VÉRIFICATION EFFICACE DE SYSTÈMES À COMPTEURS À L'AIDE DE RELAXATIONS

EFFICIENT VERIFICATION OF COUNTER SYSTEMS THROUGH RELAXATIONS

par

Philip Offtermatt

Thèse présentée au Département d'informatique
en vue de l'obtention du grade de philosophiæ doctor (Ph.D.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 2 août 2023

Le 2 août 2023

Thèse déposée par Philip Offtermatt

Membres du jury

Prof. Michael Blondin, *Directeur*
Département d'informatique

Prof. Filip Mazowiecki, *Codirecteur*
Uniwersytet Warszawski, Wydział Matematyki, Informatyki i Mechaniki

Prof. Ranko Lazić, *Membre externe*
University of Warwick, Department of Computer Science

Prof. Manuel Lafond, *Membre interne*
Département d'informatique

Prof. Dave Touchette, *Président-rapporteur*
Département d'informatique

Sommaire

Les systèmes à compteurs sont des modèles utilisés afin de raisonner sur les systèmes de divers domaines tels l'analyse de programmes concurrents ou distribués, et la découverte et la vérification de systèmes d'affaires. Nous étudions des problèmes bien établis de différentes classes de systèmes à compteurs. Cette thèse se penche sur trois systèmes particuliers :

— les réseaux de Petri, qui sont un type de modèle pour les systèmes discrets à événements concurrents et séquentiels ;

— les «réseaux de processus», qui forment une sous-classe des réseaux de Petri adaptée à la modélisation et au raisonnement des processus d'affaires ;

— les automates continus à un compteur, un nouveau modèle qui combine une sémantique continue à celles des automates à un compteur.

Pour les réseaux de Petri, nous nous concentrons sur les propriétés d'accessibilité et de couverture. Nous utilisons des algorithmes de parcours de graphes, avec des relaxations de réseaux de Petri comme heuristiques, afin d'obtenir de nouveaux algorithmes de semi-décision pour l'accessibilité et la couverture, et nous évaluons positivement un prototype.

Pour les «réseaux de processus», nous nous concentrons sur le problème de validité, une notion de correction bien établie pour ces réseaux. Nous caractérisions précisément la complexité calculatoire jusqu'ici largement ouverte de trois variantes du problème de validité. En nous basant sur nos résultats, nous développons des techniques pour vérifier la validité en pratique, à l'aide de relaxations d'accessibilité dans les réseaux de Petri.

Enfin, nous introduisons le nouveau modèle d'automates continus à un compteur.

Sommaire

Ce modèle est une variante naturelle des automates à un compteur, qui permet de raisonner de manière hybride en combinant des éléments continus et discrets. Nous caractérisons la complexité exacte du problème d'accessibilité dans plusieurs variantes du modèle.

Mots-clés: vérification algorithmique, réseaux de Petri, systèmes à compteurs, théorie de la complexité

Abstract

Counter systems are popular models used to reason about systems in various fields such as the analysis of concurrent or distributed programs and the discovery and verification of business processes. We study well-established problems on various classes of counter systems. This thesis focusses on three particular systems, namely

— Petri nets, which are a type of model for discrete systems with concurrent and sequential events,

— workflow nets, which form a subclass of Petri nets that is suited for modelling and reasoning about business processes, and

— continuous one-counter automata, a novel model that combines continuous semantics with one-counter automata.

For Petri nets, we focus on reachability and coverability properties. We utilize directed search algorithms, using relaxations of Petri nets as heuristics, to obtain novel semi-decision algorithms for reachability and coverability, and positively evaluate a prototype implementation.

For workflow nets, we focus on the problem of soundness, a well-established correctness notion for such nets. We precisely characterize the previously widely-open complexity of three variants of soundness. Based on our insights, we develop techniques to verify soundness in practice, based on reachability relaxation of Petri nets.

Lastly, we introduce the novel model of continuous one-counter automata. This model is a natural variant of one-counter automata, which allows reasoning in a hybrid manner combining continuous and discrete elements. We characterize the exact complexity of the reachability problem in several variants of the model.

Keywords: Algorithmic verification, Petri nets, counter systems, complexity

Acknowledgements

First and foremost, I want to fondly thank my advisors Michael Blondin and Filip Mazowiecki. Under their guidance and supervision, I was able to follow my research interests while they helped me with their insights, so I felt neither constrained nor directionless. I learned tremendous amounts from both of them. More than just being excellent mentors, I also consider both of them friends.

I also want to thank my co-authors and colleagues. Thanks go to Christoph Haase, Guillermo Pérez, Tim Leys, Alex Sansfaçon-Buchanan, and Piotr Hofman for fruitful collaborations. In particular, Christoph also took on the role of a mentor in the early phases of my research, where I was thrown in the cold water, and he contributed greatly in enabling me to learn how to swim.

I want to thank my office mates over the years, notably Diego Oliveira, Felix Vigneault, David Purser, Arka Gosh, and Ismaël Jecker. In particular, Diego and Felix played a great part in making me feel welcome when I arrived in Canada at the start of 2020.

Next, I want to thank my circle of friends that kept in touch even when I moved over oceans, and made me feel at home anywhere I went. I would like to thank my friends Ray, Flo, Maxim, Alex, Fuffy and Jonah.

Most importantly, I want to thank my parents for their unconditional support and love. They always encouraged me to follow my passions and walk my path in life, and I am where I am today because of them.

Table of Contents

Sommaire iii

Abstract v

Acknowledgements vi

Table of Contents vii

List of Figures x

List of Listings xii

List of Algorithms xiii

Introduction 1

1 Preliminaries 6

1.1 Notation 6

1.2 Complexity Classes 6

1.2.1 Classes below polynomial time 7

1.2.2 P and NP 7

1.2.3 Classes above nondeterministic polynomial time 8

2 Petri nets and their Relaxations 9

2.1 Notable Decision Problems 12

2.2 Applications 15

2.2.1 Concurrent Program Analysis 15
2.2.2 Program Synthesis 17
2.3 Petri Net Relaxations 20
2.3.1 Continuous Petri Nets 20
2.3.2 Marking Equation over $\mathbb{N}$ 22
2.3.3 Marking Equation over $\mathbb{Q}_{\geq 0}$ 23

3 Efficient Procedures for Reachability in Petri nets using Relaxations 25
3.1 Directed Reachability 25
3.2 Applying Directed Search to Petri Nets 33
3.3 Experimental Evaluation 36
3.3.1 Benchmarks 37
3.3.2 Tool Overview 38
3.3.3 Results 39

4 Workflow Nets and Soundness 46

5 The Complexity of Soundness in Workflow Nets 55
5.1 k-Soundness 55
5.2 Generalised Soundness 60
5.3 Structural Soundness 64
5.4 Characterizing the set of sound numbers 64

6 Verifying Generalised and Structural Soundness 66
6.1 Generalised Soundness 67
6.1.1 $\mathbb{Z}$-Boundedness 67
6.1.2 Continuous Soundness 68
6.2 Structural Soundness 71
6.3 Free-Choice Workflow Nets 72
6.4 Experimental Evaluation 73
6.4.1 Free-Choice Nets 73
6.4.2 Synthetic Instances 75

7 Continuous One-Counter Automata **78**
7.1 Continuous One-Counter Automata (COCA) 80
7.2 Reachability in COCA is in NC^2 83
7.2.1 Checking $\mathrm{Post}_{p,q}(v)$ for emptiness 84
7.2.2 Characterizing $\mathrm{Post}_{p,q}(v)$ in relation to $\overline{\mathrm{Post}_{p,q}(v)}$ 84
7.2.3 Identifying $\inf \overline{\mathrm{Post}_{p,q}(a)}$ and $\sup \overline{\mathrm{Post}_{p,q}(a)}$ 85
7.2.4 Testing membership of a and the endpoints of $\overline{\mathrm{Post}_{p,q}(a)}$ . . . 86
7.2.5 Extending results to equality tests 87
7.3 Reachability in parametric COCA is NP-complete 88
7.3.1 Encoding paths 91
7.3.2 Encoding cycles 91
7.3.3 Combining paths and cycles 92
7.4 Continuous Semantics of Automata with more Counters 93

Conclusion **96**

Bibliography **113**

A Directed Reachability for Infinite-State Systems **114**

B The Complexity of Soundness in Workflow Nets **145**

C Verifying Generalised and Structural Soundness of Workflow Nets via Relaxations **163**

D Continuous One-Counter Automata **201**

List of Figures

2.1 A Petri net with places p_1, p_2 and p_3 and transitions t_1, t_2, t_3 and t_4. 11
2.2 An excerpt of the reachability graph for the Petri net of Figure 2.1. . 12
2.3 The Petri net from Figure 2.1, modified in order to answer coverability queries via reachability queries. 14
2.4 An example of how a Petri net can be used to model a program. . . . 16
2.5 A Petri net modelling the API of Listing 2.1. 18
2.6 An example of a Petri that that behaves different when viewed as a continuous Petri net. 21
2.7 An example of a Petri which allows different behaviour under the semantics where we allow negative token counts. 22

3.1 A maze which serves as an example of a graph. 28
3.2 The graph of Figure 3.1 annotated with heuristic values. 32
3.3 Cumulative number of (positive) coverability instances decided over time (x-axis is in log-scale). 42
3.4 Cumulative number of reachability instances decided over time for the SYPET suite (x-axis is in log scale). 43
3.5 Cumulative number of reachability instances decided over time for the RANDOM-WALK suite (both axes are in log scale). 44
3.6 Length of the returned witness per tool. 45

4.1 An example of a workflow net that represents an application process. 47
4.2 A slightly altered version of the net in Figure 4.1. 48
4.3 A workflow net N' which is k-sound if and only if the net N is 1-sound. 49

4.4 A workflow net modelling the distribution of a single task among two out of three resources. 50
4.5 An illustration of the free-choice property in Petri nets. 54

5.1 A sketch of the reduction from reachability in reversible Petri nets to 1-soundness in workflow nets. 59
5.2 A visualization of Steinitz Lemma in dimension $d = 2$. 62
5.3 A sketch of the reduction from reachability in conservative Petri nets to generalised soundness of workflow nets. 63
5.4 A sketch of the reduction from structural soundness to 1-soundness. . 65

6.1 A sketch of the reduction from tautology of 3-DNF formulas to continuous soundness in Petri nets. 69
6.2 The results of experiments on chained free-choice instances. 74
6.3 The synthetic families of workflow net instances used in our experiments. 75
6.4 Results of our experiments for generalised soundness. 76
6.5 Results of our experiments for quasi-soundness. 77

7.1 Gadgets of the reduction from 3-SAT, for guessing an assignment of variable x_i. 89
7.2 Gadgets of the reduction from 3-SAT, for checking whether a clause C_j is satisfied. 90
7.3 A gadget for transforming a two-counter machine with zero tests to a 3-CVASS with tests for equality with 0 and 1. 94
7.4 A gadget that moves from state p to state q and multiplies the first counter by 3 while preserving the value stored in the third counter. . 95

List of Listings

2.1 A small sample of methods from the library `java.awt.geom`. . . . 18
2.2 One possible implementation of the function `Area rotate`. 18

List of Algorithms

1 Directed search algorithm. 30

Introduction

When designing and developing systems, it is crucial to ensure that they are free from errors[1]. We are particularly interested in verifying systems that perform some computation. We view computation in a broad mathematical sense here, so while software may include the most prominent examples of computational systems, this can also include hardware or business processes. Naturally, the problem of ensuring correctness of such systems has attracted a vast amount of research, see e.g. [10, 48] for a textbook and a survey respectively. We focus here on algorithmic verification, that is, using algorithms that analyse a system and, for example, point out errors or guarantee their absence. This is in contrast to other techniques for the avoidance of errors such as peer review or testing, which have a more direct link to the system development process itself.

One important theorem that at first glance seems to make algorithmic verification a pointless task was given by Rice in his thesis in 1951.

Theorem 1: Rice's Theorem [92]

All non-trivial, semantic properties of programs are undecidable.

In the context of this theorem, non-trivial means that the property is satisfied by some programs, but not by others, and semantic means that the property is not strictly syntactic. For example, "Does the program contain an if-statement?" is a syntactic property, whereas "Does the program always evaluate the condition of one particular if-statement to 'true'?" is a semantic property.

While Rice's theorem seems restrictive, there are two key insights which still allow

1. See https://www5.in.tum.de/~huckle/bugse.html (accessed on February 6th, 2023). for an archive of various software errors and their impacts.

us to approach program verification in an algorithmic manner:

— While Rice's theorem prohibits us from developing an algorithm that decides a property on all programs, it does not make claims about the decidability of properties on *restricted subclasses* of programs. For example, it may be possible to decide whether any given loop-free program satisfies a certain property.

— Even if, for a certain class of programs and a given property, we cannot decide whether the property holds for a given program of the class, we may be able to give a slightly weaker procedure. Such a procedure may return either "no", in which case the property surely does not hold, or "maybe", in which case the property may or may not hold. Similarly, we could aim to design procedures answering either "yes" or "maybe". While such procedures are not decision algorithms, they can still be very useful in practice - for example, even if a procedure does not catch all theoretically possible defects, if it detects the defects that most often occur in practice, using it can still increase confidence in the verified system.

Two examples of techniques that are successfully applied in the development of software systems are *static analysis* and *symbolic execution*, see for example recent surveys [11, 50]. These techniques typically operate immediately on the program in question.

Static analysis makes conclusions about programs without executing the program itself, for example detecting information leakage by tracing the flow of data. Symbolic execution executes the program, but uses symbolic instead of concrete values for variables in order to make assertions about the behaviour of the program for many input values at once.

A more general category of techniques includes those that are *model-based*. They do not operate on the system under investigation, but rather on a model that describes the functional aspects of the system and, importantly, is mathematically precise. Then, we can make conclusions about the system by observing the model. For example, if we can prove that certain behaviour cannot occur in the model, we can conclude that this behaviour cannot occur in the system itself. Note that this insight critically relies on two assumptions:

(1) We are able to model the system in an unambiguous way which preserves the property under investigation.

(2) For the model we have chosen, it is possible to decide the property under investigation efficiently.

These two assumptions have contrary requirements. Modelling a system in a precise manner requires powerful models, but powerful models tend to bring high computational complexity for many relevant decision problems.

In my work, I aim to enrich existing knowledge about some particularly relevant models that are powerful enough to represent complex systems. In these models relevant decision problems tend to be computationally hard, as well as theoretically interesting. The goal is to approach these decision problems and work towards finding practically efficient procedures, and further to provide theoretical insights that are useful to the field at large. A particular focus lies on using problems that are provably easy as stepping stones to approach computationally hard problems.

For software development in the real world, it may seem like we are by nature dealing with a class of programs where Rice's theorem does not apply. After all, real machines are physically constrained to have only a finite amount of memory. For example, while there are infinitely many integers in the mathematical sense, real-world programming languages tend to disagree, where there might only be 2^{32} different integers. This is still an immense number, but it might be manageable in some contexts. So if there are only finitely many states, then we may simply be able to check all of them, given sufficient time and resources.

However, even disregarding the practicability of such an approach, a relatively recent development challenges this point of view. For the past 50 years, Moore's law has held, which predicts a doubling of components per integrated circuit every two years. As this proposed law continued to hold, the capabilities of new processors have increased exponentially over time. However, experts have recently voiced doubts regarding how much longer this will be the case [79]. As increases in processor speeds become slower, it is predicted that the development will shift from faster processors to *more* processors. This development can already be seen in large supercomputers, which in general do not possess one large, very fast processor, but should instead be seen as a large cluster of many interconnected processors. It is not unreasonable to

think that a similar trend will follow in the world of personal computers or embedded systems. A second trend that incentivizes distributed networks where many processors work in parallel is the Internet of things. This describes a paradigm shift in which increasingly, not just humans, but machines, are connected with each other [9, 78].

These developments mean that the analysis of systems is often no longer constrained to a fixed number of participating entities. Instead, systems must work correctly in the presence of an arbitrary number of participants. This shows the importance of a particular class of models, which we call *infinite-state systems*. As the name suggests, infinite-state systems are those models that do not have a finite state space. For example, we might consider modelling a communication protocol involving an unspecified number of participants as an infinite-state system. Even if each participant has a finite set of states, there are infinitely many global states, as there are infinitely many possibilities for the number of participants.

To give an idea of the different types of models, notable examples of finite-state models are finite state machines, (finite) control flow graphs, and Turing machines with a tape restricted to linear size (also called linear bounded automata). For infinite-state models, notable examples are counter machines (that is, automata enriched with one or more counters), pushdown automata, and Petri nets. We are particularly interested in models with counters. More precisely, this thesis places a large focus on Petri nets, and particularly the subclass of workflow nets. We will define these models in more detail in Chapter 2.

Let us briefly justify this particular focus. Petri nets are a well-established and popular model to represent and verify various systems. Among other applications areas, they have been used for the analysis of concurrent programs [40, 47, 69], as workflow nets in the area of business process management [6, 72, 102], and have even found use in computational biology [70] and to model train control systems for German railways [67]. The advantage of Petri nets is that they allow modelling complex concurrent and parallel behaviour, while providing a graphical representation that is comparatively easy to understand. A survey of several applications of Petri nets can be found in Section 2.2.

This thesis is structured as follows: In Chapter 1, we introduce mathematical notation and various notable complexity classes. In Chapter 2, we introduce the

model of Petri nets and notable decision problems for them, and survey the state-of-the-art regarding the complexity of these problems. We also introduce the notion of reachability relaxations for Petri nets and survey some notable relaxations. In Chapter 3, we present the results of [21], where we propose a novel semi-decision procedure for Petri nets based on relaxations.

In Chapter 4, we introduce the model of workflow nets, which are a subclass of Petri nets with some additional mild restrictions. We also introduce several variants of a correctness notion called soundness for workflow nets, and give an overview of the existing literature with respect to these variants of soundness. In Chapter 5, we present the results of [25], where we determine the exact computational complexities of the soundness notions introduced in Chapter 4.

In Chapter 4, we introduce the notion of free-choice nets, which are another subclass of Petri nets for which some decision problems become easier. We introduce free-choice workflow nets, which are relevant in practice due to the fact that they are more amenable to analysis than standard workflow nets, and state some known results for the complexity of various problems on free-choice nets, where it differs from the complexity of those problems on Petri nets. In Chapter 6, we present the results of [27], where we propose practical procedures for verifying soundness of workflow nets, in particular showing that several notions of soundness collapse on free-choice workflow nets.

In Chapter 7, we survey the existing literature on applying continuous semantics to various types of counter systems. In Chapter 7, we present the results of [23], where we introduce the novel model of continuous one-counter automata, which are an application of continuous semantics to a certain type of counter system with a single counter.

Many proofs and details in the thesis are omitted and deferred to an appendix, which contains preprints of the publications this thesis is based on. When a proof is moved to the appendix, there will be a link to it in the main text. This is aimed at making this thesis comparably light to read, while still providing the interested reader with the full details, albeit outside the main text. Mathematical results by this author are marked with † to better distinguish them from existing work that is presented for context.

Chapter 1

Preliminaries

In this chapter, we introduce mathematical notation used throughout this thesis.

1.1 Notation

We write $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$ to denote, respectively, the naturals including zero, integers and rationals. We restrict these sets by writing a subscript, for example, $\mathbb{N}_{\geq 1}$ denotes the naturals excluding zero. For a set of numbers N and a finite set S, we write N^S to mean a vector of size $|S|$ with elements from N. Equivalently, this is a mapping $S \to N$, which we index by elements of S. We denote as $\vec{0}$ the vector containing only zeros, with the domain being implied by the context. We extend operations such as subtraction, addition and comparison to vectors by applying them component-wise. For example, $\vec{A} \leq \vec{B}$ means that $a_i \leq b_i$ for all i. We differ from the component-wise interpretation for strict inequalities, and instead define $\vec{A} < \vec{B}$ to mean that $\vec{A} \leq \vec{B}$ and $\vec{A} \neq \vec{B}$, and similarly for $\vec{A} > \vec{B}$.

1.2 Complexity Classes

Next, let us briefly recall several important complexity classes that have connections to the work presented in this proposal. The classes are presented in ascending order of complexity. For further reference on computational complexity, see [89, 96].

1.2.1 Classes below polynomial time

The smallest complexity class that is of interest to us contains the problems which can be solved by nondeterministic Turing machines with space logarithmic in the size of the input. We call that class *NL*, short for nondeterministic logarithmic space. Despite its low computational power, this class comprises several classical problems. For instance 2SAT, that is, satisfiability of a formula given in conjunctive normal form with two literals per clause is NL-complete. Another NL-complete problem is reachability between two nodes of a directed graph.

Nick's Class, or *NC* for short, is the class of problems that can be solved efficiently in a parallel manner. More formally, problems in it can be solved by a parallel random-access Turing machine with polynomially many heads in polylogarithmic time, where one can view a PRAM as a Turing machine with multiple heads that can move to arbitrary parts of the tape in each step. Random access means here that the heads of the Turing machine can make "jumps" on the tape instead of being limited to making one step left or right. An alternative view is to see NC as the class of problems that can be decided by a log-space uniform Boolean circuit with polylogarithmic depth and polynomial width.

We may further divide NC by the order of the power in the polylogarithmic time. Formally, NC^i is the class problems that can be solved by a PRAM with polynomially many heads in time $O(log^i(n))$. Notably, $\mathrm{NC}^1 \subseteq \mathrm{NL} \subseteq \mathrm{NC}^2$.

1.2.2 P and NP

P describes the class of problems that can be decided by a deterministic Turing machine in a polynomial number of steps. Linear programming (LP), that is, deciding whether a system of linear equations has a solution over the reals, is a P-complete problem.

To contrast this, *NP* describes the class of problems that can be decided in polynomial time by a nondeterministic Turing machine. One of the most fundamental problems of computer science, the Boolean satisfiability (SAT) problem, is NP-complete. Another notable problem in NP is integer linear programming (ILP), which is defined as determining whether there exists an integer solution to a system of linear

equations.

It is known that NC $\subseteq$ P, but it is open whether NC $\subsetneq$ P or NC = P hold. Further, P $\subseteq$ NP is trivially true, but it is unknown whether P $\subsetneq$ NP or P = NP hold.

1.2.3 Classes above nondeterministic polynomial time

PSPACE is the complexity class of problems that can be solved by a deterministic Turing machine using only a polynomially large section of the tape. It is known that NP $\subseteq$ PSPACE, but again, whether the inclusion is proper is unknown. However, it is known that NL $\subsetneq$ PSPACE.

EXPSPACE is the class of problems solvable by a deterministic Turing machine using at most an exponentially large section of the tape. It is known that PSPACE $\subsetneq$ EXPSPACE.

Let us define the last two complexity classes we are interested in within this work. Let $F_1(n) = 2n$, and for $k > 1$ let

$$F_k(n) = \underbrace{F_{k-1} \circ \cdots \circ F_{k-1}(1)}_{n \text{ times}}.$$

We further define $F_\omega(n) = F_n(n)$.

The complexity class F_k is the class of problems solvable in at most $F_k(g(n))$ steps, where n is the size of the input and g is a function from level F_i with $i < k$, closed under composition and primitive recursion.

We call the complexity class F_3 TOWER. The name comes from the fact that $F_3(n)$ is a tower of the form $2^{2^{\cdots^2}}$ of height n. We call the complexity class F_ω Ackermann.

Chapter 2

Petri nets and their Relaxations

When choosing a model to represent a system of interest, there is a trade-off between expressivity and computational complexity. Suitability of a model depends not only on the system whose behaviour we wish to model, but also on the types of questions we wish to ask.

Petri nets are a particularly useful model. As we will see later, they allow modelling business processes and distributed systems. Formally, a Petri net is a tuple $N = (P, T, f)$ where

— P is a finite set of *places*,

— T is a finite set of *transitions* which is disjoint from P, and

— $f\colon (P \times T) \cup (T \times P) \to \mathbb{N}$ is the *flow function*.

When we represent Petri nets graphically, we draw places as circles (○), transitions as boxes (□), and the flow function as directed edges between places and transitions. If there is no edge from a place to a transition (or vice versa), we assume the flow function to have value 0. Otherwise, we will graphically denote the value of the flow function on the arrow, with the assumption that the value is 1 if the number is omitted.

We define the *size* of N as $|N| = |P| + |T|$, and the *norm* of N as $\|N\| = \|F\| + 1$, where $\|F\| = \max_{(a,b)\in(P\times T)\cup(T\times P)} f(a, b)$.

Places can hold any nonnegative number of *tokens*, drawn as small filled circles (•). We formally describe this using a *marking* $\mathcal{M} \in \mathbb{N}^P$, where $\mathcal{M}(p)$ denotes

the number of tokens in place p.

In the following, let t be a transition. We define some important concepts:

— We denote the *pre* of t as ${}^\bullet t$, sometimes also called *guard* of t. Formally, this is the vector ${}^\bullet t \in \mathbb{N}^P$ such that ${}^\bullet t(p) = f(p,t)$ for all $p \in P$.

— Similarly, we denote the *post* of t as $t^\bullet$, where $t^\bullet \in \mathbb{N}^p$ is the vector such that $t^\bullet(p) = f(t,p)$ for all $p \in P$.

— We define the *effect* of t as $\boldsymbol{\delta}_t = t^\bullet - {}^\bullet t$.

— A marking $\mathcal{M}$ *enables* t if $\mathcal{M} \geq {}^\bullet t$.

— If t is enabled in a marking $\mathcal{M}$, then we can *fire* it in $\mathcal{M}$, which results in a successor marking $\mathcal{M}'$ where $\mathcal{M}' = \mathcal{M} + \boldsymbol{\delta}_t$. We write $\mathcal{M} \xrightarrow{t} \mathcal{M}'$ to mean that $\mathcal{M}'$ is the successor marking resulting from firing t in $\mathcal{M}$. To put it more formally, $\mathcal{M} \xrightarrow{t} \mathcal{M}'$ if and only if $\mathcal{M} \geq {}^\bullet t$ and $\mathcal{M}' = \mathcal{M} + \boldsymbol{\delta}_t$.

Intuitively, the guard of a transition determines in which markings it can be fired, while the effect determines how it will affect the number of tokens in each place. One can imagine firing a transition to be in two steps: First, the transition removes tokens from places according to its incoming edges (which make up its guard); then it puts tokens into places according to its outgoing edges. Importantly, after removing tokens according to the guard, no place may have a negative number of tokens.

Figure 2.1 shows an example of a Petri net with a marking. The marking has two tokens in place p_1, one token in p_2 and no tokens in p_3. We write such a marking as $\{p_1 : 2, p_2 : 1, p_3 : 0\}$. For ease of notation, we usually omit specifying places with no tokens, that is, we equivalently write the marking as $\{p_1 : 2, p_2 : 1\}$.

Since ${}^\bullet t_3 = \{p_1 : 2\}$, it holds that t_3 is enabled in $\{p_1 : 2, p_2 : 1\}$, so we may fire it. This will mean taking two tokens out of p_1, and putting one token back. Therefore, we get that $\{p_1 : 2, p_2 : 1\} \xrightarrow{t_3} \{p_1 : 1, p_2 : 1\}$.

