} \phi'$. Second, note that regressing a clock comparison $f(\vec{\rho}, \sigma) \bowtie r$ results in a formula $f(\vec{\rho}, S_0) \bowtie r'$, where

³ In particular, we disallow quantifying over terms of sort *time*.

$r' \in \mathbb{R}$. As all clock values are initialized to 0, we can directly replace each such comparison by $\top$ or $\perp$, e.g., $f(\vec{\rho}, S_0) > 5$ is simplified to $\perp$. Finally, with the unique name assumption, we can interpret predicate symbols as propositional variables, thereby obtaining a propositional set of sentences $\mathcal{D}'_0$ and a propositional formula ϕ' such that $\mathcal{D}'_0 \models \phi'$ iff $\mathcal{D}_0 \cup \mathcal{D}_c \models \phi$. $\blacktriangleleft$

5.2 Regionalization

To show decidability of the reachability problem for clocked BATs, we use an approach based on abstraction for timed automata inspired by [3, 25]. We construct a finite abstraction of the otherwise infinite tree of situations of a clocked BAT $\mathcal{D}$ using a time-abstract bisimulation relation over situations:

► **Definition 12** (Time-abstract Bisimulation). *An equivalence relation $R \subseteq \mathcal{S} \times \mathcal{S}$ is a $\mathcal{K}$ -relative time-abstract bisimulation on $\mathcal{S}$ if $(\sigma_1, \sigma_2) \in R$ implies*

1. *For every $\mathcal{K}$ -bounded clocked formula ϕ uniform in s : $\mathcal{D} \models \phi[\sigma_1]$ iff $\mathcal{D} \models \phi[\sigma_2]$.*
2. *For every action type $A \neq \text{wait}$ and ground arguments $\vec{\rho}$, $(\text{do}(A(\vec{\rho}), \sigma_1), \text{do}(A(\vec{\rho}), \sigma_2)) \in R$.*
3. *For every $\tau_1 \in \mathbb{R}_{\geq 0}$, there is a $\tau_2 \in \mathbb{R}_{\geq 0}$ such that $(\text{do}(\text{wait}(\tau_1), \sigma_1), \text{do}(\text{wait}(\tau_2), \sigma_2)) \in R$.*

Intuitively, two situations are bisimilar if they represent the same state of the world (i.e., they satisfy the same $\mathcal{K}$ -bounded clocked formulas) and if applying the same action in both situations results again in bisimilar situations.

In the following, we construct such a bisimulation by applying regionalization to situations. Generally, a *clock valuation over clocks C* is a function $\nu : C \rightarrow \mathbb{R}_{\geq 0}$. For situations, we can define clock valuations as follows:

► **Definition 13** (Clock Valuation). *Let $\mathcal{D}$ be a clocked BAT with (situation-suppressed) clocks $\mathcal{C}$ and σ a situation. The clock valuation $\nu_\sigma^\mathcal{D}$ of σ is the clock valuation over $\mathcal{C}$ with $\nu_\sigma^\mathcal{D}(\omega) = \tau$ iff $\mathcal{D} \models \omega[\sigma] = \tau$ for a ground clock term ω .*

We will omit the superscript $\mathcal{D}$ if $\mathcal{D}$ is clear from context. It is sometimes convenient to write a clock valuation ν as a set of pairs $\{(c, v) \mid c \in C, v = \nu(c)\} \subseteq C \times \mathbb{R}_{\geq 0}$. We also write $\nu + \tau$ for $\{(c, v + \tau) \mid c \in C, v = \nu(c)\}$ where all clocks are incremented by τ . We can show that every situation has a unique clock valuation:

► **Lemma 14.** *If $\mathcal{D}$ is a clocked BAT and σ a situation, then ν_σ always exists and is unique.*

We use *regionalization* to obtain an abstraction of clock valuations. We briefly outline the intuition behind this idea: Given a constant $\mathcal{K} \in \mathbb{N}$ that is the maximal constant of sort *time* mentioned in $\mathcal{D}$ and in the reachability query ϕ , we can collect all clock values with a value larger than $\mathcal{K}$ in a single class $\top$. Furthermore, as clocks are only compared to natural numbers, for a single clock it suffices to observe only integer parts and whether fractional parts differ from zero to derive a finite set of equivalence classes for clock valuations $v \in [0, \mathcal{K}]$. In the presence of multiple clocks, the order between the fractional parts needs to be taken into account as well [1]. We define the *fractional part* $\text{fract}(v)$ of $v \in [0, \mathcal{K}] \cup \{\top\}$ as $v - \lfloor v \rfloor$ if $v \in [0, \mathcal{K}]$ and zero otherwise. This leads to the definition of clock regions:

► **Definition 15** (Clock Regions). *Given a maximal constant $\mathcal{K}$, let $V = [0, \mathcal{K}] \cup \{\top\}$. We define the region equivalence as the equivalence relation $\sim_{\mathcal{K}}$ on V such that $u \sim_{\mathcal{K}} v$ if*

- $u = v = \top$, or
- $u, v \neq \top$, $\lceil u \rceil = \lceil v \rceil$, and $\lfloor u \rfloor = \lfloor v \rfloor$.

A region is an equivalence class of $\sim_K$. We extend region equivalence to clock valuations. Let ν, ν' be two clock valuations over $\mathcal{C}$, then ν and ν' are region-equivalent, written $\nu \cong_K \nu'$ iff (1) for every $c \in \mathcal{C}$, $\nu(c) \sim_K \nu'(c)$, and (2) for every $c, c' \in \mathcal{C}$, $\text{fract}(\nu(c)) \leq \text{fract}(\nu'(c'))$ iff $\text{fract}(\nu'(c)) \leq \text{fract}(\nu(c'))$.

Note that for a fixed K , there is only a finite number of regions. Based on region equivalence wrt. K , we can now define an equivalence relation on situations:

- **Definition 16.** Let $\approx_{\mathcal{DK}} \subseteq \mathcal{S} \times \mathcal{S}$ be an equivalence relation such that $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$ iff
1. for every fluent $R \in \mathcal{F}$, $\mathcal{D} \models R(\vec{\rho}, \sigma_1)$ iff $\mathcal{D} \models R(\vec{\rho}, \sigma_2)$,
 2. there is a bijection $f : \nu_{\sigma_1} \rightarrow \nu_{\sigma_2}$ such that
 - a. if $f(c_1, v_1) = (c_2, v_2)$, then $c_1 = c_2$ and $v_1 \sim_K v_2$
 - b. if $f(c_1, v_1) = (c_2, v_2)$ and $f(c'_1, v'_1) = (c'_2, v'_2)$, then $\text{fract}(v_1) \leq \text{fract}(v'_1)$ iff $\text{fract}(v_2) \leq \text{fract}(v'_2)$.

Two situations are equivalent if they satisfy the same fluents and agree on the clock regions. For a situation σ , we write $[\sigma]_{\approx_{\mathcal{DK}}}$ for the equivalence class of σ wrt. $\approx_{\mathcal{DK}}$. We denote the set of equivalence classes resulting from $\approx_{\mathcal{DK}}$ as $\mathcal{S}/\approx_{\mathcal{DK}}$. Equivalence directly extends to clocked formulas:

- **Lemma 17.** Let $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$ and let ϕ be a K -bounded clocked formula uniform in s . Then $\mathcal{D} \models \phi[\sigma_1]$ iff $\mathcal{D} \models \phi[\sigma_2]$.

We use a well-known result from [3] which states that it is always possible to maintain equivalence wrt. $\cong_K$ between clock valuations for a given time increment:

- **Proposition 18.** Let ν_1 and ν_2 be two clock valuations over a set of clocks $\mathcal{C}$ such that $\nu_1 \cong_K \nu_2$. Then for all $\tau \in \mathbb{R}_{\geq 0}$ there exists a $\tau' \in \mathbb{R}_{\geq 0}$ such that $\nu_1 + \tau \cong_K \nu_2 + \tau'$.

Therefore, for two situations with $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$ and any action $\text{wait}(\tau_1)$, it is always possible to find a τ_2 such that $\text{do}(\text{wait}(\tau_1), \sigma_1) \approx_{\mathcal{DK}} \text{do}(\text{wait}(\tau_2), \sigma_2)$. Based on this, we can show that $\approx_{\mathcal{DK}}$ satisfies all criteria of Definition 12:

- **Theorem 19.** $\approx_{\mathcal{DK}}$ is a time-abstract bisimulation.

Hence, it suffices to argue about the resulting equivalence classes to determine entailment with respect to a given $\mathcal{D}$. This enables us to construct a time-abstract transition system as a finite abstraction for the otherwise infinite tree of situations, where states collect $\approx_{\mathcal{DK}}$ -equivalent situations connected by executable actions from $\mathcal{D}$:

- **Definition 20 (Time-abstract Transition System).** Let $\mathcal{D}$ be a clocked BAT. We define the TS $\mathcal{T}_{\mathcal{D}} = (Q, q_0, \rightarrow)$ as follows:

- The set of states Q consists of equivalence classes of $\approx_{\mathcal{DK}}$, i.e., $Q \subseteq \mathcal{S}/\approx_{\mathcal{DK}}$;
- The initial state is the equivalence class $q_0 = [S_0]_{\approx_{\mathcal{DK}}}$;
- $[\sigma]_{\approx_{\mathcal{DK}}} \rightarrow [\sigma']_{\approx_{\mathcal{DK}}}$ if σ' is an executable successor situation of σ , i.e., if $\sigma' = \text{do}(\alpha, \sigma)$ and $\mathcal{D} \models \text{Exec}(\sigma')$.

It is easy to see that it is sufficient to consider $\mathcal{T}_{\mathcal{D}}$ for checking reachable situations:

- **Lemma 21.** $[\sigma]_{\approx_{\mathcal{DK}}}$ is reachable in $\mathcal{T}_{\mathcal{D}}$ iff there is some $\sigma' \in [\sigma]_{\approx_{\mathcal{DK}}}$ with $\mathcal{D} \models \text{Exec}(\sigma')$.

In contrast to the tree of situations, the time-abstract transition system $\mathcal{T}_{\mathcal{D}}$ is finite:

- **Lemma 22.** The time-abstract transition system $\mathcal{T}_{\mathcal{D}}$ has finitely many states.

Hence, the time-abstract transition system $\mathcal{T}_{\mathcal{D}}$ is useful for deciding reachability: every reachable situation is also reachable in $\mathcal{T}_{\mathcal{D}}$, which is a finite transition system and can be explored exhaustively. It remains to be shown that we can actually construct $\mathcal{T}_{\mathcal{D}}$.

■ **Algorithm 1** Computation of the time-abstract transition system $\mathcal{T}_{\mathcal{D}}$ for a clocked BAT $\mathcal{D}$.

```

 $Open \leftarrow \{S_0\}; V \leftarrow \{[S_0]_{\approx_{\mathcal{D}\mathcal{K}}}\}; E \leftarrow \emptyset$ 
while  $Open \neq \emptyset$  do
   $\sigma \leftarrow \text{POP}(Open)$ 
   $Acts \leftarrow \{A(\vec{\rho}) \mid A \in \mathcal{A} \setminus \{wait\}, \vec{\rho} \in \mathcal{O}^{|\mathcal{A}|}\} \cup \{wait(\tau) \mid \tau \in \text{TSuccs}(\sigma, \mathcal{K})\}$ 
  for all  $\alpha \in Acts$  do
    if  $\mathcal{D}_0 \cup \mathcal{D}_c \models \mathcal{R}[\text{Poss}(\alpha, \sigma)]$  then
      if  $[\text{do}(\alpha, \sigma)]_{\approx_{\mathcal{D}\mathcal{K}}} \notin V$  then
         $V \leftarrow V \cup \{[\text{do}(\alpha, \sigma)]_{\approx_{\mathcal{D}\mathcal{K}}}\}$ 
         $Open \leftarrow Open \cup \{\text{do}(\alpha, \sigma)\}$ 
       $E \leftarrow E \cup \{([\sigma]_{\approx_{\mathcal{D}\mathcal{K}}}, [\text{do}(\alpha, \sigma)]_{\approx_{\mathcal{D}\mathcal{K}}})\}$ 
return  $(V, [S_0]_{\approx_{\mathcal{D}\mathcal{K}}}, E)$ 

```

■ **Algorithm 2** Time successor computation for a situation σ .

```

function  $\text{TSuccs}(\sigma, K)$ 
   $\nu \leftarrow \nu_{\sigma}; last \leftarrow 0; Succs \leftarrow \{last\}$ 
  while  $\exists(c, v) \in \nu : v \leq K$  do
     $\mu \leftarrow \max_{(c, v) \in \nu} \text{fract}(v)$ 
    if  $\exists(c, v) \in \nu : \text{fract}(v) = 0$  then  $incr \leftarrow \frac{1-\mu}{2}$ 
    else  $incr \leftarrow 1 - \mu$ 
     $last \leftarrow last + incr$ 
     $Succs \leftarrow Succs \cup \{last\}$ 
     $\nu \leftarrow \nu + incr$ 
  return  $Succs$ 

```

5.3 Decidability

We now describe an algorithm for constructing $\mathcal{T}_{\mathcal{D}}$, which is shown in Algorithm 1. Starting with the initial state $[S_0]_{\approx_{\mathcal{D}\mathcal{K}}}$, the algorithm iteratively adds a state $[\sigma]_{\approx_{\mathcal{D}\mathcal{K}}}$ to $\mathcal{T}_{\mathcal{D}}$ whenever σ is reachable by a possible single action from an existing state, i.e., if $\sigma = \text{do}(\alpha, \sigma')$ and $\mathcal{D} \models \text{Poss}(\alpha, \sigma')$ for some state $[\sigma']_{\approx_{\mathcal{D}\mathcal{K}}}$ of $\mathcal{T}_{\mathcal{D}}$. It does so in a breadth-first search manner by keeping a list of *open* states $Open$, which consists of states that need to be expanded. The algorithm computes all possible successors by enumerating all possible ground action terms. For the action *wait*, it includes a canonical timepoint τ as a representative for the equivalence class $[\nu_{\sigma} + \tau]_{\approx_{\mathcal{K}}}$. These canonical time successors $\text{TSuccs}(\sigma, \mathcal{K})$ are computed using Algorithm 2 based on a well-known method [25], which satisfies the following property:

► **Proposition 23.** *Let ν be a clock valuation, $\tau \in \mathbb{R}_{\geq 0}$, and $Succs$ the time successors computed by Algorithm 2. Then there is a $\tau' \in Succs$ such that $\nu + \tau \cong_{\mathcal{K}} \nu + \tau'$.*

The reachability problem is indeed decidable:

► **Lemma 24.** *Let σ_1 and σ_2 be two situations. Then it is decidable whether $\sigma_1 \approx_{\mathcal{D}\mathcal{K}} \sigma_2$.*

► **Lemma 25.** *For a clocked BAT $\mathcal{D}$, Algorithm 1 terminates and returns the time-abstract transition system $\mathcal{T}_{\mathcal{D}}$.*

► **Theorem 26.** *Reachability for clocked BATs is decidable.*

Proof. By Lemma 21, it is sufficient to consider the time-abstract transition system $\mathcal{T}_{\mathcal{D}}$. By Lemma 25, Algorithm 1 terminates and computes $\mathcal{T}_{\mathcal{D}}$. We now show how to use $\mathcal{T}_{\mathcal{D}}$ for deciding the reachability problem of some property ϕ with maximal constant $\leq \mathcal{K}$: By

construction, every state of $\mathcal{T}_{\mathcal{D}}$ is an equivalence class wrt. $\approx_{\mathcal{D}\mathcal{K}}$ of executable situations, of which there are finitely many. By Lemma 17, it is sufficient to consider one situation $\sigma^* \in [\sigma]_{\approx_{\mathcal{D}\mathcal{K}}}$ for each such state. Furthermore, by Theorem 10, we can check whether $\mathcal{D} \models \phi[\sigma^*]$ by regression. If such a state exists, return “reachable”, otherwise return “not reachable”. ◀

► **Example 27.** We can check the following reachability queries:

$$\begin{aligned}\phi_1 &:= \forall o. IsFull(o) \\ \phi_2 &:= \forall o. Strong(o) \equiv WantStrong(o) \\ \phi_3 &:= c_{glob}() \leq 1 \wedge Strong(Mug_1)\end{aligned}$$

A situation satisfying ϕ_1 is reachable, e.g., with

$$\sigma_1 = \text{do}([sBrew(Pot); wait(1); eBrew(Pot); sPour(Pot, Mug_1); ePour(Pot, Mug_1); \\ sPour(Pot, Mug_2); ePour(Pot, Mug_2)], S_0)$$

For ϕ_2 , a different action sequence is required, as the pot may not be filled with strong and weak coffee at the same time. Instead, we brew coffee directly in a mug:

$$\sigma_2 = \text{do}([sBrew(Pot); sBrew(Mug_1); wait(1); eBrew(Pot); wait(2); eBrew(Mug_1); \\ sPour(Pot, Mug_2); wait(1); ePour(Pot, Mug_2)], S_0)$$

Finally, there is no reachable situation that satisfies ϕ_3 , as it would require providing strong coffee after one time unit, while it takes at least two time units to prepare strong coffee.

6 Realizing Golog Programs

In this section, we show how we can utilize reachability for clocked BATs to decide realizability for a GOLOG program Δ . The goal is to check whether the non-determinisms in Δ can be resolved and action timepoints can be determined such that the resulting action sequence is executable and results in a final configuration of the program. Similar to the above, we create a transition system $\mathcal{T}_{\Delta}$ from a given program Δ and check whether a final program state is reachable in $\mathcal{T}_{\Delta}$.

6.1 Golog Transition Semantics

We consider programs in CONGOLOG [11] based on clocked BATs. A *program* is a pair $\Delta = (\mathcal{D}, \delta)$, where $\mathcal{D}$ is a BAT and δ is a program expression built from the following grammar:

$$\delta ::= \alpha \mid \phi? \mid \delta; \delta \mid \delta | \delta \mid \pi x. \delta \mid \delta^* \mid \delta || \delta$$

Program expressions consist of actions α , tests $\phi?$, sequence of actions $\delta_1; \delta_2$, non-deterministic branching $\delta_1 | \delta_2$, non-deterministic choice of argument $\pi x. \delta$, non-deterministic iteration δ^* , and concurrent execution $\delta_1 || \delta_2$. We require that each action term is time-suppressed (i.e., $wait()$ has no argument) and each test $\phi?$ tests a situation-suppressed clocked formula ϕ . We write $\text{nil} := \text{TRUE?}$ for the empty program that always succeeds. As usual, conditionals and loops can be defined as macros, i.e., **if** ϕ **do** δ_1 **else** δ_2 **fi** $:= [\phi?; \delta_1][\neg\phi?; \delta_2]$ and **while** ϕ **do** δ **done** $:= [\phi?; \delta]^*; \neg\phi?$. We call $\Delta = (\mathcal{D}, \delta)$ a *clocked program* if $\mathcal{D}$ is a

clocked BAT and we assume that $\mathcal{K} \in \mathbb{N}$ is the maximal constant mentioned in $\mathcal{D}$ and δ . We adapt the GOLOG transition semantics from CONGOLOG such that each transition step follows a single primitive action, corresponding to *synchronized conditionals* known from INDIGOLOG [12]. We also extend the relation Trans to include the action such that $\text{Trans}(\delta, s, a, \delta', s')$ is true if there is a single-step transition with action a in the program δ starting in situation s and resulting in situation s' with remaining program δ' . Finally, we assume that the program δ does not fix the time of occurrence for *wait* and instead the time is determined in the transition semantics. Therefore, *wait* actions in δ do not have a time argument t and the timepoint is instead fixed in the transition rule for a *wait* action.⁴ We also write $\text{Trans}^*(\delta, s, \vec{a}, \delta', s')$ for the transitive closure of Trans such that $\vec{a}$ is the sequence of actions resulting from following the transitions according to Trans :

$$\begin{aligned}
\text{Trans}(a, s, a, \delta', s') &\equiv \text{Poss}(a, s) \wedge \delta' = \text{nil} \wedge s' = \text{do}(a, s) \\
\text{Trans}(\text{wait}(), s, a, \delta', s') &\equiv \exists t. a = \text{wait}(t) \wedge \delta' = \text{nil} \wedge s' = \text{do}(a, s) \\
\text{Trans}(\phi?, s, a, \delta', s') &\equiv \text{FALSE} \\
\text{Trans}(\delta_1; \delta_2, s, a, \delta', s') &\equiv \exists \gamma. \delta' = (\gamma; \delta_2) \wedge \text{Trans}(\delta_1, s, a, \gamma, s') \\
&\quad \vee \text{Final}(\delta_1, s) \wedge \text{Trans}(\delta_2, s, a, \delta', s') \\
\text{Trans}(\delta_1 | \delta_2, s, a, \delta', s') &\equiv \text{Trans}(\delta_1, s, a, \delta', s') \vee \text{Trans}(\delta_2, s, a, \delta', s') \\
\text{Trans}(\pi x. \delta, s, a, \delta', s') &\equiv \exists v. \text{Trans}(\delta_v^x, s, a, \delta', s') \\
\text{Trans}(\delta^*, s, a, \delta', s') &\equiv \exists \gamma. (\delta' = \gamma; \delta^*) \wedge \text{Trans}(\delta, s, a, \gamma, s') \\
\text{Trans}(\delta_1 \| \delta_2, s, a, \delta', s') &\equiv \exists \gamma. \delta' = (\gamma \| \delta_2) \wedge \text{Trans}(\delta_1, s, a, \gamma, s') \\
&\quad \vee \exists \gamma. \delta' = (\delta_1 \| \gamma) \wedge \text{Trans}(\delta_2, s, a, \gamma, s')
\end{aligned}$$

$\text{Final}(\delta, s)$ is true if program δ may terminate in situation s :

$$\begin{aligned}
\text{Final}(a, s) &\equiv \text{FALSE} & \text{Final}(\phi?, s) &\equiv \phi[s] \\
\text{Final}(\delta_1; \delta_2, s) &\equiv \text{Final}(\delta_1, s) \wedge \text{Final}(\delta_2, s) \\
\text{Final}(\delta_1 | \delta_2, s) &\equiv \text{Final}(\delta_1, s) \vee \text{Final}(\delta_2, s) \\
\text{Final}(\pi x. \delta, s) &\equiv \exists v. \text{Final}(\delta_v^x, s) & \text{Final}(\delta^*, s) &\equiv \text{TRUE} \\
\text{Final}(\delta_1 \| \delta_2, s) &\equiv \text{Final}(\delta_1, s) \wedge \text{Final}(\delta_2, s)
\end{aligned}$$

► **Example 28.** We consider the following program δ :

$$\begin{aligned}
&(s\text{Brew}(\text{Pot}); \exists m. (\text{WantStrong}(m)) \equiv c_{\text{brew}}(\text{Pot}) \geq 2?; e\text{Brew}(\text{Pot}); \\
&\quad \text{while } (\exists m. \neg \text{IsFull}(m)) \text{ do } \pi m. s\text{Pour}(\text{Pot}, m); e\text{Pour}(\text{Pot}, m); \text{done}) \parallel \text{wait}()^*
\end{aligned}$$

The program first brews coffee. If there is a mug that should be filled with strong coffee, it waits until $c_{\text{brew}}(\text{Pot}) \geq 2$, otherwise it stops brewing earlier. Afterwards, it fills all mugs.

6.2 Program Realization

We describe how to construct a transition system $\mathcal{T}_\Delta$ to determine a realization of a given program Δ similar to $\mathcal{T}_\mathcal{D}$ from above. We first formally define program realization:

⁴ We may still enforce a certain time delay by using clock formulas in tests $\phi?$, as demonstrated in Example 28.

■ **Algorithm 3** Construction of the time-abstract transition system $\mathcal{T}_\Delta$.

```

 $Open \leftarrow \{(S_0, \delta_0)\}; V \leftarrow \emptyset; E \leftarrow \emptyset$ 
while  $Open \neq \emptyset$  do
   $(\sigma, \delta) \leftarrow \text{POP}(Open)$ 
   $Acts \leftarrow \{A(\vec{\rho}) \mid A \in \mathcal{A} \setminus \{wait\}, \vec{\rho} \in \mathcal{O}^{|\mathcal{A}|}\} \cup \{wait(\tau) \mid \tau \in \text{TSUCCS}(\sigma, \mathcal{K})\}$ 
  for all  $\alpha \in Acts$  do
    if  $\mathcal{D} \models \text{Trans}(\delta, \sigma, \alpha, \delta', \sigma')$  then
      if  $([\sigma']_{\approx_{\mathcal{DK}}}, \delta') \notin V$  then
         $V \leftarrow V \cup \{([\sigma']_{\approx_{\mathcal{DK}}}, \delta')\}$ 
         $Open \leftarrow Open \cup \{(\sigma', \delta')\}$ 
       $E \leftarrow E \cup \{([\sigma]_{\approx_{\mathcal{DK}}}, \delta), ([\sigma']_{\approx_{\mathcal{DK}}}, \delta')\}$ 
return  $(V, ([S_0]_{\approx_{\mathcal{DK}}}, \delta_0), E)$ 

```

► **Definition 29** (Program Realization). Let $\mathcal{D}$ be a BAT and δ a program. An action sequence $\vec{\alpha} = \langle \alpha_1, \dots, \alpha_n \rangle$ is a realization of δ wrt. $\mathcal{D}$ if $\mathcal{D} \models \text{Trans}^*(\delta, S_0, \vec{\alpha}, \delta', \sigma') \wedge \text{Final}(\delta', \sigma')$. The corresponding decision problem is to decide whether such a realization exists.

First, we can show that the equivalence relation $\approx_{\mathcal{DK}}$ can also be applied to the transition semantics, i.e., bisimilar situations cannot be distinguished by possible program transitions:

- **Lemma 30.** Let σ_1, σ_2 be two situations with $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$ and $\mathcal{D} \models \text{Trans}(\delta, \sigma_1, \alpha, \delta^*, \sigma'_1)$.
1. If $\alpha = \text{wait}(\tau_1)$ for some $\tau_1 \in \mathbb{R}_{\geq 0}$, then there is $\tau_2 \in \text{TSUCCS}(\sigma_2, \mathcal{K})$ such that $\text{Trans}(\delta, \sigma_2, \text{wait}(\tau_2), \delta^*, \sigma'_2)$ and $\sigma'_1 \approx_{\mathcal{DK}} \sigma'_2$.
 2. If $\alpha = A(\vec{\rho})$ for some action type $A \neq \text{wait}$, then $\text{Trans}(\delta, \sigma_2, \alpha, \delta^*, \sigma'_2)$ and $\sigma'_1 \approx_{\mathcal{DK}} \sigma'_2$.

This allows us to use time-abstract bisimulations for determining realizability. We adapt Algorithm 1 such that states also contain the remaining program. The resulting algorithm is shown in Algorithm 3. The algorithm is correct, i.e., the transition system contains a final program configuration iff a realization exists:

► **Lemma 31.** Let $\Delta = (\mathcal{D}, \delta)$ be a program over a clocked BAT $\mathcal{D}$ and $\mathcal{T}_\Delta$ the transition system returned by Algorithm 3. Then $\mathcal{T}_\Delta$ contains a state $([\sigma']_{\approx_{\mathcal{DK}}}, \delta')$ reachable from $([S_0]_{\approx_{\mathcal{DK}}}, \delta)$ with $\mathcal{D} \models \text{Final}(\delta', \sigma')$ iff there is a realization of program Δ .

Moreover, the transition system is finite:

► **Lemma 32.** $\mathcal{T}_\Delta$ has finitely many states.

Proof. As $\mathcal{O}$ and $\mathcal{A}$ are finite, each state only has finitely many successors. Furthermore, as there are only finitely many fluents $\mathcal{F}$ and objects $\mathcal{O}$, $\approx_{\mathcal{DK}}$ has finitely many equivalence classes. Finally, there are finitely many reachable sub-programs from δ [8]. Therefore, the time-abstract transition system $\mathcal{T}_\Delta$ has finitely many states. ◀

Combining the results, we obtain:

► **Theorem 33.** The realization problem for a clocked program $\Delta = (\mathcal{D}, \delta)$ is decidable.

Proof. By Lemma 31, it is sufficient to check whether there is a reachable state in $\mathcal{T}_\Delta$ that corresponds to a final program configuration. By Lemma 32, $\mathcal{T}_\Delta$ is finite and so explicitly constructing it is feasible. Finally, checking whether $\mathcal{D} \models \text{Trans}(\delta, \sigma, \alpha, \delta', \sigma')$ reduces to checking entailment of clocked formulas uniform in σ (in particular, checking $\mathcal{D} \models \text{Poss}(\alpha, \sigma)$ and $\mathcal{D} \models \phi[\sigma]$ for tests ϕ). By Theorem 11, this is decidable. ◀

We conclude by presenting a realization of the example:

► **Example 34.** The following sequence is a realization of the program from Example 28:

```
sBrew(Pot); wait(2); eBrew(Pot); sPour(Pot, Mug1);
ePour(Pot, Mug1); sPour(Pot, Mug2); ePour(Pot, Mug2);
```

However, the program does not fill every mug with the correct kind of coffee. In fact, let $\delta' := \delta; \forall m(\text{Strong}(m) \equiv \text{WantStrong}(m))?$ be the program δ augmented with a check whether every mug was filled with the right kind of coffee. Then, no realization of δ' exists: As the program brews coffee only once, it fills all mugs with the same coffee.

7 Conclusion

We investigated how continuous time can be modeled in the situation calculus (SC), focusing on *reachability*, which asks whether there is a reachable situation that satisfies some given formula ϕ , and *program realization*, the task to determine an action sequence corresponding to a successful execution of a given GOLOG program. For these problems, the standard approaches for modeling time in the SC are unsuitable, because reachability (and thus also realizability) is undecidable. We presented an alternative approach based on *clocks*, i.e., real-valued fluents with restricted successor state axioms, and described sound and complete algorithms that explicitly construct a finite transition system, showing that both problems are decidable.

Our approach is restricted to basic action theories with a finite set of objects. Since quantifiers can be eliminated by grounding, our setting is essentially that of the propositional situation calculus [10], modulo the real-valued time of *wait*. For future work, it may be interesting to extend this approach towards more expressive formalisms that retain decidability, e.g., *bounded action theories* [14], or the two-variable fragment of the SC [17], which has been used for verifying GOLOG programs over local-effect [9] and acyclic [31] BATs, and for synthesizing GOLOG policies against uncontrollable environments with LTL_f goals [18]. Extending our notion of time to the synthesis setting is a promising direction.

References

- 1 Rajeev Alur. Timed automata. In Nicolas Halbwachs and Doron Peled, editors, *Computer Aided Verification (CAV)*, Lecture Notes in Computer Science, pages 8–22, Berlin, Heidelberg, 1999. Springer. doi:10.1007/3-540-48683-6_3.
- 2 Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, Lecture Notes in Computer Science, pages 209–229, Berlin, Heidelberg, 1993. Springer. doi:10.1007/3-540-57318-6_30.
- 3 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994. doi:10.1016/0304-3975(94)90010-8.
- 4 Rajeev Alur and P. Madhusudan. Decision Problems for Timed Automata: A Survey. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems*, Lecture Notes in Computer Science, pages 1–24. Springer, Berlin, Heidelberg, 2004. doi:10.1007/978-3-540-30080-9_1.
- 5 Vitaliy Batusov, Giuseppe De Giacomo, and Mikhail Soutchanski. Hybrid temporal situation calculus. In Marie-Jean Meurs and Frank Rudzicz, editors, *Advances in Artificial Intelligence*, Lecture Notes in Computer Science, pages 173–185, Cham, 2019. Springer. doi:10.1007/978-3-030-18305-9_14.

- 6 Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Updatable timed automata. *Theoretical Computer Science*, 321(2):291–345, August 2004. doi:10.1016/j.tcs.2004.04.003.
- 7 Béatrice Bérard and Catherine Dufourd. Timed automata and additive clock constraints. *Information Processing Letters*, 75(1):1–7, July 2000. doi:10.1016/S0020-0190(00)00075-2.
- 8 Jens Claßen and Gerhard Lakemeyer. A logic for non-terminating Golog programs. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 589–599. AAAI Press, 2008. URL: <https://aaai.org/papers/KR08-058-a-logic-for-non-terminating-golog-programs/>.
- 9 Jens Claßen, Martin Liebenberg, Gerhard Lakemeyer, and Benjamin Zarriß. Exploring the boundaries of decidable verification of non-terminating Golog programs. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1012–1019. AAAI Press, 2014. doi:10.1609/AAAI.V28I1.8875.
- 10 Giuseppe De Giacomo, Yves Lespérance, Matteo Mancanelli, and Gianmarco Parretti. Reactive synthesis for Golog specifications in the propositional situation calculus. In *Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2026.
- 11 Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121:109–169, 2000. doi:10.1016/S0004-3702(00)00031-X.
- 12 Giuseppe De Giacomo, Yves Lespérance, Hector J. Levesque, and Sebastian Sardina. IndiGolog: A high-level programming language for embedded reasoning agents. In *Multi-Agent Programming*. Springer, 2009. doi:10.1007/978-0-387-89299-3.
- 13 Giuseppe De Giacomo, Yves Lespérance, and Fabio Patrizi. Bounded situation calculus action theories and decidable verification. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*, May 2012.
- 14 Giuseppe De Giacomo, Yves Lespérance, and Fabio Patrizi. Bounded situation calculus action theories. *Artificial Intelligence*, 237:172–203, August 2016. doi:10.1016/j.artint.2016.04.006.
- 15 Alberto Finzi and Fiora Pirri. Representing flexible temporal behaviors in the situation calculus. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 436–441, 2005. URL: <http://ijcai.org/Proceedings/05/Papers/1562.pdf>.
- 16 Michael Gelfond, Vladimir Lifschitz, and Arkady Rabinov. What Are the Limitations of the Situation Calculus? In *Automated Reasoning: Essays in Honor of Woody Bledsoe*, pages 167–179. Springer Netherlands, Dordrecht, 1991. doi:10.1007/978-94-011-3488-0_8.
- 17 Yilan Gu and Mikhail Soutchanski. A description logic based situation calculus. *Annals of Mathematics and Artificial Intelligence*, 58(1):3–83, February 2010. doi:10.1007/s10472-010-9176-z.
- 18 Till Hofmann and Jens Claßen. LTLf synthesis on first-order agent programs in nondeterministic environments. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI)*, volume 39, pages 14976–14986, 2025. doi:10.1609/aaai.v39i14.33642.
- 19 Till Hofmann, Stefan Schupp, and Gerhard Lakemeyer. Decidable reasoning about time in finite-domain situation calculus theories, 2026. doi:10.48550/arXiv.2402.03164.
- 20 Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, March 1986. doi:10.1007/BF03037383.
- 21 John McCarthy. Situations, actions, and causal laws. Technical Report, Stanford University, 1963.
- 22 Rob Miller. A case study in reasoning about actions and continuous change. In *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI)*, page 21, 1996.
- 23 Marvin Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Englewood Cliffs, N.J, 1967.

- 24 Erik T. Mueller. Event Calculus. In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, *Foundations of Artificial Intelligence*, volume 3 of *Handbook of Knowledge Representation*, pages 671–708. Elsevier, January 2008. doi:10.1016/S1574-6526(07)03017-9.
- 25 Joël Ouaknine and James Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science*, 3(1):1–27, February 2007. Publisher: Episciences.org. doi:10.2168/LMCS-3(1:8)2007.
- 26 Javier Pinto and Raymond Reiter. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, 14(2-4):251–268, 1995. doi:10.1007/BF01530822.
- 27 Fiora Pirri and Ray Reiter. Some contributions to the metatheory of the situation calculus. *Journal of the ACM*, 46(3):325–361, 1999. doi:10.1145/316542.316545.
- 28 Raymond Reiter. Natural Actions, Concurrency and Continuous Time in the Situation Calculus. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 2–13, San Francisco, CA, 1996. Morgan Kaufmann Publishers Inc.
- 29 Raymond Reiter. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT Press, 2001.
- 30 Kristof Van Belleghem, Marc Denecker, and Danny De Schreye. On the relation between situation calculus and event calculus. *The Journal of Logic Programming*, 31(1):3–37, April 1997. doi:10.1016/S0743-1066(96)00123-9.
- 31 Benjamin Zarri   and Jens Cla  en. Decidable verification of Golog programs over non-local effect actions. In Dale Schuurmans and Michael Wellman, editors, *Proceedings of the thirtieth AAAI conference on artificial intelligence (AAAI 2016)*, pages 1109–1115. AAAI Press, 2016. doi:10.1609/AAAI.V30I1.10109.

A Proofs

► **Theorem 2.** *The reachability problem for a real-valued BAT is undecidable.*

Proof. Let $Incrs = \{(p, c, q)_i\}_i$ be the finite set of increment instructions, where p is the instruction label, $c \in \{1, 2\}$ is the index of the counter to be incremented, and q is the next instruction label. Similarly, let $Decrs = \{(p, c, q, r)_i\}_i$ be the finite set of decrement instructions, where p is the instruction label, $c \in \{1, 2\}$ is the index of the counter to be decremented, q is the jump instruction if the condition is true, and r is the jump instruction otherwise. We define a BAT $\mathcal{D}_M$ corresponding to a 2CM $\mathcal{M}$ as follows: We use the unary relational fluent $Next(i)$ to describe the next instruction i , two nullary functional fluents f_1, f_2 for counter values (we omit parentheses and write f_1 instead of $f_1()$), and the nullary relational fluent $Halt$ which is true iff $\mathcal{M}$ halted. We define the BAT $\mathcal{D}_M$ as follows. Initially, both counting fluents are set to zero and the next instruction is s_0 : $\mathcal{D}_0 = \{f_1 = 0, f_2 = 0, Next(i) \equiv i = s_0\}$. An action is possible if it is the next instruction: $Poss(i) \equiv Next(i)$. The SSA for each counting fluent $f_i \in \{f_1, f_2\}$ works as follows: If the last instruction was an increment instruction from $Incrs$ and it incremented c_i , then the fluent f_i is incremented as well. Similarly, if the last instruction was a decrement instruction and the counter value was larger than zero, then the fluent is decremented, otherwise it remains unchanged: $f_i(\text{do}(a, s)) = n \equiv \bigvee_{(p,c,q) \in Incrs} a = p \wedge c = i \wedge n = f_i + 1 \vee \bigvee_{(p,c,q,r) \in Decrs} a = p \wedge c = i \wedge (f_i > 0 \wedge n = f_i - 1 \vee f_i = 0 \wedge n = 0) \vee f_i = n \wedge \bigwedge_{(p,c,q) \in Incrs} (a \neq p \vee c \neq i) \wedge \bigwedge_{(p,c,q,r) \in Decrs} (a \neq p \vee c \neq i)$. The SSA for $Next$ works as follows: If the last instruction was an increment instruction that sets the next instruction to q , then set $Next$ accordingly. Similarly, if the last instruction was a decrement instruction, then branch on the value of the corresponding counter (i.e., whether $f_i > 0$) and set the next instruction accordingly: $Next(i) \equiv \bigvee_{(p,c,q) \in Incrs} a = p \wedge i = q \vee \bigvee_{(p,c,q,r) \in Decrs} a = p \wedge ((c = 1 \wedge f_1 > 0 \vee c = 2 \wedge f_2 > 0) \wedge i = q \vee (c = 1 \wedge f_1 = 0 \vee c = 2 \wedge f_2 = 0) \wedge i = r)$.

By construction, $\mathcal{D}_M \models \exists s \text{ Exec}(s) \wedge Halt(s)$ iff the 2CM reaches **HALT**. ◀

► **Theorem 3.** *The time-bounded reachability problem for a real-valued BAT and a time bound $b \geq 2$ is undecidable.*

Proof. The construction is the same as in Theorem 2, except that counting fluents work differently: A counter value u is encoded by a fluent value 2^{1-u} . Hence, every increment of f_i is replaced by an update to $\frac{f_i}{2}$ and every decrement of f_i is replaced by an update to $2f_i$. Furthermore, a comparison $f_i = 0$ is replaced by $f_i = 2$ and $f_i > 0$ is replaced by $f_i < 2$. Hence, the initial situation is $\mathcal{D}_0 = \{f_1 = 2, f_2 = 2, \text{Next}(i) \equiv i = s_0\}$. The precondition axiom is: $\text{Poss}(i) \equiv \text{Next}(i)$. The SSA for each counting fluent $f_i \in \{f_1, f_2\}$ is defined as: $f_i(\text{do}(a, s)) = n \equiv \bigvee_{(p,c,q) \in \text{Incrs}} a = p \wedge c = i \wedge n = \frac{f_i}{2} \vee \bigvee_{(p,c,q,r) \in \text{Decrs}} a = p \wedge c = i \wedge (f_i < 2 \wedge n = 2f_i \vee f_i = 2 \wedge n = 2) \vee f_i = n \wedge \bigwedge_{(p,c,q) \in \text{Incrs}} (a \neq p \vee c \neq i) \wedge \bigwedge_{(p,c,q,r) \in \text{Decrs}} (a \neq p \vee c \neq i)$. The SSA for Next is defined as: $\text{Next}(i) \equiv \bigvee_{(p,c,q) \in \text{Incrs}} a = p \wedge i = q \vee \bigvee_{(p,c,q,r) \in \text{Decrs}} a = p \wedge ((c = 1 \wedge f_1 < 2 \vee c = 2 \wedge f_2 < 2) \wedge i = q \vee (c = 1 \wedge f_1 = 2 \vee c = 2 \wedge f_2 = 2) \wedge i = r)$

By construction, $\mathcal{D} \models \exists s \text{Exec}(s) \wedge \text{Bounded}_2(s) \wedge \text{Halt}(s)$ iff the 2CM reaches the instruction **HALT**. ◀

► **Theorem 4.** *The reachability problem for timed BATs without real-valued functional fluents is undecidable.*

Proof. By reduction of the language emptiness problem for TAs with 4 clocks extended with additive clock constraints of the form $c_1 + c_2 = 1$, which is undecidable [7]. Let $\mathcal{A} = (L, l_0, L_F, \Sigma_{\mathcal{A}}, X, E)$ be such a TA. For an edge $e = (l, a, g, Y, l')$, we write l_e, g_e, Y_e, l'_e for its components, and for each clock $x \in X$, we write $R_x := \{e \in E \mid x \in Y_e\}$ for the set of edges resetting x . We define a BAT $\mathcal{D}_{\mathcal{A}}$ corresponding to $\mathcal{A}$ as follows. For each edge $e \in E$, there is one action type $E_e(t)$ with occurrence time t . For each location $l \in L$, there is a relational fluent $\text{Loc}_l(s)$. Finally, for each clock $x \in X$, there is a process $\text{Reset}_x(t, s)$, where t is an argument of sort time. Let $\text{start}(S_0) = 0$. Initially, $\mathcal{A}$ is in location l_0 , and every clock s was last reset at time 0: $\text{Loc}_{l_0}(S_0), \neg \text{Loc}_l(S_0)$ for each $l \neq l_0$, $\text{Reset}_x(t, S_0) \equiv t = 0$ for each $x \in X$. For a clock constraint g and variable t of sort time, the formula $\mu_g(t, s)$ is defined by

- $\mu_{x \bowtie r}(t, s) := \exists t_x. \text{Reset}_x(t_x, s) \wedge t - t_x \bowtie r$
- $\mu_{x+y=1}(t, s) := \exists t_x, t_y. \text{Reset}_x(t_x, s) \wedge \text{Reset}_y(t_y, s) \wedge (t - t_x) + (t - t_y) = 1$
- $\mu_{g_1 \wedge g_2} := \mu_{g_1}(t, s) \wedge \mu_{g_2}(t, s)$

An edge action is possible iff time does not decrease, $\mathcal{A}$ is in the source location, and the guard holds at the occurrence time: $\text{Poss}(E_e(t), s) \equiv t \geq \text{start}(s) \wedge \text{Loc}_{l_e}(s) \wedge \mu_{g_e}(t, s)$. The SSA for Loc_l states that $\mathcal{A}$ is in location l after an action iff that action is an edge with target l : $\text{Loc}_l(\text{do}(a, s)) \equiv \bigvee_{e \in E, l'_e = l} \exists t. a = E_e(t)$. The SSA for Reset_x states that the last reset time of x is the occurrence time of the current action if that action resets x , and is unchanged otherwise: $\text{Reset}_x(t, \text{do}(a, s)) \equiv \bigvee_{e \in R_x} a = E_e(t) \vee \text{Reset}_x(t, s) \wedge \bigwedge_{e \in R_x} \forall t'. a \neq E_e(t')$. By construction, $\mathcal{D}_{\mathcal{A}} \models \exists s. \text{Exec}(s) \wedge \bigvee_{l \in L_F} \text{Loc}_l(s)$ iff there is an accepting timed word of $\mathcal{A}$. ◀

► **Theorem 10.** *Let ϕ be a regressable formula. Then $\mathcal{R}[\phi]$ is uniform in S_0 and $\mathcal{D} \models \phi$ iff $\mathcal{D}_0 \cup \mathcal{D}_c \models \mathcal{R}[\phi]$.*

Proof. By induction on the maximal length of all situation terms in ϕ and structural sub-induction on ϕ .

Base case. Let $\sigma = S_0$. Then $\mathcal{R}[\phi] = \phi$ and $\mathcal{D} \models \phi$ iff $\mathcal{D}_0 \cup \mathcal{D}_c \models \phi$ as ϕ is uniform in S_0 .

Induction step. Let $\sigma = \text{do}(\alpha, \sigma')$. Here, we only consider the case of formulas of the form $\phi = f(\vec{\rho}, \sigma) \bowtie r$ and refer to [27] for the other cases. If $\alpha = \text{wait}(t)$ for some $t \in \mathbb{R}_{\geq 0}$, then $\mathcal{R}[f(\vec{\rho}, \sigma) \bowtie r] = \mathcal{R}[f(\vec{\rho}, \sigma') \bowtie r']$ with $r' = r - t$. By induction, $\mathcal{D}_0 \cup \mathcal{D}_c \models \mathcal{R}[\phi_f(\vec{\rho}, \alpha, \sigma) \bowtie r']$ iff $\mathcal{D} \models \phi_f(\vec{\rho}, \alpha, \sigma) \bowtie r'$. Otherwise (if α is not *wait*), by the definition

of the SSA for f , $\mathcal{D} \models f(\vec{\rho}, \text{do}(\alpha, \sigma)) = y \equiv \phi_f(\vec{\rho}, \alpha, \sigma) \wedge y = 0 \vee \neg \phi_f(\vec{\rho}, \alpha, \sigma) \wedge y = f(\vec{\rho}, \sigma)$ and so $\mathcal{D} \models f(\vec{\rho}, \text{do}(\alpha, \sigma)) \bowtie r \equiv \phi_f(\vec{\rho}, \alpha, \sigma) \wedge 0 \bowtie r \vee \neg \phi_f(\vec{\rho}, \alpha, \sigma) \wedge f(\vec{\rho}, \sigma) \bowtie r$. By induction, this holds iff $\mathcal{D}_0 \cup \mathcal{D}_c \models \mathcal{R}[\phi_f(\vec{\rho}, \alpha, \sigma) \wedge 0 \bowtie r \vee \neg \phi_f(\vec{\rho}, \alpha, \sigma) \wedge f(\vec{\rho}, \sigma) \bowtie r]$ and so $\mathcal{D} \models f(\vec{\rho}, \text{do}(\alpha, \sigma)) \bowtie r$ iff $\mathcal{D}_0 \cup \mathcal{D}_c \models \mathcal{R}[f(\vec{\rho}, \text{do}(\alpha, \sigma)) \bowtie r]$. $\blacktriangleleft$

► **Lemma 14.** *If $\mathcal{D}$ is a clocked BAT and σ a situation, then ν_σ always exists and is unique.*

Proof. By definition of clocked BATs, each clock is initialized to 0. Furthermore, once the initial value is fixed, the SSAs uniquely define each clock value in all successor situations. $\blacktriangleleft$

► **Lemma 17.** *Let $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$ and let ϕ be a $\mathcal{K}$ -bounded clocked formula uniform in s . Then $\mathcal{D} \models \phi[\sigma_1]$ iff $\mathcal{D} \models \phi[\sigma_2]$.*

Proof. Let $f : \nu_{\sigma_1} \rightarrow \nu_{\sigma_2}$ be a bijection witnessing $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$. We show that $\mathcal{D} \models \phi[\sigma_1]$ iff $\mathcal{D} \models \phi[\sigma_2]$ by structural induction on ϕ .

- Let $\phi = R(\vec{\rho}, s)$ for some fluent predicate R . By Definition 12, it directly follows that $\mathcal{D} \models \phi[\sigma_1]$ iff $\mathcal{D} \models \phi[\sigma_2]$.
- Let $\phi = \omega \bowtie \tau$ for some ground situation-suppressed clock term $\omega \in \mathcal{C}$ and $\tau \in \mathbb{N}$. Let $\nu_{\sigma_1}(\omega) = v_1$ and $\nu_{\sigma_2}(\omega) = v_2$. As f is a witness for $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$, it follows that $f(\omega, v_1) = (\omega, v_2)$ and $v_1 \sim_{\mathcal{K}} v_2$. By Definition 15, $v_1 \bowtie \tau$ iff $v_2 \bowtie \tau$ and so $\mathcal{D} \models \omega[\sigma_1] \bowtie \tau$ iff $\mathcal{D} \models \omega[\sigma_2] \bowtie \tau$.
- The cases for Boolean connectives and quantification follow by induction. $\blacktriangleleft$

► **Theorem 19.** *$\approx_{\mathcal{DK}}$ is a time-abstract bisimulation.*

Proof. We show that $\approx_{\mathcal{DK}}$ satisfies the criteria from Definition 12. Let $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$ such that $f : \nu_{\sigma_1} \rightarrow \nu_{\sigma_2}$ is a bijection witnessing $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$.