We can naturally treat $\xrightarrow{t}$ as a relation between markings. Then let us generalize this relation to not depend on a single transition, that is, we define $\rightarrow := \bigcup_{t \in T} \xrightarrow{t}$. Further, we extend this to sequences of transitions, which we call *runs* for brevity. Let $\pi = t_1 t_2 \ldots t_n$ be a sequence of transitions, then we write $\mathcal{M} \xrightarrow{\pi} \mathcal{M}'$ if there exist markings $\mathcal{M}_1 \ldots \mathcal{M}_{n-1}$ such that $\mathcal{M} \xrightarrow{t_1} \mathcal{M}_1 \xrightarrow{t_2} \ldots \xrightarrow{t_{n-1}} \mathcal{M}_{n-1} \xrightarrow{t_n} \mathcal{M}'$. For the special case of the empty sequence ε, we have that $\mathcal{M} \xrightarrow{\varepsilon} \mathcal{M}$ for all $\mathcal{M}$. Similarly,

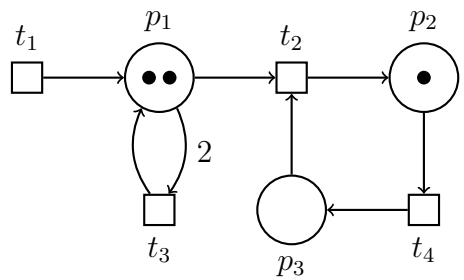


Figure 2.1: A Petri net with places p_1, p_2 and p_3 and transitions t_1, t_2, t_3 and t_4. The marking $\{p_1 : 2, p_2 : 1, p_3 : 0\} = \{p_1 : 2, p_2 : 1\}$ is drawn. Transitions t_1, t_3 and t_4 are enabled, while t_2 is not.

we can generalize enabledness of transitions to sequences of transitions, where π is enabled in $\mathcal{M}$ if there exists $\mathcal{M}'$ such that $\mathcal{M} \xrightarrow{\pi} \mathcal{M}'$. We write $\xrightarrow{*}$ to denote the reflexive, transitive closure of $\rightarrow$. We call $\rightarrow$ the *step relation*, and $\xrightarrow{*}$ the *reachability relation*. When $\mathcal{M}'$ is reachable from $\mathcal{M}$, that is, $\mathcal{M} \xrightarrow{*} \mathcal{M}'$, by definition of the reachability relation there must exist some (potentially empty) sequence of markings $\mathcal{M}_1 \ldots \mathcal{M}_{n-1}$ such that $\mathcal{M} \rightarrow \mathcal{M}_1 \rightarrow \cdots \rightarrow \mathcal{M}_{n-1} \rightarrow \mathcal{M}'$. We call the sequence $\mathcal{M}\mathcal{M}_1 \ldots \mathcal{M}_{n-1}\mathcal{M}'$ a *witness* for reachability between $\mathcal{M}$ and $\mathcal{M}'$. Note that when one wants to prove reachability between two markings, one way is to simply provide a witness for it.

Coming back to our example, we already observed that when starting in marking $\{p_1 : 2, p_2 : 1\}$, we can reach the marking $\{p_1 : 1, p_2 : 1\}$ by firing t_3. It is easy to see that the sequence $t_4 t_2$ is then enabled from that marking, and we get that $\{p_1 : 2, p_2 : 1\} \xrightarrow{t_3} \{p_1 : 1, p_2 : 1\} \xrightarrow{t_4 t_2} \{p_2 : 1\}$. In consequence, it holds that from $\{p_1 : 2, p_2 : 1\}$ we can reach $\{p_2 : 1\}$, and a possible witness is $\{p_1 : 2, p_2 : 1\}\{p_1 : 1, p_2 : 1\}\{p_1 : 1, p_3 : 1\}\{p_2 : 1\}$.

A Petri net $N = (P, T, F)$ gives rise to a so-called reachability graph G. Formally, we define $G = (V, E)$ where $V = \mathbb{N}^P$ and $E = \{(\mathcal{M}, \mathcal{M}') \in \mathbb{N}^P \times \mathbb{N}^P \mid \mathcal{M} \rightarrow \mathcal{M}'\}$. Intuitively, the reachability graph is a graph of all markings, and edges go from a marking to its successor markings with respect to all transitions of the net. Observe that the reachability graph can in general be infinite.

The reachability graph for the running Petri net example of Figure 2.1 is given in Figure 2.2. The witness $\{p_1 : 2, p_2 : 1\}\{p_1 : 1, p_2 : 1\}\{p_1 : 1, p_3 : 1\}\{p_2 : 1\}$ is colored

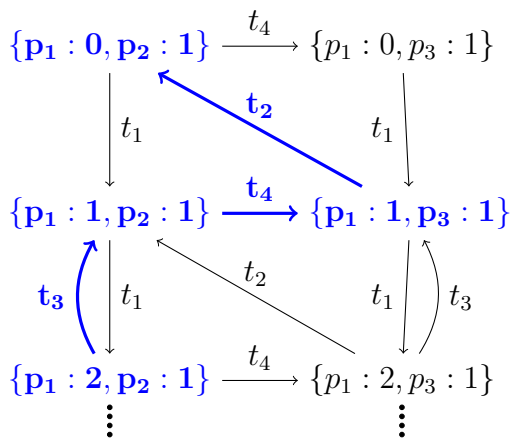


Figure 2.2: An excerpt of the reachability graph for the Petri net of Figure 2.1. One possible witness for reachability from $\{p_1 : 2, p_2 : 1\}$ to $\{p_1 : 0, p_2 : 1\}$ is drawn **bold** and colored in blue.

in the reachability graph, to demonstrate that a witness for reachability in the Petri net corresponds to a path of the reachability graph.

2.1 Notable Decision Problems

Now that we have introduced what Petri nets are, we aim to give a short overview over some interesting decision problems.

The first is the reachability problem. This problem is defined as follows:

Definition 1: Reachability Problem for Petri nets [87, Section IV.A]

Given a Petri net $N = (P, T, F)$ and two markings $\mathcal{S}, \mathcal{T}$, does $\mathcal{S} \xrightarrow{*} \mathcal{T}$ hold?

The reachability problem has a long history. It was first shown decidable by Mayr [80, 81] as early as 1981. The original algorithm was simplified later by Kosaraju [71] in 1982, and by Lambert [73] in 1992. Up to this point, all algorithms were essentially based on a particular tree decomposition, sometimes called Kosaraju-Lambert-Mayr-Sacerdote-Tenney (KLMST) decomposition. A more recent development occurred in 2012, when Leroux [74] provided another algorithm for the

reachability problem, the first such algorithm not based on the KLMST decomposition. Since decidability is well-known, the natural next step was to determine an upper bound of the complexity of the reachability problem. Most recently, Leroux and Schmitz [76] proved an Ackermannian upper bound.

In terms of lower-bound complexity, in 1976 Lipton [77] proved EXPSPACE-hardness, which at the time left a large gap, as not even decidability was yet known. While the next forty years brought improvements to the upper bound, this EXPSPACE lower-bound was only improved in 2019, when Czerwiński et al. proved TOWER-hardness. Most recently, the complexity gap was finally closed in 2021. Two teams, one consisting of Leroux, the other of Czerwiński and Orlikowski, proved independently that the problem has an Ackermannian lower bound [37, 75]. Thus, reachability is Ackermann-complete.

A second well-studied problem for Petri nets is the *coverability problem*. The formal definition is as follows:

Definition 2: Coverability Problem for Petri nets [87, Section IV.3]

Given a Petri net $N = (P, T, F)$ and two markings $\mathcal{S}, \mathcal{T}$, does there exist a marking $\mathcal{L}$ such that $\mathcal{S} \xrightarrow{*} \mathcal{L}$ and $\mathcal{T} \leq \mathcal{L}$?

Hence, the coverability problem asks whether we can reach a marking where each place has at least as many tokens as in the target marking.

In contrast to the Ackermann-hardness of the reachability problem, the coverability problem was shown EXPSPACE-complete in the 1970s [34, 91].

It is easy to give a polynomial-time reduction from coverability from $\mathcal{S}$ to $\mathcal{T}$ to reachability from $\mathcal{S}$ to $\mathcal{T}$ in a slightly modified version of the net: For each place p, add a single transition t which takes one token from p, and adds no tokens anywhere. Then, we may ask for reachability from $\mathcal{S}$ to $\mathcal{T}$ in this new net. If, in the original net, we are able to reach some marking that is greater or equal than $\mathcal{T}$, then we are also able to reach that marking in the modified net, and afterwards use the newly added transitions that remove tokens in order to reach $\mathcal{T}$. For the other direction, if we are able to reach $\mathcal{T}$ in the modified net, then in the original net we can, by replaying the run without the newly added transitions, reach a marking that is greater or equal to $\mathcal{T}$. Firing the newly added transitions only removes tokens, thus they can never be

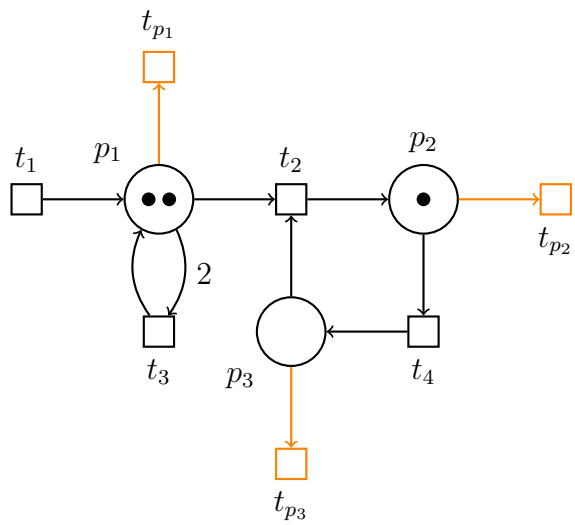


Figure 2.3: The Petri net from Figure 2.1, modified in order to answer coverability queries via reachability queries. The newly introduced transitions t_{p_1}, t_{p_2}, t_{p_3} are drawn in orange.

necessary to execute the run with just the original transitions. An example for the reduction is shown in Figure 2.3.

Let us briefly define some additional problems on Petri nets which have been considered in the literature:

— *Quasi-liveness*[1] [87, Section IV.C]: We say that a transition t is *quasi-live* from marking $\mathcal{S}$ if and only if there exists a marking $\mathcal{M}$ such that $\mathcal{S} \xrightarrow{*} \mathcal{M}$ and $\mathcal{M}$ enables t. We further say that a net N is *quasi-live* from a marking $\mathcal{S}$ if all transitions of N are quasi-live from $\mathcal{S}$. Quasi-liveness can be reduced to coverability.

— *Liveness* [87, Section IV.C]: A transition t is *live* from marking $\mathcal{S}$ if and only if t is quasi-live from all markings $\mathcal{M}$ such that $\mathcal{S} \xrightarrow{*} \mathcal{M}$. A net N is *live* if and only if all its transitions are live. Liveness is interreducible with the reachability problem, and thus is Ackermann-complete [62, Theorem 1].

— *Boundedness* [87, Section IV.B.]: A net N is *bounded* from a given marking $\mathcal{S}$ if there exists some $b \in \mathbb{N}$ such that for all $\mathcal{M}$, $\mathcal{S} \xrightarrow{*} \mathcal{M}$ implies $\mathcal{M}(p) \leq b$ for all $p \in P$. Boundedness is EXPSPACE-complete [77, 91].

The focus of this thesis in the context of Petri nets lies on coverability and reachability problems. To justify this focus, in the following we take an in-depth look at two applications of coverability and reachability.

2.2 Applications

We explore two applications of Petri nets, one widely mentioned in the literature, and one more recently introduced. This section is based on the presentation in [20, Appendix A], which is part of an extended version of [21], a paper authored by the thesis author and colleagues. A survey of additional classical applications of Petri nets can be found in [87].

2.2.1 Concurrent Program Analysis

The first application we discuss concerns concurrent program analysis.

Let us consider the short program given in Figure 2.4. If we assume only one thread at a time executes `fun`, it is clearly impossible to reach the error state. However,

1. In [87], this property is called $L1$-liveness.

```
   def fun()
1  s = 1
2  s = 0
3  if s == 1:
4    raise Err()
```

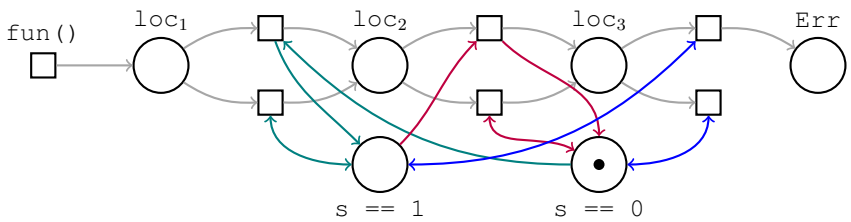


Figure 2.4: An example of how a Petri net can be used to model a program. Top: A simple program, meant to be executed by many threads simultaneously. Bottom: A Petri net representing the program. It holds that an error can occur in the program if and only if from marking {`s == 1` : 1} or {`s == 0` : 1} we can cover the marking {`Err` : 1}. Colors have no semantic meaning, and only serve to enhance the presentation.

we are interested in the case where `s` is a shared Boolean variable (with arbitrary initial value), and arbitrarily many threads may run the program at once, with no guarantee on the ordering of the execution between the threads. Then it is clear that even two threads running the program concurrently suffice to produce an error. One thread may execute Line 1 and 2 and before it executes Line 3, the other thread may execute Line 1, which means if the first thread now executes Line 3, the value of `s` will be 1 and the thread will reach Line 4.

Our goal is to algorithmically answer the question "Can an error be thrown in the program?". In 1992, German and Sistla [61] have shown that checking safety properties for systems in which an unbounded number of non-recursive Boolean programs are run in parallel reduces to checking coverability in a Petri net. Let us continue our example and take a look at what the Petri net for the program of Figure 2.4 looks like.

The Petri net at the bottom of Figure 2.4 models the program, and it is constructed after the procedure from [61]. The two places `s == 1` and `s == 0` are used to hold

the current value of the shared variable. The structure of the net ensures that the number of tokens among these two places never changes. So if we start with one token in either of the places, we will always remain with one token among the two places. In general, one can think of this as a type of one-hot encoding, that is, we view the possible values for the shared variable as a binary vector, where exactly one component is set to 1. Then it is easy to see how to handle not only binary variables, but variables with any finite number of possible values.

Further, there is one place for each line of the program. Intuitively, each token in such a place corresponds to one thread that is currently waiting to execute the respective line. As such, when a line reads or writes the value of a variable, there needs to be one transition per possible value of the variable that models the behaviour when a thread executes the line. For example, a token in loc_1 may move to loc_2 by taking the token for `s` from its current place and putting it into place `s == 1`. This corresponds to executing Line 1 of the program, which sets `s` to 1. Further, to model that an arbitrary number of threads may nondeterministically execute the program, we have the transition `fun()`, which simply adds a token in the place for Line 1. This corresponds to a new thread starting its execution of the program. Therefore, we do not need to a-priori restrict the execution to a particular number of threads.

By construction, it holds that an error may occur in the execution of the program if and only if from the marking $\{\texttt{s == 1} : 1\}$ or $\{\texttt{s == 0} : 1\}$ we can cover the marking $\{\texttt{Err} : 1\}$.

2.2.2 Program Synthesis

An application that was introduced relatively recently is the use of Petri nets for component-based synthesis [57]. In this task, we are given an API composed of functions and their type signatures, but no insight into the semantics of these functions. Our goal is to use the functions in the API to synthesize a non-branching function which satisfies a given target type signature and which should pass a set of given unit tests.

As a concrete example, let us consider the `java.awt.geom` library, of which we show an excerpt in Listing 2.1. Let us consider the target signature `Area`

```
java.awt.geom
new AffineTransformation()
Shape Shape.createTransformedShape(AffineTransformation)
String Point2D.ToString()
double Point2D.getX()
double Point2D.getY()
void AffineTransformation.setToRotation(double, double, double)
void AffineTransformation.invert()
Area Area.createTransformedArea(AffineTransformation)
```

Listing 2.1: A small sample of methods from the library `java.awt.geom`.

`rotate(Area area, Point2D p, double angle)`. Naturally, we want our synthesized function to rotate the `area` around point `p` by an angle of `angle`. An implementation of the function is shown in Listing 2.2. It makes use only of the methods in the `java.awt.geom` library, and it needs no branching or recursion.

```
1 Area rotate(Area area, Point2D point, double angle) {
2   AffineTransform at = new AffineTransform();
3   double x = point.getX();
4   double y = point.getY();
5   at.setToRotation(angle, x, y);
6   Area result = area.createTransformedArea(at);
7   return result;
8 }
```

Listing 2.2: One possible implementation of the function `Area rotate(Area area, Point2D point, double angle)` using methods from the `java.awt.geom` library. Adapted from [57].

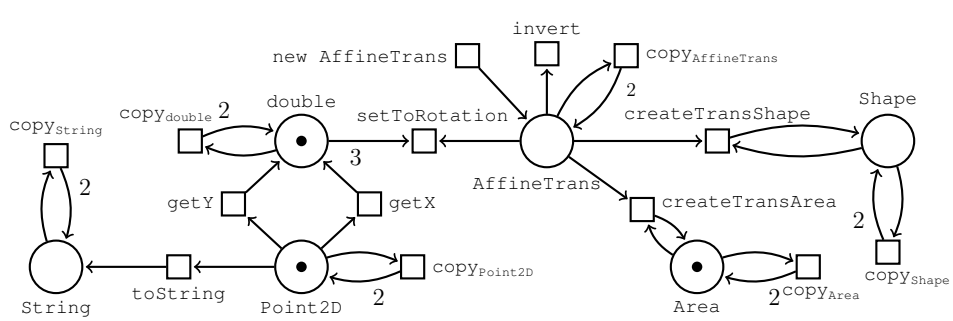


Figure 2.5: A Petri net modelling the API of Listing 2.1. Adapted from [57].

In [57], the authors propose modelling the API as a Petri net, in order to construct program sketches that type-check. The Petri net corresponding to the excerpt of the `java.awt.geom` library in Listing 2.1 is shown in Figure 2.5. Let us describe the construction in some detail. Each type that occurs in the API is represented by a place in the Petri net. Methods of the API are modelled as transitions of the net. First, note that a method of a class can be seen as a function taking one object of that class as an additional input. For example, the method `double Point2D.getX()` can be seen as a function `double getX(Point2D)`. So in words, the function takes one `Point2D` as an input, and returns one `double`. We model this in the Petri net as a transition consuming one token from `Point2D` and producing one token in `double`. Note that this does not yet accurately model the behaviour in the API, since the `Point2D` is in reality not consumed - as seen in Listing 2.2, we may call `getX` as well as `getY` on the same object. We model this by adding *copy* transitions, which take one token from any place and put two tokens back into that place.

The initial marking corresponds to the input parameters of the target signature, so as we want to synthesize a program with the signature `Area rotate(Area, Point2D, double)`, the initial marking $\mathcal{M}$ puts one token each into the places `Area`, `Point2D` and `double`. Following the same logic, the target marking $\mathcal{M}'$ is the marking with a single token in `Area`, since our function returns one object of that type. Now, witnesses for reachability from $\mathcal{M}$ to $\mathcal{M}'$ correspond to program sketches that typecheck.

Of course, finding typechecking program candidates is only the first step in the synthesis of programs. Further steps involve determining which variable should be used as which input, since tokens do not keep identities of any form. For example, when we have two variables of type `Point2D`, and the Petri net uses transition `GetY`, it is not clear which variable we should apply the function to. Further, program candidates need to be tested (and discarded if they fail to pass the provided unit tests). Solving this is out of scope for us, but more detail can be found in [57].

2.3 Petri Net Relaxations

As pointed out in Section 2.1, the reachability and the coverability problems both have prohibitive complexity, despite being decidable. This has sparked the development of so-called relaxations of Petri nets.

Intuitively, reachability and coverability are hard in Petri nets due to two reasons:

1. Markings must be non-negative, that is, we cannot fire transitions that are not enabled.
2. Transitions must be fired fully, that is, we cannot fire transitions by a fraction.

If we relax either or both of these requirements, we obtain an overapproximation of reachability. Every marking that is reachable under the normal Petri net semantics is reachable in the relaxation. This can be useful to quickly rule out reachability/coverability for some cases in practice: If a given target marking is unreachable/uncoverable under relaxed reachability, it surely is unreachable/uncoverable under the standard reachability semantics. However, the converse is not true, as the relaxed reachability semantics may allow markings to be reachable that are unreachable under the standard semantics.

Helpfully, the reachability problem in these relaxations is not only decidable, but has much lower complexity than standard reachability. In the following, let us describe the various relaxations in more detail.

2.3.1 Continuous Petri Nets

The first relaxation we present is to allow transitions to be fired by fractional amounts. We simply redefine the step relation $\rightarrow$, as well as the domain of markings. To do so, let us define the domain of markings as $\mathbb{Q}^P_{\geq 0}$, that is, mappings of places to rational numbers. Further, for a transition t and $\beta \in (0, 1]$, we define continuous reachability, denoted as $\rightarrow_{\mathbb{Q}_{\geq 0}}$, such that $\mathcal{M} \xrightarrow{\beta t}_{\mathbb{Q}_{\geq 0}} \mathcal{M}'$ holds if and only if $\mathcal{M} \geq \beta \cdot {}^{\bullet}t$ and $\mathcal{M}' = \mathcal{M} + \beta \boldsymbol{\delta}_t$, where $\mathcal{M}, \mathcal{M}' \in \mathbb{Q}^P$. Intuitively, when we fire a transition this allows us to scale it by a factor $\beta \in (0, 1]$, but we still have to ensure that the token count remains nonnegative after we remove the tokens according to the guard. Consequently, a *continuous run* is a sequence $\beta_1 t_1 \ldots \beta_n t_n$ where $\beta_i \in (0, 1]$ and t_i is

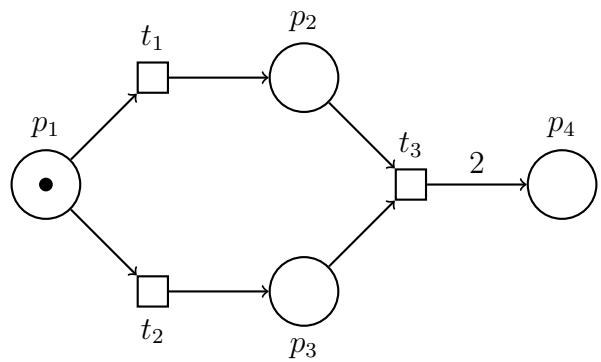


Figure 2.6: An example of a Petri that that behaves different when viewed as a continuous Petri net. When we view it as a normal Petri net, from initial marking $\{p_1 : 1\}$ we can fire only either t_1 or t_2, and the marking $\{p_4 : 1\}$ is unreachable. However, when we treat it as a continuous Petri net, then we can reach the marking $\{p_4 : 1\}$ by firing $0.5t_1$, $0.5t_2$ and $0.5t_3$ in this order.

a transition for all i.

A *continuous Petri net* is a Petri net where we use the step relation $\rightarrow_{\mathbb{Q}_{\geq 0}}$. An example of a Petri net which behaves differently when viewed as a continuous Petri net is shown in Figure 2.6. The initial marking is $\{p_1 : 1\}$. When we treat the net as a normal Petri net, then only either t_1 or t_2 may be fired, but not both. However, when we treat it as a continuous Petri net, the marking $\{p_4 : 1\}$ is reachable, since

$$\{p_1 : 1\} \xrightarrow{0.5t_1}_{\mathbb{Q}_{\geq 0}} \{p_1 : 0.5, p_2 : 0.5\} \xrightarrow{0.5t_2}_{\mathbb{Q}_{\geq 0}} \{p_2 : 0.5, p_3 : 0.5\} \xrightarrow{0.5t_3}_{\mathbb{Q}_{\geq 0}} \{p_4 : 1\}.$$

The model was first introduced with a slightly different definition by Alla and David in 1987 [38]. In 2013, Fraca and Haddad [58] provided complexities for many interesting problems in continuous Petri nets. Reachability and coverability are both P-complete. However, it is not clear whether the given polynomial-time algorithm is indeed the best way to approach the problem. In [17], continuous Petri net reachability is efficiently encoded as a logical formula. While the fragment of logic used cannot be solved in polynomial time, it appears that for practical purposes, it is more efficient, presumably due to the impressive performance of logic solvers like Z3 [86].

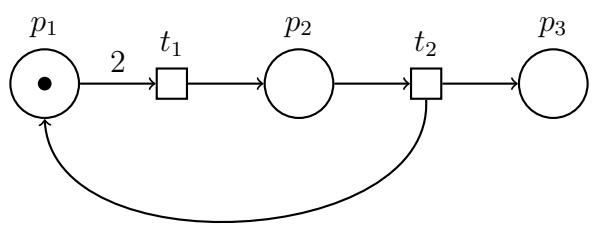


Figure 2.7: An example of a Petri which allows different behaviour under the semantics where we allow negative token counts. It holds that $\{p_1 : 1\} \xrightarrow{*}_{\mathbb{Z}} \{p_3 : 1\}$. However, $\{p_1 : 1\} \xrightarrow{*} \{p_3 : 1\}$ does not hold, that is, reachability does not hold under the standard Petri net step relation.

2.3.2 Marking Equation over $\mathbb{N}$

Here, we allow transitions to be fired even if their guard is not satisfied. Formally, we define the step relation $\rightarrow_{\mathbb{Z}}$ such that $\mathcal{M} \xrightarrow{t}_{\mathbb{Z}} \mathcal{M}'$ if and only if $\mathcal{M}' = \mathcal{M} + \boldsymbol{\delta}_t$. We also call this relation $\mathbb{Z}$-reachability.

It is easy to see that for this step relation, the order in which transitions are fired does not matter, only the overall effect of the transitions. This allows characterizing reachability under this step relation as the existence of an integral solution to a system of linear equations. For example, let us consider the Petri net from Figure 2.7, with the depicted initial marking $\{p_1 : 1\}$. Under the standard step relation, we are stuck, since transition t_1 can only be fired with two tokens in p_1, and t_2 required one token in p_2. However, under the step relation $\rightarrow_{\mathbb{Z}}$, we have that $\{p_1 : 1\} \xrightarrow{t_1}_{\mathbb{Z}} \{p_1 : -1, p_2 : 1\} \xrightarrow{t_2}_{\mathbb{Z}} \{p_3 : 1\}$. In words, we can fire t_1, which temporarily leaves us with -1 tokens in p_1, and afterwards firing t_2 puts one token into p_1, which means we end up with 0 tokens in that place.

Proposition 2: Reachability under $\rightarrow_{\mathbb{Z}}$.

Given a Petri net $N = (P, T, F)$ and two markings $\mathcal{S}, \mathcal{T}$, it holds that $\mathcal{S} \xrightarrow{*}_{\mathbb{Z}} \mathcal{T}$ if and only if the following equation with free variable $\vec{\sigma} \in \mathbb{N}^T$ has a solution:

$$\mathcal{T} = \mathcal{S} + \sum_{t \in T} \vec{\sigma}(t) \boldsymbol{\delta}_t \tag{2.1}$$

This follows trivially from the definition of $\rightarrow_{\mathbb{Z}}$.

We call Equation (2.1) the *marking equation over* $\mathbb{N}$, which is also the name we give the relaxation. This equation is a classical tool used to aid analysis of reachability in Petri nets [52, 87].

Intuitively, we have one free variable per transition, and its value determines how often we fire the corresponding transition. Since, as we pointed our earlier, the order in which we fire transitions does not matter, this is all the information we need to realize reachability over $\rightarrow_{\mathbb{Z}}$. One important fact to keep in mind is that we choose $\sigma_1 \dots \sigma_{|T|} \in \mathbb{N}$ to be natural numbers. This reflects the fact that we do not, as is the case for continuous Petri nets, allow transitions to be fired in a fractional manner.

Equation (2.1) can be solved via integer linear programming, which is an NP-complete problem. While NP-complete problems are sometimes seen as intractable (since the general worst-case runtime of all known algorithms is exponential), recall that we use this to approximate standard Petri net reachability, which is known to be Ackermann-complete, which means there is still an enormous gap in complexity between the two.

Further, let us define the notion of $\mathbb{Z}$-*boundedness*, which applies the concept of $\mathbb{Z}$-reachability in the context of boundedness, as defined in Chapter 2. We say that N is Z-bounded from a marking $\mathcal{M}$ if there exists $b \in \mathbb{N}$ such that $\mathcal{M} \xrightarrow{*}_{\mathbb{Z}} \mathcal{M}'$ implies that $\mathcal{M}'(p) \leq b$ for any $p \in P$.

In fact, $\mathbb{Z}$-boundedness is independent of the initial marking, as it is equivalent to any sum of transition effects having at least one positive as well as no negative components, regardless of the initial marking. However, for uniformity with boundedness, we still define it in terms of a specific initial marking.

2.3.3 Marking Equation over $\mathbb{Q}_{\geq 0}$

The last relaxation is obtained when we allow both negative token counts and firing transitions in a fractional manner. Essentially, we obtain a combination of the two previously mentioned relaxations. Formally, we again define a new step relation $\rightarrow_{\mathbb{Q}}$: $\mathcal{M} \xrightarrow{t}_{\mathbb{Q}} \mathcal{M}'$ if and only if there exists $\beta \in (0, 1]$ such that $\mathcal{M}' = \mathcal{M} + \beta \boldsymbol{\delta}_t$, and we define the domain of markings as $\mathbb{Q}^P$.