1. Let ϕ be a clocked formula uniform in s . It directly follows from Lemma 17 that $\mathcal{D} \models \phi[\sigma_1]$ iff $\mathcal{D} \models \phi[\sigma_2]$.
2. Let $A \neq \text{wait}$ be some action type, $\vec{\rho}$ a ground tuple of arguments. Let $\sigma'_1 = \text{do}(A(\vec{\rho}), \sigma_1)$ and $\sigma'_2 = \text{do}(A(\vec{\rho}), \sigma_2)$. We show that $\sigma'_1 \approx_{\mathcal{DK}} \sigma'_2$:
 - a. Let $R \in \mathcal{F}$ be a relational fluent and $\vec{\rho}'$ a ground tuple of arguments. We show that $\mathcal{D} \models R(\vec{\rho}', \sigma'_1)$ iff $\mathcal{D} \models R(\vec{\rho}', \sigma'_2)$. There is an SSA for R of the form $R(\vec{\sigma}, \text{do}(a, s)) \equiv \phi_R(\vec{\sigma}, a, s)$, where $\phi_R(\vec{\sigma}, a, s)$ is a clocked formula uniform in s . As $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$, it directly follows from Lemma 17 that $\mathcal{D} \models \phi_R(\vec{\rho}', A(\vec{\rho}), \sigma_1)$ iff $\mathcal{D} \models \phi_R(\vec{\rho}', A(\vec{\rho}), \sigma_2)$ and therefore $\mathcal{D} \models R(\vec{\rho}', \sigma'_1)$ iff $\mathcal{D} \models R(\vec{\rho}', \sigma'_2)$.
 - b. We construct a bijection $f' : \nu_{\sigma'_1} \rightarrow \nu_{\sigma'_2}$ witnessing $\approx_{\mathcal{DK}}$. Let c be a functional fluent and $\omega = c(\vec{\rho}')$. There is an SSA of the form $c(\vec{\sigma}, \text{do}(a, s)) = y \equiv \exists t. a = \text{wait}(t) \wedge y = c(\vec{\sigma}, s) + t \vee \neg \exists t. a = \text{wait}(t) \wedge (\phi_c(\vec{\sigma}, a, s) \wedge y = 0 \vee \neg \phi_c(\vec{\sigma}, a, s) \wedge y = c(\vec{\sigma}, s))$. Thus, $\omega[\sigma'_1] = 0$ if $\mathcal{D} \models \phi_c(\vec{\rho}, \alpha, \sigma_1)$ and $\omega[\sigma'_1] = \omega[\sigma_1]$ otherwise, similarly for $\omega[\sigma'_2]$. Now, as $\phi_c(\vec{\rho}, a, s)$ is a clocked formula uniform in s , by Lemma 17, $\mathcal{D} \models \phi_c(\vec{\rho}, \alpha, \sigma_1)$ iff $\mathcal{D} \models \phi_c(\vec{\rho}, \alpha, \sigma_2)$. Hence, for each ω , we can set $f'(\omega, 0) = (\omega, 0)$ if $\mathcal{D} \models \phi_c(\vec{\rho}, \alpha, \sigma_1)$ and $f'(\omega, \nu_{\sigma_1}(\omega)) = (\omega, \nu_{\sigma_2}(\omega))$ otherwise.
3. Let $\tau_1 \in \mathbb{R}_{\geq 0}$ and $\sigma'_1 = \text{do}(\text{wait}(\tau_1), \sigma_1)$. As $\nu_{\sigma_1} \cong_{\mathcal{K}} \nu_{\sigma_2}$, it follows with Proposition 18 that there is some $\tau_2 \in \mathbb{R}_{\geq 0}$ such that $\nu_{\sigma_1} + \tau_1 \cong_{\mathcal{K}} \nu_{\sigma_2} + \tau_2$. Let $\sigma'_2 = \text{do}(\text{wait}(\tau_2), \sigma_2)$. We show that $\sigma'_1 \approx_{\mathcal{DK}} \sigma'_2$:
 - a. Let $R \in \mathcal{F}$ be a relational fluent and $\vec{\rho}'$ a ground tuple of arguments. We show that $\mathcal{D} \models R(\vec{\rho}', \sigma'_1)$ iff $\mathcal{D} \models R(\vec{\rho}', \sigma'_2)$. There is an SSA for R of the form $R(\vec{\sigma}, \text{do}(a, s)) \equiv \phi_R(\vec{\sigma}, a, s)$, where $\phi_R(\vec{\sigma}, a, s)$ is a clocked formula uniform in s . As $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$, it directly follows from Lemma 17 that $\mathcal{D} \models \phi_R(\vec{\rho}', \text{wait}(\tau_1), \sigma_1)$ iff $\mathcal{D} \models \phi_R(\vec{\rho}', \text{wait}(\tau_2), \sigma_2)$ and therefore $\mathcal{D} \models R(\vec{\rho}', \sigma'_1)$ iff $\mathcal{D} \models R(\vec{\rho}', \sigma'_2)$.

- b. Let c be a functional fluent and $\omega = c(\vec{\rho}')$. By definition of the SSA for c , it follows that $\nu_{\sigma'_1}(\omega) = \nu_{\sigma_1}(\omega) + \tau_1$ and $\nu_{\sigma'_2}(\omega) = \nu_{\sigma_2}(\omega) + \tau_2$. We can set $f'(\omega, \nu_{\sigma'_1}(\omega)) = (\omega, \nu_{\sigma'_2}(\omega))$. As $\nu_{\sigma_1} + \tau_1 \cong_{\mathcal{K}} \nu_{\sigma_2} + \tau_2$, f' is a bijection witnessing $\approx_{\mathcal{DK}}$. $\blacktriangleleft$

► **Lemma 21.** $[\sigma]_{\approx_{\mathcal{DK}}}$ is reachable in $\mathcal{T}_{\mathcal{D}}$ iff there is some $\sigma' \in [\sigma]_{\approx_{\mathcal{DK}}}$ with $\mathcal{D} \models \text{Exec}(\sigma')$.

Proof.

⇐: By induction on the number n of do operators in σ .

Base case. Let $n = 0$. Therefore, $\sigma = S_0$. By definition, $[S_0]_{\approx_{\mathcal{DK}}}$ is reachable in $\mathcal{T}_{\mathcal{D}}$.

Induction step. Let $\sigma = \text{do}(\alpha, \sigma^*)$ such that $\mathcal{D} \models \text{Exec}(\sigma)$. By induction, $[\sigma^*]_{\approx_{\mathcal{DK}}}$ is reachable from $[S_0]_{\approx_{\mathcal{DK}}}$ in $\mathcal{T}_{\mathcal{D}}$. By definition of $\mathcal{T}_{\mathcal{D}}$, $[\sigma^*]_{\approx_{\mathcal{DK}}} \rightarrow [\sigma]_{\approx_{\mathcal{DK}}}$ is a transition in $\mathcal{T}_{\mathcal{D}}$ and so $[\sigma]_{\approx_{\mathcal{DK}}}$ is also reachable in $\mathcal{T}_{\mathcal{D}}$.

⇒: Follows immediately by the definition of $\mathcal{T}_{\mathcal{D}}$. $\blacktriangleleft$

► **Lemma 22.** The time-abstract transition system $\mathcal{T}_{\mathcal{D}}$ has finitely many states.

Proof. As there are only finitely many fluents $\mathcal{F}$, objects $\mathcal{O}$, and clock regions, $\approx_{\mathcal{DK}}$ has finitely many equivalence classes. Also, as $\mathcal{O}$ and $\mathcal{A}$ are finite, each state only has finitely many successors. Therefore, the time-abstract transition system $\mathcal{T}_{\mathcal{D}}$ has finitely many states. $\blacktriangleleft$

► **Lemma 24.** Let σ_1 and σ_2 be two situations. Then it is decidable whether $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$.

Proof. For every fluent $R \in \mathcal{F}$ and parameters $\vec{\rho}$, $\mathcal{D} \models R(\vec{\rho}, \sigma_1)$ iff $\mathcal{D}_0 \cup \mathcal{D}_c \models \mathcal{R}[R(\vec{\rho}, \sigma_1)]$, similarly for σ_2 . Following Theorem 11, checking whether $\mathcal{D}_0 \cup \mathcal{D}_c \models \mathcal{R}[R(\vec{\rho}, \sigma)]$ is decidable. Furthermore, because $\mathcal{D}$ has finitely many objects and finitely many fluents, there are only finitely many fluent atoms that need to be checked. Finally, checking for the existence of a bijection $f : \nu_{\sigma_1} \rightarrow \nu_{\sigma_2}$ can be done by enumerating all possible bijections (of which there are finitely many) and checking whether all requirements are satisfied. $\blacktriangleleft$

► **Lemma 25.** For a clocked BAT $\mathcal{D}$, Algorithm 1 terminates and returns the time-abstract transition system $\mathcal{T}_{\mathcal{D}}$.

Proof. Let $\mathcal{T}_{\mathcal{D}}$ be the time-abstract TS of a BAT $\mathcal{D}$ and T the TS returned by Algorithm 1. We show for every path p starting in $[S_0]_{\approx_{\mathcal{DK}}}$ that p is a path in $\mathcal{T}_{\mathcal{D}}$ iff p is a path in T .

⇒: Let $p = [S_0]_{\approx_{\mathcal{DK}}} \rightarrow^* [\sigma]_{\approx_{\mathcal{DK}}}$ be a path in $\mathcal{T}_{\mathcal{D}}$. We show by induction on the length n of p that p is a path in T .

Base case. Let $n = 0$, i.e., p is the empty path, which is also a path in T .

Induction step. Let $p = [S_0]_{\approx_{\mathcal{DK}}} \rightarrow^* [\sigma^*]_{\approx_{\mathcal{DK}}} \rightarrow [\sigma]_{\approx_{\mathcal{DK}}}$. By induction, $[S_0]_{\approx_{\mathcal{DK}}} \rightarrow^* [\sigma^*]_{\approx_{\mathcal{DK}}}$ is a path in T . It remains to be shown that $[\sigma^*]_{\approx_{\mathcal{DK}}} \rightarrow [\sigma]_{\approx_{\mathcal{DK}}}$ is a transition in T . Clearly, $\sigma = \text{do}(\alpha, \sigma^*)$ for some α . We distinguish two cases:

1. Assume $\alpha = A(\vec{\rho})$ for some action type $A \in \mathcal{A} \setminus \{\text{wait}\}$. α is added to Acts in line 4. By definition of $\mathcal{T}_{\mathcal{D}}$, $\mathcal{D} \models \text{Exec}(\sigma)$ and so $\mathcal{D} \models \text{Poss}(\alpha, \sigma^*)$. Hence, $[\sigma]_{\approx_{\mathcal{DK}}} \in V$ (either it is added in line 8 or it was in V before) and $([\sigma^*]_{\approx_{\mathcal{DK}}}, [\sigma]_{\approx_{\mathcal{DK}}})$ is added to E in line 10.
2. Otherwise, $\alpha = \text{wait}(\tau)$ for some $\tau \in \mathbb{R}_{\geq 0}$. By Proposition 23, there is a $\tau' \in \text{Succs}$ such that $\nu_{\sigma^*} + \tau \cong_{\mathcal{K}} \nu_{\sigma^*} + \tau'$ and so $\alpha' = \text{wait}(\tau')$ is added to Acts in line 4. Also, $\text{do}(\alpha, \sigma^*) \approx_{\mathcal{DK}} \text{do}(\alpha', \sigma^*)$, and so $[\text{do}(\alpha, \sigma^*)]_{\approx_{\mathcal{DK}}} = [\text{do}(\alpha', \sigma^*)]_{\approx_{\mathcal{DK}}} = [\sigma]_{\approx_{\mathcal{DK}}}$. Furthermore, as the precondition axiom of wait is $\text{Poss}(\text{wait}(t), s) \equiv t \geq 0$, it is clear that $\mathcal{D}_0 \cup \mathcal{D}_c \models \mathcal{R}[\text{Poss}(\alpha', \sigma^*)]$. Therefore, $[\sigma]_{\approx_{\mathcal{DK}}} \in V$ (either it is added in line 8 or it was already in V before) and $([\sigma^*]_{\approx_{\mathcal{DK}}}, [\sigma]_{\approx_{\mathcal{DK}}})$ is added to E in line 10.

⇐: Let $p = [S_0]_{\approx_{\mathcal{DK}}} \rightarrow^* [\sigma]_{\approx_{\mathcal{DK}}}$ be a path in T . We show by induction on the length $n = |p|$ of p that (1) $\mathcal{D} \models \text{Exec}(\sigma)$ and (2) p is a path in $\mathcal{T}_{\mathcal{D}}$.

Base case. Let $n = 0$, i.e., p is the empty path. Clearly, the empty path is also a path in $\mathcal{T}_D$. Also, by definition, $D \models \text{Exec}(S_0)$.

Induction step. Let $p = [S_0]_{\approx_{\mathcal{DK}}} \rightarrow^* [\sigma^*]_{\approx_{\mathcal{DK}}} \rightarrow [\sigma]_{\approx_{\mathcal{DK}}}$. By induction, $D \models \text{Exec}(\sigma^*)$ and $[S_0]_{\approx_{\mathcal{DK}}} \rightarrow^* [\sigma^*]_{\approx_{\mathcal{DK}}}$ is a path in $\mathcal{T}_D$. It remains to be shown that $D \models \text{Exec}(\sigma)$ and $[\sigma^*]_{\approx_{\mathcal{DK}}} \rightarrow [\sigma]_{\approx_{\mathcal{DK}}}$ is a transition in $\mathcal{T}_D$. Clearly, $\sigma = \text{do}(\alpha, \sigma^*)$ for some ground action term α . From the definition of Algorithm 1, it directly follows that $D \models \text{Poss}(\alpha, \sigma^*)$ and so $D \models \text{Exec}(\sigma)$. As $D \models \text{Exec}(\sigma)$, by definition of $\mathcal{T}_D$, $[\sigma^*]_{\approx_{\mathcal{DK}}} \rightarrow [\sigma]_{\approx_{\mathcal{DK}}}$ is a transition in $\mathcal{T}_D$.

We now show that the algorithm terminates. By Theorem 11 checking whether $D_0 \cup D_c \models \mathcal{R}[\text{Poss}(\alpha, \sigma)]$ is decidable. Also, by Lemma 24, checking for each new situation $\text{do}(\alpha, \sigma)$ whether there is an existing state $[\sigma]_{\approx_{\mathcal{DK}}} \in V$ (line 7) is decidable. Furthermore, by Lemma 22, $\mathcal{T}_D$ is finite, and so the loop eventually terminates. $\blacktriangleleft$

► **Lemma 30.** *Let σ_1, σ_2 be two situations with $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$ and $D \models \text{Trans}(\delta, \sigma_1, \alpha, \delta^*, \sigma'_1)$.*

1. *If $\alpha = \text{wait}(\tau_1)$ for some $\tau_1 \in \mathbb{R}_{\geq 0}$, then there is $\tau_2 \in \text{TSUCCS}(\sigma_2, K)$ such that $\text{Trans}(\delta, \sigma_2, \text{wait}(\tau_2), \delta^*, \sigma'_2)$ and $\sigma'_1 \approx_{\mathcal{DK}} \sigma'_2$.*
2. *If $\alpha = A(\vec{\rho})$ for some action type $A \neq \text{wait}$, then $\text{Trans}(\delta, \sigma_2, \alpha, \delta^*, \sigma'_2)$ and $\sigma'_1 \approx_{\mathcal{DK}} \sigma'_2$.*

Proof. First, note that by Lemma 17, for every clocked formula ϕ uniform in s , $D \models \phi[\sigma_1]$ iff $D \models \phi[\sigma_2]$ and so it directly follows that $D \models \text{Final}(\delta, \sigma_1)$ iff $D \models \text{Final}(\delta, \sigma_2)$.

1. Let $\alpha = \text{wait}(\tau_1)$ for some $\tau_1 \in \mathbb{R}_{\geq 0}$. By Proposition 23, there is a $\tau_2 \in \text{TSUCCS}(\sigma_2, K)$ such that $\nu_{\sigma_1} + \tau_1 \cong_{\mathcal{K}} \nu_{\sigma_2} + \tau_2$. Let $\sigma'_2 = \text{do}(\text{wait}(\tau_2), \sigma_2)$. With $D \models \text{Poss}(\text{wait}(t), s) \equiv \top$, it follows that $\text{Trans}(\delta, \sigma_2, \text{wait}(\tau_2), \delta^*, \sigma'_2)$. It remains to be shown that $\sigma'_1 \approx_{\mathcal{DK}} \sigma'_2$. Analogously to Theorem 19:
 - a. Let $R \in \mathcal{F}$ be a relational fluent and $\vec{\rho}'$ a ground tuple of arguments. We show that $D \models R(\vec{\rho}', \sigma'_1)$ iff $D \models R(\vec{\rho}', \sigma'_2)$. There is an SSA for R of the form $R(\vec{\sigma}, \text{do}(a, s)) \equiv \phi_R(\vec{\sigma}, a, s)$, where $\phi_R(\vec{\sigma}, a, s)$ is a clocked formula uniform in s . As $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$, it directly follows from Lemma 17 that $D \models \phi_R(\vec{\rho}', \text{wait}(\tau_1), \sigma_1)$ iff $D \models \phi_R(\vec{\rho}', \text{wait}(\tau_2), \sigma_2)$ and therefore $D \models R(\vec{\rho}', \sigma'_1)$ iff $D \models R(\vec{\rho}', \sigma'_2)$.
 - b. Let c be a func. fluent and $\omega = c(\vec{\rho}')$. By definition of the SSA for c , it follows that $\nu_{\sigma'_1}(\omega) = \nu_{\sigma_1}(\omega) + \tau_1$ and $\nu_{\sigma'_2}(\omega) = \nu_{\sigma_2}(\omega) + \tau_2$. Thus, we can set $f'(\omega, \nu_{\sigma'_1}(\omega)) = (\omega, \nu_{\sigma'_2}(\omega))$. As $\nu_{\sigma_1} + \tau_1 \cong_{\mathcal{K}} \nu_{\sigma_2} + \tau_2$, it follows that f' is a bijection witnessing $\approx_{\mathcal{DK}}$.
2. Let $\alpha = A(\vec{\rho})$ for some action type $A \neq \text{wait}$ and so $\sigma'_1 = \text{do}(\alpha, \sigma_1)$. There is a precondition axiom of the form $\text{Poss}(A(\vec{\sigma}), s) \equiv \phi_A(\vec{\sigma}, s)$ where $\phi_A(\vec{\sigma}, s)$ is a clocked formula uniform in s . As $\text{Trans}(\delta, \sigma_1, \alpha, \delta^*, \sigma'_1)$, it is clear that $D \models \phi_A(\vec{\rho}, \sigma_1)$ and with $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$, also $D \models \phi_A(\vec{\rho}, \sigma_2)$. Therefore, $D \models \text{Poss}(A(\vec{\rho}), \sigma_2)$ and so $D \models \text{Trans}(\delta, \sigma_2, \alpha, \delta^*, \sigma'_2)$ with $\sigma'_2 = \text{do}(\alpha, \sigma_2)$. It remains to be shown that $\sigma'_1 \approx_{\mathcal{DK}} \sigma'_2$. Analogously to Theorem 19:
 - a. Let $R \in \mathcal{F}$ be a relational fluent and $\vec{\rho}'$ a ground tuple of arguments. We show that $D \models R(\vec{\rho}', \sigma'_1)$ iff $D \models R(\vec{\rho}', \sigma'_2)$. There is an SSA for R of the form $R(\vec{\sigma}, \text{do}(a, s)) \equiv \phi_R(\vec{\sigma}, a, s)$, where $\phi_R(\vec{\sigma}, a, s)$ is a clocked formula uniform in s . As $\sigma_1 \approx_{\mathcal{DK}} \sigma_2$, it directly follows from Lemma 17 that $D \models \phi_R(\vec{\rho}', A(\vec{\rho}), \sigma_1)$ iff $D \models \phi_R(\vec{\rho}', A(\vec{\rho}), \sigma_2)$ and therefore $D \models R(\vec{\rho}', \sigma'_1)$ iff $D \models R(\vec{\rho}', \sigma'_2)$.
 - b. We construct a bijection $f' : \nu_{\sigma'_1} \rightarrow \nu_{\sigma'_2}$ witnessing $\approx_{\mathcal{DK}}$. Let c be a functional fluent and $\omega = c(\vec{\rho}')$. There is an SSA of the form $c(\vec{\sigma}, \text{do}(a, s)) = y \equiv \exists t. a = \text{wait}(t) \wedge y = c(\vec{\sigma}, s) + t \vee \neg \exists t. a = \text{wait}(t) \wedge (\phi_c(\vec{\sigma}, a, s) \wedge y = 0 \vee \neg \phi_c(\vec{\sigma}, a, s) \wedge y = c(\vec{\sigma}, s))$. Thus, $\omega[\sigma'_1] = 0$ if $D \models \phi_c(\vec{\rho}, \alpha, \sigma_1)$ and $\omega[\sigma'_1] = \omega[\sigma_1]$ otherwise, similarly for $\omega[\sigma'_2]$. Now, as $\phi_c(\vec{\rho}, a, s)$ is a clocked formula uniform in s , by Lemma 17, $D \models \phi_c(\vec{\rho}, \alpha, \sigma_1)$ iff $D \models \phi_c(\vec{\rho}, \alpha, \sigma_2)$. Hence, for each ω , we can set $f'(\omega, 0) = (\omega, 0)$ if $D \models \phi_c(\vec{\rho}, \alpha, \sigma_1)$ and $f'(\omega, \nu_{\sigma_1}(\omega)) = (\omega, \nu_{\sigma_2}(\omega))$ otherwise. $\blacktriangleleft$

► **Lemma 31.** *Let $\Delta = (\mathcal{D}, \delta)$ be a program over a clocked BAT $\mathcal{D}$ and $\mathcal{T}_\Delta$ the transition system returned by Algorithm 3. Then $\mathcal{T}_\Delta$ contains a state $([\sigma']_{\approx_{\mathcal{DK}}}, \delta')$ reachable from $([S_0]_{\approx_{\mathcal{DK}}}, \delta)$ with $\mathcal{D} \models \text{Final}(\delta', \sigma')$ iff there is a realization of program Δ .*

Proof. We first show that $([\sigma]_{\approx_{\mathcal{DK}}}, \delta')$ is reachable in $\mathcal{T}_\Delta$ iff $\mathcal{D} \models \text{Trans}^*(\delta, S_0, \vec{\alpha}, \delta', \sigma)$.

⇒: Let $p = ([S_0]_{\approx_{\mathcal{DK}}}, \delta) \rightarrow^* ([\sigma']_{\approx_{\mathcal{DK}}}, \delta')$ be a path in $\mathcal{T}_\Delta$. We show by induction on the length n of p that $\mathcal{D} \models \text{Trans}^*(\delta, S_0, \vec{\alpha}, \delta', \sigma')$ for some $\vec{\alpha}$ with $|\vec{\alpha}| = n$:

Base case. Let $n = 0$ and therefore $\sigma' = S_0$ and $\delta' = \delta$. Clearly, $\mathcal{D} \models \text{Trans}^*(\delta, S_0, \langle \rangle, \delta, S_0)$.

Induction step. Let $p = ([S_0]_{\approx_{\mathcal{DK}}}, \delta) \rightarrow^* ([\sigma^*]_{\approx_{\mathcal{DK}}}, \delta^*) \rightarrow ([\sigma']_{\approx_{\mathcal{DK}}}, \delta')$. By induction, $\mathcal{D} \models \text{Trans}^*(\delta, S_0, \vec{\alpha}, \delta^*, \sigma^*)$ for some $\vec{\alpha}$ with $|\vec{\alpha}| = n$. By construction of $\mathcal{T}_\Delta$, $\mathcal{D} \models \text{Trans}(\delta^*, \sigma^*, \alpha, \delta', \sigma')$ for some α and therefore also $\mathcal{D} \models \text{Trans}^*(\delta, S_0, \vec{\alpha} \cdot \alpha, \delta', \sigma')$.

⇐: Let $\mathcal{D} \models \text{Trans}^*(\delta, S_0, \vec{\alpha}, \delta', \sigma')$. We show by induction on the length n of $\vec{\alpha}$ that there is a path $p = ([S_0]_{\approx_{\mathcal{DK}}}, \delta) \rightarrow^* ([\sigma']_{\approx_{\mathcal{DK}}}, \delta')$ in $\mathcal{T}_\Delta$ with length n .

Base case. Let $n = 0$ and therefore $\sigma' = S_0$ and $\delta' = \delta$. Clearly, there is an empty path in $\mathcal{T}_\Delta$ starting in $([S_0]_{\approx_{\mathcal{DK}}}, \delta)$.

Induction step. Let $\mathcal{D} \models \text{Trans}^*(\delta, S_0, \vec{\alpha}, \delta^*, \sigma^*)$ for some $\vec{\alpha}$ with $|\vec{\alpha}| = n$ and let $\mathcal{D} \models \text{Trans}(\delta^*, \sigma^*, \alpha, \delta', \sigma')$. By induction, there is a path $p = ([S_0]_{\approx_{\mathcal{DK}}}, \delta) \rightarrow^* ([\sigma^*]_{\approx_{\mathcal{DK}}}, \delta^*)$ with length n . By Lemma 30, there is a α' such that $\mathcal{D} \models \text{Trans}(\delta^*, \sigma^*, \alpha', \delta', \sigma'')$ and $\sigma'' \approx_{\mathcal{DK}} \sigma'$. Hence, $([\sigma^*]_{\approx_{\mathcal{DK}}}, \delta^*) \rightarrow ([\sigma']_{\approx_{\mathcal{DK}}}, \delta')$ is a transition in $\mathcal{T}_\Delta$.