This relaxation behaves almost exactly like the marking equation over $\mathbb{N}$, with the one difference being that we allow the transition multiplicity variables $\beta_1 \dots \beta_{|T|}$ to take on values from $\mathbb{Q}_{\geq 0}$ instead of from $\mathbb{N}$. More formally, we restate Proposition 2 for this relaxation as follows:

Proposition 3: Reachability under $\rightarrow_{\mathbb{Q}}$.

Given a Petri net $N = (P, T, F)$ and two markings $\mathcal{S}, \mathcal{T}$, it holds that $\mathcal{S} \xrightarrow{*}_{\mathbb{Q}} \mathcal{T}$ if and only if the following equation with free variable $\vec{\beta} \in \mathbb{Q}^T_{\geq 0}$ has a solution:

$$\mathcal{T} = \mathcal{S} + \sum_{t \in T} \vec{\beta}(t) \boldsymbol{\delta}_t \tag{2.2}$$

Again, a proof is trivial by the definition of $\rightarrow_{\mathbb{Q}}$.

Now, solving the equation given by the theorem amounts to linear programming, which is P-complete.

Chapter 3

Efficient Procedures for Reachability in Petri nets using Relaxations

In this chapter, we present the results of [21], a paper that resulted from a collaboration with my supervisor and another researcher. The paper was published in the proceedings of TACAS'21. The main result is a novel approach to semi-decide reachability and coverability in Petri nets, which is based on established directed search algorithms like A^*. The motivation of moving to semi-decision procedures is that both reachability and coverability have prohibitive complexity in general, as described in Section 2.1. Below, we first give a brief introduction to the concept of directed reachability, then we introduce heuristics to guide directed search on Petri nets that are based on the relaxations of Section 2.3. Lastly, we present an experimental evaluation of a prototype implementation of the proposed procedure, which was accepted as an artifact at TACAS'21 [22].

3.1 Directed Reachability

In the literature, overapproximations of reachability have typically been used as ways to ensure unreachability of a marking:

Observation 4

If a marking is unreachable in an overapproximation of reachability, then it is not reachable.

This insight immediately translates to coverability as well. One possible use of this fact is that it is sometimes immediately possible to show that the target marking is unreachable from the initial marking. For example, Esparza et al. implemented this approach in the tool `Petrinizer` [52] to perform coverability analysis (but the approach can be extended in a straightforward manner to handle reachability). Without further modification, this approach can however only decide negative coverability instances, that is, those where the target marking cannot be covered.

One possible modification to mitigate this issue is to use the reachability overapproximation as an oracle for pruning in the so-called backward algorithm. This algorithm was introduced in 1996 by Abdulla et al. in their seminal paper on infinite-state systems [7]. It can only be used to decide coverability, not reachability. The backward algorithm relies on the fact that the set of target markings is upward-closed, which means it is sufficient to store a set of minimal representatives (called base) such that the union of the upward-closures of these representatives yields the set of all markings that can cover the target marking. The upward closure of a marking $\mathcal{M}$ is the set of all $\mathcal{M}'$ with $\mathcal{M} \leq \mathcal{M}'$. Intuitively, the algorithm has as an invariant that in the i-th iteration, its set of representatives are the smallest markings from which one can cover the target marking in at most i steps. By monotonicity of coverability, all markings larger than any of the representatives can also cover the target marking.

One problem of the backward algorithm is that the set of representatives can grow doubly exponentially large [31]. To mitigate this, reachability overapproximations can be used as a powerful tool to ensure that the set of representatives is as small as possible: If a representative cannot be covered by the initial marking in the overapproximation, then it cannot be covered in the standard semantics, so we may discard it. This approach was initially proposed in the thesis of Strazny [99]. However, there only what we call *static* overapproximations are considered. These overapproximations are typically fast to compute, but provide rather low accuracy. In 2016,

Blondin et al. independently discovered this approach of combining the backwards algorithm with pruning arising from overapproximations [17, 18], and it was later refined by Geffroy et al. [60]. However, the authors use continuous Petri nets instead of static overapproximations, which yields a more accurate heuristic with appreciable complexity.

The existing approaches have two downsides. Because they use the backward algorithm, they can only be used to decide coverability and are unable to decide reachability. Further, they are tailored towards deciding negative coverability instances, as pointed out in [17]. This leaves a large gap: overapproximations have been shown to be useful for deciding coverability, but similar procedures are lacking for reachability.

The work presented in the rest of this section aims to close this gap. Let us introduce more formally how we define weighted graphs for the purpose of this section.

Formally, a labelled, weighted, directed graph is a tuple $G = (V, A, E, \mu)$ where

— V is a possibly infinite set of *vertices*,

— A is a finite set of *actions* that are used to label edges,

— $E \subseteq V \times A \times V$ is the set of *edges*, and

— $\mu : E \to \mathbb{Q}_{>0}$ is the *weight function* mapping edges to their (positive) weights.

For technical reasons, we assume that the weight function has a minimum weight. A *path* π of the graph is a sequence of nodes $v_1 \cdots v_n$ and actions $a_1 \cdots a_{n-1}$ such that $(v_i, a_i, v_{i+1}) \in E$ for all i. Further, π is a *path from v to w* if $v = v_1$ and $w = v_n$. We also define *infinite paths*: π is an infinite path if it is an infinite sequence of nodes $v_1 \cdots$ and actions $a_1 \cdots$ where all finite subsequences are paths. We say that a (finite or infinite) path is *simple* if it contains no node twice. In some graphs, the actions of a path can be uniquely determined by its nodes or similarly, its nodes can be uniquely determined by the initial node and the sequence of actions. If that is the case, we may omit the redundant information, that is, we only list the nodes of the path or the initial node and the actions of the path. We say the label of π is $a_1 \dots a_{n-1}$. Further, its weight is the sum of the weights of its edges, that is, $\mu(\pi) = \sum_{1 \leq i < n} \mu(v_i, a_i, v_{i+1})$.

As an example, consider Figure 3.1. It shows a graph that models a maze with walls. These walls are drawn as black lines, and serve only to illustrate the structure of

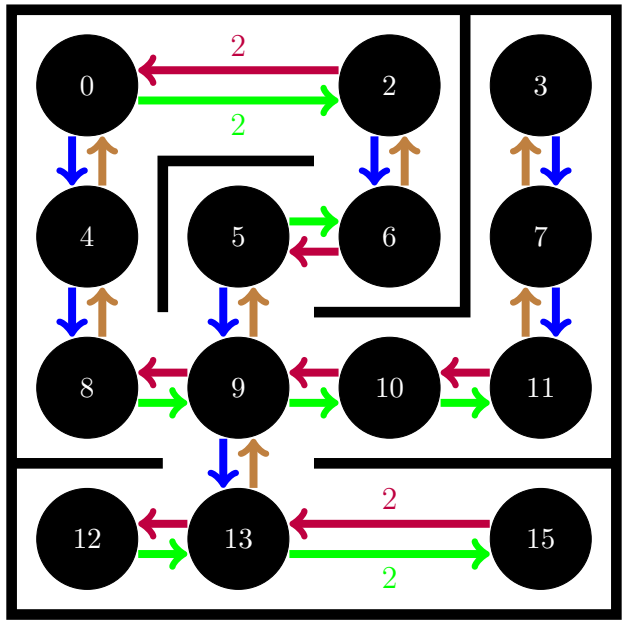


Figure 3.1: A maze which serves as an example of a graph. Thick black lines serve only to illustrate the structure of the maze, and do not belong to the graph. Vertices are drawn in solid black, with vertice names in white. The actions associated with edges are shown in colours: up in brown, right in green, down in blue, and left in purple. Edge weights are drawn next to the respective arrows, where we assume the weight is 1 if it is omitted in the graph.

the graph, that is, they do not belong to the graph. The set of nodes is $\{0, \ldots, 15\} \setminus \{1, 14\}$. There are four actions: up (colored brown), right (colored green), down (colored blue), and left (colored purple). Each edge is labelled with an action, and drawn in the corresponding color. We draw no number next to edges if their weight is 1, otherwise we draw the respective number. To give an example of the definitions of paths introduced above, the path π with nodes $0, 2, 6, 5, 9, 10, 11, 7, 3$ and actions right, down, left, down, right,right, up,up is a path from 0 to 3. Its weight is $2 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 9$.

We define the distance $dist_G(v, w)$ between two nodes $v, w \in V$ of the graph G as the weight of a shortest path from v to w, or $dist_G(v, w) = \infty$ if there is no such path. In our example graph, $dist_G(0, 3) = 7$, since there exists a path with less weight than the one we examined before, namely the path with nodes $0, 4, 8, 9, 10, 11, 7, 3$.

Computing the distance between two given nodes is an interesting problem on graphs, with applications in artificial intelligence, logistics, network routing, and many more [93]. Formally, we define the problem as follows:

Definition 3: Shortest path problem for graphs

Given a graph $G = (V, A, E, \mu)$, and nodes s and t, compute $dist_G(s, t)$.

We are particularly interested in tackling the problem on infinite graphs, because in general, Petri net reachability graphs can in general be infinite. Let us explain the general algorithmic scheme for directed search, which is shared by all directed search algorithms we investigate in this section.

In Algorithm 1, we show a sketch of a directed search algorithm. The algorithm takes as an input a selection function S, whose purpose we will explain later, as well as a graph $G = (V, E)$, and start and target nodes s and t. It uses two auxiliary data structures:

— A map $g : V \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ which stores for each node the known shortest distance from the start. Note that initially, we have only information about the starting node s (which has distance 0 to itself), while we assume all other nodes are unreachable, signified by assigning them distance ∞. As the algorithm proceeds, more nodes will be found to be reachable, and their g-values will be updated.

— A set F, also called *frontier*, which stores a set of candidate nodes that we wish to expand next. Initially, s is the only candidate.

In each iteration, one node is chosen from the frontier in order to be expanded next. How this is done is determined by the *selection strategy*. For instance, the selection strategy might always choose the node most recently added to the frontier, or the node with the shortest known distance from the start. Once a node v is chosen to be expanded, the algorithm checks whether it is the target node. If not, the algorithm then iterates over all the successors of v. A successor w is only added to the frontier if its distance is improved by reaching it via v. Note that the first time some node w is encountered, this is trivially true, since initially we define $g(w) = \infty$ for $w \neq s$. One may wonder why we need to add nodes we have seen before to the

Input: Selection function S, graph $G = (V, E, A, \mu)$, initial node s, target node t

```
// map of nodes to the best currently known distance from s
g := [s ↦ 0, v ↦ ∞ : v ≠ s]
// current frontier or workset of nodes
F := {s}
while F ≠ ∅ do
    // choose the next node according to the selection function
    v := arg min_{w∈F} S(g, w)
    if v = t then return g(t)
    for (v, a, w) ∈ E do
        // expand v: go over all successors w of v
        if g(v) + μ(v, a, w) < g(w) then
            // we found a better way to go to w (via v)
            // → update the distance for w and add it to the frontier
            g(w) := g(v) + μ(v, a, w)
            F    := F ∪ {w}
    // remove v from the frontier
    F := F \ {v}
// frontier became empty without finding target → unreachable
return ∞
```

Algorithm 1: Directed search algorithm.

frontier again. Intuitively, the first time we encounter a node, we may not do so via a shortest path from the start. Thus, we may need to add a node to the frontier again if we have found a better path to reach it since it was last expanded.

One important note is that whether the directed search algorithm is guaranteed to compute a shortest path from start to target depends on the utilized selection strategy. Let us survey some classical selection strategies from the literature.

Dijkstra's Selection Strategy. If we choose our selection strategy such that it picks the node with lowest g-value, that is, $S(g, w) = g(w)$, we obtain Dijkstra's algorithm, introduced by Edsger W. Dijkstra in a short note [45]. Dijkstra's algorithm is guaranteed to find shortest paths. However, it is easy to see that it will expand all nodes with distance strictly smaller than the target node.

For example, consider applying Dijkstra's algorithm to the graph of Figure 3.1, where the initial node is 0 and the target node is 3. In the first step, the frontier only contains 0. The node will be expanded and its two successors $2, 4$ are added to the frontier. Further, $g(2) = 2$ and $g(4) = 1$. In the next step, the algorithm will choose to expand node 4, since it has the lower g-value. Running Dijkstra's algorithm will output distance 7, stemming from the path $0, 4, 8, 9, 10, 11, 7, 3$. However, all nodes 14 of the graph will need to be explored.

Greedy Best-First Strategy. Instead of choosing the node with the lowest known distance from the start, we can choose the node with the lowest distance *to the target*. This is called the greedy best-first strategy, which results in the Greedy Best-First Search algorithm, or GBFS for short. Naturally, we do not know the distance to the target accurately, but for many problem domains, we are able to estimate it. We formally require a *heuristic function* $h : V \to \mathbb{Q}_{\geq 0}$ which maps vertices to their estimated distance to the target, and take $S(g, w) = h(w)$.

One possible heuristic for the maze example from Figure 3.1 would be to use the distance to the target node, *assuming there are no walls*. This can be computed simply by summing the vertical and horizontal distance of the two nodes. In the literature, this type of heuristic is sometimes called the *Manhattan distance*[1].

1. Named for the gridlike arrangement of streets in Manhattan.

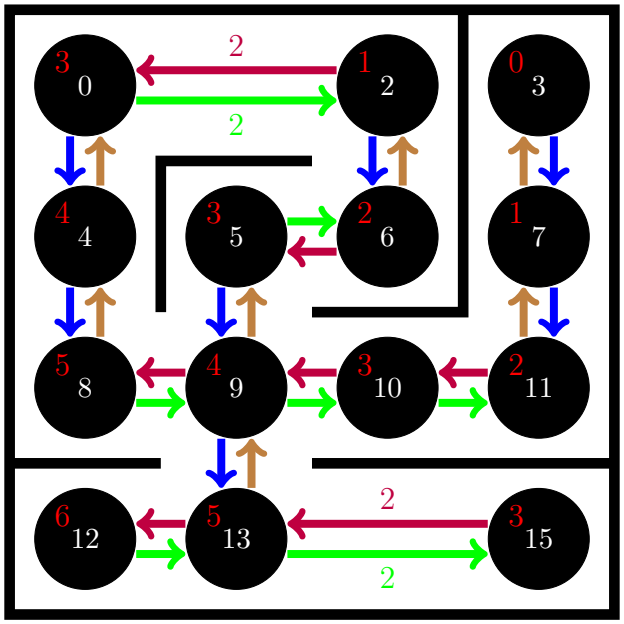


Figure 3.2: The graph of Figure 3.1 annotated with heuristic values. We draw the Manhattan distance of nodes to the target node 3 in red in the top-left corner of each node. For example, node 0 has estimated distance 3.

As an example, let us consider again the maze from Figure 3.1. In Figure 3.2, we show the same maze, but where the Manhattan distance to node 3 has been annotated in red in the top-left corner of each node. If we take $s = 0$ and $t = 3$, that is, we start at s and want to go to t, the frontier after the first step will contain nodes 1 and 4. Since we estimate that the distance from node 2 to node 3 is 1, compared to the estimation of 4 from node 4. So we will expand node 2. Running GBFS with the Manhattan distance as heuristic will output distance 9, with the underlying path being $0, 2, 6, 5, 9, 10, 11, 7, 3$. It is easy to see that this is not a shortest path from the start to the target. However, only 10 nodes will be explored (all but nodes $8, 12, 13$ and 15).

A* Selection Strategy. The A* selection strategy again needs a heuristic function h, defined in the same way as for GBFS. If we run the directed search algorithm with this strategy, we call the resulting algorithm the A* algorithm.

The algorithm combines the advantages of Dijkstra's algorithm with those of

GBFS. It tends to need to explore few nodes, and if the heuristic function we choose fulfills some simple criteria, it is also guaranteed to compute the optimal distance. This selection strategy is defined as follows: We select the node v from the frontier that minimizes $g(v) + h(v)$, that is, we minimize the sum of the known distance from the start to the node plus the estimated distance from the node to the target. For A^* to be guaranteed to compute the optimal distance, it is required that for all $v \in V$, it holds that $h(v) \leq dist_G(v, t)$. Intuitively, we simply require the heuristic to never overestimate the cost to the target. This is a rather natural condition, since many heuristics are derived by removing or simplifying away obstacles. Recall the maze example, where one natural heuristic is to take as an estimation the distance if we assume there are no walls in the maze.

If a second mild condition holds, then it can even be shown that A^* is guaranteed to expand the fewest nodes out of a class of A^*-like algorithms [39]. This condition is called *consistency*: For all $(v, a, w) \in E$, it holds that $h(v) \leq \mu(v, a, w) + h(w)$.

As an example, let us again take the frontier after one iteration of the algorithm when we start at node 0 and want to reach node 3. The two nodes in the frontier will be 2 and 4, where $g(2) + h(2) = 2 + 1 = 3$ and $g(4) + h(4) = 1 + 4 = 5$. So the A^* algorithm will choose to explore node 2.

The A^* algorithm has a long history of use, particularly in the field of artificial intelligence. It was originally introduced by Hart, Nilsson and Raphael in 1968 [63] in a seminal paper that has been cited in over 10,000 works. Since then, it has been applied to robotic movement planning, logistics and network routing, among other areas [93].

3.2 Applying Directed Search to Petri Nets

Next, let us show how directed search algorithms can be used to tackle the reachability problem (and by extension the coverability problem) for Petri nets. Recall that in this problem, we are given a Petri net N and two markings $\mathcal{S}, \mathcal{T}$, and the goal is to determine whether $\mathcal{S} \xrightarrow{*} \mathcal{T}$. Recall further that we defined the reachability graph of a Petri net as a graph where nodes are markings and where edges in the graph correspond to reachability in one step in the Petri net. Then it is clear that

the reachability problem for Petri nets can be reduced to the reachability problem in the underlying reachability graph.

This insight has frequently been applied in the literature in order to approach the reachability problem. One notable example is LoLA [95], where depth-first search is used to solve reachability queries. Notably, this does not require a heuristic, and is not guaranteed to terminate even if the target is reachable (recall that the reachability graph is infinite in general). It has also been suggested in [100] to use A^* to tackle the reachability problem. However, they propose heuristics with which A^* is not guaranteed to compute an optimal distance. Similarly related, [41] suggests utilizing the marking equation over $\mathbb{Q}$ to help find witnesses for reachability. However, it is suggested to use an ad-hoc algorithm that is not guaranteed to return shortest paths.

In hindsight, this is an obvious gap in the literature which we close by systematically instantiating directed reachability algorithms with heuristics derived from reachability overapproximations. In [21], we make the following key observation which extends on Observation 4:

Observation 5

If a marking $\mathcal{M}$ is reachable from an initial marking in an overapproximation, then the length of a shortest witnessing path in the overapproximation lower bounds the length of a shortest path reaching $\mathcal{M}$.

Essentially, this ensures that the distance in an overapproximation must be at most the distance in the Petri net itself. The intuition behind the proof is that overapproximations only allow more behaviour, never less, so a path witnessing reachability in the Petri net also witnesses it in the overapproximation. Note that this means we can immediately derive a heuristic from a reachability overapproximation.

Let us define a *distance under-approximation* of a Petri net N as a function $d : \mathbb{N}^P \times \mathbb{N}^P \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ such that for all markings $\mathcal{M}, \mathcal{M}', \mathcal{M}'' \in \mathbb{N}^P$ it holds that

- $d(\mathcal{M}, \mathcal{M}') \leq dist_N(\mathcal{M}, \mathcal{M}')$ and
- $d(\mathcal{M}, \mathcal{M}'') \leq d(\mathcal{M}, \mathcal{M}') + d(\mathcal{M}', \mathcal{M}'')$.

Then let us define the heuristic $h(\mathcal{M}) = d(\mathcal{M}, \mathcal{T})$, where $\mathcal{T}$ is the target marking. Proving that h satisfies consistency, and thus guarantees optimality when used with A^*, is straightforward.

Proposition 6: [21, Reformulation of Proposition 1] †

The heuristic h is consistent.

Proof: Appendix A, Page 8

One natural next question is whether for our particular class of heuristics arising from reachability overapproximations, we can make any guarantee about the performance of greedy best-first search. Unlike Dijkstra's algorithm and A^*, GBFS does not guarantee termination when run on infinite graphs where the target is reachable. In reality, our setting is such that termination is guaranteed. To show this, we introduce the novel notion of an *unbounded* heuristic.

We say that a heuristic h is *unbounded* (for a given graph G) if for any infinite simple path $v_1 v_2 \dots$ of G and for any bound $b \in \mathbb{Q}_{\geq 0}$, there is some index j such that $h(v_j) \geq b$.

Theorem 7: [21, Theorem 1] †

Greedy best-first search always terminates when run with an unbounded heuristic on a locally-finite graph in which the target is reachable.

Proof: Appendix A, Page 7

Intuitively, on any infinite path, an unbounded heuristic must reach arbitrarily high values. If the target is reachable, it must be reachable via a finite path, on which each node may have some arbitrarily large but fixed heuristic value. Thus, the heuristic on any infinite path will at some point become larger than the heuristic for the next node on the path to the target, thus we will eventually explore the path to the target. Note that this intuitive argument relies on a subtle yet important assumption: That the graph is locally finite, that is, each node has finitely many incoming and outgoing edges. This assumption holds in our setting, since Petri nets have finitely many transitions.

So GBFS is ensured to terminate for reachable targets, even on infinite graphs. Now, all that is left is to show that for any Petri net, the heuristics arising from reachability overapproximations are distance underapproximations, and that they are unbounded.

For $\mathbb{G} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{Q}_{\geq 0}\}$, and a Petri net $N = (P, T, F)$, let us define the function $dist_{\mathbb{G}} : \mathbb{N}^P \times \mathbb{N}^P \rightarrow_{\mathbb{G}} \mathbb{Q}_{\geq 0}$ such that

$$dist_{\mathbb{G}}(\mathcal{M}, \mathcal{M}') = min \left\{ \sum_{i=1}^{n} \beta_i \boldsymbol{\delta}_i \mid \beta_1 t_1 \ldots \beta_n t_n \text{ is a sequence s.t. } \mathcal{M} \xrightarrow{\beta_1 t_1 \ldots \beta_n t_n}_{\mathbb{G}} \mathcal{M}' \right\}.$$

For example, $dist_{\mathbb{Z}}$ is the distance in the reachability overapproximation over $\mathbb{Z}$. We claim that $dist_{\mathbb{G}}$ is a distance underapproximation.

Theorem 8: [21, Theorem 2] †

Let N be a Petri net and $\mathcal{T}$ the target marking. For $\mathbb{G} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{Q}_{\geq 0}\}$, it holds that $dist_{\mathbb{G}}$ is a distance underapproximation. Further, the heuristic $h_{\mathbb{G}}(\mathcal{M}) = dist_{\mathbb{G}}(\mathcal{M}, \mathcal{T})$ is unbounded.

Proof: Appendix A, Page 9, continued on Page 25

Proving the first part, that is, that $dist_{\mathbb{G}}$ is a distance underapproximation, is straightforward. The proof of the second part is more technically challenging and makes use of the fact that markings are elements of $\mathbb{N}^P$, thus well-quasi-ordered. By well-quasi-orderedness, any infinite sequence of markings must contain an infinite increasing subsequence, which, intuitively, corresponds to markings that must have greater and greater heuristic distance from the target, which yields unboundedness of the heuristic.

3.3 Experimental Evaluation

For the procedure presented in the previous section, a prototype implementation was developed in C#, with additional benchmarking infrastructure written in PYTHON. The tool is called FASTFORWARD, and the artifact used for the evaluation is publicly available [22]. The directed search algorithm, three different selection strategies (Dijkstra's Algorithm, GBFS and A*) and three reachability overapproximations (Continuous Petri nets, marking equation over $\mathbb{N}$, marking equation over $\mathbb{Q}$)

are implemented in it. We use the tool GUROBI[2] to compute both marking equations, since it is a state-of-the-art solver for (integer) linear programming that offers an academic license. To decide reachability in continuous Petri nets, we use the Z3 SMT solver [86], a well-known state-of-the-art solver used across related fields [15]. FASTFORWARD also implements some minor technical optimizations, whose details we omit here.

The rest of this section is structured as follows. First will be an overview of the benchmark instances we evaluate our approach on. Second, we discuss the other tools we compare with FASTFORWARD. Lastly, we show the results of our evaluation.

3.3.1 Benchmarks

For benchmarks, we used benchmarks from three different suites. The first suite, called COVERABILITY, is collected from five different classical benchmark suites from the literature [13, 47, 59, 68, 69] that have been used to evaluate many tools in the past [19, 53, 60]. Since the novel aspects of our approach are tailored towards deciding positive instances, we only use positive instances and omit the negative ones. As the complexity of the Petri net reachability problem is so high, focussing on positive instances like this is a reasonable restriction, as the approach may be run in parallel with one specializing on negative instances.

A second suite originates from queries used in program synthesis with Petri nets [57]. This suite contains instances with very high branching factors, that is, high average out-degree of the nodes in the expanded region of the reachability graph. Each instance has between 23 and 187 unguarded transitions. These can all naturally always be fired, so each corresponds to one outgoing edge from all markings. Further, markings tend to enable some unguarded transitions too. For example, the initial markings have out-degrees ranging from 30 to 300. To put these numbers into perspective, Chess and Go have similar branching factors. These are two games that have in the past been viewed as major challenges for artificial intelligence due to their large search spaces. For Chess, the average branching factor is roughly 35, while for

2. Gurobi can be found at http://www.gurobi.com (accessed on February 6th, 2023). For this work, we utilized Gurobi version 9.0.2, released in April of 2020, which was the most recent version at the time.

Go it is around 350 [93].

Lastly, the third suite consists of reachability instances derived from the coverability instances in the first suite. This was done in order to lay a greater focus on the reachability problem, as our approach is one of the few to handle not only coverability, but also reachability queries. To obtain somewhat realistic reachability instances which are as close as possible to queries that might be interesting in practice, we used a simple procedure. For each of the five suites forming the whole of our coverability instances, we picked the largest quarter of nets and performed random walks of lengths 20, 25, 30, 35, 40, 50, 60, 75, 90 and 100. Each time, the resulting marking was saved as the target, so we generated ten reachability instances for each of the coverability instances. Afterwards, we removed all instances where any tool reported a shortest path of length 20 or less. This was done because even with long random walks, the distance to the target tends to be small, and many of the instances turned out to be trivial. By removing those instances with very short witnesses, we can ensure that we end up with a benchmark suite of the most challenging instances.

The following table summarizes the three benchmark suites:

Suite	**Size**	**Number of places**				**Number of transitions**			
		min.	med.	mean	max.	min.	med.	mean	max.
COVERABILITY	61	16	87	226	2826	14	181	1519	27370
SYPET	30	65	251	320	1199	537	2307	2646	8340
RANDOM WALKS	127	52	306	531	2826	60	3137	5885	27370

3.3.2 Tool Overview

On reachability instances, there are two possible competitors. The first is LoLA [95]. This is a tool that has been in development for over 20 years, and continues to compete with great success in the Model Checking Competition, with the category most relevant for our comparison being the reachability category. In that category, LoLA has achieved second place in each of the past three years [3]. The second tool is KReach [46], a recent implementation of Kosaraju's classical algorithm

3. The results of the Model Checking Competition can be found at https://mcc.lip6.fr/2022/ (accessed on February 6th, 2023)

for Petri net reachability. As the algorithm is complete, it has impractical worst-case runtime, and in our evaluation, KReach could not solve a single instance. Therefore, we omit it from the results here.

For coverability instances, there is more tool support. We compare our implementation to four tools that implement algorithms tailored towards coverability:

— LoLA [95], which uses a dedicated coverability algorithm when run on coverability instances;

— BFC [69], a tool implementing an approach based on so-called widening;

— ICover [60], a tool using the backward algorithm for well-structured transition systems together with pruning based on continuous Petri net reachability;

— mist [59], where we choose to run its backward algorithm (which itself is based on [7]).

For FastForward, our preliminary evaluation suggested the heuristic with the best trade-off between computational effort and accuracy to be the marking equation over $\mathbb{Q}$, that is, the marking equation over the rationals. Therefore, we decided to include three instantiations of FastForward:

— FF(A^*, $\mathbb{Q}_{\geq 0}$) denotes FastForward with the A^* selection strategy and reachability over $\mathbb{Q}$ as heuristic;

— FF(GBFS, $\mathbb{Q}_{\geq 0}$) denotes FastForward with the greedy best-first selection strategy and reachability over $\mathbb{Q}$ as heuristic; and

— FF(Dijkstra) denotes FastForward with Dijkstra's selection strategy.

3.3.3 Results

Figure 3.3 shows the results on the coverability suite. The x-axis shows the time t, while the y-axis shows the number of instances decided after time t. Let us discuss the plot in some detail. Firstly, FastForward needs several hundred milliseconds to decide even trivial instances. This is regardless of the selection strategy, and holds true even for Dijkstra's. This could point towards a lack of optimization in parsing the nets, which can even take a majority of the time for very small examples. However, disregarding the first 0.5 seconds, we can clearly see that FastForward

performs well. At the end of the one-minute timeout, FF(A^*, $\mathbb{Q}_{\geq 0}$) has solved the most instances, totalling two more than BFC, the next closest competitor. No other tool comes close to either of these two. As one might expect, Dijkstra's selection strategy has troubles finding targets that are trivial for FF(A^*, $\mathbb{Q}_{\geq 0}$) and FF(GBFS, $\mathbb{Q}_{\geq 0}$) to find. However, Dijkstra's selection strategy decides more instances than FF(GBFS, $\mathbb{Q}_{\geq 0}$) after one minute. One possible reason for this behaviour is that GBFS might get mislead for many iterations if the heuristic is particularly far off on one net, but for many nets where the heuristic is good, it can find a path very fast. One very encouraging fact is that even though FASTFORWARD is not using an algorithm specifically tailored towards coverability instances, which the other tools do, it still manages to obtain very competitive performance on these instances.

Figure 3.4 shows the results on the SYPET suite. Here, LOLA is the only competitor (as KREACH was excluded), but it is immediately clear that its performance on this particular suite cannot match even that of Dijkstra's algorithm. We see again that FF(A^*, $\mathbb{Q}_{\geq 0}$) decides the most instances. The difference to FF(GBFS, $\mathbb{Q}_{\geq 0}$) is small, however.

Figure 3.5 shows the behaviour on the RANDOM WALKS benchmark suite. Note that for this plot, both axes are in log-scale. Here, the approach with the best performance is FF(GBFS, $\mathbb{Q}_{\geq 0}$), with no other approach deciding even one tenth as many instances. LOLA again decides the fewest instances. One explanation for the particularly good performance of FF(GBFS, $\mathbb{Q}_{\geq 0}$) might be that this benchmark suite was specifically engineered to have long shortest paths (recall that we discarded instances where a shortest path is known to have length 20 or less). Such long paths are less typical for the other benchmark suites.

Lastly, let us conclude by discussing the relationship between the tool used and the optimality of the witness that is returned. Figure 3.6 shows the length of the returned witness, compared to the actual shortest witness. Naturally, we omit tools that guarantee shortest witnesses (among the tools we benchmarked, those are FF(A^*, $\mathbb{Q}_{\geq 0}$), FF(Dijkstra), and MIST). ICOVER is left out because it does not return a witness nor its length, instead returning only true or false, depending on whether the target is coverable. So the only three tools that return witnesses, but are not guaranteed to return shortest ones by design are FF(GBFS, $\mathbb{Q}_{\geq 0}$), LOLA and BFC.

One minor caveat is that BFC uses a different format which introduces additional transitions and which might lead to it returning witnesses that are optimal in its translated setting, but suboptimal in the actual Petri net.

In the figure, a value at $y = 0$ means that the returned witness was optimal, while high values mean that it was much longer. The figure shows that while all tools, FF(GBFS, $\mathbb{Q}_{\geq 0}$) in particular, occasionally return shortest paths, this is far from guaranteed. The largest difference between the length of the returned witness and a shortest one is for a witness returned by LoLA, where LoLA returned a witness that is 53 transitions longer than an optimal one. For a different perspective, LoLA was able to return a shortest witness on 28 out of 43 instances, while FF(GBFS, $\mathbb{Q}_{\geq 0}$) returned a shortest witness on 60 out of 83 instances. Note that the number of total instances differs between the two as we only count instances where a shortest witness is known, that is, where an approach terminated that guarantees a shortest witness.

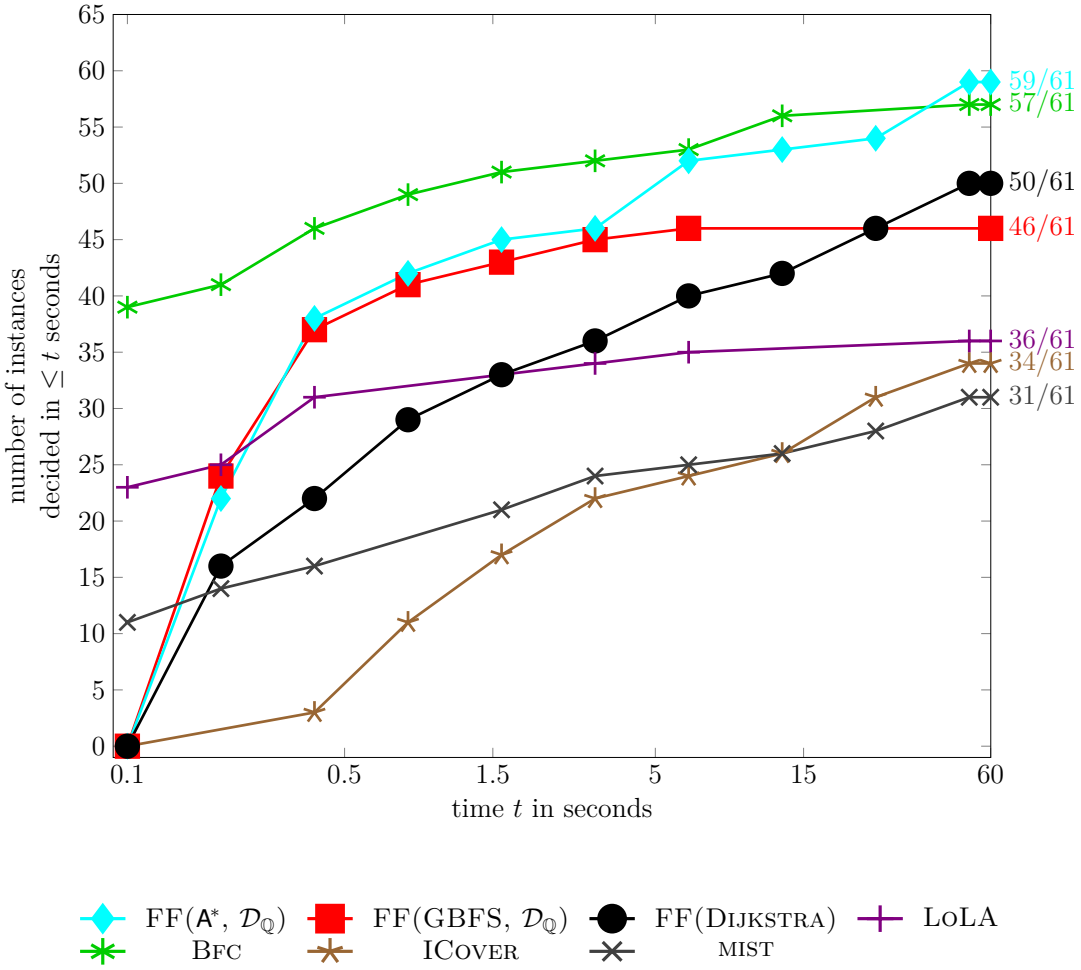


Figure 3.3: Cumulative number of (positive) coverability instances decided over time (x-axis is in log-scale).

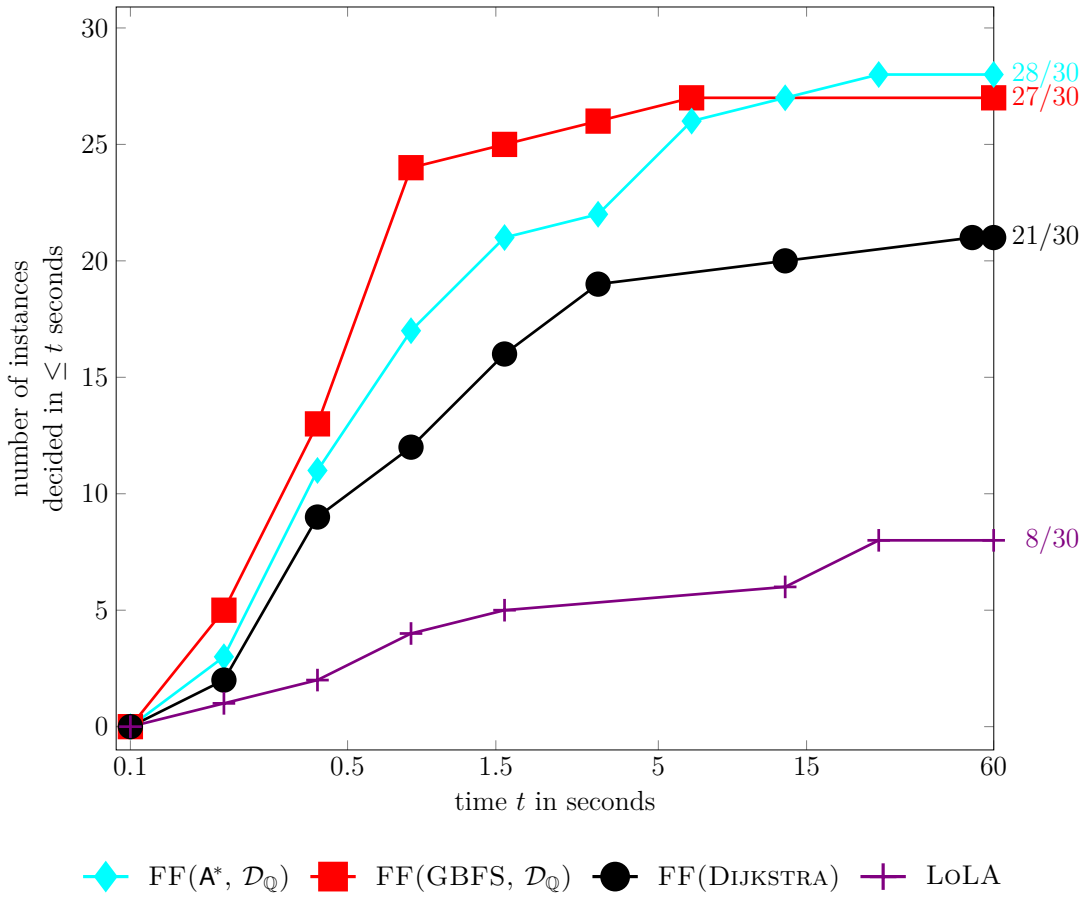


Figure 3.4: Cumulative number of reachability instances decided over time for the SYPET suite (x-axis is in log scale).

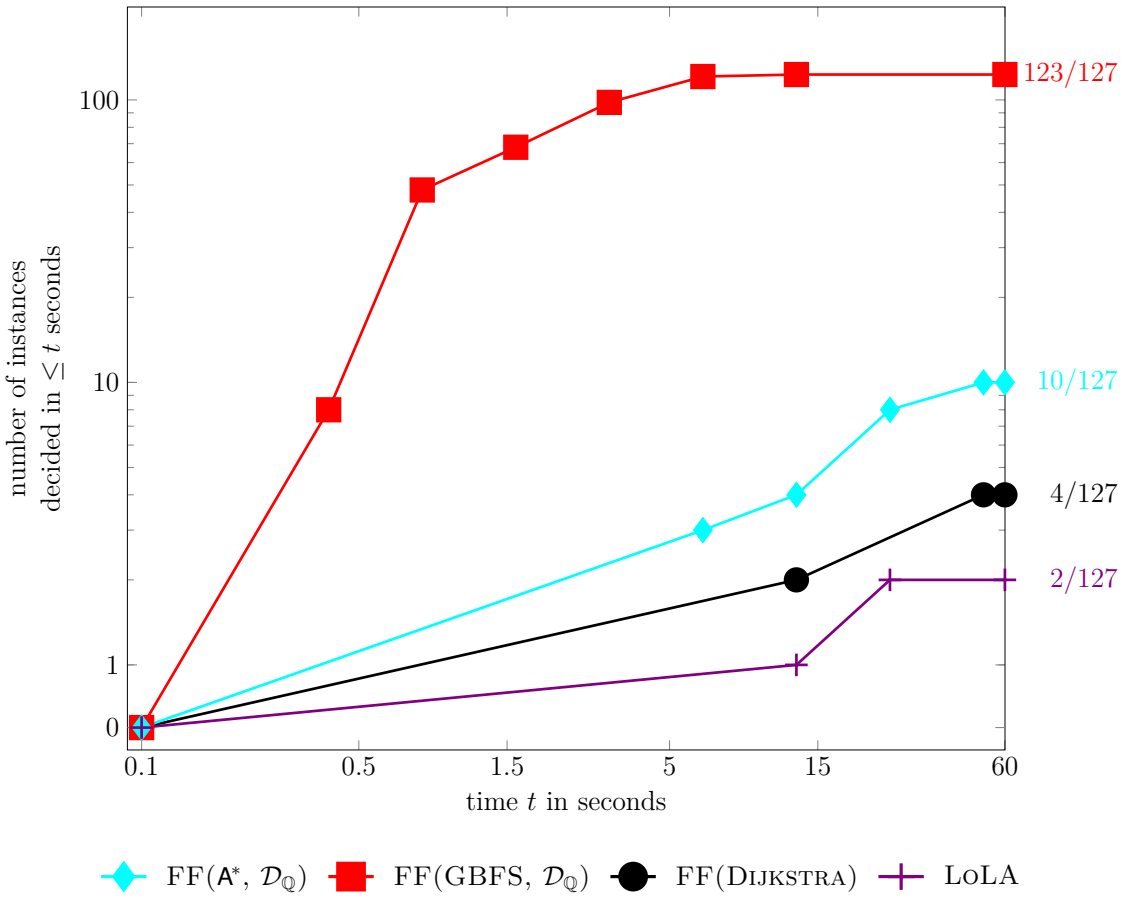


Figure 3.5: Cumulative number of reachability instances decided over time for the RANDOM-WALK suite (both axes are in log scale).

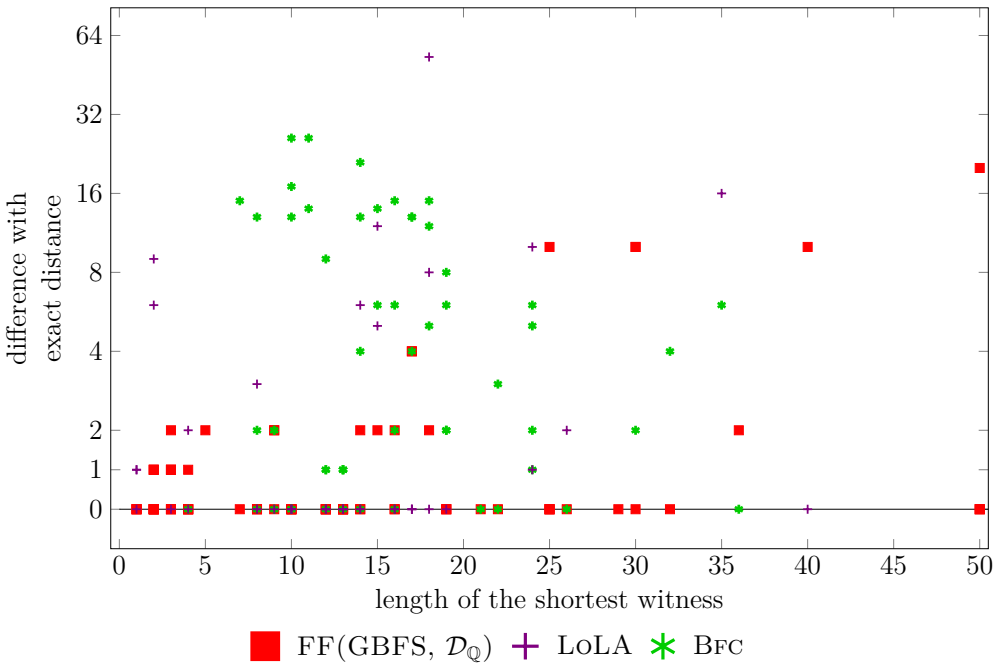


Figure 3.6: Length of the returned witness, per tool, compared to the length of a shortest witness. ICover is left out as it does not return witnesses. FF(A*, $\mathcal{D}_{\mathbb{Q}}$), FF(Dijkstra) and mist are left out as they are guaranteed to return shortest witnesses.

Chapter 4

Workflow Nets and Soundness

Aside from verifying and synthesizing software, one of the most impactful applications of Petri nets is in modelling and verifying *business processes*, sometimes also called workflows. An example of a workflow could be the process used by a company to deal with incoming applications. Once an application is received, in parallel the legal department checks whether the applicant fulfils the legal requirements to be employed by the company, and the hiring manager checks whether the applicant is a suitable candidate. Once both the legal department and the hiring manager have done their checks, a decision is made: The candidate is either accepted, rejected, or no clear decision can be made, and the application undergoes another round of review.

The importance of these processes is illustrated by the wide variety of languages that exist to model them: among others, workflows can be modelled using YAWL [103], BPMN [44], EPC [94], and UML activity diagrams [49]. Apart from being manually generated by human process designers, such models are also generated automatically, for example by taking so-called event logs and deriving a workflow model for them using process mining [101]. While different modelling languages have different levels of expressiveness, they all essentially focus on being able to express concurrent and sequential execution of tasks.

Many formalisms for modelling workflows can be translated into a special type of Petri nets called *workflow nets*, which very naturally capture many of the primitives available in the modelling languages mentioned earlier [43, 44, 72]. Consequently,

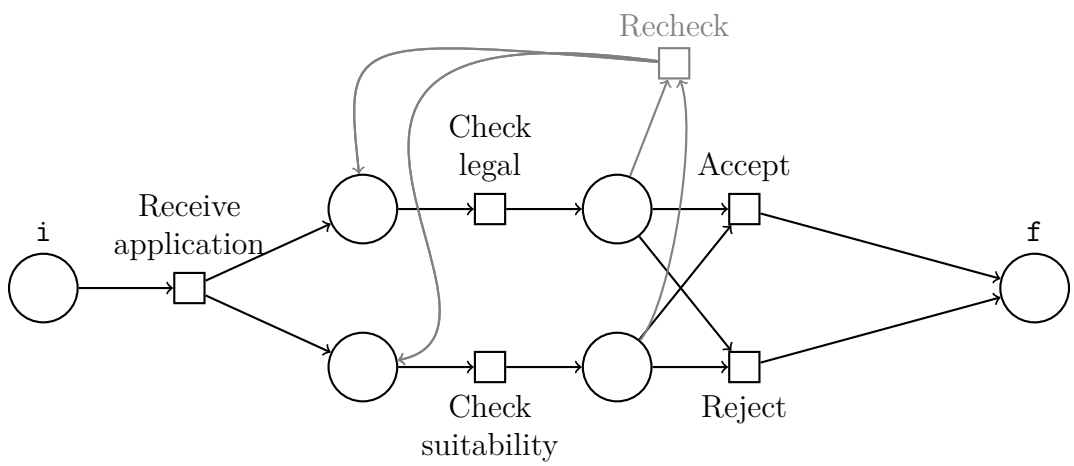


Figure 4.1: An example of a workflow net that represents an application process. When an application is received, first the legal requirements to employ the candidate, and the suitability of the candidate are checked in any order. Then, a decision is made, which can be to either accept the candidate, reject the candidate, or to check the application again.

Petri net-based techniques are ubiquitous in many tools for the modelling and analysis of business processes and workflows [14, 104, 105].

Formally, a *workflow net* $N = (P, T, f)$ is a Petri net that additionally satisfies the following conditions:

— There exists an *initial place* $\mathtt{i} \in P$ such that $f(t, \mathtt{i}) = 0$ for all $t \in T$,

— there exists a *final place* $\mathtt{f} \in P$ such that $f(\mathtt{f}, t) = 0$ for all $t \in T$,

— in the graph (V, E) with vertices $V = P \cup T$ and edges $E\{(u, v) \mid f(u, v) > 0\}$, any vertex lies on a path from $\mathtt{i}$ to $\mathtt{f}$.

Intuitively, $\mathtt{i}$ signifies the start of the process, and $\mathtt{f}$ signifies the end.

A workflow net for the example of a company processing job applications, as described above, is shown in Figure 4.1. The transitions are labelled with the names of tasks they represent. When starting the process, thus from marking $\{\mathtt{i}\colon 1\}$, one possible behaviour would be firing $t_{\text{Receive application}}, t_{\text{Check legal}}, t_{\text{Check suitability}}$, and t_{Accept}, which ends in marking $\{\mathtt{f}\colon 1\}$.

However, not all workflow nets are well-behaved. For example, consider the net shown in Figure 4.2. From marking $\{\mathtt{i}\colon 1\}$, it enables firing the transition

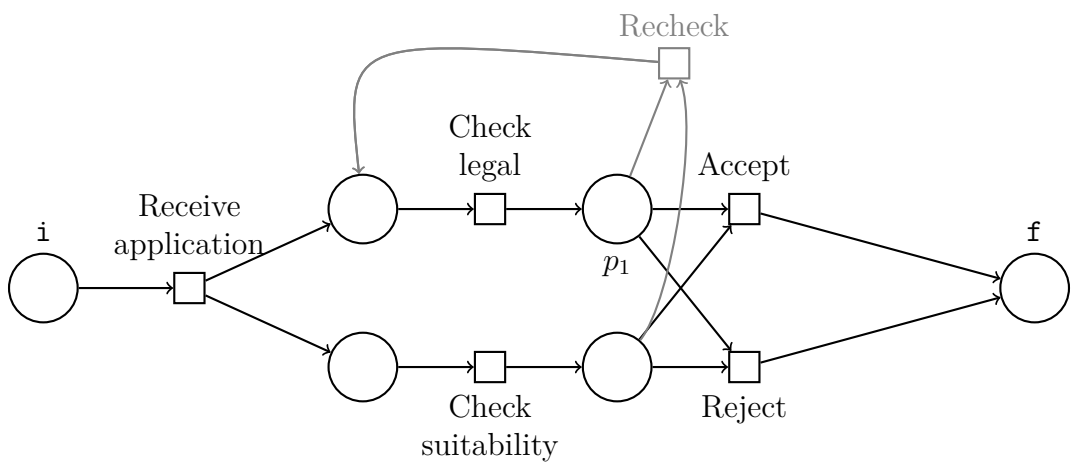


Figure 4.2: A slightly altered version of the net in Figure 4.1 which had one of the outgoing arcs of transition t_{Recheck} removed. This net is *not* k-sound for any $k \in \mathbb{N}$, and thus neither generalised sound nor structurally sound.

sequence $t_{\text{Receive application}}, t_{\text{Check legal}}, t_{\text{Check suitability}}, t_{\text{Recheck}}, t_{\text{Check legal}}$, which leads to marking $\{p_1 \colon 1\}$. From that marking, no more transitions are enabled. This seems like defective behaviour: The process can never end correctly, since it is impossible to reach the marking $\{\mathtt{f} \colon 1\}$, which corresponds to the end of the process. To avoid such defective behaviour, we introduce here a correctness condition for workflow nets called *k-soundness*, which was originally defined by van Hee et al. in [64].

Definition 4: The k-soundness Problem for workflow nets [64, Definition 3]

A workflow net is k-sound if and only if for any $\mathcal{M}$ such that $\{\mathtt{i} \colon k\} \xrightarrow{*} \mathcal{M}$, it holds that $\mathcal{M} \xrightarrow{*} \{\mathtt{f} \colon k\}$.

A slight variation of 1-soundness, named here *classical soundness*, is also presented in the literature. It is satisfied for a workflow net $N = (P, T, f)$ if and only if N is 1-sound, and that further each transition is quasi-live from $\{\mathtt{i} \colon 1\}$.

It turns out that 1-soundness reduces to classical soundness by an exponential-time procedure. For each transition t of net N, we can check whether the marking ${}^\bullet t$ is coverable from $\{\mathtt{i} \colon 1\}$. This is equivalent to checking whether a marking which

Figure 4.3: A workflow net N' which is k-sound if and only if the net N, shown in color, is 1-sound.

enables t is reachable from $\{\mathtt{i}\colon 1\}$, thus checking quasi-liveness of t. Then we can remove transitions for which this is not the case, since they are never enabled in any marking from the initial marking, and thus we are left with a net where classical soundness holds if and only if 1-soundness holds.

Let us also briefly argue why k-soundness can be reduced to 1-soundness. Given a workflow net N for which we want to check k-soundness, we can construct a workflow net N' that is 1-sound if and only if N is k-sound by adding two places $\mathtt{i}'$ and $\mathtt{f}'$ and two transitions $t_{\mathtt{i}}$ and $t_{\mathtt{f}}$. We further adjust f by adding $f(\mathtt{i}', t_{\mathtt{i}}) = f(t_{\mathtt{f}}, \mathtt{f}') = k$ and $f(t_{\mathtt{i}}, \mathtt{i}) = f(\mathtt{f}, t_{\mathtt{f}}) = 1$. Note that for N', the initial and final place are $\mathtt{i}'$ and $\mathtt{f}'$. Then note that $\{\mathtt{i}'\colon 1\} \xrightarrow{t_{\mathtt{i}}} \{\mathtt{i}\colon k\}$ and $\{\mathtt{f}\colon k\} \xrightarrow{t_{\mathtt{f}}} \{\mathtt{f}\colon 1\}$. Figure 4.3 illustrates the construction.

We can similarly reduce 1-soundness to k-soundness by reusing the same construction, but having instead $f(\mathtt{i}', t_{\mathtt{i}}) = f(t_{\mathtt{f}}, \mathtt{f}') = k$ and $f(t_{\mathtt{i}}, \mathtt{i}) = f(\mathtt{f}, t_{\mathtt{f}}) = 1$, thus the problems are interreducible.

The parameterized nature of k-soundness immediately gives rise to two related properties, called *generalised soundness* and *structural soundness*.

Definition 5: The generalised soundness problem for workflow nets [64, Definition 3]

A workflow net is *generalised sound* if and only if it is k-sound for all $k \geq 1$.

Definition 6: The strucural soundness problem for workflow nets [36]

A workflow net is *structurally sound* if and only if there exists a $k \geq 1$ such that it is k-sound.

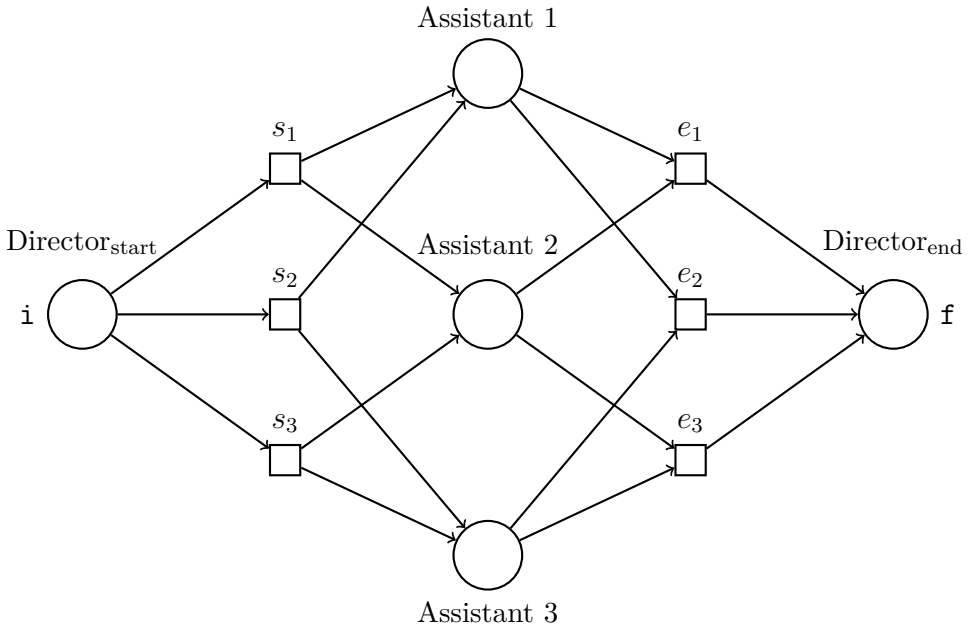


Figure 4.4: A workflow net modelling the distribution of a single task among two out of three resources. The net is 1-sound, but not k-sound for any other k. Thus, it is structurally sound, but not generalised sound.