Now, as $([\sigma]_{\approx_{\mathcal{DK}}}, \delta')$ is reachable in $\mathcal{T}_\Delta$ iff $\mathcal{D} \models \text{Trans}^*(\delta, S_0, \vec{\alpha}, \delta', \sigma)$ for some $\vec{\alpha}$, it follows with Lemma 17 that $\mathcal{D} \models \text{Final}(\delta', \sigma)$ iff $\vec{\alpha}$ is a realization of Δ . ◀

Earth Observation Satellite Constellation Planning with Matheuristics and Metaheuristics Combination

Romain Barrault¹ 

DTIS, ONERA, Université de Toulouse, 31000, Toulouse, France

Cedric Pralet 

DTIS, ONERA, Université de Toulouse, 31000, Toulouse, France

Gauthier Picard 

DTIS, ONERA, Université de Toulouse, 31000, Toulouse, France

Eric Sawyer 

CNES, Toulouse, France

Abstract

A standard problem in the field of Earth observation is the scheduling of the observations of an agile satellite constellation. Given a set of end-user requests over Points Of Interest (POIs), it consists in selecting observations among the candidate ones, attributing each of them to a satellite, and defining the sequence of observations planned for each satellite given operational constraints. The latter are related to the visibility windows available to observe the POIs and the time-dependent maneuvers required to reorient the observation instrument between two POIs. They result in a highly combinatorial problem that must be solved in a restricted amount of time. To solve such a complex problem, we propose an approach that combines matheuristics to filter the observation tasks and metaheuristics to schedule them. Firstly, we solve a Sequential Ordering Problem for each satellite to get a giant tour visiting all the visible POIs. From this giant tour, we exploit a Linear Programming Model to compute the best set of observations under several tour length constraints. Finally, we schedule the selected observations based on a Large Neighborhood Search. This three-step method notoriously improves the solution quality when compared to a baseline scheduling approach.

2012 ACM Subject Classification Computing methodologies → Planning and scheduling; Applied computing → Operations research; Computing methodologies → Combinatorial algorithms; Mathematics of computing → Combinatorial optimization

Keywords and phrases Scheduling, Earth Observation Satellite, Matheuristic, Metaheuristic, Linear Programming

Digital Object Identifier 10.4230/OASICS.TIME.2026.12

1 Introduction

Earth observation constellations have become essential assets for a wide range of applications, from climate monitoring and disaster management to urban planning and security. The increasing number of satellites and the growing demand for timely, high-resolution data lead to a challenging planning and scheduling problem: each satellite must be assigned a sequence of observation tasks that respects strict temporal windows, resource constraints, and orbital dynamics while maximizing the overall scientific and economic value of the acquired imagery. This Multi-Satellite Earth Observation Scheduling Problem (MSEOSP) is intrinsically hard, as it can be formulated as a time-dependent Team Orienteering Problem with Time Windows (TD-TOP-TW) [10], where both the transition times and the profit of each observation vary with the exact acquisition time. Consequently, the problem is highly combinatorial

¹ corresponding author



© Romain Barrault, Cedric Pralet, Gauthier Picard, and Eric Sawyer;
licensed under Creative Commons License CC-BY 4.0

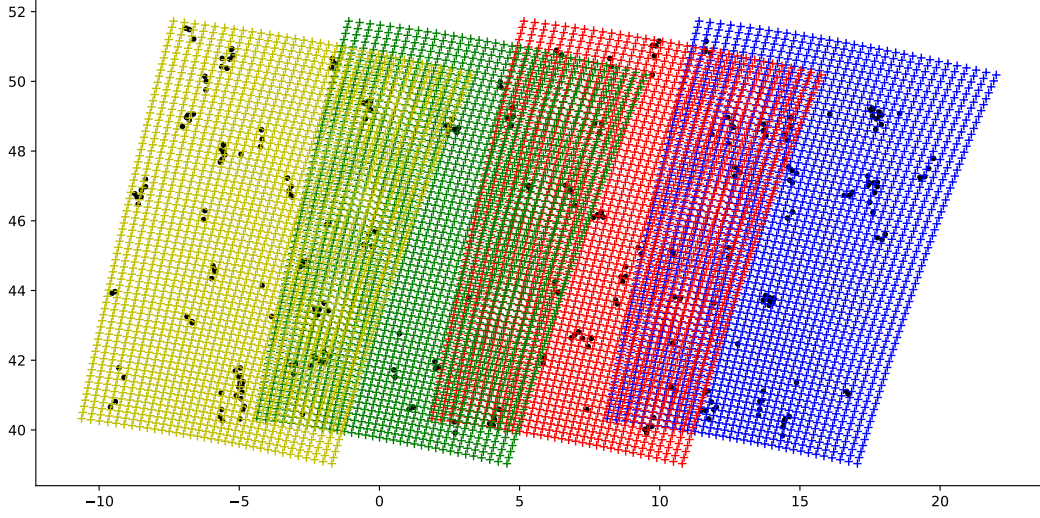
33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 12; pp. 12:1–12:17

OpenAccess Series in Informatics



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



■ **Figure 1** Sample problem instance represented in a Geographic Coordinates System: x-axis is latitude, y-axis is longitude. Black dots correspond to POIs. The colored meshings illustrate the areas successively overflight by four different satellites evenly separated on the same orbit, with a shift of the area overflight between two successive satellites due to the rotation of Earth.

(NP-hard), and highly sensitive to the interplay between multiple satellites, making exact solution methods impractical for realistic instance sizes and motivating the development of dedicated heuristic and metaheuristic methods.

In this paper, we propose a novel approach combining a matheuristic approach to select a subset of the (large) set of candidate observations for a constellation, and a metaheuristic scheduler, based on Large Neighborhood Search (LNS) [18, 19], to schedule the selected observations. The idea of the selector is to exploit the locations of the POIs and a simplified transition model in a Sequential Ordering Problem (SOP) solver in order to generate a so-called *giant tour* that then constrains a MILP aiming to select a set of observations maximizing the total reward. On this point, we note that giant tours are already used for vehicle routing problems [6], team orienteering problems [5], and even team orienteering problems with time windows [1]. In these related works, a unique giant tour visiting a subset of customers is split to get one tour per vehicle. Given that each satellite can see different POIs, the giant tour approach we propose is different: we build several giant tours (one per vehicle) taking into account some precedence constraints, and then select sub-tours of these giant tours to try and satisfy the temporal constraints. To further restrict which observations are selected based on the SOP ordering, we also consider sub-tours for sub-time frames to generate capacity constraints. Once a selection is provided by the selector, we use a LNS to schedule the observations, depending on their time windows and transition times.

The remainder of the paper is structured as follows. Section 2 retraces previous and current works on similar problems in the literature. Section 3 presents the core concepts and formalizes the problem. Section 4 overviews the global approach we follow. Next sections zoom in the main building blocks of our method: a matheuristic selector combining sequential ordering problem solving and an integer linear program that selects candidate observations to schedule, in Section 5, and a metaheuristics scheduler based on a large neighborhood search aimed to schedule the selected observations, in Section 6. Our approach is experimentally evaluated on a constellation composed of four agile low-orbit satellites, on scenarios where

the meteorological conditions (cloud coverage) impact the quality of the observations. We compare our approach to a baseline LNS running for 20 minutes in Section 6.2. Finally, Section 7 concludes the paper with perspectives.

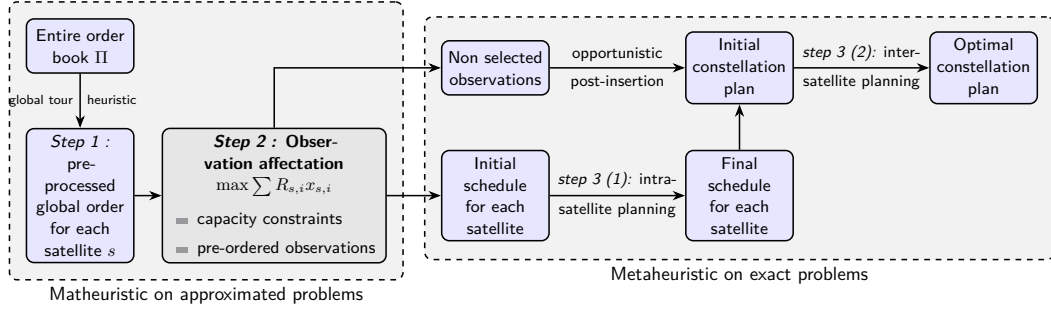
2 Background

The Multi-Satellite Earth Observation Scheduling Problem is a Time-Dependent Team Orienteering Problem with Time Windows and Time-Dependent Profit (TD-TOP-TW-TDP). The latter is an extension of TD-TOP-TW, where the goal is to design a set of routes for a fleet of vehicles to maximize the total collected profit while respecting a total time budget, time windows at visited locations, and time-dependent travel times, while maximizing the sum of time-dependent profits (*e.g.*, profit increasing with service time) [14]. Due to its computational complexity, exact methods, such as Mixed Integer Linear Programming (MILP) formulations, are typically only viable for small-scale instances [14]. For larger, more realistic scenarios, metaheuristics are the dominant solution approach. Effective metaheuristic techniques include Variable Neighborhood Search (VNS) [14], Iterated Local Search (ILS) [11], and nature-inspired algorithms like the Hybrid Artificial Bee Colony (HABC) algorithm, which are designed to balance exploration and exploitation of the search space [21]. These methods often employ tailored local search operations and insertion heuristics to efficiently handle the dynamic nature of travel times and the profit gained, which can depend on the vehicle's arrival and service time at a location [11]. Exact approaches based on Dynamic Programming (DP) and extensions of algorithms like A^* search have also been adapted for related time-dependent shortest path and TSP problems with time windows, often using concepts like partially time-expanded networks for improved performance [9].

In the space domain, Earth Observation Satellite Scheduling Problems require the maximization of total priority/profit from observation requests while respecting time windows (target visibility periods) and intricate operational constraints (*e.g.*, memory capacity, power availability, slew angle/maneuver time, and non-overlap of tasks) [7]. The core techniques mirror those for the TD-TOP-TW, centered around Mixed-Integer Linear Programming (MILP) for optimal solutions on small instances, and metaheuristics for large-scale real-world problems. Reinforcement learning techniques have also recently proven to be competitive in the context of multi-satellite scheduling [20, 16]. For large constellations or long planning horizons, decomposition methods and hybrid metaheuristics are crucial, often combining Local Search (like Iterated Local Search or Simulated Annealing) with greedy construction heuristics and calls to Constraint Programming (CP) solvers for solving sub-problems [2, 12]. MILP formulations often represent the problem using a time-space network or as a variation of the Traveling Salesman Problem with Time Windows (TSPTW) or Orienteering Problem [15]. The next section details the problem's features.

3 Problem Definition

We consider the problem of scheduling tasks for a low orbit agile EOS constellation. In this problem, observation requests over specific POIs are submitted to the system. Each satellite of the constellation overflies different candidate POIs along its orbit, but corresponding areas may overlap, implying a need to assign each observation to only one satellite (see Fig. 1). The observations also come up with an individual reward which depends on a cloud cover forecast and the time at which each satellite can observe the corresponding POI (time-dependent profit). The purpose is to compute the best constellation schedule, that means the set of individual schedules for each satellite which maximizes the total sum of rewards collected by the constellation.



■ **Figure 2** Constellation scheduling algorithm workflow.

3.1 Input Data: Satellite Observation Areas

We define our case study in order to measure the efficiency of the allocation part of our method without drastically increasing the problem complexity. Firstly, we define input data related to a single agile satellite that can point left, right, forward, or backward thanks to gyroscopic actuators. For this satellite, we consider a time horizon $[0, H_0]$ where $H_0 = N \cdot \delta_T$ is defined from a timestep δ_T and a number steps N . Then, we determine the Earth portion visible by the satellite during time frame $[0, H_0]$. This portion is centered on the satellite ground track and bounded in two dimensions: along track because of the restricted time horizon, and across track because of a maximum angle α_{max} beyond which observation quality is considered as too low. Any point within this area may be a candidate POI for the satellite (also called a target). For each target, we can also compute a Time Window (TW) during which the satellite can point to the target while having a visibility angle less than α_{max} (the farther the POI from the satellite ground track, the smaller the time window). Finally, each observation is associated with an individual reward in interval $[0.2, 1]$. This reward is obtained from a decreasing function of the local cloud cover forecast provided by real weather data.

Once the data for a single satellite is computed, we derive data for an entire EOS constellation with the following process. We define a set of satellites $\mathcal{S}$ along the same orbit, separated by a duration Δ_T . Basically, Δ_T is the time required for a satellite to reach the same orbital position as its predecessor. In our case, we consider a unique orbital plane and four satellites evenly spaced on this plane ($\Delta_T = 25$ minutes in the experiments). Satellite number s is considered over time frame $[s \times \Delta_T, s \times \Delta_T + H_0]$. From the set of Earth portions visible for the satellites, we get a set of potential candidate POIs.

3.2 Input data: Candidate POIs within the Observation Areas

For each satellite s , we generate a set of N_s candidate POIs within its area, using some meshing to avoid having very close candidate POIs. The i th candidate POI for satellite s is referred to as $P_{s,i}$, and the visibility window of this POI by satellite s is denoted by $W_{s,i} = [Ostart_{s,i}, Oend_{s,i}]$. For this POI, we also compute an individual reward $R_{s,i}$. Note that in our settings, each candidate POI p may be viewed by several satellites, that is we may have $p = P_{s,i}$ for one satellite s and $p = P_{s',i'}$ for another satellite s' . However, due to the time-dependent cloud cover, the observation reward for p may depend on the satellite (*i.e.*, $R_{s,i}$ not necessarily equal to $R_{s',i'}$). In the following, we denote Π the set of positions for the POI visible by at least one satellite ($\Pi = \cup_{s \in \mathcal{S}, i \in [1..N_s]} P_{s,i}$). At this point, the set of candidate POIs makes up for an instance of our problem, as represented in Figure 1.

3.3 Optimization Problem

As an input, we consider a set of POIs Π defining the so-called *Order Book*, and for each satellite s the set of candidate POIs ($\{P_{s,i} \mid i \in [1..N_s]\}$), their rewards ($\{R_{s,i} \mid i \in [1..N_s]\}$), and their time windows ($\{W_{s,i} \mid i \in [1..N_s]\}$). An additional input data comes from the time-dependent maneuvers between successive observations, in order for each satellite to point its observation instrument towards the right directions. We denote by $tt_{s,i,j,t}$ the duration of the maneuver by satellite $s \in \mathcal{S}$ when maneuvering from observation number $i \in [1..N_s]$ to observation number j , for a maneuver starting at time t . This transition function is time-dependent since it depends on the current position of the considered satellite on its orbit.

► **Definition 1.** *The Multi-Satellite Earth Observation Scheduling Problem (MSEOSP), for $\zeta = |\mathcal{S}|$ satellites, consists in finding ζ sequences of observations $\sigma_s = [\sigma_{s,1}, \dots, \sigma_{s,K_s}]$ such that:*

- *each candidate observation in Π appears at most once over all sequences in σ_s ;*
- *for each satellite $s \in \mathcal{S}$, all observations are performed during the available time window ($W_{s,i}$), while respecting the required transition times; formally, if number i is the first element of σ_s , the earliest start time for the corresponding observation is $t_{s,i} = Ostart_{s,i}$; if i and j are two consecutive elements in σ_s , then we have $t_{s,j} = \max(Ostart_{s,j}, t_{s,i} + tt(s,i,j,t_{s,i}))$; finally, the transition constraints are satisfied if and only if condition $t_{s,i} \leq Oend_{s,i}$ is satisfied for every observation number i belonging to σ_s ;*
- *the total collected reward $\sum_{s \in \mathcal{S}, i \in \sigma_s} R_{s,i}$ is maximized.*

In practice, the problem is usually over-constrained, that is it is usually not possible to observe all the candidate POIs.

4 Global Search Approach

To solve the optimization problem presented before, we exploit a two-step approach involving a matheuristic that computes an estimation of the optimal solution of our problem given coarse-grain constraints, and a metaheuristic that adapts this solution given the detailed constraints. The global search architecture is illustrated in Figure 2.

4.1 Step 1: Resolution of a Sequential Ordering Problem

First, for each satellite s , we compute a relevant observation order containing all the POIs it is able to see, in order to build a giant tour for s . Several methods can be used to get such a giant tour. A baseline method is the Earliest Start First (ESF, also referred to as *Earliest Due Date* [8]) heuristic that returns observations sorted by increasing window start times ($Ostart_{s,i}$), so that the satellite tries to observe any POI as soon as it is visible. In practice, this may lead to very bad solutions where along its orbit, the satellite is moving very often between the left and right parts of its ground track. This is why we exploit another method based on the resolution of a problem known as the *Sequential Ordering Problem* (SOP).

► **Definition 2.** *Given a weighted directed graph $G = (V, E)$ where we denote by $(c_e)_{e \in E}$ the set of costs over the edges of the graph, and a set of precedence constraints $\mathcal{P} \subset E$, the Sequential Ordering Problem (SOP) consists in the following optimization problem:*

$$\min_{\sigma \in E} \sum_{e \in \sigma} c_e \quad (1)$$

$$s.t. \quad \sigma = [(\sigma_1, \sigma_2), (\sigma_2, \sigma_3) \dots (\sigma_{|V|-1}, \sigma_{|V|})] \quad (2)$$

$$\forall i \in [1..|V|], \sigma_i \in V \quad (3)$$

$$\forall v \in V, \exists! i \in [1..|V|] s.t. \sigma_i = v \quad (4)$$

$$\forall (v, v') \in \mathcal{P}, indexOf(v, \sigma) < indexOf(v', \sigma) \quad (5)$$

This can be viewed as an *Asymmetric Traveling Salesman Problem* with added precedence constraints implementing the impossibility for certain nodes to follow other ones in the resulting sequence (constraint (5)). There is however no time window for visiting the nodes, nor time-dependency on the transition costs, hence this is an approximated version of our problem.

4.2 Step 2: Matheuristic to Dispatch POIs to Satellites

The SOPs solved provide us with a giant tour for each satellite. We then exploit these giant tours in a MILP model that assigns observations to satellites while taking into account the following constraints.

- Each POI may be assigned to at most one satellite.
- Given giant tour G_s for satellite s , the total cost associated with a selection of observations O_s for s corresponds to the total transition time (or cost) of the sub-tour of G_s induced by O_s . At this search step, this constraint allows us to get free of the decisions concerning POI observation ordering.
- The total time (or total cost) consumed by an observation sequence must be lower than a pre-determined value which is the duration between the earliest start time and the latest end time of candidate POI observations (overload checking, as in constraint programming methods). We also add local constraints on the observation sequence cost (more details later on this point).

To compute the total cost of a sequence while using only linear constraints in a MILP, we approximate the time-dependent transition times as time-independent transition times based on a specific policy, such as an *optimistic policy* considering the minimum transition duration among a set of possible discrete start times for the corresponding maneuver, or a *pessimistic policy* considering the maximum transition duration.

4.3 Step 3: Metaheuristic-based Scheduling

At the end of the previous step, we obtain a good subset of candidate POIs assigned to each satellite. At that time, we resort to existing scheduling algorithms described in [3] to

1. plan observations without changing the assigned satellite first, and
2. then consider exchanging acquisitions between the satellites.

The purpose is to first exploit the MILP result as a warm start for observation assignment, and then improve for the best the total plan quality. Basically, the existing scheduler introduced by Barrault et al. generates an initial plan thanks to a greedy method called **GreedySch** that iteratively inserts observations in the current plan. In this scheduler, the next observation to be inserted is selected based on heuristic values that depend on both the individual observation rewards and the additional maneuver durations required to insert observations. To improve the resulting plan, we resort to a *Large Neighborhood Search*

algorithm, namely `LNS to schedule`, which iteratively destroys and repairs the current plan based on these same heuristic values. In the first part of the process, this algorithm is computed at the same time locally on all satellites, and in the second part a single `LNS to schedule` is performed to destroy and repair the plans in a multi-satellite context. This is done during a pre-determined amount of time, and in the end, the plan that got the best reward sum is returned.

5 Matheuristics for Solving a Relaxed Routing Problem with Capacities

To quickly produce a first good quality solution, we resort to a *matheuristic* [4], that is to a technique that exploits mathematical programming techniques (MILP techniques in our case) to get heuristic solutions. The MILP model proposed in this context attempts to quickly produce good quality solutions by going beyond greedy heuristics that fail having a global view of the problem. Before detailing this MILP model, we present how we compute a global tour over all visible POIs from each satellite's perspective, based on the resolution of sequential ordering problems. The global tours obtained are then exploited in a MILP model that avoids considering scheduling decisions and focuses on pure observation selection and assignment and then we refer to the obtained sequences to select and attribute observations from the order book with respect to satellite capacity constraints.

5.1 Sequential Ordering Problem (SOP)

In our process, the SOP solved can be seen as a problem relaxation where the decisions are focused on the routing part, that is on the transition times between observations, while the time windows available to perform the observations are (almost) completely discarded. This SOP is solved thanks to the LKH solver presented in [13]. For a satellite s that sees POIs numbered from 1 to N_s , as we are only looking for a sequence of observations and not a tour over them, we add a fictitious node numbered 0 which makes up for both the departure and the ending of our sequence. This gives us a weighted directed graph $G = ([0..N_s], E)$ where the edges are weighted by an approximation of the transition times between the observations. A key part in the problem definition is the establishment of a set of mandatory precedence constraints $\mathcal{P}_s$ between the observations of satellite s . More precisely, for two distinct observation numbers $i, j \in [1..N_s]$, if the maneuver towards j starting from i at time $Ostart_{s,i}$ arrives after the end of the window of j (after $Oend_{s,j}$), then j must precede i in the resulting sequence, that is

$$(Ostart_{s,i} + tt(s, i, j, Ostart_{s,i}) > Oend_{s,j}) \rightarrow ((j, i) \in \mathcal{P}_s)$$

The previous rule may however lead to both $(j, i) \in \mathcal{P}_s$ and $(i, j) \in \mathcal{P}_s$ for certain observation couples with very narrow and overlapping time windows. In this case, to avoid precedence cycles, only one of the two precedence constraints is kept in $\mathcal{P}_s$. We then compute all edge weights with the optimistic transition times policy.

Executing LKH over the SOP built provides us with an observation sequence that contains all the observations, satisfies the precedence constraints, and minimizes the total tour cost. Figure 3 displays the kind of sequence returned, for a satellite whose ground track traverses the center of the area from the top to the bottom. We can see on the figure that in the solution found, the satellite limits large maneuvers going backward with regards to its ground track. This is quite natural since backward maneuvers, requiring the satellite to move counter to its orbital motion, are usually longer for such a system.

5.2 MILP Model

From now on, for satellite s , the index $i \in [1..N_s]$ of a candidate observation corresponds to position of this observation in the order returned by the SOP solver. Moreover, $P_{s,i}$ still denotes the i -th candidate observation point of satellite s and Π denotes the set of candidate POIs over all the satellites. To select a subset of the candidate observations and assign each of them to a satellite, we implement the following MILP that takes into account some global capacity constraints related to the time windows. This model is detailed thereafter.

$$\text{maximize } \sum_{s \in \mathcal{S}} \sum_{i \in [1..N_s]} R_{s,i} \cdot x_{s,i} \quad (6)$$

s.t.

$$\forall s \in \mathcal{S}, \forall i \in [1..N_s], x_{s,i} = \sum_{j > i} next_{s,i,j} \quad (7)$$

$$\forall s \in \mathcal{S}, \forall i \in [1..N_s], x_{s,i} = \sum_{j < i} next_{s,j,i} \quad (8)$$

$$\forall s \in \mathcal{S}, \sum_{j > 0} next_{s,0,j} = 1 \quad (9)$$

$$\forall s \in \mathcal{S}, \sum_{j \leq N_s} next_{s,j,N_s+1} = 1 \quad (10)$$

$$\forall p \in \Pi, \sum_{s \in \mathcal{S}, i \in [1..N_s] \text{ s.t. } P_{s,i}=p} x_{s,i} \leq 1 \quad (11)$$

$$\forall s \in \mathcal{S}, \forall w \in \mathcal{W}_s, \sum_{(i,j) \in U} next_{s,i,j} \cdot T_{s,i,j} \leq Dur_w \quad (12)$$

where $U = \{(i,j) \in [1..N_s]^2 \mid i < j, W_{s,i} \subseteq w \vee W_{s,j} \subseteq w\}$

$$\forall s \in \mathcal{S}, \forall i \in [1..N_s], x_{s,i} \in \{0,1\} \quad (13)$$

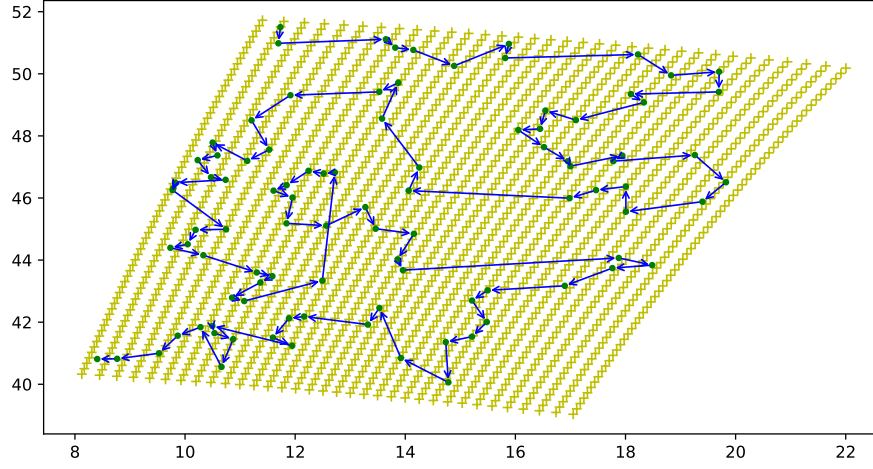
$$\forall s \in \mathcal{S}, \forall i, j \in [0..N_s + 1] \text{ s.t. } i < j, next_{s,i,j} \in \{0,1\} \quad (14)$$

For each satellite $s \in \mathcal{S}$ and observation point $p = P_{s,i} \in \Pi$, $x_{s,i}$ is a Boolean variable that takes value 1 if and only if p is selected in the observation sequence of satellite s . Variables $next_{s,i,j}$ are Boolean variables equal to 1 if and only if:

- points $P_{s,i}$ and $P_{s,j}$ are both selected for satellite s ,
- for each $k \in]i; j[$, $P_{s,k}$ is not selected for satellite s , that is $P_{s,j}$ is the next point selected after $P_{s,i}$ in the observation sequence returned by the SOP solver.