Let us demonstrate soundness on a few examples. The net in Figure 4.1 is k-sound for all k, thus both generalised sound and structurally sound. Next, let us look at the net in Figure 4.2 and show that it is not k-sound for any k. Recall we demonstrated that $\{\mathtt{i}\colon 1\} \xrightarrow{*} \{p_1\colon 1\}$, which is a deadlock and thus $\{p_1\colon 1\} \not\xrightarrow{*} \{\mathtt{f}\colon 1\}$. Observe that it follows that for any k, $\{\mathtt{i}\colon k\} \xrightarrow{*} \{p_1\colon k\}$, which is also a deadlock. So $\{p_1\colon k\} \not\xrightarrow{*} \{\mathtt{f}\colon 1\}$, thus the net is not k-sound for any k. So the net is neither generalised sound nor structurally sound. Lastly, we take a look at the net in Figure 4.4. Observe that the net is 1-sound: From the initial marking $\{\mathtt{i}\colon 1\}$, the three transitions s_1, s_2 and s_3 are enabled. If we fire s_i for some i, then afterwards only transition e_i is enabled, and firing it leads to the correct target marking $\{\mathtt{f}\colon 1\}$. However, the net is not 2-sound. Notice that

$$\{\mathtt{i}\colon 2\} \xrightarrow{s_1 s_e} \{\text{Assistant } 1\colon 1, \text{Assistant } 2\colon 2, \text{Assistant } 3\colon 1\} \xrightarrow{e_2} \{\text{Assistant } 2\colon 2, \mathtt{f}\colon 1\}.$$

Note that the final marking along this run is a deadlock, thus from that marking, we cannot reach $\{\mathtt{f}\colon 2\}$. So the net is not 2-sound, thus also not generalised sound.

All three soundness problems have received attention in the literature. The original variant of soundness is classical soundness. The notion was first defined by van der Aalst in [3], and already a characterization was given, which shows that classical soundness is equivalent to *boundedness* and *liveness* of the so-called *short-circuited net*. Given a workflow net $N = (P, T, f)$, the short-circuited net N_{sc} is defined as $N_{\text{sc}} = (P, T', f')$, where $T' = T \cup \{t_{\text{sc}}\}$, and $f'(a, b) = f(a, b)$ for all $a, b \in P \cup T$, and additionally $f'(\mathtt{f}, t_{\text{sc}}) = f'(t_{\text{sc}}, \mathtt{i}) = 1$. Intuitively, the short-circuited net simply adds a single transition, which consumes one token from the final place and puts one token into the initial place. Note that the short-circuited net is not a workflow net, since transition t_{sc} violates the requirements that no transition consumes from $\mathtt{f}$, and that no transition produces in $\mathtt{i}$.

Theorem 9: Characterization of classical soundness [3, Theorem 11]

A workflow net N is classically sound if and only if N_{sc} is live and bounded from $\{\mathtt{i}\colon 1\}$.

Let us give some rough intuition behind the result. If N is classically sound,

then from any marking reachable from $\{\mathtt{i}\colon 1\}$, we can reach $\{\mathtt{f}\colon 1\}$. Then t_{sc} can be fired in N_{sc}, and we are back to the initial marking. So N_{sc} is live. Additionally, assume for contradiction N_{sc} is not bounded from $\{\mathtt{i}\colon 1\}$, but N is classically sound. Because Petri nets have a finite number of places, unboundedness of N_{sc} from $\{\mathtt{i}\colon 1\}$ is equivalent to having $\{\mathtt{i}\colon 1\} \xrightarrow{\pi} \mathcal{M} \xrightarrow{\pi'} \mathcal{M} + \mathcal{M}'$ in N_{sc}, with $\mathcal{M}' > \vec{0}$ and some runs π and π'. Take the first time we fire t_{sc} among the whole run $\pi\pi'$. We either fire t_{sc} from $\{\mathtt{f}\colon 1\}$ to $\{\mathtt{i}\colon 1\}$, in which case we can shorten the run to omit behaviour up to that point since we are back to the initial marking, or we fire it from $\{\mathtt{f}\colon 1\} + \mathcal{N}$ with $\mathcal{N} > 0$. In the latter case, we have $\{\mathtt{i}\colon 1\} \xrightarrow{*} \{\mathtt{f}\colon 1\} + \mathcal{N}$ in N, since we fired only transitions that are also in N. However, $\mathtt{f}$ has no outgoing arcs by definition, and all transitions need to be on a path from $\mathtt{i}$ to $\mathtt{f}$, so in particular all transitions need to have an outgoing arc. So intuitively, from $\{\mathtt{f}\colon 1\} + \mathcal{N}$, we cannot reach $\{\mathtt{f}\colon 1\}$, thus N is unsound. In case t_{sc} does not appear in π and π', all transitions of the run are also available in N, thus we have $\{\mathtt{i}\colon 1\} \xrightarrow{\pi} \mathcal{M} \xrightarrow{\pi'} \mathcal{M} + \mathcal{M}'$ in N. Since we assume N to be classically sound, it must hold that $\mathcal{M} \xrightarrow{\rho} \{\mathtt{f}\colon 1\}$ in N for some run ρ. It follows that $\mathcal{M} + \mathcal{M}' \xrightarrow{\rho} \{\mathtt{f}\colon 1\} + \mathcal{M}'$, and we can repeat the argument from above to show that N is not sound, since in N from $\{\mathtt{f}\colon 1\} + \mathcal{M}'$ we cannot reach $\{\mathtt{f}\colon 1\}$.

For the other direction, assume that for N_{sc} it holds that t_{sc} is live and the net is bounded from $\{\mathtt{i}\colon 1\}$. By liveness of t_{sc}, from any marking reachable from $\{\mathtt{i}\colon 1\}$ in N_{sc}, we can reach a marking $\{\mathtt{f}\colon 1\} + \mathcal{M}$ for some $\mathcal{M} \geq \vec{0}$, since that is the only way to enable t_{sc}. We can then fire t_{sc} to get back to $\{\mathtt{i}\colon 1\} + \mathcal{M}$. By repeating the run, we get $\{\mathtt{i}\colon 1\} + \mathcal{M} \xrightarrow{*} \{\mathtt{i}\colon 1\} + 2\mathcal{M} \xrightarrow{*} \{\mathtt{i}\colon 1\} + 3\mathcal{M}/cdots$. The only way this does not violate boundedness is to have $\mathcal{M} = \vec{0}$. Thus, from any marking reachable from $\{\mathtt{i}\colon 1\}$ in N_{sc}, we can reach $\{\mathtt{f}\colon 1\}$, which means N is classically sound.

The characterization shows decidability, since boundedness can be decided in EXPSPACE, and liveness is Ackermann-complete. However, the exact complexity of the problem was left open, but suggested to be EXPSPACE-hard, though no proof appears to the best of our knowledge. For example, in [6], it is mentioned that "[...] soundness is decidable but also EXPSPACE-hard ([1])", yet [1] merely states that "it may be intractable to decide soundness. (For arbitrary [workflow]-nets liveness and boundedness are decidable but also EXPSPACE-hard [...])". Naively, one could even expect the problem to inherit the Ackermann-hardness from liveness, but

note that this is not necessarily the case. For example, for the subclass of free-choice nets, testing liveness is coNP-complete [42, Theorem 4.28] and testing boundedness is EXPSPACE-hard, but testing simultaneously liveness and boundedness can be done in polynomial time [42, Corollary 6.18].

Generalised soundness was defined several years later by van Hee et al. in [64]. They were motivated by their observation that k-soundness is not compositional. Observe that one can replace a transition in a workflow net by another workflow net, and the result will again be a workflow net. However, even if both inputs are k-sound, this does *not* necessarily result in a k-sound workflow net; thus k-soundness is not compositional. Consider that in a k-sound net, it may be possible to fire a transition twice in succession, which would input $2k$ tokens into the workflow net that replaces the transition. Even if that net is k-sound, it may not be $2k$-sound, so may be able to get stuck and not reach the final marking. Thus, k-soundness is not compositional. However, if both inputs are generalised sound, the result will be generalised sound as well, thus generalised soundness *is* compositional in this regard. Decidability of the problem was established in [65], but the paper does not give any bound on the complexity.

For structural soundness, the situation is similar. The problem was first defined in [12], and later shown decidable in [36], but no other complexity bounds are known. In [36], the problem is motivated by noting that generalised soundness is a very restricting requirement, and structural soundness can serve as a more permissive variant. To summarize:

Observation 10

In the literature, it was shown that classical, generalised and structural soundness have an Ackermannian upper-bound on their complexity, but no lower bounds have been proven, leaving the complexity of all three problems widely open.

Let us briefly introduce the notion of *nonredundancy*. We say a workflow net N is nonredundant if for each transition t, there exists $k \in \mathbb{N}$ such that t is quasi-live from $\{\mathtt{i} : k\}$. This can be checked by using reachability over $\mathbb{Z}$, and transitions for which this does not hold can never be fired, no matter the initial number of tokens, so we

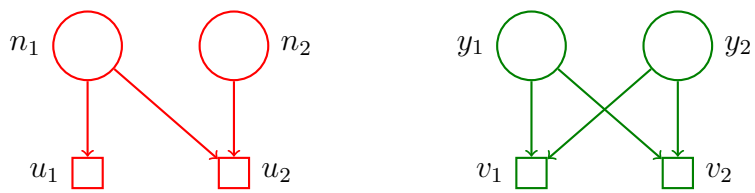


Figure 4.5: An illustration of the free-choice property in Petri nets. Left, in red: A Petri net that is *not* free-choice. Right, in green: A free-choice Petri net.

may simply remove them. We usually assume that workflow nets are nonredundant.

The high complexities associated with many decision problems on Petri nets have sparked the search for subclasses that are expressive enough to model interesting and complex systems, yet are easier to analyse than standard Petri nets. A particularly notable subclass are *free-choice Petri nets* [42]. Intuitively, this subclass aims to separate *concurrency* (cases where multiple transitions can occur in any order) from *choice* (cases where only one of a set of transitions may occur).

Formally, a free-choice Petri net (P, T, f) is a Petri net where $f(a, b) \in \{0, 1\}$ for all $a, b \in P \cup T$, and such that further for all pairs of transitions $t_1, t_2 \in T$ it holds that if there exists p such that ${}^\bullet t_1(p) = {}^\bullet t_2(p) = 1$, then ${}^\bullet t_1 = {}^\bullet t_2$. In words, the flow function can only have values 0 or 1, and if two transitions both consume a token from the same place, then their guards need to be identical.

Figure 4.5 illustrates the concept. The Petri net on the left-hand side is not free-choice, as ${}^\bullet u_1(n_1) = {}^\bullet u_2(n_1) = 1$, but ${}^\bullet u_1(n_2) \neq {}^\bullet u_2(n_2)$. The Petri net on the right-hand side is free-choice, since ${}^\bullet v_1 = {}^\bullet v_2$.

Free-choice Petri nets turn out to be far easier to analyse than standard Petri nets for some properties: For example, in standard Petri nets, the liveness problem is equivalent to the reachability problem [62, Theorem 1], thus Ackermann-hard [37], but the same problem is coNP-complete in free-choice nets [42, Theorem 4.28]. It turns out that reachability, however, remains as hard as in standard Petri nets, thus Ackermann-hard.

Notably, soundness is known to be decidable in polynomial time for workflow nets that are free-choice [3, Theorem 12]. This has led to intense study of free-choice workflow nets, see for example [4, 51].

Chapter 5

The Complexity of Soundness in Workflow Nets

This section presents results from [25], a paper that resulted from joint work with my two supervisors, and which was published in the proceedings of LICS'22.

There, we consider the three variants of soundness in workflow nets as defined in Chapter 4, namely k-soundness, generalised soundness, and structural soundness. As stated in Observation 10, the exact complexities of all three problems were left widely open in the previous literature. Our three main results are that:

— k-soundness is EXPSPACE-complete,

— generalised soundness is PSPACE-complete, and

— structural soundness is EXPSPACE-complete.

As a corollary to the EXPSPACE-completeness of k-soundness, we obtain the same complexity for classical soundness. We further show how to compute a representation of the set of numbers for which a given net is sound in exponential time.

5.1 k-Soundness

We will focus on 1-soundness and classical soundness in this section, since k-soundness can be reduced to 1-soundness, as demonstrated in Figure 4.3.

Recall Theorem 9, which characterizes classical soundness in terms of liveness and boundedness of the short-circuited net. Unfortunately, liveness is equivalent to reachability in Petri nets, thus Ackermann-hard [37, 62].

Thus, we look to replace liveness with conditions that are easier to decide. We make a simple observation relating classical soundness to quasi-liveness, boundedness and cyclicity.

Let us briefly introduce the latter property. Cyclicity asks whether from a given marking $\mathcal{S}$, it is the case that from any marking $\mathcal{M}$ such that $\mathcal{S} \xrightarrow{*} \mathcal{M}$, it holds that $\mathcal{M} \xrightarrow{*} \mathcal{S}$. The problem is EXPSPACE-complete. Membership was shown by Bouziane and Finkel in 1997 [30], while EXPSPACE-hardness is folklore.

Lemma 11: [25, Corollary 3.4] †

A workflow net N is classically sound if and only if N_{sc} is bounded and cyclic, and all transitions are quasi-live from $\{\mathtt{i} : 1\}$.

Proof: Appendix B, Page 4

Proving the characterization is straightforward. The proof goes along the lines of the intuition we outlined for Theorem 9, but the simple observation that liveness can be replaced by quasi-liveness and cyclicity in this characterization is valuable: Quasi-liveness and cyclicity are in EXPSPACE [30, 91]. For an intuition of why we can replace liveness by cyclicity and quasi-liveness, it is the case that boundedness together with liveness of t_{sc} already imply cyclicity, and liveness on its own clearly implies quasi-liveness.

The characterization extends to 1-soundness, up to a technical detail. A net N is 1-sound if and only if N_{sc} is cyclic and bounded and some transition of N consumes exactly one token from $\mathtt{i}$.

Lemma 12: [25, Lemma 3.3] †

A workflow net N is 1-sound if and only if N_{sc} is bounded and cyclic from $\{\mathtt{i} : 1\}$, and some transition $t \in T$ satisfies ${}^{\bullet}t = \{\mathtt{i} : 1\}$.

Proof: Appendix B, Page 4

Thus, we obtain the following result:

Theorem 13: [25, Theorem 3.5] †

The 1-soundness and classical soundness problems are in EXPSPACE.

Proof: Appendix B, Page 4

Next, we turn to the lower bound. For hardness, we give the following result:

Theorem 14: [25, Theorem 3.10] †

The 1-soundness and classical soundness problems are EXPSPACE-hard.

Proof: Appendix B, Page 5, continued on Page 14

Let us give a very rough sketch of the proof strategy. We will give the intuition behind the hardness of 1-soundness in the following, and later explain what needs to be added to the reduction for classical soundness.

The reduction is from reachability in *reversible Petri nets*. A reversible Petri net is a Petri nets such that for each transition t, there exists a transition t_{reverse} such that ${}^\bullet t_{\text{reverse}} = t^\bullet$ and $t^\bullet_{\text{reverse}} = {}^\bullet t$. Reachability in reversible Petri nets is EXPSPACE-complete [34, 82].

In the following, let $N = (P, T, f)$ be a reversible Petri net and let $\mathcal{S}$, $\mathcal{T}$ be markings. Our goal is to construct a workflow net N' such that N' is 1-sound if and only if in N, $\mathcal{S} \xrightarrow{*} \mathcal{T}$. For now, let us start by taking $N' = N$, and in the following, we are gradually going to describe the additions we make to N'.

From the upper bound for reachability in reversible Petri nets, we extract a useful lemma which places a bound on the number of tokens needed in markings along runs to reach $\mathcal{T}$.

Lemma 15: [82, Lemma 3]

Let $n := \text{size}(N, \mathcal{S}, \mathcal{T})$. There exists $c_n \in 2^{2^{O(n)}}$ such that if $\mathcal{S} \xrightarrow{*} \mathcal{T}$, then there exists a c_n-bounded run ρ such that $\mathcal{S} \xrightarrow{\rho} \mathcal{T}$.

Intuitively, a c_n-bounded run is a run which has at most c_n tokens per place in each marking along the run. This simple statement allows a useful construction: For each place p, let us add a "budget place" p', where the number of tokens in p' is how

many more tokens can be present in p in a c_n-bounded run. For ease of notation, let us collectively refer to the set of budget places as P'.

Defining the behaviour of P' more formally, we ensure that except during some setup and final steps, for any reachable marking $\mathcal{M}$ the invariant $\mathcal{M}(p)+\mathcal{M}(p') = c_n$ holds. We achieve this as follows: initially, $p' = c_n$ for $p' \in P'$. Further, for any transition $t \in T$, we set $f(p', t) = f(t, p)$ and $f(t, p') = f(p, t)$. Thus, whenever a token is added to p, one token is removed from p'. Similarly, whenever a token is consumed from p, a token is added to p'.

For a marking $\mathcal{M}$ of N, let us denote as $\mathcal{C}_\mathcal{M}$ the marking of N' where for each place $p \in P$, $\mathcal{C}_\mathcal{M}(p) = \mathcal{M}(p)$, and $\mathcal{C}_\mathcal{M}(p') = c_n - \mathcal{M}(p)$. Note that for any markings $\mathcal{M}, \mathcal{M}'$ and any run π, $\mathcal{M} \xrightarrow{\pi} \mathcal{M}'$ in N if and only if $\mathcal{C}_\mathcal{M} \xrightarrow{\pi} \mathcal{C}'_\mathcal{M}$ in N'.

We can have a transition that is enabled if and only if the current marking in P is exactly $\mathcal{T}$ by having it consume $\mathcal{C}_\mathcal{T}$. Then it is easy to imagine how we can continue to construct N'. We add two places $\mathtt{i}$ and $\mathtt{f}$ and two transitions $t_\mathtt{i}$ and $t_\mathtt{f}$. Transition $t_\mathtt{i}$ consumes a token from $\mathtt{i}$ and produces the marking $\mathcal{C}_\mathcal{S}$, and $t_\mathtt{f}$ consumes $\mathcal{C}_\mathcal{T}$ and produces a token in $\mathtt{f}$. It holds that if $\mathcal{S} \xrightarrow{\pi} \mathcal{T}$ in N, then $\{\mathtt{i}\colon 1\} \xrightarrow{t_\mathtt{i}} \mathcal{C}_\mathcal{S} \xrightarrow{\pi} \mathcal{C}_\mathcal{T} \xrightarrow{t_\mathtt{f}} \{\mathtt{f}\colon 1\}$ in N'. Let us briefly argue why reversibility is important, and consequently why this reduction does not work starting from an arbitrary Petri net. Assume that for some marking $\mathcal{M}$ of N, it holds that $\mathcal{S} \xrightarrow{*} \mathcal{M}$ and $\mathcal{S} \xrightarrow{*} \mathcal{T}$ but $\mathcal{M} \not\xrightarrow{*} \mathcal{T}$. Then $\mathcal{C}_\mathcal{S} \xrightarrow{*} \mathcal{C}_\mathcal{M}$ and $\mathcal{C}_\mathcal{M} \not\xrightarrow{*} \mathcal{T}$. This would mean N' is unsound, even though $\mathcal{S} \xrightarrow{*} \mathcal{T}$ in N. However, note that the assumption is impossible by reversibility, since $\mathcal{S} \xrightarrow{*} \mathcal{M}$ implies $\mathcal{M} \xrightarrow{*} \mathcal{S}$. So if $\mathcal{S} \xrightarrow{*} \mathcal{T}$, then $\mathcal{M} \xrightarrow{*} \mathcal{T}$.

The construction is sketched in Figure 5.1. Note that we skip over one of the main technical challenges here: We cannot simply write c_n on arcs, as it is doubly-exponential in the size of N, thus even when encoding arc weights in binary, this would result in an exponential blow-up in the size of N' compared to N. Thus, we need to construct a small gadget which produces c_n, and further the gadget should not automatically make N' violate soundness, which would for example be the case if the gadget had a way to get stuck if transitions are fired incorrectly. Such a gadget can be extracted from [82, Lemmas 6 and 8]. Very roughly, the idea is to use a set of places to keep a state, meaning only one of multiple places has a token at the same time, and to use other places to store a counter. Switching from one state to the next

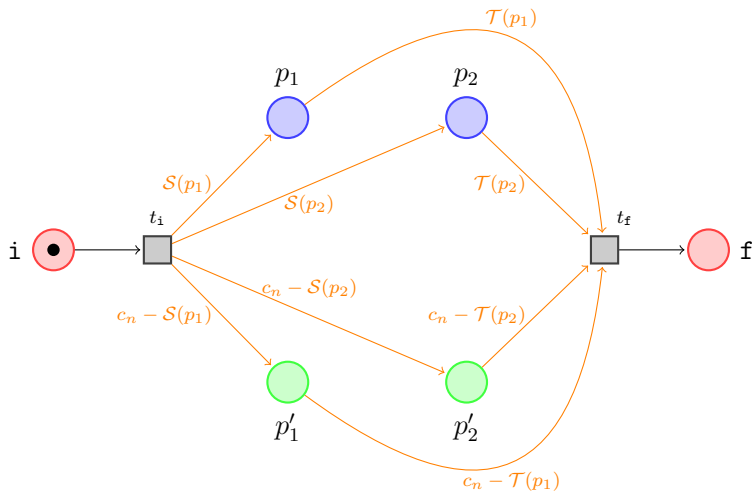


Figure 5.1: A sketch of the reduction from reachability in reversible Petri nets to 1-soundness in workflow nets. We sketch a workflow net N' which is 1-sound if and only if $\mathcal{S} \to \mathcal{S}'$ in the reversible Petri net $N = (P, T, F)$. In the example, $P = \{p_1, p_2\}$. The original places are blue, their copies are green, and other new places are red. We omit the transitions in T_1 that originated from T (recall that these transitions are modified to consume and produce tokens also in green places). We only provide a rough sketch by writing the intuitive meaning of the gadgets that add/remove c_n tokens, here drawn as orange arcs.

causes the number of tokens in the counter places to be multiplied, and switching back is only possible when at the same time decreasing the number of tokens in the counter places accordingly. Then, reaching the final state corresponds to having multiplied the counter by the right number c_n.

Finally, to adapt the construction to show hardness of classical soundness instead of 1-soundness, we need to ensure that all transitions of N' are quasi-live if $\mathcal{S} \xrightarrow{*} \mathcal{T}$ in N. Let B be the maximal number of tokens consumed by any transition in T. Then we ensure quasi-liveness by adding an alternative execution path consisting of two transitions, t_{simple} and t_{simple2}. First, t_{simple} consumes a token from `i` and saturates each place among P and P' with B tokens, thus enabling all transitions in T, and then t_{simple2} consumes B tokens from places in P and P' and produces a token in `f`.

5.2 Generalised Soundness

Recall that for generalised soundness, we claim that it is PSPACE-complete. Let us first show the membership result.

Theorem 16: [25, Theorem 5.1] †

Generalised soundness is in PSPACE.

Proof: Appendix B, Page 8

In the following, let $N = (P, T, f)$ be a workflow net with initial place $\mathtt{i}$ and final place $\mathtt{f}$.

Recall the notion of $\mathbb{Z}$-boundedness as defined in Section 2.3.2, which we will use in the following. We are ready to outline our proof for PSPACE-membership of generalised soundness, which proceeds in three steps.

We show that:

1. There exists a bound $b \in \mathbb{N}$ at most exponential in $\|N\|$ such that if N is not generalised sound, it holds that N is k-unsound for some $k \leq b$ [25, Lemmas 5.6 and 5.8].
2. For all $k \in \mathbb{N}$, if N is $\mathbb{Z}$-unbounded from $\{\mathtt{i} \colon k\}$, then N is not generalised sound [25, Lemma 5.9].
3. For all $k \in N$ there exists a bound $b_k \in \mathbb{N}$ at most polynomial in $k + \|N\|$ such that if N is $\mathbb{Z}$-bounded from $\{\mathtt{i} \colon k\}$, then for any marking $\mathcal{M}$ such that $\{\mathtt{i} \colon k\} \xrightarrow{*} \mathcal{M}$, it holds that $\|\mathcal{M}\| \leq b_k$ [25, Lemma 5.10].

Let us outline how the three points give us a polynomial-space algorithm for deciding generalised soundness. Notice that by the first point, generalised soundness is equivalent to k-soundness for all $1 \leq k \leq b$. Thus, let us iterate over all such k and check k-soundness for each. Note that as b is exponentially large, we can write it in polynomial space by a binary encoding.

One might be temped to use the procedure for k-soundness as a black-box, but recall that the problem is EXPSPACE-hard. Thus, we cannot use the existing procedure, but we instead use some additional restrictions from the second and third point to check k-soundness. Note that if any marking reachable from $\{\mathtt{i} \colon k\}$ has more than

b_k tokens in any place, then N is $\mathbb{Z}$-unbounded from $\{\mathtt{i}\colon k\}$ by the third point, and thus not generalised sound by the second point. Thus, we can iterate through all reachable markings from $\{\mathtt{i}\colon k\}$; if any are too big and have more than b_k tokens in any place, we know that N is not generalised sound. We have that b_k is at most polynomial in $k+\|N\|+|N|$ by the second point, and k is at most exponential in $\|N\|$ by the third point. Thus, all reachable configurations can be stored in polynomial space, and we get PSPACE membership.

Let us give the brief ideas behind the proofs of the three points:

Proof of 1.) The idea is to relate reachability with many tokens to reachability over $\mathbb{Z}$ [65, Lemma 12], to relate reachability over $\mathbb{Z}$ to solutions of an integer linear program [25, Claim 5.7], and lastly to use known results on the size of minimal solutions to integer linear programs [25, Lemma 4.3].

Proof of 2.) If N is $\mathbb{Z}$-unbounded from $\{\mathtt{i}\colon k\}$, it is easy to show that under the mild technical assumption of nonredundancy, this translates to unboundedness from $\{\mathtt{i}\colon k+\ell\}$ for some $\ell \in \mathbb{N}$. Intuitively, with enough initial budget, we can saturate the net and transform a run that witnesses $\mathbb{Z}$-unboundedness into one that witnesses unboundedness. We can use Lemma 12 and the reduction from k-soundness to 1-soundness of Figure 5.4 to show that N is then not $(k+\ell)$-sound, thus not generalised sound.

Proof of 3.) The proof makes use of Steinitz Lemma [98]. We refer to [25, Lemma 4.5] for the formal statement, and instead restrict ourselves to giving the intuition. A graphical representation of the lemma can be seen in Figure 5.2. Intuitively, let z be a point. Let $x_0, \ldots, x_n \in \mathbb{Z}^d$ be vectors such that $\sum_{i=0}^{n} x_i = z$. Equivalently, we can think that the vectors form a path from 0 to z. This is the initial situation shown on the left-hand side of Figure 5.2. Then the vectors can be reordered such that x_0 remains the first vector, and further no intermediate point along the path lies outside a band around the straight-line from 0 to z. Note that after reordering, the vectors will still be a path from 0 to z, since reordering does not change their sum. The width of the band depends polynomially on the dimension d and the size of vectors x_i, but crucially, neither on z nor n. Then the idea is as follows: If $\{i : k\} \xrightarrow{t_1 t_2 \ldots t_n} \mathcal{M}$ with $\|M\| > b_k$, then the vectors $\{i : k\}, t_1, t_2, \ldots, t_n$ form a path from 0 to $\mathcal{M}$. Intuitively, as $\mathcal{M}$ is large, many

transitions must be used to reach it, so n must be big. Thus, there must be many points on the path. However, we can reorder the vectors so that all points are in a band of polynomial width around the straight line from 0 to $\mathcal{M}$. With enough points along the path, thus in the band, we will necessarily have two points such that one is greater than the other, since the band itself has a positive slope and is of limited width. Then the path of vectors between these two points witnesses $\mathbb{Z}$-unboundedness, since their sum is nonnegative on all components and positive on at least one component. In the picture, after reordering, the path that would witness unboundedness consists only of the vector x_2, since it is the first vector that causes a strict increase.

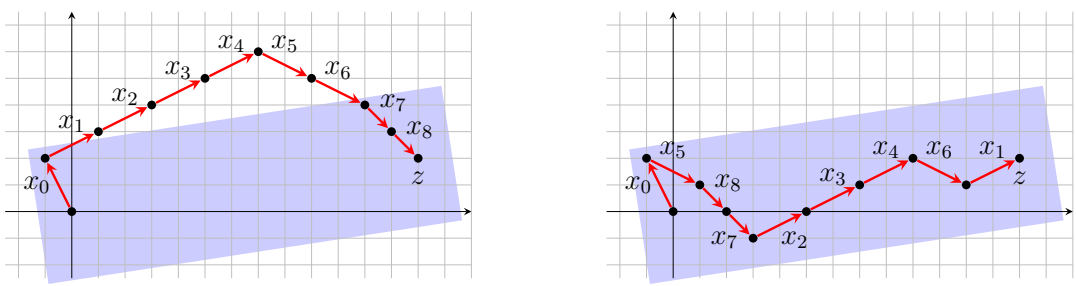


Figure 5.2: A visualization of Steinitz Lemma in dimension $d = 2$. The vectors $x_0, \ldots, x_n \in \mathbb{Z}^d$ form a path from point 0 to point z. The colored background highlights points that are within some bounded distance from the line 0 to z (the bound depends on d and x_i, but not on z). In the right picture, the vectors are reordered so that they all fit within the bound. The additional constraints are that the first vector x_0 remains first; and, *in some way*, the points are getting closer to z.

Next, let us tackle PSPACE-hardness.

Theorem 17: [25, Theorem 5.11] †

Generalised soundness is PSPACE-hard.

Proof: Appendix B, Page 11, continued on Page 16

Here, we reduce from reachability in *conservative Petri nets*. We say that a Petri net $N = (P, T, f)$ is *conservative* if all transitions preserve the number of tokens in the net. Formally, we require that for any pair of marking $\mathcal{M}, \mathcal{M}'$, if $\mathcal{M} \xrightarrow{*} \mathcal{M}'$,

then $\sum_{p \in P} \mathcal{M}(p) = \sum_{p \in P} \mathcal{M}'(p)$. Reachability in conservative Petri nets is PSPACE-complete [83].

In the following, let us assume that $N = (P, T, f)$ is a conservative Petri net, and $\mathcal{S}, \mathcal{T}$ are given markings. Then we will construct a Petri net $N' = (P', T', f')$ such that $\mathcal{S} \xrightarrow{*} \mathcal{T}$ in N if and only if N' is generalised sound. Observe that it is easy to rule out reachability when $\sum_{p \in P} \mathcal{S}(p) = \sum_{p \in P} \mathcal{T}(p)$, as conservativeness of N means that reachability does not hold. Thus, we can assume the markings have the same number of tokens. For ease of notation, let us write $c = \sum_{p \in P} \mathcal{S}(p)$.

The intuition behind our construction of N' is that it is easy to reset N, since we know that at any point in time, there are exactly c tokens. Thus, resetting means collecting c tokens in a new auxiliary place, then consuming them and producing again $\mathcal{S}$.

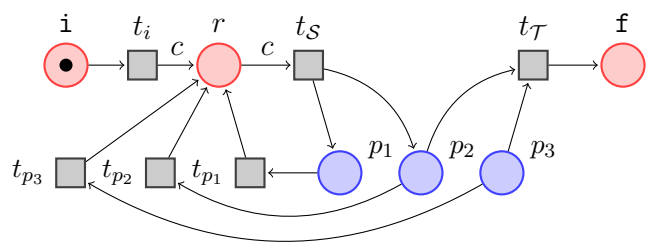


Figure 5.3: A sketch of the reduction from reachability in conservative Petri nets to generalised soundness of workflow nets. We draw a workflow net N' which is generalised sound iff $\mathcal{S} \to \mathcal{T}$ in the conservative Petri net $N = (P, T, F)$. Here, $P = \{p_1, p_2, p_3\}$, $\mathcal{S} = \{p_1 \colon 1, p_2 \colon 1\}$, $\mathcal{T} = \{p_2 \colon 1, p_3 \colon 1\}$ and thus $c = 2$. The original places are blue and the new places are red. We omit the original transitions (from T) in the picture.

Figure 5.3 shows a sketch of the construction. Let us give the idea behind it. Transition $t_{\mathcal{S}}$ consumes c tokens from place r and produces the initial marking $\mathcal{S}$ in places from P, while transition $t_{\mathcal{T}}$ consumes $\mathcal{T}$ from P and produces a single token in `f`. It is easy to see that N' is 1-sound if and only if $\mathcal{S} \to \mathcal{T}$ in N, since the reset gadget consisting of transitions t_{p_1}, t_{p_2} and t_{p_3} can "clear out" N' if it reaches a configuration from which $\mathcal{T}$ is not reachable, and put back $\mathcal{S}$ via transition $t_{\mathcal{S}}$. Thus, if $\mathcal{S} \not\to \mathcal{T}$, then N' is surely not generalised sound, as it is not even 1-sound. On the other hand, if $\mathcal{S} \xrightarrow{\pi} \mathcal{T}$ for some π, then for k larger than 1, we can essentially play run π in N'

k times, and by a similar argument as for 1-soundness reset the net if we reach a marking that cannot cover $\mathcal{T}$.

5.3 Structural Soundness

We show that structural soundness is EXPSPACE-complete.

Theorem 18: [25, Theorem 6.1] †

Structural soundness is in EXPSPACE.

Proof: Appendix B, Page 11

The proof reuses some ideas from the upper bound for generalised soundness. We establish a connection between $\mathbb{Z}$-reachability and integer linear programs which allows us to bound the size of the smallest number for which a net N is sound, if there are any. That bound b is exponential in $\|N\|$, thus we can write it in polynomial space. Thus, it suffices to check k-soundness for all $k \leq b$. Here, we invoke as a black-box our previous result of EXPSPACE-membership of k-soundness, see Theorem 13. If N is k-sound for any such k, it is structurally sound, otherwise it is not.

Theorem 19: [25, Theorem 6.4] †

Structural soundness is EXPSPACE-hard.

Proof: Appendix B, Page 12

We reduce from 1-soundness, which we have previously shown EXPSPACE-hard, see Theorem 13. The idea of the reduction is straightforward. Take a net N for which we want to check 1-soundness. We construct N' as a copy of N, and slightly modify it to ensure it is not sound for any $k \geq 2$; while not impacting 1-soundness. Then N' is structurally sound if and only if N is 1-sound. The reduction is shown in Figure 5.4.

5.4 Characterizing the set of sound numbers

The final contribution to this chapter is to characterize the set of sound numbers. For a workflow net N, we define $\text{Sound}_N = \{k \mid N \text{ is } k\text{-sound}\}$. We expand on an

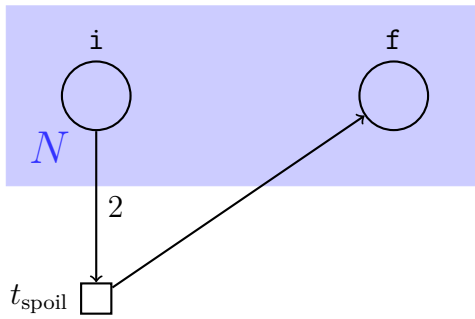


Figure 5.4: A sketch of the reduction from structural soundness to 1-soundness. We show a workflow net N' which is 1-sound if and only if the net N, shown in color, is 1-sound. However, N' is not k-sound for any $k \neq 1$, regardless of whether N is k-sound.

observation made in [36, Lemmas 2.2 and 2.3], and show that Sound_N is *closed under subtraction with positive results*, that is, for all $a, b \in \mathbb{N}$ with $a > b$, if N is a-sound and b-sound, then it is $(a - b)$-sound.

As a corollary, we can give a characterization of Sound_N.

Corollary 20: [25, Corollary 7.2] †

If $\text{Sound}_N \neq \varnothing$, there exist $e \in \mathbb{N}_{>0}$, $\ell \in \mathbb{N}_{>0} \cup \{\infty\}$ such that $\text{Sound}_N = \{i \cdot f \mid 1 \leq i < k\}$.

We reuse ideas from the upper bounds for structural soundness generalised soundness to give bounds exponential in $\|N\|$ on the size of e and ℓ, respectively, and compute them in exponential space by trying all possible candidates, due to the EXPSPACE-membership of k-soundness.

Chapter 6

Verifying Generalised and Structural Soundness

We established exact complexity bounds for generalised and structural soundness in Chapter 5, namely we have shown that generalised soundness is PSPACE-complete and structural soundness is EXPSPACE-complete. As a consequence of the high complexity of both problems, our upper bounds do not seem to yield practically relevant decision procedures. Thus, the goal of this chapter is to develop semi-decision procedures for generalised and structural soundness that are efficient in practice.

We present results from [27], which is a paper that resulted from joint work with my two supervisors, and which was published in the proceedings of CAV'22. The main contributions of the work are novel semi-decision procedures for generalised and structural soundness, which are based on the Petri net relaxations introduced in Section 2.3. We introduce a novel notion called continuous soundness which is a necessary requirement for generalised soundness. We show that continuous soundness is coNP-complete. For structural soundness, our main contribution is to relate a known necessary requirement called *structural quasi-soundness* to reachability in continuous Petri nets. Another major result of this work is that we show an equivalence of the different soundness notions on free-choice nets. There, the notions of $\{k$-, generalised, structural, continuous$\}$ soundness are equivalent. Lastly, we evaluate a prototype implementation of our approaches against existing tools for workflow nets, on a large set of existing benchmarks and on new families of synthetic benchmark instances pro-

posed by us. The prototype is an extension of this author's tool FastForward and was accepted as an artifact at CAV'22 [88].

6.1 Generalised Soundness

We state two necessary conditions for generalised soundness. One is $\mathbb{Z}$-boundedness, as defined in Section 2.3.2, and which we already showed to be a necessary condition for generalised soundness in Section 5.2. Our contribution here is to give a polynomial-time algorithm for checking whether for a net N, it holds that there exists $k \in \mathbb{N}_{>0}$ such that N is $\mathbb{Z}$-unbounded from $\{\mathtt{i} : k\}$.

The second necessary condition we define is called *continuous soundness*, which relates to continuous Petri nets as presented in Section 2.3.1. We show that continuous soundness is coNP-complete.

6.1.1 $\mathbb{Z}$-Boundedness

We have already shown that $\mathbb{Z}$-boundedness is a necessary condition in Section 5.2. It remains to establish the complexity of testing $\mathbb{Z}$-boundedness. We make the straightforward observation that $\mathbb{Z}$-boundedness is independent of the initial marking.

Proposition 21: [27, Proposition 4] †

A Petri net N is $\mathbb{Z}$-unbounded from a marking $\mathcal{M}$ if and only if there exists $\mathcal{M}' > \vec{0}$ such that $\vec{0} \xrightarrow{*}_{\mathbb{Z}} \mathcal{M}'$.

Proof: Appendix C, Page 8

It is easy to relate the fact that $\vec{0} \xrightarrow{*}_{\mathbb{Z}} \mathcal{M}' > \vec{0}$ to feasibility of an integer linear program as done in Proposition 2. However, this yields an NP-procedure, as integer linear programming is well-known to be NP-complete. We instead observe that we can translate the condition into a linear program. Intuitively, we show that there exists $\mathcal{M}' > \vec{0}$ such that $\vec{0} \xrightarrow{*}_{\mathbb{Z}} \mathcal{M}'$ if and only if there exists $\mathcal{M}'' > \vec{0}$ such that $\vec{0} \xrightarrow{*}_{\mathbb{Q}_{\geq 0}} \mathcal{M}''$ [27, Proposition 5]. Recall that $\rightarrow_{\mathbb{Q}_{\geq 0}}$ is reachability with negative token counts and fractional transition firings as defined in Section 2.3.3. Reachability

under $\rightarrow_{\mathbb{Q}_{\geq 0}}$ can be formulated as a linear program by Proposition 3. Thus, $\mathbb{Z}$-unboundedness can be decided in polynomial time.

6.1.2 Continuous Soundness

We introduce a continuous variant of 1-soundness based on continuous reachability.

Definition 7: Continuous Soundness †

A workflow net N is *continuously sound* if for all markings $\mathcal{M}$ such that $\{\mathtt{i}\colon 1\} \rightarrow_{\mathbb{Q}_{\geq 0}} \mathcal{M}$, it holds that $\mathcal{M} \rightarrow_{\mathbb{Q}_{\geq 0}} \{\mathtt{f}\colon 1\}$.

The formulation of continuous soundness looks slightly deceiving. One might think that continuous soundness is related to 1-soundness, because it reasons about markings $\{\mathtt{i}\colon 1\}$ and $\{\mathtt{f}\colon 1\}$. Unfortunately, such a relation does not hold in general. It is easy to construct a net that is 1-sound but not continuously sound, and likewise a net that is continuously sound, but not 1-sound. However, continuous soundness is a necessary condition for generalised soundness.

Theorem 22: [27, Theorem 1] †

If a workflow net N is continuously unsound, then it is also generalised unsound.

Proof: Appendix C, Page 8

A proof of the statement is straightforward. It relies on the fact that, given markings $\mathcal{M}, \mathcal{M}'$, it holds that $\mathcal{M} \xrightarrow{*}_{\mathbb{Q}_{\geq 0}} \mathcal{M}'$ if and only if there exists $b \in \mathbb{N}_{>0}$ such that $b \cdot \mathcal{M} \xrightarrow{*} b \cdot \mathcal{M}'$.

Next, we establish the complexity of continuous soundness.

Theorem 23: [27, Theorem 2] †

Continuous soundness is coNP-complete.

Proof: Appendix C, Page 9, continued on Page 26

Membership in coNP is established by reducing continuous soundness to checking *inclusion* in continuous Petri nets, which is known to be coNP-complete [18, Proposition 4.6]. Intuitively, we check whether the set $\{\mathcal{M} \mid \{\mathtt{i}\colon 1\} \xrightarrow{*}_{\mathbb{Q}_{\geq 0}} \mathcal{M} \text{ in } N\}$ is

included in the set $\{\mathcal{M} \mid \{\mathtt{f}\colon 1\} \xrightarrow{*}_{\mathbb{Q}_{\geq 0}} \mathcal{M} \text{ in } N'\}$. Here, N' is N where we flip the direction of all arcs in the flow function, that is, $f'(a,b) = f(b,a)$.

Hardness is shown by a reduction from checking whether a Boolean formula given in disjunctive normal form with three variables per clause (3-DNF) is a tautology, which is a canonical coNP-complete problem. The reduction is adapted from [109, Corollary 1], where it is used to show that deciding 1-soundness in acyclic and bounded workflow nets is coNP-complete. We modify the reduction slightly to work for continuous soundness. Thus, given a formula φ in 3-DNF, we aim to construct a workflow net N_φ that is continuously sound if and only if φ is a tautology. The proof is similar to the proof of the original reduction, but requires a more careful analysis of N_φ.

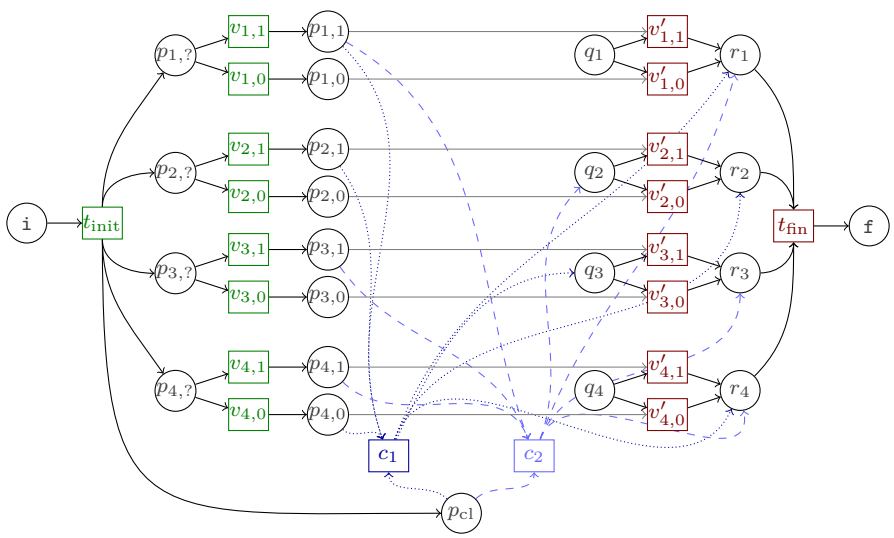


Figure 6.1: A sketch of the reduction from tautology of 3-DNF formulas to continuous soundness in Petri nets. We show a workflow net N_φ such that N_φ is continuously sound iff $\varphi = (x_1 \wedge x_2 \wedge \neg x_4) \vee (x_1 \wedge x_3 \wedge x_4)$ is a tautology. Places and transitions contain their names. Arcs corresponding to the first and second clauses are respectively dotted and dashed. Transitions colored green are responsible for guessing assignments; transitions in shades of blue are responsible for checking whether the formula is satisfied; and transitions colored red are responsible for cleaning up the net.

A sketch of the construction is shown in Figure 6.1. The idea of behind it is as follows. Let us focus on the case where transitions are only fired fully, that is, with weight 1, as it allows understanding the reduction more easily. Then, we can view

runs in N_φ as proceeding in three phases:

— The run guesses an assignment to the variables of φ.

— The run checks whether any clause of φ is satisfied by the assignment, thus whether the formula is satisfied.

— The run cleans up the net and reaches the correct final marking.

These phases are distinguished by distinct colors in Figure 6.1. Let us focus on the case where transitions are always fired with weight 1 first. The green transitions serve to guess an assignment for each variable, putting a token in $p_{j,1}$ if variable x_j is assigned true, and a token in $p_{j,0}$ if it is assigned false. Then, the blue transitions serve to check whether a clause of φ is enabled. Each transition c_j corresponds to one clause c_j, and consumes tokens from places according to the literals in c_j. Thus, if no clause is satisfied, and φ is not a tautology, we can reach a deadlock by guessing an assignment that satisfies no clause, thus enables no transition c_j. If some c_j is enabled, after firing it, the assignment places $p_{i,?}, p_{i,1}$ and $p_{i,0}$ will be emptied and place r_i will carry a token for each variable x_i occurring in c_j, For each variable $x_{i'}$ which does not appear in c_j, the assignment places corresponding to $x_{i'}$ will still collectively hold one token. No matter what assignment we guess for $x_{i'}$, we can move the token to place $r_{i'}$ by firing transition $v'_{i',1}$ or $v'_{i',0}$. Finally, we reach a marking that marks all places r_i, and fire t_{fin} to reach $\{\mathtt{f}\colon 1\}$. Showing that the reduction is correct even when we allow firing transitions partially is more technical. It involves showing that in any run, for each variable x_i certain invariants hold on the numbers of tokens in the places corresponding to x_i.

Next, we show that not only is $\mathbb{Z}$-boundedness necessary for generalised soundness, it is also necessary for continuous soundness.

Proposition 24: [27, Proposition 7] †

Let N be a nonredundant workflow net and $\mathcal{M}$ be a marking. If N is $\mathbb{Z}$-unbounded from $\mathcal{M}$, then N is not continuously sound.

Proof: Appendix C, Page 10

Conversely, continuous soundness is not necessary for $\mathbb{Z}$-boundedness. Thus, we can order our two necessary conditions by coarseness: $\mathbb{Z}$-unboundedness is necessary

for continuous soundness, which in turn is necessary for generalised soundness.

6.2 Structural Soundness

For structural soundness, helpful insights were presented in [36]. A straightforward necessary condition stated there is *structural quasi-soundness*. We say that a workflow net N is *k-quasi-sound* if $\{\mathtt{i}\colon k\} \xrightarrow{*} \{\mathtt{f}\colon k\}$. Further, N is *structurally quasi-sound* if there exists a $k \in \mathbb{N}_{>0}$ such that N is k-quasi-sound. Clearly, k-quasi-soundness is a necessary condition for k-soundness, and thus structural quasi-soundness is a necessary condition for structural soundness [27, Proposition 8]. Our second observation is less straightforward. Let us define k_N as the smallest number such that N is k-quasi-sound or ∞ if N is not structurally quasi-sound. Then we observe that structural soundness requires k_N-soundness [36, Theorem 2.1].

In [36, Lemma 2.1], the problem of deciding structural quasi-soundness and computing k_N is reduced to Petri net reachability [36, Lemma 2.1]. We make the key observation that it is not necessary to invoke Petri net reachability, but that we can instead use continuous reachability to decide structural quasi-soundness.

Proposition 25: [27, Reformulation of Proposition 9] †

Let N be a workflow net. N is structurally quasi-sound if and only if $\{\mathtt{i}\colon 1\} \xrightarrow{*}_{\mathbb{Q}_{\geq 0}} \{\mathtt{f}\colon 1\}$.

Proof: Appendix C, Page 11

Further, it is not only possible to decide structural quasi-soundness via continuous reachability, but we can use reachability relaxations to obtain lower bounds on the size of k_N. Using $\mathbb{Z}$-reachability, let us define $k_{N,\mathbb{Z}}$ as the smallest $k \in \mathbb{N}$ such that $\{\mathtt{i}\colon k\} \xrightarrow{*}_{\mathbb{Z}} \{\mathtt{f}\colon k\}$, or ∞ if there is no such k.

We similarly want to define $k_{N,\mathbb{Q}_{\geq 0}}$ by making use of continuous reachability. However, we need an additional constraint on the run from $\{\mathtt{i}\colon k\}$ to $\{\mathtt{f}\colon k\}$, since otherwise if there exists a run π for $k > 1$, then scaling π with $1/k$ would yield a run from $\{\mathtt{i}\colon 1\}$ to $\{\mathtt{f}\colon 1\}$. To address this issue, we define $k_{N,\mathbb{Q}_{\geq 0}}$ as the smallest $k \in \mathbb{N}$ such that there exists a run $\pi = \beta_1 t_1 \ldots \beta_n t_n$ with $\{\mathtt{i}\colon k\} \xrightarrow{\pi}_{\mathbb{Q}_{\geq 0}} \{\mathtt{f}\colon k\}$ and such that for

each $t \in T$, $\left(\sum_{i \in \{x \in 1,\ldots,n \mid t_x = t\}} \beta_i\right) \in \mathbb{N}$. Intuitively, π can still fire transitions partially, but we require that the total amount by which each transition is fired is a natural number. We can show that $k_{N,\mathbb{Z}} \leq k_{N,\mathbb{Q}_{\geq 0}} \leq k_N$, and similar to our approximations for generalised soundness, we get a hierarchy of approximations for k_N. Further, we show that it is possible to compute $k_{N,\mathbb{Z}}$ by solving an integer linear program, and $k_{N,\mathbb{Q}_{\geq 0}}$ by solving an existential mixed linear arithmetic formula [27, Proposition 10].

6.3 Free-Choice Workflow Nets

Recall that free-choice Petri nets, as introduced in Chapter 4, form a subclass of Petri nets that is more amenable to analysis for some problems such as boundedness and liveness. The restrictions of workflow nets can be naturally applied to free-choice nets, and we end up with the class of free-choice workflow nets. It turns out that many processes in practice can be modelled naturally as free-choice workflow nets [2], thus avoiding the extensive computational complexity of checking soundness in the general case, as shown in Chapter 5. It is known that classical soundness (as well as 1-soundness) in free-choice workflow nets can be checked in polynomial time [3, Theorem 12]. Further, it is known that 1-soundness implies generalised soundness [90, Corollary 4.3]. As generalised soundness also trivially implies 1-soundness, the two notions are thus equivalent for free-choice nets. However, no such equivalence is known for structural soundness. We show that indeed all four notions of soundness we work with in this thesis are equivalent.

Theorem 26: [27, Theorem 3] †

Let N be a free-choice workflow net. The following are equivalent:

— N is 1-sound,

— N is generalised sound,

— N is structurally sound,

— N is continuously sound.

Proof: Appendix C, Page 12

The proof is technical and crucially relies on the free-choice property. We reason about the sets of transitions that are live and quasi-live from certain markings, and show that 1) from any marking $\mathcal{M}$, it is possible to reach a marking $\mathcal{M}'$ such that all transitions that are quasi-live are also live, and 2) that the same transitions are quasi-live and live from multiples of $\mathcal{M}'$. Roughly, this allows us to argue that witnesses to 1-unsoundness are equivalently witnesses for k-unsoundness for any k and for continuous unsoundness, and vice versa.

It is thus the case that for free-choice nets, one can use any procedure for any of the four types to decide soundness.

6.4 Experimental Evaluation

We implemented our approaches for generalised and structural soundness in C#, as an extension to our tool FastForward [21]. To semi-decide generalised soundness, we implemented procedures for checking $\mathbb{Z}$-boundedness and continuous soundness, utilizing the SMT solver Z3 [86]. For structural soundness, we implemented a procedure for computing $k_{N,\mathbb{Q}_{\geq 0}}$ and deciding structural quasi-soundness, also utilizing Z3. We evaluate our prototype implementations on multiple benchmark suites. All experiments were run on the same machine, equipped with an 8-Core Intel® Core™ i7-7700 CPU @ 3.60GHz with Ubuntu 18.04. We limited memory to ∼8GB, and time to 120s for each instance.

6.4.1 Free-Choice Nets

Our first benchmark suite is based on a standard benchmark suite containing 1386 free-choice Petri nets. It was originally proposed in [54], and has been used to evaluate various procedures on workflow nets since then [33, 55, 84]. We utilize the benchmark suite as a basis to generate larger, *chained* instances by combining nets from the original suite. The resulting nets are still free-choice, and a chained instance is sound if and only if the original instances it was built from were sound. This allows us to evaluate our approach for scalability on large instances. We are interested in checking 1-soundness on these instances. As they are free-choice, note that this

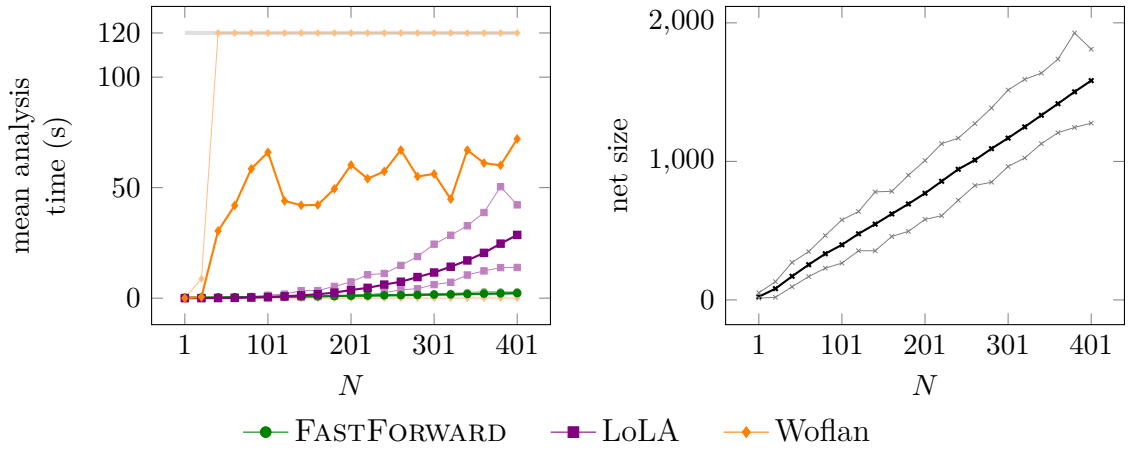


Figure 6.2: The results of experiments on chained free-choice instances. The x-value denotes the number N of chained nets. Dark thick lines denote the mean, and light thin lines of the same color denote the minimum and maximum, respectively. For Woflan, the minimum line is slightly below the line of this work. For this work, the minimum and maximum lines are very close to the mean. *Left:* The y-value denotes time for checking soundness of the 20 nets for each N. Marks on the gray line at $120s$ denote timeouts. *Right:* The y-value denotes the size of generated nets, that is, the number of places and transitions.

is equivalent to checking either of structural soundness, generalised soundness or continuous soundness.