The objective function defined in Equation (6) corresponds to the total reward collected. Constraints (7) and (8) are standard flow constraints ensuring that each selected observation has a unique predecessor and successor in the sequence of selected observations. The constraints use two additional fictitious nodes: one node numbered 0 for the start of the observation plan, and one node numbered $N_s + 1$ for its end. Constraints (9) and (10) are particular cases for the start and end of the sequence. Constraint (11) models that each POI cannot be observed by more than one satellite.

Constraints (12) are capacity constraints over a set of specific time windows. More precisely, in these constraints, $T_{s,i,j}$ is the same approximated transition time between $P_{s,i}$ and $P_{s,j}$ as explained in Section 4. We also denote by $\mathcal{W}_s$ a set of time windows $w = [Wstart_w, Wend_w]$, of duration Dur_w , over which we attempt to prevent overloads in terms of transition times for satellite s . Set $\mathcal{W}_s$ is built as follows.



■ **Figure 3** Observation order (blue arrows) returned by the LKH solver over the visible set of observations (green dots) from a single satellite's perspective.

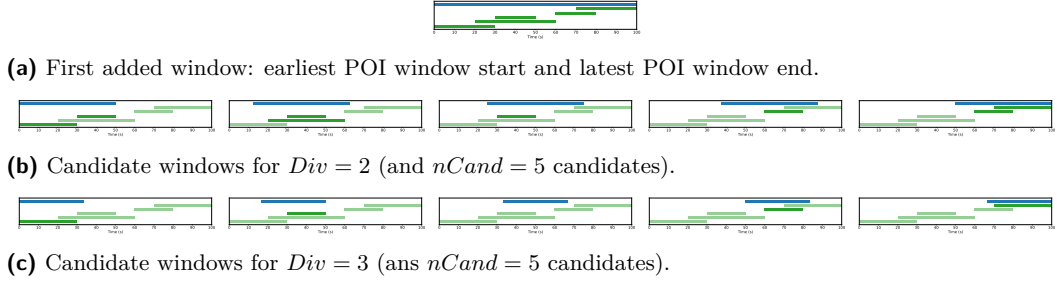
- The first added window is the larger one $\tilde{w}$:
 $\tilde{w} = [Wstart_{\tilde{w}}, Wend_{\tilde{w}}] = [\min_i Ostart_{s,i}, \max_i Oend_{s,i}]$ so that the resulting constraint concerns all observations for satellite s .
- We generate other windows iteratively. For this, we initiate an integer $Div = 2$. We create $nCand$ candidate windows of length $l = \frac{Wend_{\tilde{w}} - Wstart_{\tilde{w}}}{Div}$. We set $nCand = 5$ after some experiments. The windows generated are spread in $\tilde{w}$, which means that they start at one of the timestamps $Wstart_{\tilde{w}} + (k - 1) \cdot step$ for $k \in [1..nCand]$, where $step = \frac{Wend_{\tilde{w}} - l - Wstart_{\tilde{w}}}{nCand - 1}$. Among all these windows w , we keep only those verifying $\exists i \in [1..N_s], W_{s,i} \subseteq w$. If there are still windows satisfying this property, we increment Div and compute this step again, otherwise we stop the window generation process.

The set of capacity constraints given in Equation (12) limits the sum of transition times performed on numerous time windows. Figure 4 exemplifies these time windows' computations.

In the end, the MILP obtained solves a time-independent version of the MSEOSP introduced in Section 1, using a pre-determined order and without taking into account detailed time windows. Solving this MILP gives us both an assignment of observations to satellites (based on the $x_{s,i}$ variables) and for each satellite, a pre-determined order on the selected set of observations (based on the $next_{s,i,j}$ variables) by following the path from node 0 to node $N_s + 1$. Of course, following the order provided by the global tour can be sub-optimal, but the goal is to define a MILP that can be quickly solved to get a first heuristic solution.

6 Metaheuristic for the Detailed Scheduling

In this section, we describe the greedy search and the Large Neighborhood Search (LNS) metaheuristic used to compute “actual” MSEOSP schedules over the satellites, where “actual” means that no input data of the MSEOSP is approximated from this point. More specifically, the time windows and the exact time-dependent transition times are both taken into account to get actually feasible solutions. Both the greedy search and LNS exploit operators that update the satellite plans at the level of a single satellite (Section 6.1), or at the level of the entire constellation (Section 6.2).



■ **Figure 4** Example of the first three iterations of sub-windows generation for a single satellite and five POI. Blue windows are the candidate windows, and green windows are POI visibility windows. Highlighted green windows represent the ones that are comprised within the candidate window. $\mathcal{W}_s$ is the set of blue windows that fully contain at least one POI window. For $Div = 2$, all candidate windows are added to $\mathcal{W}_s$. For $Div = 3$, only the third candidate is not added to $\mathcal{W}_s$. The process would go on by incrementing Div until no candidate window can be added to $\mathcal{W}_s$.

6.1 Intra-satellite Moves

When the previously described MILP model returns all observations selected for a given satellite, there is a need to obtain an actual schedule satisfying all the MSEOSP constraints. This part describes two different ways to do so.

6.1.1 Greedy Repair Heuristics

6.1.1.1 Selection-based Greedy Repair Heuristic

This heuristic is described by algorithm 1. It starts either from an empty satellite schedule or one that is simply not full of observations, and fills this schedule according to a greedy policy that considers only the observations selected for the satellite in the MILP solution. The best observation to insert at any time is the one with the minimum insertion penalty. For this, the algorithm tests the addition of each unplanned observation at each possible position in the current schedule and computes an associated insertion penalty. The penalty value obtained for the insertion of observation $i \in [1..N_s]$ between two observations j and k in schedule σ_s is computed as:

$$\frac{tt(s, j, i, t_{s,j}) + tt(s, i, k, t_{s,i}) - tt(s, j, k, t_{s,j})}{R_{s,i}}$$

where $t_{s,j}$ describes the current observation time for j in σ_s and $t_{s,i} = t_{s,j} + tt(s, j, i, t_{s,j})$ gives the observation time of i after insertion (without considering the time window of i). The penalty value therefore takes into account both the temporal cost for inserting each acquisition at any position and the reward provided by the insertion. The downside of this method is that it ignores the formerly computed global order on $\sigma_s \cup \overline{\sigma_s}$. At any time during the process, the feasibility of an observation plan is verified using a **Earliest Start heuristic**.

6.1.1.2 Tour-based Greedy Repair Heuristic

The second greedy heuristic is very similar to the former one except that it only considers schedules respecting the global order induced by the values obtained for the $next_{s,i,j}$ variables at the level of the matheuristic (cf. Section 5). In this so-called *Tour-based Greedy Repair Heuristic*, a single insertion position is tested at each step for each unplanned observation. This approach considers a restricted set of schedules but is much faster. It also gives hindsight on the quality of the global tour's restriction to the current schedule.

Algorithm 1 Greedy Repair Heuristic.

Require: current schedule σ_s
 unplanned observations $\overline{\sigma_s}$

```

1:  $full \leftarrow false$ 
2: while  $!full$  do
3:    $\tilde{p} \leftarrow \infty$ 
4:    $full \leftarrow true$ 
5:   for  $acq \in \overline{\sigma_s}$  do
6:     for  $position \leq len(\sigma_s)$  do
7:       if  $insertable(i, position, \sigma_s)$  then
8:          $full \leftarrow false$ 
9:          $p \leftarrow penalty(acq, position, \sigma_s)$ 
10:        if  $p \leq \tilde{p}$  then
11:           $\tilde{p} \leftarrow p$ 
12:           $updateBestAcq(acq, position)$ 
13:   if  $!full$  then
14:      $insertBestAcq(\sigma, bestAcq, bestPosition)$ 
15:      $removeAcq(bestAcq, \overline{\sigma})$ 

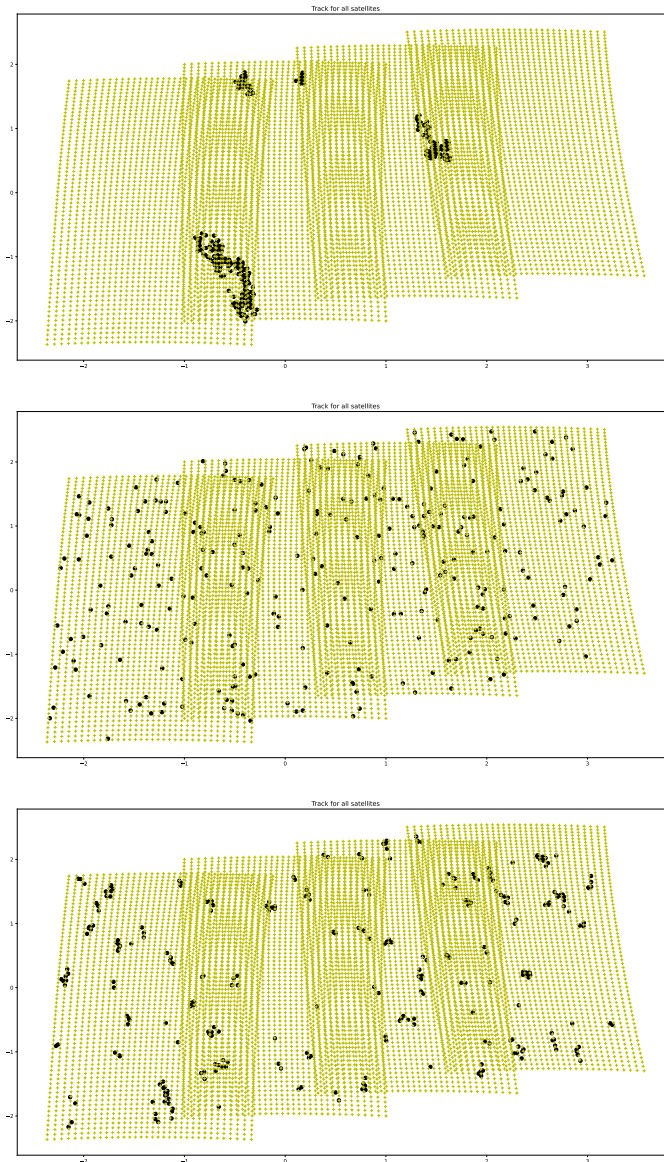
```

6.1.2 Randomized Greedy Destroy Heuristic

Once the greedy search process produced a full solution σ_s for each satellite s , the LNS part of the algorithm applies *destroy* and *repair* operations to iteratively enhance the schedule quality for s . The destroy operator used in this purpose removes K observations from the current schedule, with K randomly chosen between 1 and $|\sigma_s|/3$. At each destroy step, one observation is randomly removed from the schedule, with a higher removal probability for observations having a higher presence penalty. Here, we use the same penalty criterion as before: each observation in the current schedule has a presence penalty based on its reward and the time gained in case of removal.

6.2 Inter-satellites Moves

When all satellite schedules are optimized, the last part of the algorithm destroys and repairs the current solution at the constellation level. Also, any observation selected by the MILP solver that did not make it in the best schedule for the corresponding satellite after intra-satellite moves is considered as unworthy and will remain unused for the rest of the algorithm. Two ideas are exploited to try and get better rewards or smaller transitions: first, it can be relevant to reassign an observation to another satellite able to perform it, and second there may be some place left to insert observations not selected initially in the solution derived from the MILP. To tackle these two points, we define an Inter-Satellite Greedy Repair Heuristic (ISGRH) and a Randomized Greedy Destroy Heuristic (RGDH). These two heuristics are very similar to their intra-satellite counterpart. The only difference is that they browse all satellite current schedules before inserting or removing any observation, which implies that the lowest or highest penalty is computed over a wider set of observations and a wider set of insertion positions. The first step following the end of intra-satellite scheduling consists in the opportunistic post-insertion of observations not selected by the MILP. Empirically, the inter-satellite moves and the insertion of observations not selected by the MILP allow us to enhance the final schedule quality, especially when problem approximations lead to some



■ **Figure 5** 2D representations of instance examples for each archetype described in 6.3. From top to bottom: *dense*, *sparse* and *mixed*. Black dots correspond to POIs and each yellow grid displays the Earth area covered by a single satellite.

erroneous decisions at the MILP level. The counterpart of this scheduling process is that unlike intra-satellite moves, these computations cannot be performed in parallel. ISGRH and RGDH are also used to define our baseline LNS method. In this context, they start computations from an empty constellation schedule, and all observations can be inserted or removed since no MILP computation is involved. As a matter of fact, our problem's specificities make it really difficult to implement and adapt a SOTA method, mainly due to the time and vehicle-dependency of the transition times, the vehicle-dependency of the rewards and the vehicle-dependency of time windows. LNS still remains among the best candidates for solving such scheduling problems [19, 17]. We have therefore presented all components of our algorithm shown in Figure 2. The next part describes its performances.

6.3 Instances

The instance set considers 36 different POI distributions. For each of them, 300 observation points are chosen in the area covered by the four satellites, as in Figure 1. In addition, in order to test our algorithms on various realistic study cases, we designed three instance archetypes in terms of POI local density:

1. *dense* ones, where there are a handful of POI hubs with a high local density,
2. *sparse* ones, where most POIs are isolated from one another,
3. *mixed* ones, as a mix of both of these types, where POIs are either isolated or part of a very narrow hub.

These archetypes are displayed on figure 5. To compute observation rewards, 24 real cloud cover scenarios from year 2024 are retrieved ². For a cloud cover scenario, the reward $R_{s,i}$ associated with an observation is equal to one in case of no cloud cover over point $P_{s,i}$, 0.2 for a full cloud cover, and quadratically decreasing in between. More precisely, the resulting reward function is a piecewise constant version of the latter one with 5 possible values. In the end, we obtain 36 different POI distributions and 24 cloud cover scenarios (corresponding to 1st and 15th of each month of 2024). To limit the size of the instance set, we consider only 2 cloud cover scenarios per POI distribution (and each cloud cover scenario is applied to exactly 3 POI distribution scenarios). This results in a total of 72 problem instances. Finally, visibility time windows are defined according to a maximum observation angle α_{max} set to 30° in accordance with operational needs.

6.4 Experimental Setup

Firstly, two policies were implemented to return a global route over all satellite visible observations: the basic ESF heuristic that visits observation following their window start times, and the LKH heuristic that solves an SOP. Secondly, once the MILP solver returns a satellite-observation assignment, each detailed satellite schedule can be built based either on the Selection-based Greedy Repair Heuristic (denoted SelGH) or on the Tour-based Greedy Repair Heuristic (denoted TourGH). This leads to four different algorithm variants: ESF-SelGH, ESF-TourGH, LKH-SelGH, and LKH-TourGH.

The maximum CPU time for each algorithmic variant is set to 1 minute. A maximum CPU time of 40 seconds is dedicated to the matheuristic (including ESF or LKH, plus the resolution of the MILP model that is stopped when the optimality gap is less than 1%). The observation assignment problem is solved thanks to CPLEX solver, and the matheuristic usually converges much faster, typically in 5 seconds. Once the matheuristic produces a solution, at time t , the remaining time (1min minus t) is equally split between the intra-satellite LNS and the inter-satellite LNS. In the latter part, inter-satellite LNS operates with the possibility to insert observations that were not selected in the observation affectation process. Empirically, it allows us to enhance the final schedule's quality in case of imprecise choices in the latter part because of problem approximations. Thanks to problem decomposition, intra-satellite LNS is performed simultaneously over each satellite by multi-threading. All schedulers are implemented in Java and executed on 20-core Intel(R) Xeon(R) CPU ES-2660 v3 @ 2.60GHz, 62GB RAM, Ubuntu 18.04.5 LTS, with an OpenJDK 11.0.9 JVM.

The four algorithmic variants are compared to a baseline *LNS* algorithm (denoted Baseline) whose operators are ISGRH and RGDH.

² <https://cds.climate.copernicus.eu/datasets/reanalysis-era5-single-levels>

■ **Table 1** Algorithm versions results on 72 instances: optimality gaps (%) at the end of any step. The best value per column is highlighted in gray.

Algorithm	initial solution				post intra-moves				post inter-moves			
	max	median	mean	min	max	median	mean	min	max	median	mean	min
Baseline (20min)	16.4	5.29	6.11	1.41	n/a	n/a	n/a	n/a	10.5	0.792	1.57	0
Baseline (1min)	16.4	5.29	6.11	1.41	n/a	n/a	n/a	n/a	10.5	1.35	1.85	0
ESF-SelGH	15.6	9.38	9.81	4.14	15.0	6.71	7.12	3.25	4.37	1.44	1.52	0
ESF-TourGH	24.0	14.8	14.8	8.66	15.5	6.81	7.13	3.25	7.91	1.60	1.87	0
LKH-SelGH	15.1	8.16	8.52	2.78	11.0	5.12	5.17	0.831	5.46	0.699	1.11	0
LKH-TourGH	20.2	10.2	10.7	2.35	10.3	4.79	5.07	0.892	6.80	0.754	1.17	0

■ **Table 2** Median value of algorithms' final result depending on the instance type. The best value per instance type is highlighted in gray.

Instance type	dense	sparse	mixed
Baseline (20min)	0.870	0.784	0.792
Baseline (1min)	1.05	1.53	1.15
LKH-SelGH	0.654	1.16	0.620
LKH-TourGH	0.685	1.15	0.485

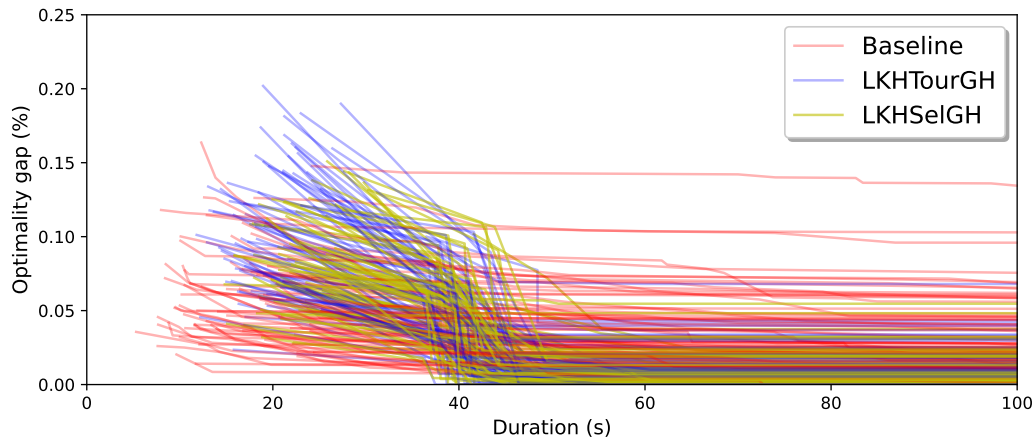
6.5 Result Analysis

Performance metrics are shown in Table 1, where we display the rewards' optimality gap: (i) after initial greedy schedule generation for each satellite, that means with inter-satellite SelGH for Baseline, and with the corresponding heuristic over each satellite's observation affectation after MILP solving for the other algorithms, (ii) after intra-satellite LNS, (iii) after inter-satellite LNS. This gap is computed as such: if bv is the best known reward on instance I with all algorithms, and algorithm A finds reward v on instance I at the end of step s , then the optimality gap (%) of A on I at the end of s is $100 \cdot (bv - v)/bv$. There are several results to point out. First, LKH-based algorithms perform better than ESF-based ones. Unfortunately, computing an initial schedule that respects the order obtained in the MILP solution gives worse results in most cases, since the initial solution reward computed with TourGH is lower than with SelGH, though it is less marked with LKH. A weakness in the algorithm architecture is that it requires intra-satellite scheduling to be competitive with a greedy heuristic that provides a solution very quickly, as shown in Figure 6. However, the overall performance of our LKH-based algorithms is very satisfying: after opportunistic post-insertion of non selected observations and inter-satellite scheduling, LKH-SelGH and LKH-TourGH beat the baseline algorithm when using the same amount of computation time. The optimality gap drop of blue and yellow curves in Figure 6 around 40 seconds represent the reward gain at the opportunistic post-insertion part of the algorithm, and the rest of inter-satellite moves provide a last chance improvement for constellation schedules. Our results also show that the instance archetype only slightly matters, as shown in Table 2.

Moreover, in terms of solution quality, LKH-SelGH and LKH-TourGH are competitive with the baseline algorithm running during 20 minutes. In our algorithms, the SOP solving takes several seconds per satellite. However, both LKH-SelGH and LKH-TourGH are able to outperform the baseline once in the inter-satellite phase. In other words, the observation plan resulting from the matheuristic steps serves as a good warm-start for the following LNS procedure.

7 Conclusion

This paper proposes a novel approach combining matheuristics and metaheuristics for solving EOS constellation scheduling problems. The main idea is to heuristically compute, for each satellite, a route visiting visible POIs, based on the resolution of SOPs and a multi-satellite



■ **Figure 6** Optimality gaps on each instance depending on CPU time for Baseline and LKH-based algorithms. One line corresponds to a single instance.

MILP that makes some problem approximations. After that, we reconstruct actually feasible schedules for the satellites, based on some called intra-satellite and inter-satellite moves. Through schedule destroy and repair operations, the algorithm obtained is able to outperform an efficient LNS algorithm in the same amount of time.

One of the bottlenecks remains the quality of the solutions produced by the MILP and the impact of some problem approximations. Indeed, our results show that the method should not be too committed to follow the pre-determined satellite-observation assignments and observation orders. One of the next steps is to build a version of our current MILP where transition costs would be artificially increased, and then to solve the initial problem while keeping the previous satellite-observation assignment decisions. The aim would be to build a more robust assignment algorithm. In this direction, we could try and learn the parameters of such an over-constrained model to get the best results. We also believe that a Constraint Programming approach would be relevant in replacement of the matheuristic part, and we would consider comparing these different approaches.

References

- 1 Youcef Amarouche, Rym Nesrine Guibadj, Elhadja Chaalal, and Aziz Moukrim. Effective neighborhood search with optimal splitting and adaptive memory for the team orienteering problem with time windows. *Computers & Operations Research*, 123:105039, 2020. doi:10.1016/j.cor.2020.105039.
- 2 Valentin Antuori, Damien T. Wojtowicz, and Emmanuel Hebrard. Solving the Agile Earth Observation Satellite Scheduling Problem with CP and Local Search. In Maria Garcia de la Banda, editor, *31st International Conference on Principles and Practice of Constraint Programming (CP 2025)*, volume 340 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:22, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CP.2025.3.
- 3 Romain Barrault, Cédric Pralet, Gauthier Picard, and Eric Sawyer. Hybridizing machine learning and optimization for planning satellite observations. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 22nd International Conference, CPAIOR 2025, Melbourne, VIC, Australia, November 10–13, 2025, Proceedings, Part I*, pages 68–85, Berlin, Heidelberg, 2025. Springer-Verlag. doi:10.1007/978-3-031-95973-8_5.