We compare three approaches for deciding soundness of free-choice nets: Our approach for deciding continuous soundness via SMT solving, the established Petri net model checker LoLA [106] and the workflow analysis tool Woflan [104]. Both Woflan and LoLA can directly check 1-soundness.

The results are shown in Figure 6.2. On the left-hand side, the y-axis shows the analysis time, and the x-axis shows the number N of chained nets. We use instances with $N \in \{1, 21, 41, \ldots, 401\}$, and have 20 instances for each N, for a total of 420 instances.

One can think of N as a size parameter, and to put this number in context, we show the size of the resulting nets on the right-hand side of Figure 6.2, where the size is the number of transitions plus the number of places.

For small instances ($N = 1$ and $N = 21$), all tools perform well. For larger in-

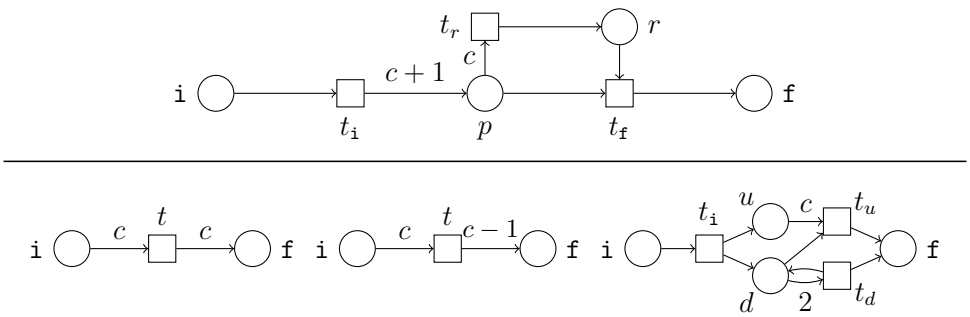


Figure 6.3: The synthetic families of workflow net instances used in our experiments. *Top*: A workflow net N_c that is c-unsound and k-sound for all $k \in [1..c-1]$. *Bottom*: Three families of instances. *Bottom left:* $N_{\text{sound-}c}$ is quasi-sound and ℓc-sound for all $\ell \in \mathbb{N}_{>0}$. *Bottom center:* $N_{\neg\text{quasi-}c}$ is not structurally quasi-sound. *Bottom right:* $N_{\neg\text{sound-}c}$ is ℓc-quasi-sound for all $\ell \in \mathbb{N}_{>0}$, but not structurally sound.

stances, Woflan times out frequently. On moderately sized instances, LoLA performs best, and is faster than our approach for $N \leq 101$. However, it seems that our approach scales better, and for $N = 401$, our approach takes a mean of $2.28s$, compared to over $30s$ for LoLA. Thus, we conclude that for large free-choice nets, deciding soundness via continuous soundness can outperform existing approaches based on deciding 1-soundess via state-space exploration.

6.4.2 Synthetic Instances

We were interested in having a wider variety of instances, in particular non-free-choice instances, where the different notions of soundness do not collapse. We introduce four families of synthetic instances, one on which generalised soundness is interesting, and three on which structural soundness is interesting. The families are purposely simple to understand, but have large state spaces, and are thus difficult to analyse with state-space exploration. All families are parameterized by a size parameter c. The description of the families uses arcs with weights, but to create challenging instances, we replace weighted arcs with larger subnets [27, A.5], while preserving (quasi-)soundness.

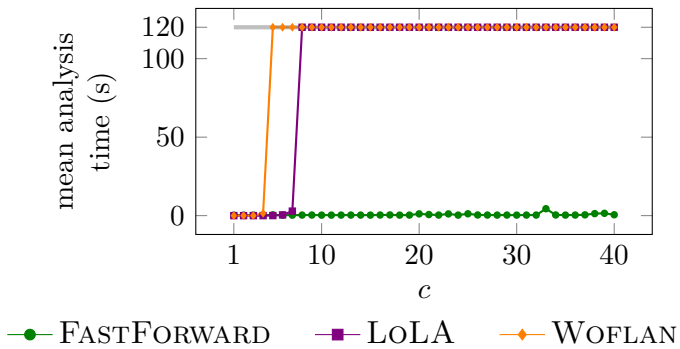


Figure 6.4: Results of our experiments for generalised soundness. We show the time to check generalised soundness of N_c for different values of c. Marks on the gray line at $120s$ denote timeouts.

Generalised Soundness. For generalised soundness, we introduce the family shown at the top of Figure 6.3. For each $c \in \mathbb{N}_{>0}$, N_c is k-sound for $k \leq c - 1$, and c-unsound. Thus, for large parameter values, this family makes it challenging to decide generalised soundness using the naive approach of iteratively checking 1-soundness, 2-soundness, ... We compare this approach, using LoLA and Woflan to perform state-space exploration to check soundness in each iteration step, with our proposed approach of checking continuous soundness. Notice that N_c is continuously unsound, thus our approach of continuous soundness is precise on this family, and proves generalised unsoundness. Figure 6.4 shows the result of the comparison. For small parameter values, state-space exploration works well, but quickly becomes intractable for larger c. For $c \geq 14$, Woflan cannot even check 1-soundness in the time limit, while LoLA can check 1- and 2-soundness up to $c \leq 28$, but cannot check 2-soundness for larger c. However, continuous soundness remains easy to verify even for $c = 40$ and beyond. At most $1s$ is needed on 34 out of 40 instances, with a mean of $0.6s$.

Structural Soundness. Recall that our decision procedure for structural soundness consists of deciding structural quasi-soundness and computing $k_{N,\mathbb{Q}_{\geq 0}}$ via continuous reachability. We want to test our procedure both on instances that are structurally quasi-sound and on instances that are not. We introduce three families of instances for which structural soundness appears challenging. The families are shown at the

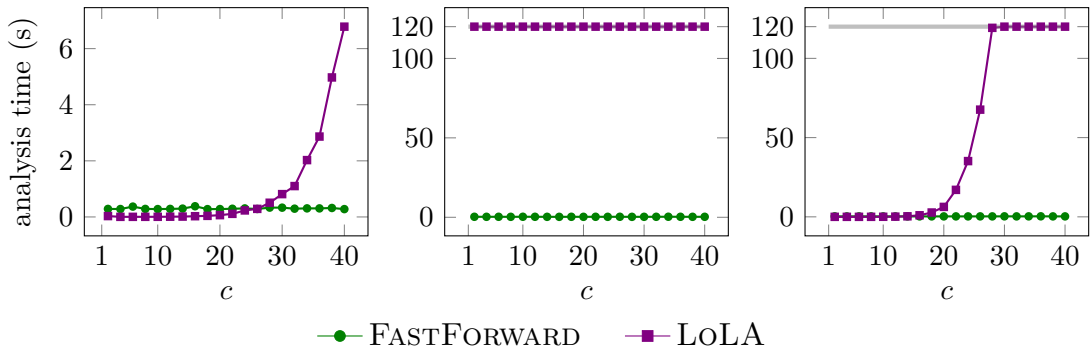


Figure 6.5: Results of our experiments for quasi-soundness. We show the time taken vs parameter c for checking structural quasi-soundness using the reduction to reachability, and utilizing our approach to compute $k_{N,\mathbb{Q}_{\geq 0}}$, for each of the three families at the bottom of Figure 6.3: $N_{\text{sound-}c}$ (*left*),$N_{\neg\text{quasi-}c}$ (*center*), $N_{\neg\text{sound-}c}$ (*right*). Note that the axis ranges differ. Marks on the gray line at $120s$ denote timeouts.

bottom of Figure 6.3, and cover different cases of quasi-soundness and soundness. We compare our approach to LoLA, which we use to decide structural quasi-soundness using a reduction Petri net reachability given in [36, Lemma 2.1]. Since Woflan does not support quasi-soundness directly, we leave it out of the comparison here. The results of the comparison are shown in Figure 6.5. For small instances, LoLA performs well on many instances, but does not scale as well as our approach for larger instances. In particular, for nets that are not structurally quasi-sound, LoLA generally times out, as it tries to explore an infinite state-space. On the other hand, continuous reachability performs well even for large c, and for $N_{\neg\text{quasi}-c}$, we can check structural quasi-soundness in under $2s$ even for $c = 20\ 000\ 000$. Further, we found that for all instances of all families, $k_{N,\mathbb{Q}_{\geq 0}} = k_N$, that is, our approximation is exact.

Chapter 7

Continuous One-Counter Automata

While Petri nets have large expressive power and are convenient for modelling many systems, we are interested in looking at some other variations of counter systems. These are models that combine some number of counters, each containing a number, with a control state. To motivate our forthcoming work on continuous one-counter automata, let us introduce in the following the models of (continuous) vector addition systems with states and counter automata.

A model that is equivalent to Petri nets is that of *Vector Addition Systems*, or VAS for short. Essentially, a VAS of dimension d is a set of transition vectors, each of dimension d. Dimensions correspond to places, and transition vectors correspond to Petri net transitions. Extending the model further, we can introduce states to VAS, thus obtaining the model of *vector addition systems with states*, or VASS for short. In that model, transitions have an effect, and additionally specify a source and target state. Like VAS, VASS are equivalent in modelling power to Petri nets [62, 66]. This equivalence does not, however, hold when we consider the continuous semantics. *Continuous VASS*, or CVASS for short, are VASS where we allow transition firings to scale the effect by a non-zero rational between zero and one, similar to the continuous semantics we defined for Petri nets in Section 2.3. While reachability in continuous Petri nets is P-complete [58], the same problem is NP-complete in CVASS [19]. The question is also interesting when we restrict the number of dimensions. For 2-VASS,

that is, VASS with 2 dimensions, reachability is PSPACE-complete [16, Theorem 2]. For CVASS with fixed dimension, the NP-hardness from the case with arbitrary dimension remains firm above 1 dimension, where the problem can be solved in polynomial time [8].

Another angle of applying continuous semantics to models similar to Petri nets is starting from so-called counter automata. A counter automaton is specified as a dimension d, a set of states and a set of transitions, almost identical to the model of VASS. A configuration contains a state, and d numbers that indicate the counter values. What differentiates the models is that in counter automata, we allow also tests on the counter values. For instance, a transition might be restricted to be only enabled if the first counter is equal to zero. This power makes many questions on this model undecidable [85]. In the literature, it is therefore common to study counter automata with restrictions. One can think of VASS as a restriction of counter automata that disallows tests on counter values. Another interesting restriction is to focus on the case with only a single counter, called *One-Counter Automata*, or OCA for short.

One-counter automata have found practical usage, for example in the verification of programs with lists [29] or the validation of XML streams [35]. The complexity of many problems has also been studied in the literature. It is known that reachability is PSPACE-complete when counter updates are encoded in binary [56], that is, when we have a succinct encoding for counter updates that takes only $\log(n)$ space to store a counter update of size n. Thus, it is natural to search for relaxations for the problem, such as applying continuous semantics to the model of OCA.

This chapter presents results from [24]. The paper resulted from joint work with my supervisors, which was merged with the work of a team independently investigating the same problem. The paper was published in the proceedings of LICS'21, and an extended version containing proofs missing from the conference publication was published in the TOCL journal [23]. Our contributions are to define the novel model of *continuous one counter automata*, or *COCA* for short, which approximates OCA. The model allows guarding states by upper and lower bound tests on the counter value. We show that

— reachability in COCA is in NC^2 when we only have the same *global* lower and upper bound test for each state, even when we additionally allow equality tests;

— in the general case, the problem is in polynomial time;

— for parametric counter updates and bound tests, reachability is NP-complete.

In the following, we first formally introduce the model. We then present the main ideas for our results on COCA with global bound tests with equality tests, and on parametric COCA. Afterwards, we give a short primer on continuous automata with more than one counter.

We do not dive into the second result of P-membership of reachability in the general case, as this result was chiefly the contribution of coauthors, but we will require a black box result that follows from the proof for the polynomial time algorithm.

7.1 Continuous One-Counter Automata (COCA)

Let us write $\mathcal{I}$ to mean the set of all intervals (closed or open) over $\mathbb{Q}$, where we allow endpoints to come from $\mathbb{Q} \cup \{\infty, -\infty\}$. For example, $(5, 8] \in \mathcal{I}$, and it denotes the set of numbers $\{v \in \mathbb{Q} \mid 5 < v \leq 8\}$. We also allow unbounded intervals, for example $[2, \infty)$ denotes the set $\{v \in \mathbb{Q} \mid 2 \leq v\}$. Importantly, the empty interval $\emptyset$ is allowed, and we can represent it, for example, as $(8, 4]$ or $[3, 3)$. We also naturally allow intervals with a single element, for example $[5, 5] = \{5\}$. Further, let us write $\overline{S}$ to denote the *interval closure* (or closure for short) of a set S. Formally, we define this as the interval $[\inf S, \sup S]$. For example, $\overline{(3, 4)} = [3, 4]$ and $\overline{[3, 5) \cup (5, 6)} = [3, 6]$.

A *continuous one-counter automaton* (or COCA for short) is a triple $C = (Q, T, \tau)$, where

— Q is a finite set of *(control) states*,

— $T \subseteq Q \times \mathbb{Z} \times Q$ is a finite set of *transitions*, and

— $\tau \in \mathcal{I}$ denotes the *global guard* of the COCA.

For a transition $t = (p, w, q)$, we denote as $\mathrm{src}(t) = p$ the *source* of t, as $\boldsymbol{\delta}_t = w$ the *effect* of t, and as $\mathrm{dest}(t) = q$ the *destination* of t. Let us define a *path* of C as a path of the graph underlying C, defined naturally with states as vertices and transitions as edges between them.

A configuration of C is a pair (p, v), where $p \in Q$ and $v \in \mathbb{Z}$. For ease of notation, we write $p(v)$ to mean a configuration (p, v). We define the step relation for a single

transition $t = (p, w, q) \in T$ as $\xrightarrow{t}$. For two configurations (p, v) and (q, v'), it holds that $(p, v) \xrightarrow{t} (q, v')$ if and only if

— $v, v' \in \tau$, and

— there exists $\alpha \in (0, 1]$ such that $v' = v + \alpha w$.

We naturally extend from the step relation for a single transition to the step relation over all transitions, which we denote by $\rightarrow = \bigcup_{t \in T} \xrightarrow{t}$. We further generalize it to sequences of transitions, that is, we define $\xrightarrow{t_1 t_2}$ such that $p(a) \xrightarrow{t_1 t_2} q(b)$ if and only if there exists $p'(a')$ such that $p(a) \xrightarrow{t_1} p'(a') \xrightarrow{t_2} q(b)$. Lastly, we denote as $\xrightarrow{*}$ the *reachability relation*. We define it such that $p(a) \xrightarrow{*} q(b)$ if and only if there exists a transition sequence $t_1 t_2 \cdots t_n$ such that $p(a) \xrightarrow{t_1 t_2 \cdots t_n} q(b)$.

We say that a *run* σ from a configuration $p(v)$ to a configuration $q(v')$ is a sequence $\alpha_1 t_1 \ldots \alpha_n t_n$, where for all i, $\alpha_i \in (0, 1]$ and $t_i \in T$ and such that there exists a sequence of configurations $q_0(v_0), q_1(v_1), \ldots, q_{n-1}(v_{n-1}), q_n(v_n)$, where

— $q_0(v_0) = p(v)$,

— $q_n(v_n) = q(v')$,

— for all $0 < i \leq n$, $\mathrm{src}(t_i) = q_{i-1}, \mathrm{dest}(t_i) = q_i$, and $v_{i-1} + \alpha_i \boldsymbol{\delta}_{t_i} = v_i$.

Furthermore, we say that σ is *admissible* if $v_0, v_1, \ldots, v_{n-1}, v_n \in \tau$. COCA are equivalent to 1-CVASS when we set $\tau = [0, \infty)$.

Next, we extend COCA where instead of one global guard, we allow different guards in different states. Formally, we define a *guarded COCA* as a triple $C = (Q, T, \tau)$, where Q and T are defined as for COCA, and $\tau \colon Q \rightarrow \mathcal{I}$ assigns an interval to each state. Configurations and runs are defined as for COCA; however, the criterion for admissibility of a run is different. We say that a run σ is *admissible* if and only if each of its configurations $p(v)$ satisfies τ, that is, $v \in \tau(q)$. It is easy to see that guarded COCA indeed generalize COCA. We can view a COCA as a guarded COCA where τ has the same value for each state.

Now, we are ready to introduce our last model, where we extend guarded COCA with parameters. For a set X we define the set $\mathcal{I}_X$ of *parameterised intervals over* X as the set of intervals where endpoints belong to $Q \cup \{-\infty, \infty\} \cup X$. Note that elements of X are not numbers, but instead symbolic variables. A *parametric, guarded COCA* (or just parametric COCA for short) is a tuple $C = (Q, T, \tau, X)$, where

— Q, as for COCA, is a finite set of states,
— X is a finite set of parameters,
— $T \subseteq Q \times (\mathbb{Z} \cup X) \times Q$ is a finite set of transitions, and
— $\tau : Q \to \mathcal{I}_X$.

A *valuation* of X is a mapping $\mu\colon X \to \mathbb{Q}$. We write $C^\mu = (Q, T^\mu, \tau^\mu)$ to denote the guarded COCA we obtain when replacing each occurrence of a parameter with its value assigned by μ. Then we say that there is a run from $p(v)$ to $q(v')$ in C if there exists a valuation μ such that there is a run from $p(v)$ to $q(v')$ in C^μ.

We similarly allow intervals to be further specified by valuations: For an interval $I \in \mathcal{I}_X$, we denote by I^μ the interval where each occurrence of a parameter x from X in I is replaced by its value assigned by μ, that is, $\mu(x)$.

For all types of COCA, we define the *post set* of a value a with respect to a path π. Formally, we write

$$\mathrm{Post}_\pi(a) = \{b \mid \mathrm{src}(\pi)(a) \xrightarrow{\pi} \mathrm{dest}(\pi)(b)\}.$$

We also define the post set of a set of paths S with common source and destination states as the union of the post sets of the paths in S, that is,

$$\mathrm{Post}_S(a) = \bigcup_{\pi \in S} \mathrm{Post}_\pi(a).$$

For ease of notation, we write $\mathrm{Post}_{p,q}(a)$ to mean the post set of a with respect to all paths between two states p and q, which we denote as

$$\mathrm{Paths}_{p,q} = \{\pi \mid \pi \text{ is a path from } p \text{ to } q\}.$$

Formally, $\mathrm{Post}_{p,q}(a) = \mathrm{Post}_{\mathrm{Paths}_{p,q}}(a)$.

Further, we define the *enabledness set* of a path π as the set

$$\mathrm{enab}(\pi) = \{a \in \mathbb{Q} \mid \mathrm{Post}_{\mathrm{src}(\pi),\mathrm{dest}(\pi)}(a) \neq \emptyset\}.$$

We extend the definition of the enabledness set to sets of paths by taking the union of the individual enabledness sets. More formally, for a set S of paths, we have that

$\mathrm{enab}(S) = \bigcup_{s \in S} \mathrm{enab}(s)$.

To conclude the formal definitions, let us give a brief recap of the models we have introduced. A COCA is a 1-CVASS, that is, CVASS with one dimension, where we may additionally have global lower and upper bounds. A guarded COCA is a COCA where, instead of global lower and upper bounds, we may have a different lower and upper bound in each state. Lastly, a parametric, guarded COCA is a guarded COCA where we allow guards and transitions to depend on a set of parameters.

7.2 Reachability in COCA is in NC^2

In this section, let us characterize the complexity of the reachability problem for COCA.

Definition 8: The Reachability Problem for COCA †

Given a COCA $C = (Q, T, \tau)$ and two configurations $p(v)$ and $q(v')$, does there exist an admissible run from $p(v)$ to $q(v')$?

We will show the following theorem:

Theorem 27: [23, Theorem 18] †

Reachability for COCA is in NC^2.

Our aim is, essentially, to derive a procedure that computes a succinct representation of $\mathrm{Post}_{p,q}(v)$ for any given p, q and v. Then, to answer a reachability query, we simply need to compute the representation of the post set and check for membership of the target value in it.

We are going to show the result in 4 steps. We show

1. that it is possible to check in NC^2 whether $\mathrm{Post}_{p,q}(v)$ is empty;
2. that if $\mathrm{Post}_{p,q}(v)$ is nonempty, then $\mathrm{Post}_{p,q}(v)$ is equal to $\overline{\mathrm{Post}_{p,q}(v)}$, except they may differ in the three points $\inf \mathrm{Post}_{p,q}(v)$, v and $\sup \mathrm{Post}_{p,q}(v)$;
3. how to compute $\overline{\mathrm{Post}_{p,q}(v)}$ in NC^2; and lastly
4. how to check membership of the three remaining points, that is, how to check whether $\inf \mathrm{Post}_{p,q}(v), v, \sup \mathrm{Post}_{p,q}(v) \in \mathrm{Post}_{p,q}(v)$.

7.2.1 Checking $\mathbf{Post}_{p,q}(v)$ for emptiness

We give a sufficient and necessary condition for $a \in \text{enab}(\pi)$ to hold. In the following, let $first(\pi)$ denote the index of the first nonnegative rule on π, and let $last(\pi)$ denote the same for the last nonnegative rule on π. Further, for an index $i \leq |\pi|$, we denote as π_i the i-th transition on π.

Lemma 28: [23, Lemma 3] †

Let $a \in \mathbb{Q}$ and $\pi \in \text{Paths}_{p,q}$. It holds that $a \in \text{enab}(\pi)$ if and only if $a \in \tau$ and any of the following conditions hold:

(a) $a \notin \{\inf \tau, \sup \tau\}$;

(b) $a = \inf \tau = \sup \tau, first(\pi) = \infty$;

(c) $a = \inf \tau < \sup \tau,\ first(\pi) \neq \infty \Rightarrow \boldsymbol{\delta}_{\pi_{first(\pi)}} > 0$;

(d) $a = \sup \tau > \inf \tau,\ first(\pi) \neq \infty \Rightarrow \boldsymbol{\delta}_{\pi_{first(\pi)}} < 0$.

Proof: Appendix D, Page 7

The proof is straightforward by careful analysis of the definition of enab(π).

Importantly, the conditions of the prior lemma are easy to check by looking at the path of π in the underlying graph. In NC^2, we can also reason about all short paths of the underlying graph, which is enough to obtain the following corollary:

Corollary 29: [23, Corollary 4] †

Given $a \in \mathbb{Z}$ and $p, q \in \mathbb{Q}$, deciding whether $a \in \text{enab}(\text{Paths}_{p,q})$, or equivalently $\text{Post}_{p,q}(a) = \emptyset$, is in NC^2.

Proof: Appendix D, Page 8

7.2.2 Characterizing $\mathbf{Post}_{p,q}(v)$ in relation to $\overline{\mathbf{Post}_{p,q}(v)}$

In this section, let us point out the relation between the post-set and its closure. We will prove that the two differ only in very few points.

The following short observation is straightforward by the fact that transition firings can be scaled.

Lemma 30: [23, Lemma 6] †

Let $b \in \mathrm{Post}_{p,q}(a)$. It holds that $(a, b) \subseteq \mathrm{Post}_{p,q}(a)$ and $(b, a) \subseteq \mathrm{Post}_{p,q}(a)$.

Proof: Appendix D, Page 8

For a brief intuition behind the result, consider that when going from $p(a)$ to $q(b)$, we can scale down all transitions in the run to make its effect smaller, and thus also go to any value between a and b.

From this, we obtain a characterization of $\mathrm{Post}_{p,q}(a)$ with respect to its closure:

Corollary 31: [23, Corollary 7] †

The set $\overline{\mathrm{Post}_{p,q}(a)}$ is a closed interval. Moreover, $\overline{\mathrm{Post}_{p,q}(a)} \setminus \mathrm{Post}_{p,q}(a) \subseteq \{\inf \overline{\mathrm{Post}_{p,q}(a)}, a, \sup \overline{\mathrm{Post}_{p,q}(a)}\}$.

Proof: Appendix D, Page 8

Thus, to compute the post-set, we need to compute its closure by finding the supremum and infimum, then decide membership of the three points $\{\inf \overline{\mathrm{Post}_{p,q}(a)}, a, \sup \overline{\mathrm{Post}_{p,q}(a)}\}$.

7.2.3 Identifying $\inf \overline{\mathrm{Post}_{p,q}(a)}$ and $\sup \overline{\mathrm{Post}_{p,q}(a)}$

Let us focus in the following on finding the supremum, as finding the infimum is symmetrical.

There are two cases, depending on whether from configuration $p(a)$, some cycle containing a transition with positive effect is admissible. Identifying whether this is the case can be done in NC^2, as we can, in parallel, check for each state p' whether (a) it is reachable from state p, (b) it has a cycle to itself that contains a positive transition, (c) it can reach state q.

(1) No cycle with positive effect is admissible: Then the supremum can be found by considering only enabled simple paths, which can be done in NC^2. Intuitively, we maximize the positive effect among paths, while ignoring the negative effects, since we can choose the scaling for transitions with negative effect arbitrarily small.

(2) A cycle with positive effect is admissible: Then the supremum is the supremum of the guard τ, since we can take the cycle until we hit the "ceiling" $\sup \tau$, and rescale so that we get arbitrarily close to it.

Finally, we obtain the following result:

Proposition 32: [23, Proposition 12] †

Computing $\inf \overline{\mathrm{Post}_{p,q}(a)}$ and $\sup \overline{\mathrm{Post}_{p,q}(a)}$ can be done in NC^2.

Proof: Appendix D, Page 11

7.2.4 Testing membership of a and the endpoints of $\overline{\mathrm{Post}_{p,q}(a)}$

Let us recall that we have shown that $\mathrm{Post}_{p,q}(a) \setminus \{a, \inf \overline{\mathrm{Post}_{p,q}(a)}, \sup \overline{\mathrm{Post}_{p,q}(a)}\} = \overline{\mathrm{Post}_{p,q}(a)} \setminus \{a, \inf \overline{\mathrm{Post}_{p,q}(a)}, \sup \overline{\mathrm{Post}_{p,q}(a)}\}$, and that we can compute a representation of $\overline{\mathrm{Post}_{p,q}(a)}$ in NC^2 by finding its endpoints, that is, supremum and infimum. Hence, towards our goal of computing a representation of $\mathrm{Post}_{p,q}(a)$, it remains to check membership of the three points $a, \inf \overline{\mathrm{Post}_{p,q}(a)}, \sup \overline{\mathrm{Post}_{p,q}(a)}$.

To check whether $a \in \mathrm{Post}_{p,q}(a)$, we show there are two cases of how a could be included in $\mathrm{Post}_{p,q}(a)$. The first case is that there is a path from p to q containing only transitions with effect zero; and the second case is that there exists a path containing both positive and negative transitions. The second case involves some additional technical requirements, as we need to make sure that the path is enabled, which might not be the case if a sits on the border of the global guard.

Checking whether $\sup \overline{\mathrm{Post}_{p,q}(a)} \in \mathrm{Post}_{p,q}(a)$ and $\inf \overline{\mathrm{Post}_{p,q}(a)} \in \mathrm{Post}_{p,q}(a)$ is in principle similar but more technical, and the characterizations are less straightforward.

We obtain the following result:

Proposition 33: [23, Proposition 15] †

Computing $\mathrm{Post}_{p,q}(a) \cap \{\inf \overline{\mathrm{Post}_{p,q}(a)}, a, \sup \overline{\mathrm{Post}_{p,q}(a)}\}$ can be done in NC^2.

Proof: Appendix D, Page 13

Finally, we are ready to state the main result, that is, that reachability in COCA is in NC^2.

Theorem 34: [23, Theorem 16] †

Given $a, a' \in \mathbb{Z}$ and $p, q \in Q$, the following can be done in NC^2: obtaining a representation of $\text{Post}_{p,q}(a)$ and testing whether $a' \in \text{Post}_{p,q}(a)$.