- 4 Marco Antonio Boschetti and Vittorio Maniezzo. Matheuristics: using mathematics for heuristic design. *4OR*, 20(2):173–208, 2022. doi:10.1007/S10288-022-00510-8.
- 5 Hermann Bouly, Duc-Cuong Dang, and Aziz Moukrim. A memetic algorithm for the team orienteering problem. In Mario Giacobini, Anthony Brabazon, Stefano Cagnoni, Gianni A. Di Caro, Rolf Drechsler, Anikó Ekárt, Anna Isabel Esparcia-Alcázar, Muddassar Farooq, Andreas Fink, Jon McCormack, Michael O’Neill, Juan Romero, Franz Rothlauf, Giovanni Squillero, A. Şima Uyar, and Shengxiang Yang, editors, *Applications of Evolutionary Computing*, pages 649–658, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-78761-7_71.
- 6 Diego Cattaruzza, Nabil Absi, and Dominique Feillet. The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science*, 50(2):676–693, 2016. doi:10.1287/TRSC.2015.0608.
- 7 Xiaoyu Chen, Gerhard Reinelt, Guangming Dai, and Andreas Spitz. A mixed integer linear programming model for multi-satellite scheduling. *European Journal of Operational Research*, 275(2):694–707, 2019. doi:10.1016/j.ejor.2018.11.058.
- 8 W. Dullaert, G.K. Janssens, K. Sirensen, and B. Vernimmen. New heuristics for the fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society*, 53(11):1232–1238, 2002. Cited by: 49. doi:10.1057/palgrave.jors.2601422.
- 9 Romain Fontaine, Jilles Dibangoye, and Christine Solnon. Exact and anytime approach for solving the time dependent traveling salesman problem with time windows. *European Journal of Operational Research*, 311(3):833–844, 2023. doi:10.1016/j.ejor.2023.06.001.
- 10 Ander Garcia, Pieter Vansteenwegen, Olatz Arbelaitz, Wouter Souffriau, and Maria Teresa Linaza. Integrating public transportation in personalised electronic tourist guides. *Computers & Operations Research*, 40(3):758–774, 2013. Transport Scheduling. doi:10.1016/j.cor.2011.03.020.
- 11 Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati Pantziou, and Nikolaos Vathis. Efficient heuristics for the time dependent team orienteering problem with time windows. In Prosenjit Gupta and Christos Zaroliagis, editors, *Applied Algorithms*, pages 152–163, Cham, 2014. Springer International Publishing. doi:10.1007/978-3-319-04126-1_13.
- 12 Al Globus, James Crawford, Jason Lohn, and Anna Pryor. A comparison of techniques for scheduling earth observing satellites. In *Proceedings of the 16th Conference on Innovative Applications of Artificial Intelligence*, IAAI’04, pages 836–843. AAAI Press, 2004.
- 13 Keld Helsgaun. An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems, December 2017. doi:10.13140/RG.2.2.25569.40807.
- 14 M. Khodadadian, A. Divsalar, C. Verbeeck, A. Gunawan, and P. Vansteenwegen. Time dependent orienteering problem with time windows and service time dependent profits. *Computers & Operations Research*, 143:105794, 2022. doi:10.1016/j.cor.2022.105794.
- 15 Minkeon Lee, Seunghyeon Yu, Kybeom Kwon, Myungshin Lee, Junghyun Lee, and Heungseob Kim. Mixed-integer linear programming model for scheduling missions and communications of multiple satellites. *Aerospace*, 11(1), 2024. doi:10.3390/aerospace11010083.
- 16 Bohua Li, Ming Chen, Lining Xing, Yingguo Chen, and Yingwu Chen. Optimizing time-dependent multi-agile earth observation satellite scheduling problem using deep q-learning and ensemble heuristics. *Information Sciences*, 712:122140, 2025. doi:10.1016/j.ins.2025.122140.
- 17 Xiaolu Liu, Gilbert Laporte, Yingwu Chen, and Renjie He. An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time. *Computers and Operations Research*, 86:41–53, 2017. Cited by: 246. doi:10.1016/j.cor.2017.04.006.
- 18 David Pisinger and Stefan Ropke. Large neighborhood search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, pages 99–127. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-319-91086-4_4.

- 19 Cédric Pralet. Iterated maximum large neighborhood search for the traveling salesman problem with time windows and its time-dependent version. *Computers & Operations Research*, 150:106078, 2023. doi:10.1016/j.cor.2022.106078.
- 20 Lili Ren, Xin Ning, and Zheng Wang. A competitive markov decision process model and a recursive reinforcement-learning algorithm for fairness scheduling of agile satellites. *Computers and Industrial Engineering*, 169, 2022. Cited by: 28. doi:10.1016/j.cie.2022.108242.
- 21 Vincent F. Yu, Parida Jewpanya, Shih-Wei Lin, and A.A.N. Perwira Redi. Team orienteering problem with time windows and time-dependent scores. *Computers & Industrial Engineering*, 127:213–224, 2019. doi:10.1016/j.cie.2018.11.044.

Multi-Agent Path Finding with Tasks Under Time Uncertainty

Alexandre Albore ✉ 

ONERA, University of Toulouse, France

Christophe Grand ✉ 

ONERA, University of Toulouse, France

Noëlie Ramuzat ✉ 

ONERA, University of Toulouse, France

Ajdin Sumic ✉ 

ONERA, University of Toulouse, France

Abstract

Multi-Agent Path Finding under Time Uncertainty (MAPF-TU) extends classical MAPF by considering uncertain traversal duration. However, existing approaches, such as the Conflict Based Search with Time Uncertainty (CBS-TU) method, assume that uncertainty only arises while agents move between locations, overlooking uncertainty induced by executing tasks. In many real-world applications, agents must not only navigate but also perform tasks of uncertain duration at specific locations, leading to additional temporal dependencies.

In this paper, we introduce Multi-Agent Path Finding with Tasks under Time Uncertainty (MAPF-TTU), a new extension of MAPF-TU in which agents must execute tasks of uncertain duration at intermediate goals. We first analyze the computational complexity of MAPF-TU and MAPF-TTU. We then propose two complementary solution approaches. The first extends CBS-TU to explicitly reason about uncertain task execution during planning. The second transforms an MAPF-TU solution into a Simple Temporal Network with Uncertainty (STNU) and computes a robust execution schedule through Strong Controllability.

Finally, we experimentally evaluate both approaches under varying levels of traversal and task time uncertainty, highlighting the tradeoff between solution quality and scalability.

2012 ACM Subject Classification Computing methodologies → Multi-agent planning; Computing methodologies → Planning under uncertainty; Computing methodologies → Temporal reasoning

Keywords and phrases Multi-Agent Path Finding, Temporal Uncertainty, Conflict Based Search, Simple Temporal Network

Digital Object Identifier 10.4230/OASICS.TIME.2026.13

Supplementary Material *Software*: https://github.com/ajdin14200/MAPF_TTU_TIME [31]
archived at `swb:1:dir:b90831099c338304e218af5159b6add58ed219b1`

Funding This work was supported by government funding administered by the French National Research Agency (ANR) under the France 2030 programme, reference number ANR-23-DMRO-0006.

1 Introduction

Multi-Agent Pathfinding (MAPF) is an NP-hard problem that consists of computing collision-free paths for a set of agents that must reach predefined goal locations from their initial positions [32]. MAPF has been extensively studied due to its relevance in many real-world applications, including transportation and logistics for warehouse automation [37, 18], robotics [29, 14], and video games [28]. Classical formulations of MAPF typically assume either discrete timesteps or deterministic continuous-time actions [36, 2], allowing the motion of the agents to be synchronized precisely and straightforwardly checked for conflicts.



© Alexandre Albore, Christophe Grand, Noëlie Ramuzat, and Ajdin Sumic;
licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: Andrea Orlandini and Sophie Pinchinat; Article No. 13; pp. 13:1–13:21

OpenAccess Series in Informatics



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

Over the last decade, a large body of work has focused on developing increasingly efficient MAPF solvers, including search-based methods, compilation approaches, prioritized planning, and hybrid techniques. Among these approaches, Conflict-Based Search (CBS) and its extensions have become particularly influential due to their balance between optimality and scalability [27, 7, 19]. CBS decomposes the problem into a high-level search that resolves conflicts between agents and a low-level search that computes individual paths under constraints. Numerous variants have subsequently extended CBS to continuous time, kinematic constraints, and execution robustness [1, 2, 8, 4, 11].

In contrast, the Prioritized Inheritance Backtracking (PIBT) algorithm is a well-known sub-optimal approach designed for large-scale MAPF instances involving thousands of agents [24]. Its main strength lies in its high computational efficiency and scalability, although this comes at the cost of lower solution quality compared to optimal approaches such as CBS.

However, the deterministic assumptions underlying classical MAPF significantly limit its applicability in realistic environments, where traversal durations may vary due to exogenous factors. Several works therefore focused on generating more robust execution strategies, either by guaranteeing feasibility under bounded delays [3, 15, 22], dynamically adapting agent priorities [30, 5], or repairing plans online during execution [16, 9, 13, 6, 20, 21].

In such settings, action durations are more naturally modeled as uncertain but bounded values rather than fixed ones. Multi-Agent Pathfinding with Time Uncertainty (MAPF-TU) extends MAPF by associating temporal intervals with edge traversals in the navigation graph [26]. Reasoning under temporal uncertainty fundamentally changes the nature of the problem. In deterministic MAPF, conflicts can be checked at precise timesteps, whereas MAPF-TU requires considering all possible realizations of uncertain durations. Two agents may conflict whenever their occupancy intervals overlap under some admissible execution scenario. Consequently, MAPF-TU requires safe plans that remain collision-free despite uncertain execution outcomes. Existing approaches typically rely on interval-based reasoning and robust conflict detection to guarantee safe execution [26, 33]. In particular, CBS_TU [26] extends CBS to reason directly over interval-valued plans and conflicts.

Although MAPF-TU considerably improves the realism of MAPF, existing formulations mainly focus on uncertainty induced by navigation actions, namely uncertain edge traversal durations. This assumption is sufficient when agents only move between locations, but it becomes inadequate in many practical applications where agents must also perform tasks at specific locations. In numerous real-world domains, these tasks may themselves exhibit uncertain durations.

Consider, for example, an automated warehouse where robots retrieve items from shelves. Uncertainty may arise not only from navigation times due to congestion, but also from picking operations because of object accessibility or grasping conditions.

Despite its practical importance, this second source of uncertainty has received little attention in the MAPF literature. Existing approaches generally assume that agents instantly complete tasks upon reaching a goal location, or that task durations are deterministic [38]. However, these assumptions may therefore lead to unrealistic schedules at execution, leading to online adaptation to avoid collisions.

In this work, we introduce a new paradigm called Multi-Agent Path Finding with Time and Task Duration Uncertainty (MAPF-TTU). In MAPF-TTU, each agent is assigned an ordered sequence of goals. Every goal except the final one requires the execution of a task of uncertain duration before the agent can proceed. Task durations are represented as temporal intervals similarly to uncertain traversal times. Consequently, uncertainty may arise both while moving between locations and while executing tasks at a location.

Introducing uncertain task durations substantially increases the complexity of temporal reasoning in MAPF. Unlike traversal uncertainty, task uncertainty occurs while an agent remains at a fixed location, as it can wait indefinitely before starting a task of uncertain duration or leaving. This introduces new temporal dependencies and conflicts, since an agent may occupy a shared resource for an uncertain duration while executing a task.

To address these challenges, we propose two complementary approaches. The first is an optimal extension of CBS_TU, called CBS_TTU, that explicitly reasons about uncertain task durations during planning. This extension augments the low-level search with explicit task execution states and interval-valued task actions, allowing the planner to compute robust solutions that account simultaneously for uncertain navigation and uncertain task execution.

Our second, suboptimal approach adopts a decoupled temporal reasoning perspective. Instead of integrating all temporal uncertainty directly into the MAPF search, we first compute a classical MAPF-TU solution and subsequently transform the resulting execution plan into a Simple Temporal Network with Uncertainty (STNU) [34]. In this transformation, traversal uncertainties and task execution uncertainties are modeled as *contingent* temporal constraints, while precedence relations induced by collisions are represented as *requirement* constraints. This enables the computation of a feasible execution policy that remains valid under all realizations of uncertainty.

The remainder of the paper is organized as follows. We first review the necessary background on MAPF, MAPF-TU, Temporal Plan Graphs, and Simple Temporal Networks with Uncertainty. We then formally define MAPF-TTU and present our two approaches. Subsequently, we provide an experimental evaluation comparing the approaches under varying levels of traversal and task uncertainty. Finally, we discuss the implications of our results and outline directions for future research on temporally robust multi-agent coordination.

2 Background

2.1 Multi-Agent Path Finding

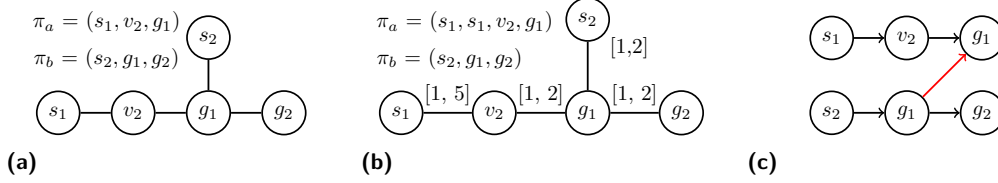
Multi-Agent Path Finding is the problem of computing collision-free paths for a set of agents moving in a shared environment from predefined start locations to goal locations [32, 29]. MAPF is generally defined as a tuple $\langle G, A, s, g \rangle$, where $G = (V, E)$ is a graph representing the environment, where vertices correspond to locations and edges represent feasible movements between adjacent locations. $A = \{a_i\}_{i \in \mathbb{N}}$ is the set of agents, while $s : A \rightarrow V$ and $g : A \rightarrow V$, respectively, maps each agent to its start and goal location.

At each timestep, an agent executes an action: it either moves to an adjacent vertex, or waits at its current location. A plan for an agent $a \in A$ is a sequence of vertices $\pi_a = (v_1, \dots, v_k)$, such that $v_1 = s(a)$ and $v_k = g(a)$. Consecutive vertices are either connected by an edge in E (move action) or identical (wait action).

The objective of MAPF is to compute a set of plans $\Pi = \{\pi_1, \dots, \pi_n\}$ such that no collisions occur between agents. Two main types of conflicts are generally considered: a *vertex conflict* occurs when two agents a and b occupy the same vertex $v \in V$ at the same timestep t and is defined as a tuple $\langle a, b, v, t \rangle$, an *edge conflict* occurs when two agents traverse the same edge $(u, v) \in E$ in opposite direction at the same timestep t and is defined as a tuple $\langle a, b, (u, v), t \rangle$. A solution to MAPF is therefore a set of collision-free plans satisfying all agents' goals.

From an MAPF solution, a directed acyclic graph can be constructed, called Temporal Plan Graph (TPG) [16], which captures the precedence relations required to preserve collision-free execution. In a TPG, vertices represent events in the agents' plans, while edges

encode temporal dependencies between these events, including both intra-agent sequencing constraints and inter-agent precedence constraints induced by conflict resolution. TPGs support online adaptation, rescheduling, and temporal reasoning during execution while preserving the safety guarantees of the original MAPF solution [30, 5, 16, 9]. Figure 1 illustrates an MAPF example with two agents and the corresponding optimal solution minimizing the makespan, transformed as a TPG.



■ **Figure 1** Figure a illustrates an MAPF problem with two agents a and b , starting respectively at nodes s_1 and s_2 and finishing at nodes g_1 and g_2 , respectively, with π_a and π_b the optimal collision-free solutions. Figure b illustrates the MAPF-TU version of the example in Figure a, where the optimal collision-free solution now requires agent a to wait one timestep on node s_1 to avoid potentially colliding with agent b in g_1 at timestep 2. Figure c illustrates the corresponding TPG of the optimal solution of Figure a with the inter-agent precedence highlighted in red, i.e., agent b (below) must pass by node g_1 before agent a (top). This TPG is also valid for Figure b.

2.2 Multi-Agent Path Finding with Time Uncertainty

The MAPF with Time Uncertainty (MAPF-TU) problem is an MAPF problem where travel duration between two positions is uncertain and can assume values within a fixed interval [26].

► **Definition 1** (MAPF-TU). *We define the Multi-Agent Path Finding problem with Time Uncertainty as a tuple $\mathcal{M} = \langle G, A, s, g, w^-, w^+ \rangle$ where:*

- $\langle G, A, s, g \rangle$ is an MAPF instance;
- $w^- : E \rightarrow \mathbb{R}^+$ is the function assigning the minimal possible duration to traverse an edge.
- $w^+ : E \rightarrow \mathbb{R}^+$ is the function returning the maximal possible duration to traverse an edge.

In addition, we refer by $[w^-(e), w^+(e)]$ the non-deterministic time interval that bounds the duration of traversing an edge $e \in E$: traversing e multiple times may lead to different travel duration each time. Hence, $w^-(e)$ and $w^+(e)$ are the lower and upper bounds of the durations associated to e .

Unlike deterministic MAPF, the exact arrival time of an agent at a vertex is no longer known precisely. Instead, each visit is associated with a temporal interval representing all possible arrival timesteps induced by uncertainty propagation, as shown in Figure 1b.

► **Definition 2** (Potential Presence). *Given an agent a and a plan $\pi_a = (v_1, \dots, v_n)$, the potential presence of a at vertex $v_k \in \pi_a$, denoted $\tau(\pi_a, v_k)$, is the interval of possible timesteps during which agent a may occupy v_k when following π_a . The potential presence is defined similarly for an edge.*

Reasoning under temporal uncertainty fundamentally changes conflict detection. In deterministic MAPF, conflicts are checked at exact timesteps, whereas MAPF-TU requires considering all possible realizations of uncertain durations.

► **Definition 3** (Potential Conflict). *Given two agents a and b , their plans π_a and π_b are in potential conflict on a vertex or edge x if $\tau(\pi_a, x) \cap \tau(\pi_b, x) \neq \emptyset$.*

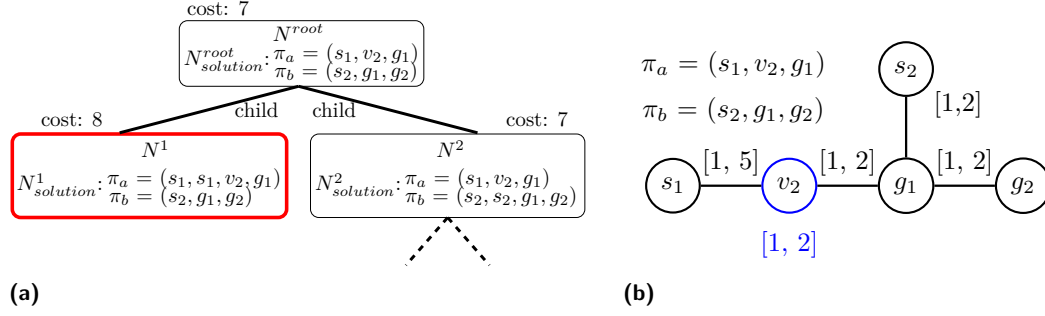


Figure 2 Figure a illustrates the execution of CBS on the provided MAPF-TU example in Figure 1b. Figure b illustrates the MAPF-TTU version of the MAPF-TU example in Figure 1b, where now agent a must perform a task at the intermediate goal node v_2 of uncertain duration $[1, 2]$. The optimal collision-free solution is given with π_a and π_b . The corresponding TPG of this solution is given in Figure 1c.

A potential conflict therefore exists whenever two potential presences overlap on the same vertex/edge. A valid MAPF-TU solution is a set of collision-free plans Π for all admissible realizations of traversal durations.

Several approaches have been proposed to solve MAPF-TU. Shahar et al. [26] extended CBS to uncertain traversal times through CBS_TU, which reasons directly over interval-valued conflicts while preserving completeness and optimality guarantees. More recently, Švancara et al. [33] proposed policy-based approaches capable of adapting agent actions dynamically according to observed traversal durations during execution.

2.3 Conflict-Based Search algorithm

CBS maintains a search tree, called *Constraint Tree* (CT), where each node represents a set of constraints and the associated solution. At each iteration, it selects the most promising node in the tree. If the node's solution has no conflicts, it is returned as the final plan. Otherwise, a conflict is resolved by branching the search tree on new nodes, each adding a constraint that forbids one side of the conflict. For each side of the conflict, a new constraint is added, and a constrained path is replanned for the involved agent. By systematically exploring the space of constraints in this way, CBS guarantees completeness and optimality. Nodes are selected according to their cost, usually defined by the makespan of the agents' plan. The first node of the tree is called *root node* and has no constraints, thus its solution contains the optimal individual path for each agent. In CBS, a low-level process computes new paths by finding the shortest path to the goal (e.g., using A*), whilst taking into consideration the set of constraints and a heuristic to guide the search. In MAPF-TU, the heuristic considers either the lower or the upper bound duration for each edge. In the present implementation, the latter option was used exclusively in order not to overestimate the delay too much.

Figure 2a shows the execution of CBS on the example in Figure 1b. The initial path computed for the agents is shown in the root CT node (N^{root}). This solution contains a vertex conflict on g_1 at $t = 2$. CBS_TU will resolve the conflict and creates two CT child nodes N^1 and N^2 , where N^1 is the collision-free solution. Please note that the search process will select N^2 first due to a lower cost and iterate until it select N^1 as the most promising node in the CT.

2.4 Simple Temporal Network with Uncertainty

A *Simple Temporal Network* (STN) is a tuple $\langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V}$ is a set of real-value variables called timepoints $\{\nu_1, \dots, \nu_n\}$, and $\mathcal{E}$ is a set of temporal constraints between timepoints called *requirements* [12]. Specifically, each requirement $r \in \mathcal{E}$ is of the form $\nu_y - \nu_x \in [l, u]$, with $\nu_x, \nu_y \in \mathcal{V}$, $l \in \mathbb{R} \cup \{-\infty\}$, and $u \in \mathbb{R} \cup \{+\infty\}$; l and u represent the minimal and maximal possible temporal distance between ν_x and ν_y . A requirement is represented as an edge $\nu_x \xrightarrow{[l, u]} \nu_y$. A reference timepoint ν_z is generally included in $\mathcal{V}$ as a fixed temporal anchor assumed to be the first executed timepoint of the network, i.e., ν_z will take a value such that each implicit constraint $\nu_z \leq \nu_x$ for each $\nu_x \in \mathcal{V}$ is satisfied. An STN is satisfiable if a timepoint assignment (schedule) exists that satisfies all the constraints. We say that a controller executes an STN when it schedules its timepoints. STNs have proven to be useful for MAPF by transforming TPGs into STNs to support execution flexibility and online adaptation in dynamic environment [16].

A *Simple Temporal Network with Uncertainty* (STNU) is an extension of the STN in which a subset of constraints is distinguished, with the values (duration) of these constraints being decided by external entities (e.g., *Nature*) and the controller can only observe them [34]. A STNU $\mathcal{X} = \langle \mathcal{V}, \mathcal{E}, \mathcal{C} \rangle$ is a tuple where $\mathcal{V}$ is now divided into controllable ($\mathcal{V}_c$) and uncontrollable ($\mathcal{V}_u$) timepoints. $\mathcal{C}$ is a set of (uncontrollable) contingent links each of the form $\nu_y - \nu_x \in [l, u]$, where $\nu_x \in \mathcal{V}_c, \nu_y \in \mathcal{V}_u$, and $0 < l \leq u$. The value (duration) $\omega = \nu_y - \nu_x$ is set by an external entity before or during the execution. Once ν_x is executed, the value of ν_y is $\nu_x + \omega$. A contingent link is depicted as $\nu_x \xrightarrow{[l, u]} \nu_y$. A *schedule* for a STNU is a mapping $\delta: \mathcal{V} \rightarrow \mathbb{R}$ from timepoints to real values.

► **Definition 4** (Situation, Projection and Solution). *Given a STNU $\mathcal{X} = \langle \mathcal{V}, \mathcal{E}, \mathcal{C} \rangle$, the situations of $\mathcal{X}$ are the set of tuples Ω defined as the Cartesian product of contingent domains: $\Omega = \times_{c_k \in \mathcal{C}} [l_k, u_k]$. Each situation $\omega = (\omega_1, \dots, \omega_n) \in \Omega$ represents one possible complete set of values for the durations of the contingent links of $\mathcal{X}$. A projection $\mathcal{X}^\omega = (\mathcal{V}, \mathcal{E} \cup \mathcal{C}^\omega)$ of $\mathcal{X}$ is an STN, where $\mathcal{C}^\omega$ replaces each contingent link c_k with the rigid requirement induced by ω_k . A solution of $\mathcal{X}^\omega$ is a schedule δ_ω satisfying all the constraints.*

In STNUs, consistency needs to be redefined: a network is now said to be *controllable* if there exists a schedule that satisfies all requirement constraints in any possible projection.

Three levels of controllability exist [34], depending on when the contingent durations are known: *Strong Controllability* (SC) assumes that observation at execution is not possible (as in conformant planning), hence, a single common schedule δ satisfying all constraints in all situations is required. *Weak Controllability* (WC) assumes all contingent durations will be known just before execution (oracle), hence, one need to guarantee a consistent schedule for each situation. *Dynamic Controllability* (DC) assumes that contingent durations are observable at execution, hence, the schedule values assignment depends on past observations only, regardless of the contingent durations still to be observed. In this paper, we will focus on the case of Strong Controllability, of which we recall the formal definition below.

► **Definition 5** (Strong Controllability). *A STNU $\mathcal{X}$ is said to be strongly controllable (SC) if there exists a schedule δ , called a strong schedule, such that for all situations ω , δ is a solution of $\mathcal{X}^\omega$.*

STNUs provide a natural formalism for reasoning about controllable and uncontrollable temporal events, making them particularly suitable for modeling uncertain traversal and task durations.

3 Multi-Agent Path Finding with Tasks under Time Uncertainty

We consider now an extension of MAPF-TU where agents must perform some task of uncertain but bounded duration in intermediate locations. To do so, we will identify those vertices where a durative task takes place as *intermediate goal locations*: they are specific to the goal assigned to each agent, and the solution plan must take them into account. We call such extension Multi-agent Path Finding with Tasks under Time Uncertainty (MAPF-TTU) and is defined as follows:

► **Definition 6** (MAPF-TTU). A MAPF-TTU is a tuple $\mathcal{P} = \langle G, A, s, g, w^-, w^+, M, t, \psi^-, \psi^+ \rangle$ where:

- $\langle G, A, s, g, w^-, w^+ \rangle$ is an MAPF-TU instance as in Definition 1.
- M is a set of intermediate goal locations $M \subseteq V$.
- $t : A \rightarrow M_a$ a function mapping agents to sets of intermediate locations, s.t. $t(a) = M_a$ is a set of intermediate goal locations assigned to agent a , where $M_a \subseteq M$
- $\psi^- : V \rightarrow \mathbb{R}^+$ is the function assigning the minimal possible duration on staying on an intermediate goal location $v \in M$;
- $\psi^+ : V \rightarrow \mathbb{R}^+$ is the function returning the maximal possible duration on staying on an intermediate goal location $v \in M$.

In addition, we refer by $[\psi^-(v), \psi^+(v)]$ the non-deterministic time interval that bounds the duration of staying in a vertex $v \in M$. Hence, $\psi^-(v)$ and $\psi^+(v)$ are the lower and upper bounds of the durations associated to v .

When an agent reaches an intermediate goal location, it must remain at the corresponding vertex during the entire task execution before continuing its plan. Consequently, uncertainty may arise both while traversing edges and while executing tasks at specific locations. Traversal uncertainty and task uncertainty are modeled independently through the functions w and ψ .

► **Definition 7** (MAPF-TTU single-agent plan). A single-agent plan for an agent a is a sequence of nodes $\pi_a = (v_1, \dots, v_n)$ that moves a from its start vertex $v_1 = s(a)$ to its goal vertex $v_n = g(a)$ such that: $\forall v \in M_a, v \in \pi_a$.

We assume that the task associated with an intermediate goal is executed only once upon the first visit of the corresponding agent.

The notion of potential presence is extended to account for both uncertain traversal durations and uncertain task durations. Intuitively, the uncertainty accumulated at a vertex corresponds to the sum of all previous traversal and task uncertainties along the path.

► **Definition 8** (MAPF-TTU Potential Presence). Given an agent a and its plan $\pi_a = (v_1, \dots, v_n)$. The potential presence of a at vertex $v_k \in \pi_a$, denoted $\tau(\pi_a, v_k)$, is defined as an interval in which a might be in vertex v_k , after traversing the edge (v_{k-1}, v_k) , with $(v_{k-1}) \in \pi_a$ s.t:

$$\begin{cases} [0, 0] & \text{if } k = 1 \\ [\tau^- + \psi^-(v_k) + w^-(v_{k-1}, v_k), \tau^+ + \psi^+(v_k) + w^+(v_{k-1}, v_k)] & \text{if } v_k \in M_a \\ [\tau^- + w^-(v_{k-1}, v_k), \tau^+ + w^+(v_{k-1}, v_k)] & \text{Otherwise} \end{cases}$$

where $\tau(\pi_a, v_{k-1}) = [\tau^-, \tau^+]$.

The potential presence is similarly defined for an edge.

We consider a set of plans to be solution for a MAPF-TTU if each agent visits its intermediate goal locations (see Definition 7) for a duration that is compatible with the potential presence (see Definition 8), in which no potential conflicts occur.

3.1 Complexity of MAPF-TTU

In this section, we analyze the computational complexity of MAPF-TTU. We first study the complexity of MAPF-TU, which, to the best of our knowledge, has not been explicitly characterized in the literature, although it can naturally be conjectured to be at least as hard as classical MAPF. We then show that introducing uncertain task durations preserves this computational hardness.

► **Definition 9.** *The MAPF-TU decision problem is defined by:*

Input: *An MAPF-TU instance $\mathcal{M} = \langle G, A, s, g, w^-, w^+ \rangle$ and a rational number κ .*

Question: *Is there a safe solution for $\mathcal{M}$ with cost at most κ ?*

► **Proposition 10.** *MAPF-TU is NP-hard.*

Proof. The NP-hardness of MAPF-TU follows directly from classical MAPF, which is NP-hard [32]. Given an MAPF instance, we can construct an equivalent MAPF-TU instance by setting: $w^-(e) = w^+(e) = 1$, for every $e \in E$. ◀

► **Definition 11.** *The MAPF-TTU decision problem is defined by:*

Input: *An MAPF-TTU instance $\mathcal{P} = \langle G, A, s, g, w^-, w^+, M, t, \psi^-, \psi^+ \rangle$ and a rational number κ .*

Question: *Is there a safe solution for $\mathcal{P}$ with cost at most κ ?*

► **Proposition 12.** *MAPF-TTU is NP-hard.*

Proof. The NP-hardness of MAPF-TTU follows immediately from the one of MAPF-TU given in Proposition 10. Any MAPF-TU instance can be transformed into an equivalent MAPF-TTU instance by setting: $M = \emptyset$, i.e., by not having any intermediate goals. The resulting MAPF-TTU instance is therefore equivalent to the original MAPF-TU instance. Since MAPF-TU is NP-hard, it follows that MAPF-TTU is also NP-hard. ◀

4 The CBS_TTU Algorithm

To solve MAPF-TTU optimally, we extend the CBS_TU algorithm proposed in [26]. The resulting algorithm, called CBS_TTU, preserves the overall structure of CBS_TU while extending the low-level search to explicitly reason about uncertain task execution.

Similarly to CBS_TU, the algorithm maintains a Constraint Tree. The high-level search remains unchanged: the algorithm iteratively selects the lowest-cost CT node, detects potential conflicts through interval overlaps, and resolves them by generating child nodes with additional constraints.

The main modification made here concerns the low-level planner. In addition to movement and wait actions, agents may now execute tasks at intermediate goal locations of uncertain duration: $[\psi^-(v), \psi^+(v)]$. During this action, the agent remains at the same location until the task is completed before continuing toward subsequent goals.

The low-level state representation is therefore augmented with a task execution status indicating whether the current sub-goal action has already been performed. Potential conflicts are detected exactly as in CBS_TU using interval overlap tests on potential presence intervals. However, these intervals may now originate both from uncertain edge traversals and uncertain task executions.

Since CBS_TTU preserves the same high-level conflict resolution mechanism as CBS_TU while extending the low-level search with additional uncertain actions, the completeness and optimality guarantees of CBS_TU are maintained.

5 The CBS_TU_STNU Algorithm

In this section, we present our STNU-based approach for MAPF-TTU called CBS_TU_STNU. We first solve the MAPF-TTU using CBS_TU to find collision free paths under uncertain edge traversal durations, then we use STNUs in order to compute a schedule by incorporating uncertain task durations to the given MAPF-TU solution.

Our method builds upon the approach proposed in [16], which extends Temporal Plan Graphs to Simple Temporal Networks in order to accommodate kinematic constraints. We further extend this approach to MAPF-TTU, with the aim of reasoning explicitly about temporal uncertainty, which arises in our benchmarks both from edge traversal and task execution durations.

5.1 Extending Temporal Plan Graphs to MAPF-TTU

Extending TPGs to MAPF-TTU is straightforward. Similarly to the classical MAPF setting, the TPG captures the precedence relations that must be preserved during execution in order to guarantee the absence of collisions. The main difference resides in the fact that events are now associated with potential presence intervals induced by uncertain traversal and task durations.

Building the TPG of a MAPF-TTU solution is shown in Algorithm 1. More precisely, each visit of an agent to a vertex in its plan is represented by a TPG node. Type-1 edges encode the sequential order of actions for each agent, while type-2 edges represent inter-agent precedence relations induced by conflict avoidance [16]. Since MAPF-TU solutions are collision-free, the potential presence intervals of two conflicting visits cannot overlap. Therefore, one interval precedes the other, which directly determines the corresponding precedence relation in the TPG.

The resulting graph compactly represents both the ordering of individual plans and the temporal dependencies required for safe execution under uncertainty. The TPG of the MAPF-TTU solution in Figure 2b is given in Figure 1c.

5.2 Extending Temporal Plan Graphs to STNUs

Given a TPG, we construct a STNU $\mathcal{X} = \langle \mathcal{V}, \mathcal{E}, \mathcal{C} \rangle$ capturing both controllable precedence relations and uncontrollable durations induced by traversal and task execution uncertainty.

Let η_i^k denote the k -th node of agent a_i in the TPG. For each node η_i^k , we decompose it into arrival, task-execution, and leaving events. We introduce two STNU timepoints: an uncontrollable timepoint $\nu_A(\eta_i^k) \in \mathcal{V}_u$ and a controllable timepoint $\nu_L(\eta_i^k) \in \mathcal{V}_c$, representing respectively the arrival and the leaving time of agent a_i at the corresponding location. However, if η_i^k corresponds to intermediate goal location $v \in M_{a_i}$, we additionally introduce two timepoints: a controllable timepoint $\nu_{Ts}(\eta_i^k) \in \mathcal{V}_c$ and an uncontrollable timepoint $\nu_{Te}(\eta_i^k) \in \mathcal{V}_u$, to represent the start and end time of the task, respectively.

An agent a_i can wait indefinitely before leaving the corresponding location or before starting a task of uncertain duration at the same location; we model this behavior by the following constraints:

$$\begin{cases} \nu_A(\eta_i^k) \xrightarrow{[0, +\infty]} \nu_{Ts}(\eta_i^k) \xrightarrow{[\psi^-(v), \psi^+(v)]} \nu_{Te}(\eta_i^k) \xrightarrow{[0, +\infty]} \nu_L(\eta_i^k) & \text{if } \eta_i^k \text{ corresponds to a } v \in M_{a_i} \\ \nu_A(\eta_i^k) \xrightarrow{[0, +\infty]} \nu_L(\eta_i^k) & \text{Otherwise} \end{cases}$$

We also introduce two reference timepoints ν_z and ν_F , respectively representing the beginning and the end of the global execution. For each agent a_i , let η_i^1 and η_i^n denote respectively the first and last node in its TPG path. We add the requirement constraints:

Algorithm 1 BuildTPGFromMAPFTU.

Input: An MAPF-TU solution $\Pi = \{\pi_1, \dots, \pi_n\}$
Output: A Temporal Plan Graph $G_T = (V_T, E_T)$

```

1  $V_T \leftarrow \emptyset, E_T \leftarrow \emptyset;$ 
2 foreach agent  $a_i$  do
3   foreach visit  $k$  in  $\pi_i$  do
4     create a TPG node  $\eta_i^k$ ;
5     annotate  $\eta_i^k$  with its vertex and potential presence interval;
6      $V_T \leftarrow V_T \cup \{\eta_i^k\};$ 
7     if  $k > 1$  then
8        $E_T \leftarrow E_T \cup \{(\eta_i^{k-1}, \eta_i^k)\};$  // intra-agent edge
9 foreach pair of agents  $a_i, a_j$  with  $i \neq j$  do
10  foreach nodes  $\eta_i^k, \eta_j^m$  visiting the same vertex do
11    if the interval of  $\eta_i^k$  precedes the interval of  $\eta_j^m$  then
12       $E_T \leftarrow E_T \cup \{(\eta_i^k, \eta_j^m)\};$  // inter-agent edge
13    else if the interval of  $\eta_j^m$  precedes the interval of  $\eta_i^k$  then
14       $E_T \leftarrow E_T \cup \{(\eta_j^m, \eta_i^k)\};$  // inter-agent edge
15 return  $G_T = (V_T, E_T);$ 

```

$\nu_z \xrightarrow{[0,0]} \nu_A(\eta_i^1)$ and $\nu_L(\eta_i^n) \xrightarrow{[0,+\infty]} \nu_F$. The first constraint synchronizes the start of every agent with the global reference timepoint, while the second ensures that ν_F occurs after all agents have completed their plans.

For every type-1 edge (η_i^k, η_i^{k+1}) in the TPG, let $e \in E$ be the corresponding traversal edge. We connect the leaving timepoint of η_i^k to the arrival timepoint of η_i^{k+1} by a dotted arrow as follows: $\nu_L(\eta_i^k) \xrightarrow{[w_-(e), w_+(e)]} \nu_A(\eta_i^{k+1})$

Finally, each type-2 edge (η_i^k, η_j^m) , representing a precedence relation induced by conflict avoidance, is translated into the requirement constraint: $\nu_L(\eta_i^k) \xrightarrow{[1,+\infty]} \nu_A(\eta_j^m)$. Please note that the lower bound defines the temporal duration safety between the moment agent a_i leaves the corresponding location and the moment agent a_j arrives at the same location.

The resulting STNU therefore captures both the temporal flexibility of the execution and the uncontrollable durations arising from navigation and task execution uncertainty. Figure 3 illustrates an example of transforming a TPG into a STNU.

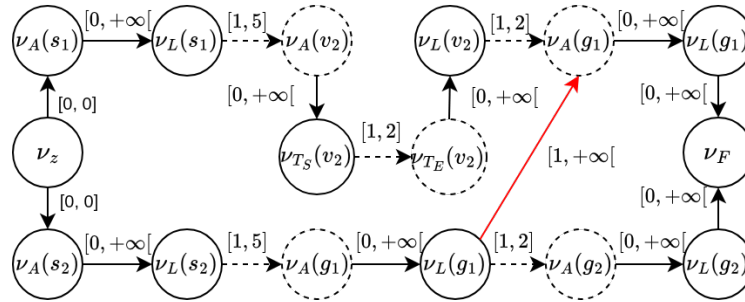


Figure 3 The transformation of the TPG from Figure 1c in a STNU. Dotted nodes depict uncontrollable timepoints of contingent links (dotted arrows), while the highlighted constraint shows the precedence relation on node g_1 .

5.3 Solving Strong Controllability

To find a solution for MAPF-TTU, we use the CBS_TU algorithm to compute a collision-free path between intermediate goals for each agent, and transform this solution into a TPG and a STNU. Then, we find a solution by looking for an assignment of the timepoints under the semantic of Strong Controllability. To do so, we use the polynomial size linear encoding of [23]. In the following, we describe the linear encoding for checking Strong Controllability.

For a requirement $\nu_x \xrightarrow{[l, u]} \nu_y$, there are three cases in which an uncontrollable timepoint is involved: (1) $\nu_y \in \mathcal{V}_u$, (2) $\nu_x \in \mathcal{V}_u$, and (3) $\nu_x, \nu_y \in \mathcal{V}_u$. A requirement in case (1) can be encoded by the linear inequality $l \leq (\nu_k + \omega_y) - \nu_x \leq u$, where ω_y represents the uncontrollable duration of the contingent link $\nu_k \xrightarrow{[l', u']} \nu_y$ with ν_y as endpoint. Other cases can be encoded similarly.

By Definition 5, a strong schedule must satisfy this system regardless of the durations $\omega_1, \dots, \omega_n$. It is shown in [34] that for a schedule to be a strong schedule, it suffices for it to satisfy the system when each duration takes one of its worst possible values, which are the lower and upper bounds of the corresponding contingent link. For example, to satisfy $l \leq (\nu_k + \omega_y) - \nu_x \leq u$ for every $\omega_y \in [l', u']$, it suffices to satisfy $l \leq (\nu_k + l') - \nu_x$ and $(\nu_k + u') - \nu_x \leq u$. This is the reason why it results in a system of linear inequalities of polynomial size.

To find the best solution, i.e., the one with the lowest makespan, we slightly modify the encoding by adding an optimization criteria on ν_F : $\text{Min}(\nu_F)$.

► **Proposition 13** (CBS_TU_STNU Completeness). *CBS_TU_STNU is complete but not optimal. Completeness follows from the completeness of CBS_TU and from the structure of the constructed STNU. Indeed, all precedence constraints introduced in the STNU are lower-bounded constraints of the form $[l, +\infty]$. Consequently, contingent delays induced by uncertain traversal and task durations can only postpone events without ever violating temporal upper bounds. Therefore, the resulting STNU is always strongly controllable.*

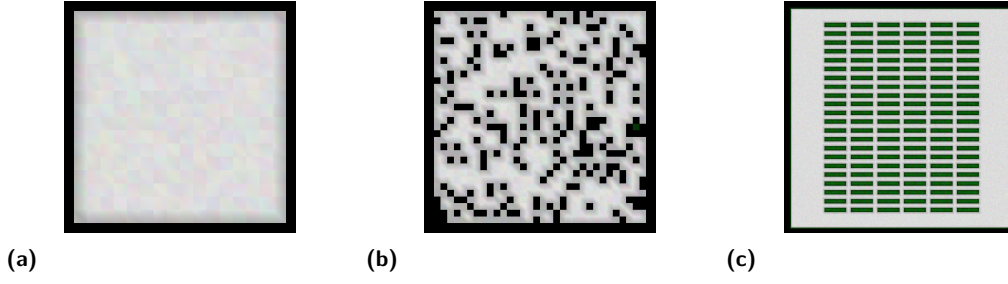
However, the approach is not optimal since temporal optimization is performed only after fixing the MAPF-TU plan and its associated precedence relations. As a result, the STNU can optimize execution times only within the flexibility of this fixed ordering, potentially excluding globally optimal MAPF-TTU solutions.

Although the resulting STNU is always strongly controllable due to the absence of restrictive upper-bound precedence constraints, the optimization process remains useful to minimize execution makespan while preserving robustness under uncertainty.

5.4 Perspective on Dynamic Controllability

Švancara et al. [33] assume that observation during execution is possible and propose a policy-based execution framework yielding solutions of improved quality for MAPF-TU. Similarly, Dynamic Controllability (DC) in STNUs allows the scheduling of controllable timepoints to depend on observations made during execution. Applying DC to MAPF-TTU could therefore enable agents to adapt their execution decisions online according to observed traversal and task durations, reducing the conservativeness of fixed robust schedules obtained under Strong Controllability (SC).

However, unlike SC, which only requires computing a fixed schedule expressible through linear constraints, DC involves observation-dependent decisions and typically requires disjunctive reasoning. As a result, efficient schedule optimization methods for DC generally rely on Mixed Integer or Non-Linear Programming approaches, making the optimization



■ **Figure 4** Figure the illustrates a the empty, b the random, and c the warehouse maps.

problem significantly more challenging [10, 35]. Consequently, finding an optimal dynamically controllable execution policy may become harder than computing the collision-free paths themselves. In this work, we therefore restrict ourselves to Strong Controllability, as MAPF applications do not always guarantee online observation capabilities.

6 Experimental Evaluation

In this section, we empirically evaluate the effectiveness of our proposed methods `CBS_TTU` and `CBS_TU_STNU`. We implemented ¹ our methods on top of the framework of [26]. Several extensions of `CBS` have been proposed to improve its efficiency while preserving completeness and optimality [7]:

- *Bypassing* nodes allow the search to skip unnecessary branching when a child node resolves a conflict without increasing the solution cost. Thus reducing the size of the constraint tree.
- *Prioritized conflicts resolution* classify conflict in three types. A conflict is *cardinal* if resolving it necessarily increases the cost of both child CT node, *semi-cardinal* if it increases the cost of only one child CT node, and *non-cardinal* otherwise. Prioritizing cardinal conflicts improves `CBS` despite overburdening the low-level search.

We particularly focus on prioritized conflict resolution, as planning paths across multiple sequential goals inherently overburdens the low-level search of `CBS`. Consequently for both methods, we evaluate a version with the two improvements (respectively called `CBS_TTU_PC` and `CBS_TU_STNU_PC`) and a version without prioritizing conflict resolution (called `CBS_TTU` and `CBS_TU_STNU`, respectively).

6.1 Environment and Benchmark

We conducted the experiments on a 20-core Intel(R) Xeon(R) CPU E5-2660 at 2.6 GHz, with 64GB RAM, running Ubuntu 18, with time-out for each run set to 300 seconds. We implemented the algorithm in Python 3.14. We compare our approach on three different types of maps from [29]: a classical empty map (i.e., without obstacles) of size 16×16 (see Figure 4a), a random map of size 32×32 (see Figure 4b), a warehouse map of size 84×170 (see Figure 4c).

We adapt the generator from [26] to generate MAPF-TTU instances. In our experiments we randomly assigned start and end positions and considered a sequence of two intermediate goals for each agent. For simplicity, the current implementation assumes that each agent

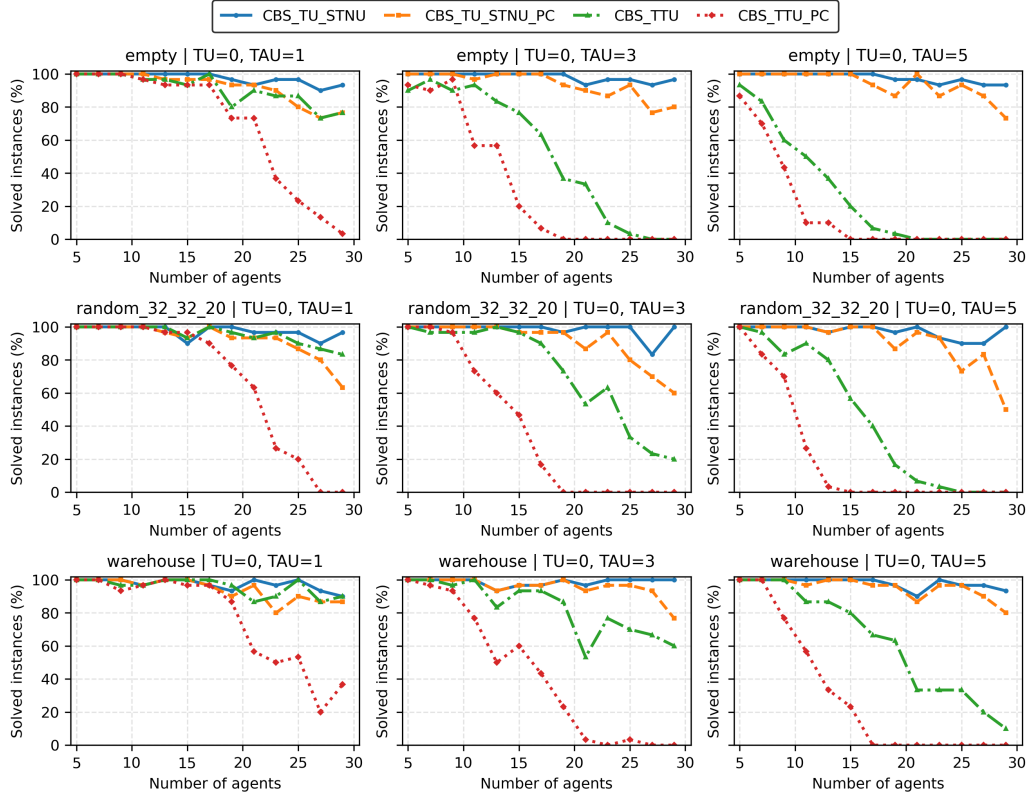
¹ The code is available at: https://github.com/ajdin14200/MAPF_TTU_TIME#

must visit its goals in the specified ordering, and does not wait before starting a task. This is a classical setting in Multi-Agent Pick-up and Delivery problems where agents must pick-up and deliver packages [25]. Please note that we extend CBS_TU so to find collision free paths for a set of goals, instead of a single goal as in the classical implementation of CBS_TU.

We distinguish the edge traversals and task executions types of uncertainty, but generate them similarly using the generator. For a given value of uncertainty U , we set $\omega^-(e)$ (resp. $\psi^-(e)$) to be a value chosen uniformly at random from the range $[1, U + 1]$, and set $\omega^+(e)$ (resp. $\psi^+(e)$) to be a value chosen uniformly at random from the range $[\omega^-(e), U + 1]$ (resp. $[\psi^-(e), U + 1]$). Thus, the parameter U allows control over the amount of uncertainty in the generated experiments, where $U = 0$ means no uncertainty. In our experimental evaluation we consider $U \in \{1, 3, 5\}$. For simplicity, we refer by TU the temporal uncertainty from traversing edges, and TAU the one from performing tasks.

For each combination of TU and TAU, we progressively increase the number of agents until the methods reach the timeout limit, generating 30 instances for each agent count.

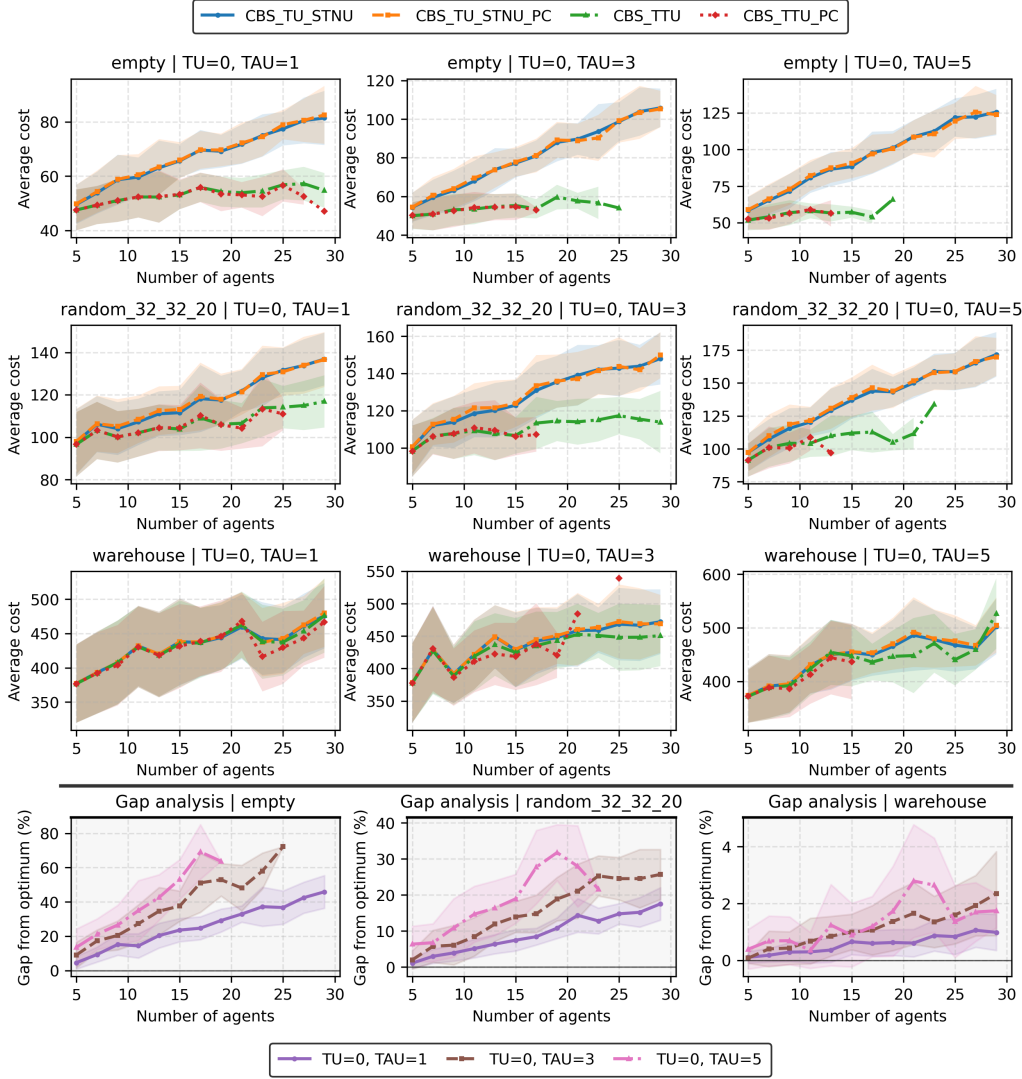
Regarding the Strong Controllability encoding, we replace the upper bound $+\infty$ with a sufficiently large constant value and solved the optimization problem using CPLEX [17] as backend solver.



■ **Figure 5** Comparison of the methods with $TU = 0$ on coverage.

6.2 Evaluation with Task Duration Uncertainty

We first evaluate the impact of task uncertainty independently from traversal uncertainty by fixing traversal uncertainty $TU = 0$, and varying task uncertainty in 1,3,5. This experiment aims to isolate the complexity introduced by uncertain task execution durations. In particular, we analyze how increasing task uncertainty affects the scalability of both methods.

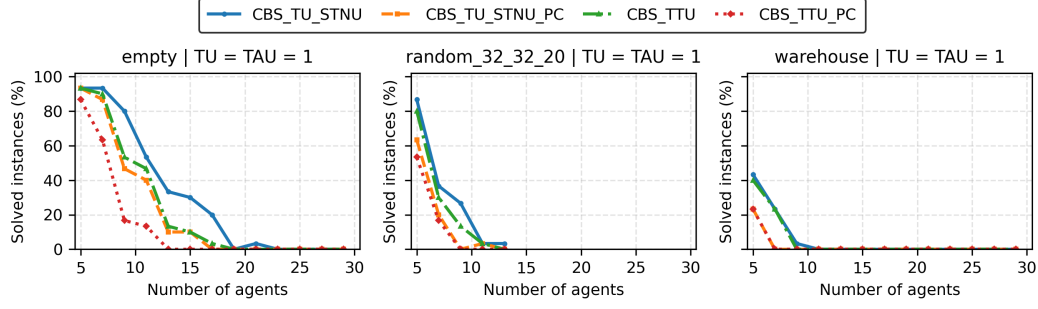


■ **Figure 6** Comparison of the methods with $TU = 0$ on cost (i.e., makespan).

Figure 5 reports the coverage (in percentage) of the different methods with respect to the number of agents. Overall, the STNU-based approaches solve more instances than CBS_TTU. Surprisingly, we observe that the prioritized conflict resolution reduces performance, suggesting that this improvement is not well suited to MAPF-TTU.

Figure 6 presents the average solution cost obtained by each method, with the corresponding standard deviation. As expected, the STNU-based approaches generate lower-quality solutions compared to the optimal methods. To quantify this difference, we compute the percentage deviation of suboptimal methods from the optimal solution cost.

The results show that CBS_TU_STNU consistently achieves a slightly smaller optimality gap than CBS_TU_STNU_PC. Consequently, the last row of Figure 6 reports the average percentage gap achieved by CBS_TU_STNU for each map, with one plot per map configuration. Interestingly, on the warehouse map, the STNU-based approach produces solutions that remain very close to the optimum.



■ **Figure 7** Comparison of the methods with $TU = TAU = 1$ on coverage.

6.3 Evaluation with Edge and Task Duration Uncertainty

We evaluate both approaches under simultaneous traversal and task uncertainty, varying both values identically in $\{1, 3, 5\}$. This evaluation studies the cumulative impact of both uncertainty sources on planning complexity and execution robustness. Combining them substantially increases potential temporal interactions between agents, as traversal uncertainty affects movement while task uncertainty impacts resource occupancy at fixed locations.

Figure 7 shows the coverage (in percentage) of the methods according to the number of agents only where $TU = TAU = 1$. We observed that adding temporal uncertainty on edges drastically impact the efficiency of the methods. For the case where $TU = TAU = 3$ and $TU = TAU = 5$, the methods show their limits under 10 agents. Overall, these results highlight that finding an optimal solution for MAPF-TTU is challenging as CBS does not scale at all.

6.4 Extension to PIBT

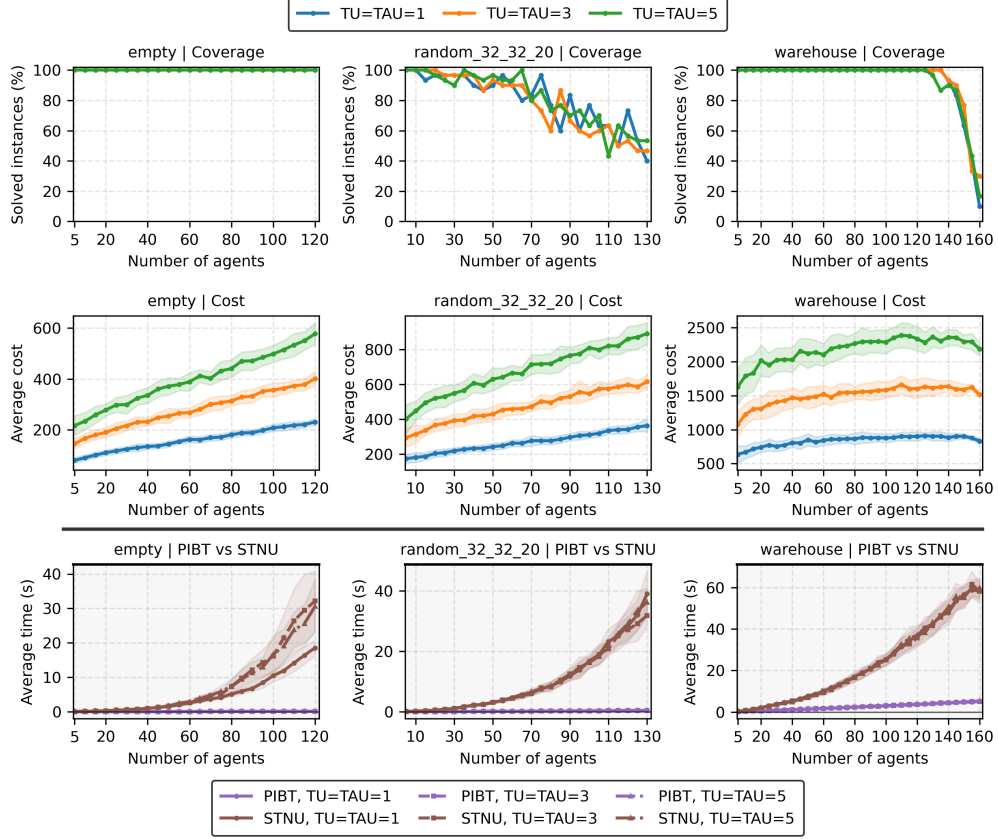
To further evaluate our STNU-based framework, we extend the experiments by replacing CBS_TU with PIBT to generate collision-free agent paths. We recall that PIBT is a suboptimal approach designed for large-scale MAPF instances involving thousands of agents [24]. Consequently, this approach, called PIBT_STNU, is overly conservative, as temporal uncertainty (both TU and TAU) is handled entirely at the STNU level rather than during the path finding process itself.

Figure 8 highlights the improvement in terms of coverage and scalability achieved by PIBT_STNU (first row), despite generating solutions of higher cost (second row). Furthermore, the third row presents a comparison of the average execution time of PIBT with that of computing an optimal strong schedule (STNU). This comparison indicates that the latter is more time-consuming.

7 Conclusion

In this paper, we introduced MAPF-TTU, a new extension of MAPF-TU in which agents must execute tasks of uncertain duration at intermediate locations. Unlike classical MAPF-TU, where uncertainty only affects traversal durations, MAPF-TTU additionally models uncertainty arising from task execution, leading to new temporal dependencies and resource conflicts between agents.

To solve this problem, we proposed two complementary approaches. First, we introduced CBS_TTU, an extension of CBS_TU that explicitly reasons about uncertain task execution during planning. Second, we proposed CBS_TU_STNU, a STNU-based approach that transforms an MAPF-TU solution into a temporally flexible execution model and optimizes its execution



■ **Figure 8** Results of PIBT_STNU on coverage (first row), cost (second row), and execution time (third row).

under uncertainty using Strong Controllability. We additionally analyzed the computational complexity of MAPF-TU and MAPF-TTU, and showed how temporal reasoning techniques can be leveraged to robustly execute MAPF plans under both traversal and task uncertainty.

Future work includes investigating Dynamic Controllability and online execution monitoring to reduce the conservativeness of Strong Controllability schedules through observation during execution. Another promising direction is the tighter integration of temporal reasoning within scalable suboptimal MAPF algorithms such as PIBT, in order to better balance solution quality, robustness, and scalability under traversal and task uncertainty. More generally, extending MAPF-TTU to online and continuously evolving coordination settings, where agents repeatedly receive new tasks during execution (i.e., lifelong settings), constitutes an important perspective for robust multi-agent coordination in dynamic environments.

References

- 1 Anton Andreychuk. Multi-agent path finding with kinematic constraints via conflict based search. In Sergei O. Kuznetsov, Aleksandr I. Panov, and Konstantin S. Yakovlev, editors, *Proceedings of the Russian Conference on Artificial Intelligence (RCAI)*, Lecture Notes in Computer Science, pages 29–45, Moscow, Russia, 2020. Springer. doi:10.1007/978-3-030-59535-7_3.
- 2 Anton Andreychuk, Konstantin Yakovlev, Pavel Surynek, Dor Atzmon, and Roni Stern. Multi-agent pathfinding with continuous time. *Artificial Intelligence*, 305:103662, 2022. doi:10.1016/J.ARTINT.2022.103662.

- 3 Dor Atzmon, Roni Stern, Ariel Felner, Glenn Wagner, Roman Barták, and Neng-Fa Zhou. Robust Multi-Agent Path Finding and Executing. *Journal of Artificial Intelligence Research*, 67:549–579, 2020. doi:10.1613/JAIR.1.11734.
- 4 Yifan Bai, Shruti Kotpalliwar, Christoforos Kanellakis, and George Nikolakopoulos. Multi-agent Path Planning Based on Conflict-Based Search (CBS) variations for Heterogeneous Robots. *Journal of Intelligent Robotic Systems*, 111(1):26, 2025. doi:10.1007/S10846-025-02229-0.
- 5 Alexander Berndt, Niels van Duijkeren, Luigi Palmieri, and Tamás Keviczky. A feedback scheme to reorder a multi-agent execution schedule by persistently optimizing a switchable action dependency graph. In Eva Onaindia Ronen I. Brafman and Shashank Shekha, editors, *Proceedings of the ICAPS Workshop on Distributed and Multi-Agent Planning (DMAP)*, 2020. arXiv:2010.05254.
- 6 Alexander Berndt, Niels van Duijkeren, Luigi Palmieri, Alexander Kleiner, and Tamás Keviczky. Receding horizon re-ordering of multi-agent execution schedules. *IEEE Transactions on Robotics*, 40:1356–1372, 2024. doi:10.1109/TR0.2023.3344051.
- 7 Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, Oded Betzalel, David Tolpin, and Solomon Eyal Shimony. ICBS: the improved conflict-based search algorithm for multi-agent pathfinding. In *Proceedings of the Annual Symposium on Combinatorial Search (SoCS)*, pages 223–225, Ein Gedi, the Dead Sea, Israel, 2015. AAAI Press. doi:10.1609/SoCS.V6I1.18343.
- 8 Yu Quan Chong, Jiaoyang Li, and Katia P. Sycara. Optimal task assignment and path planning using conflict-based search with precedence and temporal constraints. In Mehdi Dastani, Jaime Simão Sichman, Natasha Alechina, and Virginia Dignum, editors, *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 2210–2212, Auckland, New Zealand, 2024. International Foundation for Autonomous Agents and Multiagent Systems. doi:10.5555/3635637.3663110.
- 9 Adem Coskun, Jason M. O’Kane, and Marco Valtorta. Deadlock-free online plan repair in multi-robot coordination with disturbances. In *Proceedings of the International Florida Artificial Intelligence Research Society Conference*, North Miami Beach, Florida, USA, 2021. doi:10.32473/FLAIRS.V34I1.128371.
- 10 Jing Cui, Peng Yu, Cheng Fang, Patrik Haslum, and Brian Charles Williams. Optimising bounds in simple temporal networks with uncertainty under dynamic controllability constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 52–60, Jerusalem, Israel, 2015. AAAI Press. URL: <http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10614>.
- 11 Guilherme Daudt, Alleff Dymytry, Mariana Kolberg, and Renan Maffei. k-robust conflict-based search with continuous time for multi-robot coordination. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10260–10266, Abu Dhabi, UAE, 2024. IEEE. doi:10.1109/IR058592.2024.10801503.
- 12 Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. doi:10.1016/0004-3702(91)90006-6.
- 13 Ying Feng, Aditty Paul, Zhe Chen, and Jiaoyang Li. A real-time rescheduling algorithm for multi-robot plan execution. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 201–209, Banff, Alberta, Canada, 2024. AAAI Press. doi:10.1609/ICAPS.V34I1.31477.
- 14 A. Gautier, A. Stephens, B. Lacerda, N. Hawes, and M. Wooldridge. Negotiated path planning for non-cooperative multi-robot systems. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 1, pages 472–480, Auckland, New Zealand, 2022. International Foundation for Autonomous Agents and Multiagent Systems.
- 15 Wolfgang Hönig, Scott Kiesel, Andrew Tinka, Joseph W. Durham, and Nora Ayanian. Persistent and robust execution of MAPF schedules in warehouses. *IEEE Robotics and Automation Letters*, 4(2):1125–1131, 2019. doi:10.1109/LRA.2019.2894217.

- 16 Wolfgang Hönig, T. K. Satish Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. Multi-agent path finding with kinematic constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 477–485, London, UK, 2016. AAAI Press. URL: <http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13183>.
- 17 IBM. ILOG CPLEX Optimization Studio. <https://www.ibm.com/products/ilog-cplex-optimization-studio>, 2024.
- 18 Tim Tsz-Kit Lau and Biswa Sengupta. The Multi-Agent Pickup and Delivery Problem: MAPF, MARL and Its Warehouse Applications, 2022. doi:10.48550/arXiv.2203.07092.
- 19 Jiaoyang Li, Ariel Felner, Eli Boyarski, Hang Ma, and Sven Koenig. Improved heuristics for multi-agent path finding with conflict-based search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2019, pages 442–449, Macao, China, 2019. doi:10.24963/IJCAI.2019/63.
- 20 Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W. Durham, T. K. Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11272–11281, virtual, 2021. AAAI Press. doi:10.1609/AAAI.V35I13.17344.
- 21 Yihao Liu, Xueyan Tang, Wentong Cai, and Jingning Li. Multi-agent path execution with uncertainty. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pages 64–72, Kananaskis, Alberta, Canada, 2024. AAAI Press. doi:10.1609/SoCS.V17I1.31543.
- 22 Hang Ma, T. K. Satish Kumar, and Sven Koenig. Multi-agent path finding with delay probabilities. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3605–3612, San Francisco, California, USA, 2017. AAAI Press. doi:10.1609/AAAI.V31I1.11035.
- 23 Frédéric Maris, Aidin Sumic, Thierry Vidal, and Bruno Zanuttini. Study of the computational complexity of the repair problem on simple temporal networks with uncertainty. In *Journées d’Intelligence Artificielle Fondamentale et Journées Francophones sur la Planification, la Décision et l’Apprentissage pour la conduite de systèmes, JIAF-JFPDA*, pages 101–109, Dijon, France, 2025. URL: <https://hal.science/hal-05189745v2/document>.
- 24 Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*, 310:103752, 2022. doi:10.1016/J.ARTINT.2022.103752.
- 25 Oren Salzman and Roni Stern. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1711–1715, Auckland, New Zealand, 2020. International Foundation for Autonomous Agents and Multiagent Systems. doi:10.5555/3398761.3398959.
- 26 Tomer Shahr, Shashank Shekhar, Dor Atzmon, Abdallah Saffidine, Brendan Juba, and Roni Stern. Safe multi-agent pathfinding with time uncertainty. *Journal of Artificial Intelligence Research*, 70:923–954, 2021. doi:10.1613/JAIR.1.12397.
- 27 Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. doi:10.1016/J.ARTINT.2014.11.006.
- 28 Devon Sigurdson, Vadim Bulitko, William Yeoh, Carlos Hernández, and Sven Koenig. Multi-agent pathfinding with real-time heuristic search. In *IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, Maastricht, Netherlands, 2018. doi:10.1109/CIG.2018.8490436.
- 29 Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Bartak. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pages 151–158, Napa, California, USA, 2019. doi:10.1609/SOCS.V10I1.18510.

- 30 Yifan Su, Rishi Veerapaneni, and Jiaoyang Li. Bidirectional temporal plan graph: Enabling switchable passing orders for more efficient multi-agent path finding plan execution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 17559–17566, Vancouver, Canada, 2024. AAAI Press. doi:10.1609/AAAI.V38I16.29706.
- 31 Ajdin Sumic and Noëlie Ramuzat. Multi-agent Path Finding with Tasks under Time Uncertainty. Software, version 1.0., ANR-23-DMRO-0006, swhId: swh:1:dir:b90831099c338304e218af5159b6add58ed219b1 (visited on 2026-09-14). URL: https://github.com/ajdin14200/MAPF_TTU_TIME, doi:10.4230/artifacts.27850.
- 32 Pavel Surynek. An optimization variant of multi-robot path planning is intractable. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1261–1263, Atlanta, Georgia, USA, 2010. doi:10.1609/AAAI.V24I1.7767.
- 33 Jiří Švancara, David Zahrádka, Mrinalini Subramanian, Roman Barták, and Miroslav Kulich. Generating safe policies for multi-agent path finding with temporal uncertainty. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*, volume 3, pages 1238–1244, Porto, Portugal, 2025.
- 34 Thierry Vidal and Hélène Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence*, 11(1):23–45, 1999. doi:10.1080/095281399146607.
- 35 Benjamin W. Wah and Dong Xin. Optimization of bounds in temporal flexible plans with dynamic controllability. *International Journal of Artificial Intelligence Tools*, 16(1):17–44, 2007. doi:10.1142/S0218213007003187.
- 36 Thayne T Walker, Nathan R Sturtevant, and Ariel Felner. Extended increasing cost tree search for non-unit cost domains. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 18, pages 534–540, 2018. doi:10.24963/IJCAI.2018/74.
- 37 Peter R. Wurman, Raffaello D’Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008.
- 38 Yuanhang Zhang, Hesheng Wang, and Zhongqiang Ren. A short summary of multi-agent combinatorial path finding with heterogeneous task duration (extended abstract). In Ariel Felner and Jiaoyang Li, editors, *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pages 301–302, Kananaskis, Alberta, Canada, 2024. AAAI Press. doi:10.1609/SoCS.V17I1.31591.

A Appendix

In the experimental section we have shown plots of the most relevant results. In this appendix, the remaining results are shared.

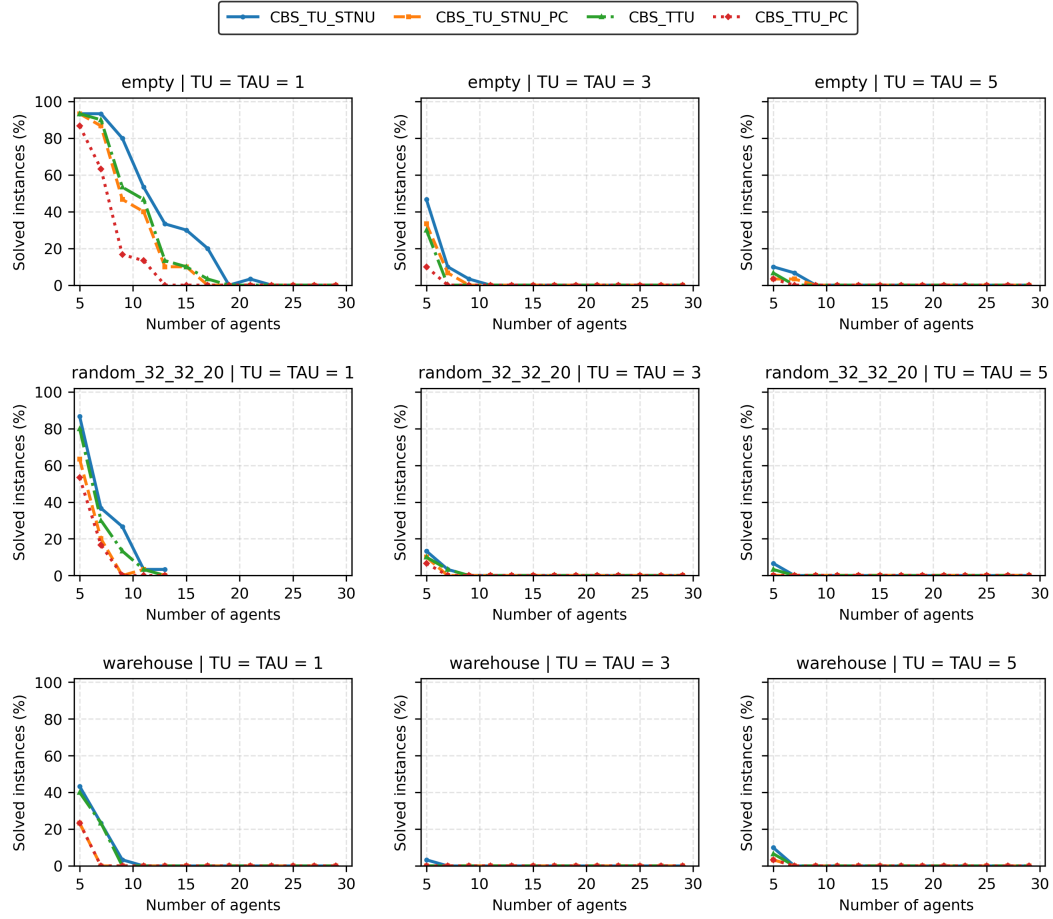
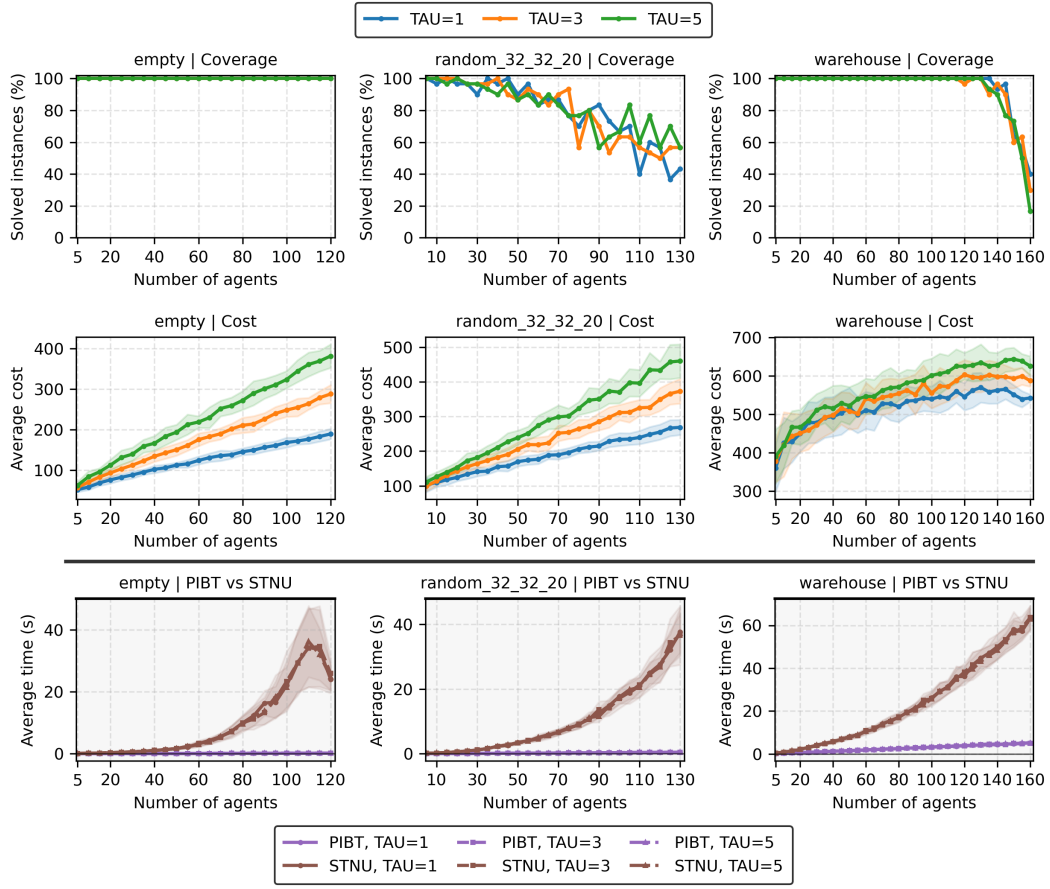


Figure 9 Complete comparison results of CBS_TTU and CBS_TU_STNU on coverage with TU and TAU in $\{1, 3, 5\}$.



■ **Figure 10** Results of PIBT with $TU = 0$.