Proof: Appendix D, Page 14

7.2.5 Extending results to equality tests

This section is dedicated to introducing an interesting extension of COCA, for which our NC^2 decision procedure can be adapted. Our extension will be to allow *equality tests* on states. In terms of expressiveness, this extension fits between COCA and guarded COCA.

Formally, a *COCA with equality tests* $C = (Q, T, \tau)$ is a guarded COCA where we require τ to be the same for each state, except for states where the guard interval contains only a single number. Formally, we require that there exists an interval $\phi \in \mathcal{I}$ such that for each state $q \in Q$ with $|\tau(q)| \neq 1$, it must hold that $\tau(q) = \phi$.

Theorem 35: [23, Theorem 18] †

Reachability for COCA with equality tests is in NC^2.

Proof: Appendix D, Page 18

The intuition is that we construct a graph with one node per equality test of the COCA, as well as two additional nodes, one for the input configuration and one for the output configuration. Edges in this graph then correspond to reachability between two configurations, in a slightly adjusted copy of the input COCA where we disallow equality tests. Since the structure of the graph simulates equality tests, it holds that reachability in the graph corresponds to reachability in the COCA with equality tests. To determine whether an edge is present, it suffices to make one COCA-reachability query. This graph has at most $|Q| + 2$ nodes, so at most $(|Q| + 2)^2$ edges. Hence, to construct it, we need to make at most quadratically many reachability queries.

As these can all be made in parallel, and NC^2 allows us to have polynomially many processors running in parallel, this graph can be constructed in NC^2. Then, it remains to simply check for reachability from the node of the initial configuration to the node of the final configuration, which can be checked in $NL \subseteq NC^2$.

7.3 Reachability in parametric COCA is NP-complete

In this section, we show that reachability in parametric COCAs is NP-complete. We show hardness holds already for the special case where the underlying control structure is acyclic, and regardless of whether parameters can occur in guards, updates, or both.

Lemma 36: [23, Theorem 46] †

Reachability for parametric COCA is NP-hard, regardless of whether parameters occur only on guards, updates, or both. Additionally, the problem is still NP-hard if the control structure is acyclic.

Proof. We give a reduction from 3-SAT, that is, deciding satisfiability of a Boolean formula in conjunctive normal form with 3 literals per clause.

Let $X = \{x_1 \dots x_n\}$ be a set of variables and $\varphi = \bigwedge_{i \leq j \leq m} C_j$ a formula over X. We give two acyclic parametric COCAs $\mathcal{P}$ and $\mathcal{P}'$, both with parameters X. Each one will

(i) guess an assignment to X, and

(ii) check whether the assignment satisfies φ.

Further, $\mathcal{P}$ uses parameters only on guards and $\mathcal{P}'$ uses them only on updates.

Part (i) is achieved by composing n copies of the gadget in Figure 7.1. $\mathcal{P}$ uses the gadget on the top, $\mathcal{P}'$ uses the gadget on the bottom. Both function analogously: (1) state p_i is entered with counter value 0; (2) the counter is set to x_i; (3) membership of the counter value in $\{0, 1\}$ is checked; and (4) the counter is reset to zero upon

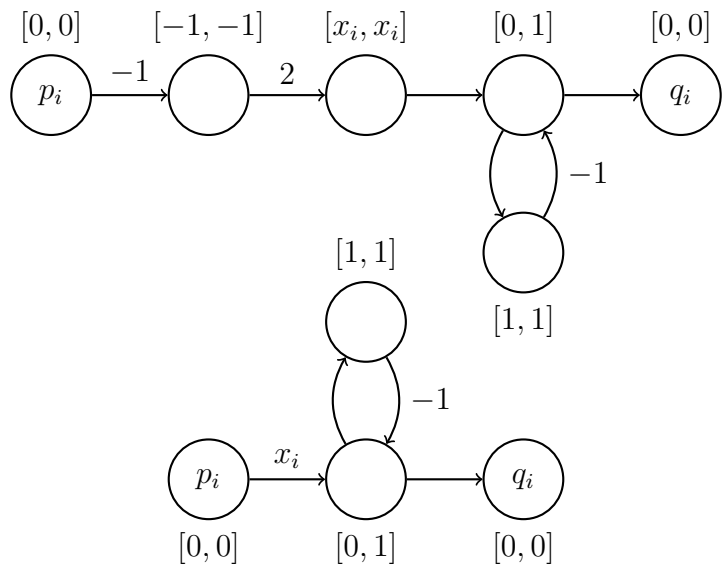


Figure 7.1: Gadgets of the reduction from 3-SAT, for guessing an assignment of variable x_i. *Top:* parameters are only used on guards. *Bottom:* parameters are only used on updates.

leaving to q_i. The only way to traverse the chain of n such gadgets from p_1 to q_n is to have $x_i \in \{0, 1\}$ for each $x_i \in X$.

Part (ii) is achieved by chaining one gadget per clause. Each gadget is similar to the one depicted in Figure 7.2 for $C_j = (x_1 \vee x_2 \vee \neg x_4)$. Again, $\mathcal{P}$ uses the gadget on the top, while $\mathcal{P}'$ uses the gadget on the bottom.

Altogether, these statements are equivalent: (1) formula φ is satisfiable; (2) there exists a valuation $\mu\colon X \to \mathbb{Q}$ such that $p_1(0) \xrightarrow{*} s_m(0)$ holds in $\mathcal{P}^\mu$; and (3) there exists a valuation $\mu'\colon X \to \mathbb{Q}$ such that $p_1(0) \xrightarrow{*} s_m(0)$ holds in $\mathcal{P}'^{\mu'}$. □

Lemma 37: [23, Theorem 43] †

Reachability for parametric COCA is in NP.

Proof: Appendix D, Page 27

Let us sketch the proof strategy here.

We reduce reachability in parametric COCA to feasibility of an existential

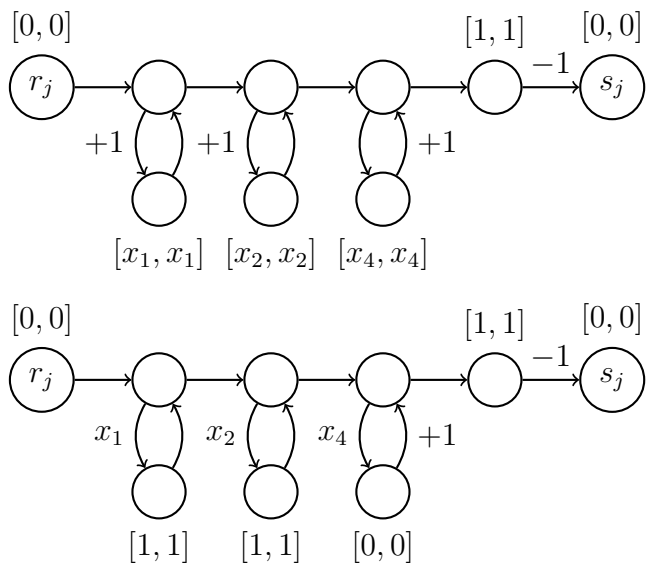


Figure 7.2: Gadgets of the reduction from 3-SAT, for checking whether clause $C_j = (x_1 \vee x_2 \vee \neg x_4)$ is satisfied. *Top:* parameters are only used on guards. *Bottom:* parameters are only used on updates.

$\mathrm{FO}(\mathbb{Q}, +, <)$ sentence. By $\mathrm{FO}(\mathbb{Q}, +, <)$, we mean the existential fragment of first-order logic over the rationals with addition and comparison. Atomic propositions of this fragment are of the form $\vec{a}^T \cdot \vec{x} \sim b$, where $\vec{x}$ is a vector of first order variables, $\vec{a} \in \mathbb{Q}^n$ and $b \in \mathbb{Q}$, and $\sim \in \{<, \leq, =\}$. Formulas can use Boolean combinations of such atomic propositions, as well as (non-negated) existential quantification over variables. Deciding whether such a formula is feasible can be done in NP [97].

We first show that if reachability holds, then there exists a witnessing path which is from a "small" linear path scheme, that is, a short series of alternation between short paths and short cycles. In the following, let C be a parametric COCA.

Lemma 38: [23, Lemma 32]

Let $p(a), q(b)$ be configurations and let μ be a valuation. If $b \in \mathrm{Post}_{p,q}(a)$ in C^μ, then there exists a path π such that $p(a) \xrightarrow{\pi} q(b)$ in C^μ where π belongs to a linear path scheme $\rho_0\theta_0^*\rho_1\theta_1^* \cdots \rho_k$ such that $k \in |Q|^{O(1)}$, $|\rho_i| \leq |Q|^{O(1)}$ for all $0 \leq i \leq k$, and finally $|\theta_j| \leq |Q|^{O(1)}$ for all $0 \leq j < k$.

Proof: Appendix D, Page 23

The result follows in a straightforward manner from techniques developed in [23, Section 4].

While this Theorem implies that reachability can be witnessed by a path from a linear path scheme, this depends on the valuation μ, which there are infinitely many of, so this is on its own not yet enough to obtain NP-membership.

Accordingly, we focus on giving small formulas that describe the reachable configurations for paths (ρ), and for cycles (θ^*). Intuitively, combining formulas for these components will allow us to give such formulas for whole linear path schemes.

7.3.1 Encoding paths

We obtain the following result:

Lemma 39: [23, Lemma 36] †

Let ρ be a path from state p to state q. There exists a $\mathrm{FO}(\mathbb{Q}, +, <)$ formula $\phi_{p(a)\xrightarrow{\rho}q(b)}$ with free variables a, b, μ which is satisfied if and only if $b \in \mathrm{Post}_{p,q}(a)$ in C^μ.

Proof: Appendix D, Page 24, Section 5.1

The result is built up from simple ingredients: It is easy to write a formula $\phi_{a \in I^\mu}(a, \mu)$ for a fixed interval $I \in \mathcal{I}_X$ that is satisfied if and only $a \in I^\mu$. From this, it is straightforward to write a formula $\phi_{p(a)\xrightarrow{t}q(b)}(a, b, \mu)$ for a fixed transition t from state p to state q which is satisfied if and only if $p(a) \xrightarrow{t} q(b)$ in C^μ.

By combining several such formulas, we can obtain the desired formula $\phi_{p(a)\xrightarrow{\rho}q(b)}$ for paths. The size of the formula grows linearly in the length of ρ, since it uses one subformula for each transition in ρ.

7.3.2 Encoding cycles

Encoding cycles is more intricate than in the case for paths. Roughly, we proceed by splitting into three cases: We iterate the cycle zero times, one time, or two or more

times. The first two cases are easy to handle, since they reduce to checking whether $a = b$ in the first case, or reusing the formula for a simple path in the second case. The third case is more challenging. Intuitively, we show that if we can iterate a cycle twice, then we can "accelerate" and iterate the cycle until some configuration on the cycle hits a guard. We obtain the following result:

Corollary 40: [23, Corollary 41] †

There exists a formula $\varphi_{\theta*}(a, b, \mu)$ which is satisfied iff $b \in \mathrm{Post}_{(\theta*)^\mu}(a)$ and which has linear size in the length of θ and the value of nonparametric updates and endpoints on θ.

Proof: Appendix D, Page 25, Section 5.2

7.3.3 Combining paths and cycles

Putting together the formulas for paths and simple cycles, we obtain formulas for linear path schemes in a straightforward manner. Given a linear path scheme $L = \rho_0\theta_0^*\rho_1\theta_1^* \cdots \rho_k$, we invoke the results from the last two sections to obtain formulas $\varphi_{\rho_0}, \varphi_{\theta_0^*}, \ldots$, and define

$$\varphi_L(a, b, \mu) = \exists a_0, a_0', \ldots, a_k, a_k' : (a_0 = a) \wedge (a_k' = b)$$
$$\wedge\, \varphi_{\rho_0}(a_0, a_0', \mu) \wedge \varphi_{\theta_0^*}(a_0', a_1, \mu) \wedge \cdots \wedge \varphi_{\rho_k}(a_k, a_k', \mu) \text{ [23, Lemma 42].}$$

Again per the results from previous sections, the size of φ_L is polynomial in the size of L.

Thus, we obtain NP membership for reachability in parametric COCAs. To check whether $p(a) \xrightarrow{*} q(b)$, we (1) guess a linear path scheme L, (2) construct φ_L, and (3) check whether $\exists\mu : \varphi_L(a, b, \mu)$ holds. This is an existential linear formula, which can be checked in NP [19], thus we are done.

Theorem 41: [23, Theorem 43] †

The existential reachability problem for parametric COCAs is in NP.

Proof: Appendix D, Page 27

7.4 Continuous Semantics of Automata with more Counters

A gap in our work is the treatment of continuous automata with more than one counter.

It is known that reachability in CVASS with zero-tests, that is, testing for equality with zero, is undecidable in 4 dimensions [19]. The proof works via a reduction from reachability in 2-VASS with tests for equality with zero. This problem is known to be undecidable [85]. Two of the four counters of the CVASS are used to store auxiliary information, and this information allows the two remaining counters to be forced to behave in a discrete manner.

One relatively unsurprising result, which to the knowledge of the author has not been proven in the literature, is that the reduction can be adapted to work for 3-CVASS with *equality tests*. Note that, in contrast to the case of 4-CVASS, where zero-tests suffice, this reduction needs tests for equality with two distinct numbers.

Theorem 42 †

Reachability for 3-CVASS with equality tests is undecidable.

Proof. We reduce from reachability in 2-VASS with zero tests. Let V be a 2-VASS with equality tests, and let $p(a_1, a_2)$ and $q(b_1, b_2)$ be configurations. Then we give a 3-CVASS with equality tests V' such that $p(a_1, a_2) \xrightarrow{*} q(b_1, b_2)$ in V if and only if $p(a_1, a_2, 0) \xrightarrow{*} q(b_1, b_2, 0)$ in V'.

First, let us simply expand V with one new counter c_3. Zero-tests of V are kept as-is, since V' also allows zero-tests. Other transitions are transformed, as depicted in Figure 7.3. Intuitively, there are two possibilities for taking the transition: either

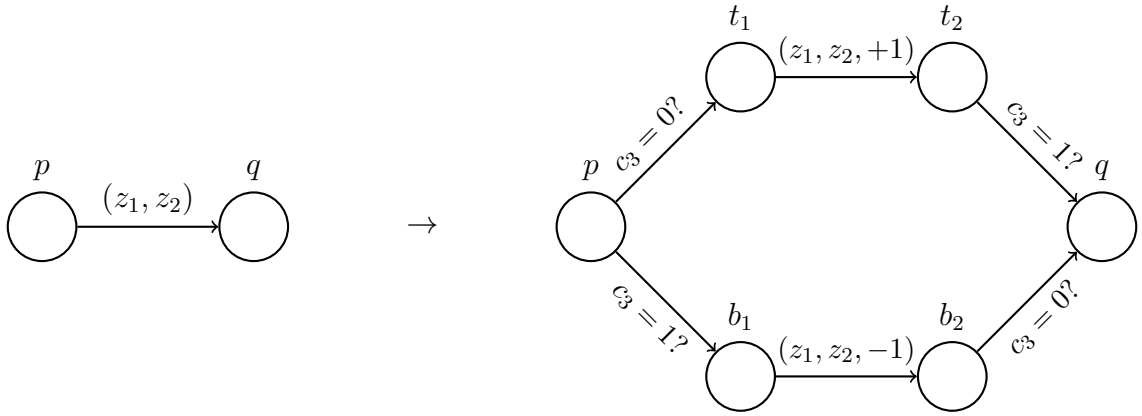


Figure 7.3: A gadget for transforming a two-counter machine with zero tests to a 3-CVASS with tests for equality with 0 and 1.

c_3 is currently 0, then the transition adds 1 to it and afterwards, checks for equality with 1; or c_3 is currently 1, then the transition subtracts 1 and subsequently checks for equality with 0.

Lastly, since this procedure means c_3 will alternate between 0 and 1 after each transition, we add a transition to each state (except for the auxiliary ones introduced when transforming transitions) which simply decrements c_3 by 1. Then $p(a_1, a_2) \xrightarrow{*} q(b_1, b_2)$ in V if and only if $p(a_1, a_2, 0) \xrightarrow{*} q(b_1, b_2, 0)$ or $p(a_1, a_2, 0) \xrightarrow{*} q(b_1, b_2, 1)$ in V', and we are done. □

It remains an open question whether, when only zero-tests are available, the reachability problem is undecidable for 3-CVASS. Both decidability and undecidability seem possible for this problem. Essentially, it is clear that 3-CVASS with zero-tests can be used to simulate a counter machine with one counter and zero tests, by using two counters to store auxiliary information. However, there may be a way to use the two auxiliary counters of this construction in a more concise way, in order to simulate a counter machine with zero tests and "1 1/2" counters.

A sketch of one interesting construction that highlights the expressiveness of 3-CVASS with zero-tests is shown in Figure 7.4. As an example, it shows how we can achieve multiplication of counter c_1 with a constant, in this example 3. The construction uses only two counters, which is crucial, as, for example, the reduction from [19] uses the invariant that one auxiliary counter always holds a known value to force

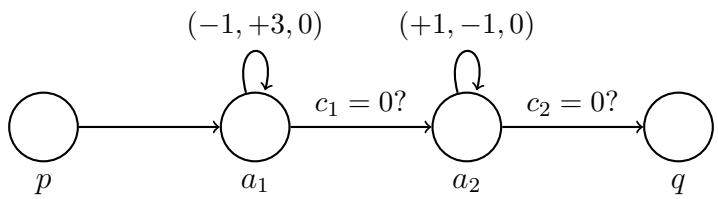


Figure 7.4: A gadget that moves from state p to state q and multiplies the first counter by 3 while preserving the value stored in the third counter.

transitions to be applied in a discrete manner. Similarly, for 3-CVASS, this means multiplication can be achieved without losing this known, auxiliary value. Decidability for this problem is thus open, and proving decidability or undecidability could lead to interesting techniques that yield insights into continuous counter automata.

Conclusion

In this thesis we have studied several problems relating to the algorithmic verification of counter systems.

We have given an efficient, practical algorithm for reachability and coverability problems in Petri nets. It is based on guided search and uses various notions of relaxed reachability in Petri nets to obtain heuristics. We have implemented the algorithm in a novel tool called FastForward, and have evaluated it positively against state-of-the-art approaches for reachability and coverability.

For workflow nets, our focus is on verifying the correctness notion of soundness, and in particular three variants called k-soundness, generalised soundness and structural soundness. We establish the exact complexities of all three problems. It turns out that k-soundness and structural soundness are EXPSPACE-complete, while generalised soundness is surprisingly PSPACE-complete, thus computationally easier.

We also provide algorithms designed to verify generalised and structural soundness in practice. Previously proposed algorithms for the two problems work by reductions to reachability in Petri nets. Our proposed algorithms are instead based on insights from the study of reachability relaxations. For generalised soundness, our approach is a semi-procedure in general, but it is precise on the widely used class of free-choice workflow nets. It is based on the novel notion of continuous soundness, a variant of soundness that we prove coNP-complete in general. We further show that continuous soundness is equivalent to generalised and structural soundness on free-choice nets. For structural soundness, our algorithm is based on continuous reachability. We extend our tool FastForward with our algorithms, and positively evaluate them against existing tools for the analysis of workflow nets.

Lastly, we study the continuous semantics in the case of a system with states and

a single counter. For this novel model, called continuous one-counter automata, we establish complexity bounds for reachability in several interesting variants. These variants differ based on the type of restrictions we place on the counter value. Reachability is in NC2 when we place global lower and upper bounds on the counter value. It is instead NP-complete when we allow separate individual lower and upper bounds for each state, and additionally allow parameters on guards or updates.

Future work. Our algorithm for reachability in Petri nets, as presented in Section 3.1, is limited to standard Petri nets. It might be interesting to extend it to the case of transfer Petri nets or reset Petri nets, which are extensions to Petri nets that allow transitions to move all, instead of a fixed number of, tokens from one place. In the case of transfer nets, those tokens are put into another place. For reset nets, they are simply removed. Extending our algorithm to these extensions seems promising, as there exist rational and integral overapproximations for them. These overapproximations might allow constructing useful heuristics similar to the work presented in this section. However, the amount of benchmarks that make use of these extensions is fairly limited, and we suspect that the heuristics may turn out to be less useful in practice than for standard Petri nets, since the nonlinear nature of resets and transfers cause executions to be very sensitive to changes in the order in which transitions are fired even when we disregard whether transitions are enabled or not.

Another related direction for future research is to find better heuristics. In practice, it turns out that the calculation of the heuristic value for encountered nodes takes up a significant percentage of the total time. One way to speed this up would be to use heuristics that can precompute some values once to speed up the many calls to the heuristic. For example, in [99] the so-called *syntactic distance* is proposed, which computes for each place locally a value, and the heuristic value for a marking is derived in a certain way from the local values for places that are marked. This means that the heuristic precomputes the local place values only once, and can then reuse them and compute the heuristic values for markings relatively fast. The existing heuristics that are *local* like this are crude, but there seems to be potential for finding more accurate heuristics that are fast to evaluate by taking advantage of some form of precomputation.

Conclusion

For our complexity analysis of soundness in workflow nets in Chapter 5, one avenue for future research is to take into account extensions of workflow nets. Some concepts like OR-joins or cancellation regions do not map to standard workflow nets, but can be modelled by extensions such as reset, transfer or inhibitor nets [107, 108]. The variants of soundness are known to be undecidable in many variants of workflow nets, for example 1-soundness is undecidable in workflow nets with reset arcs or inhibitor arcs [5, Corollary 6.3], and generalised soundness is undecidable in workflow nets with inhibitor arcs [5, Proposition 6.2]. One open question is whether generalised soundness is decidable in workflow nets with reset arcs. It seems plausible that it could be, as generalised soundness is easier than soundness for standard workflow nets. However, the techniques we employed in our proof for the upper bound of generalised soundness in standard Petri nets seem hard to adjust. We made use of the fact that reachability from large initial k has connections to reachability over $\mathbb{Z}$, but this relationship seems much harder to establish with reset arcs, as runs in reset nets are more "sensitive" to changes in the order of transition firings. This is contrary to what happens in standard nets, where reordering the transitions of a run might make it only fireable over $\mathbb{Z}$, not $\mathbb{N}$, but does not change the marking that the run results in.

An interesting direction for future research related to our practical algorithms for generalised and structural soundness in Chapter 6 is to find other classes of workflow nets where soundness may be easier to decide. Some attempts in this direction already exist in the literature. For example, the question of k-soundness for acyclic workflow nets is co-NP-complete [109]. One promising idea along those lines seems to be to extend the notion of termination complexity, as defined for a related but slightly different model in [32], to workflow nets. One can define the class of *terminating* workflow nets, which are nets where the length of all runs is bounded by a function on the size of the target marking. In particular, this disallows the existence of infinite runs, and thus circumvents the PSPACE lower bound of generalised soundness, and the EXPSPACE lower bound of k-soundness. Thus, one may find more efficient procedures for this subclass. Currently, I am working on a submission exploring this research direction with one of my supervisors and another colleague.

Promising future research also seems possible in the investigation of continuous

systems with a small, fixed number of counters, for example continuous automata with two or three counters. For three counters, recall that we show in Section 7.4 that reachability is undecidable with equality tests, but it is unknown whether this holds when only zero tests are available, and solving this open problem could lead to interesting new techniques for continuous automata in general.

Another interesting question is whether we can find efficient practical procedures for parametric COCA. Recall that we solve the problem by checking a formula from $\mathrm{FO}(\mathbb{Q}, +, <)$. However, we need to guess the formula, since it corresponds to one of many possible linear path schemes. It appears that this causes the problem to be hard to tackle in practice. Finding other approaches for the problem that appear more promising to implement in practice could be a direction for future research.

Bibliography

[1] W. M. Van der Aalst, «Structural characterizations of sound workflow nets», *Computing science reports*, vol. 96(23), 1996, pp. 18–22.

[2] W. M. van der Aalst, «Using free-choice nets for process mining and business process management», in «2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS)», IEEE, 2021, pp. 9–15.

[3] W. M. P. van der Aalst, «Verification of workflow nets», in «Proc. 18th International Conference on Application and Theory of Petri Nets (ICATPN)», vol. 1248, 1997, pp. 407–426, doi:10.1007/3-540-63139-9_48.

[4] W. M. P. van der Aalst, «Using free-choice nets for process mining and business process management», in «2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS)», 2021, pp. 9–15, doi:10.15439/2021F002.

[5] W. M. P. van der Aalst, K. M. van Hee, A. H. M. ter Hofstede, N. Sidorova, H. M. W. Verbeek, M. Voorhoeve and M. T. Wynn, «Soundness of workflow nets: classification, decidability, and analysis», *Formal Aspects of Computing*, vol. 23(3), 2011, pp. 333–363, doi:10.1007/s00165-010-0161-4.

[6] W. V. der Aalst, «Interorganizational workflows: An approach based on message sequence charts and Petri nets», *Systems Analysis, Modelling, Simulation*, vol. 34(3), 1999, pp. 335–367.

[7] P. A. Abdulla, K. Cerans, B. Jonsson and Yih-Kuen Tsay, «General decidability theorems for infinite-state systems», in «Proceedings 11th An-

nual IEEE Symposium on Logic in Computer Science», 1996, pp. 313–321, doi:10.1109/LICS.1996.561359.

[8] S. Almagor, A. Ghosh, T. Leys and G. A. Perez, «The geometry of reachability in continuous vector addition systems with states», *arXiv preprint arXiv:2210.00785*, 2022. Unpublished.

[9] L. Atzori, A. Iera and G. Morabito, «The internet of things: A survey», *Computer Networks*, vol. 54(15), 2010, pp. 2787–2805, ISSN 1389-1286, doi:10.1016/j.comnet.2010.05.010.

[10] C. Baier and J.-P. Katoen, *Principles of model checking*, MIT press, 2008.

[11] R. Baldoni, E. Coppa, D. C. D'Elia, C. Demetrescu and I. Finocchi, «A survey of symbolic execution techniques», *ACM Comput. Surv.*, vol. 51(3), 2018.

[12] K. Barkaoui and L. Petrucci, «Structural analysis of workflow nets with shared resources», in «Proc. Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM)», vol. 98/7, 1998, pp. 82–95.

[13] A. Barth, J. C. Mitchell, A. Datta and S. Sundaram, «Privacy and utility in business processes», in «Proc. 20th IEEE Computer Security Foundations Symposium (CSF)», IEEE Computer Society, 2007, pp. 279–294, doi:10.1109/CSF.2007.26.

[14] A. Berti, S. J. Van Zelst and W. van der Aalst, «Process mining for python (pm4py): bridging the gap between process-and data science», 2019.

[15] N. Bjorner, «Smt solvers for testing, program analysis and verification at microsoft», in «2009 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing», 2009, pp. 15–15, doi:10.1109/SYNASC.2009.64.

[16] M. Blondin, A. Finkel, S. Göller, C. Haase and P. McKenzie, «Reachability in two-dimensional vector addition systems with states is PSPACE-complete», in «Proc. 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)», 2015, pp. 32–43, doi:10.1109/LICS.2015.14.

[17] M. Blondin, A. Finkel, C. Haase and S. Haddad, «Approaching the coverability problem continuously», in «Tools and Algorithms for the Construction and Analysis of Systems», edited by M. Chechik and J.-F. Raskin, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016, ISBN 978-3-662-49674-9, pp. 480–496.

[18] M. Blondin, A. Finkel, C. Haase and S. Haddad, «The logical view on continuous Petri nets», *ACM Trans. Comput. Logic*, vol. 18(3), August 2017, ISSN 1529-3785, doi:10.1145/3105908.

[19] M. Blondin and C. Haase, «Logics for continuous reachability in Petri nets and vector addition systems with states», in «2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)», 2017, pp. 1–12, doi:10.1109/LICS.2017.8005068.

[20] M. Blondin, C. Haase and P. Offtermatt, «Directed reachability for infinite-state systems», *CoRR*, vol. abs/2010.07912, 2020. "Preprint on arXiv. Published at TACAS 2021.", 2010.07912.

[21] M. Blondin, C. Haase and P. Offtermatt, «Directed reachability for infinite-state systems», in «Tools and Algorithms for the Construction and Analysis of Systems», edited by J. F. Groote and K. G. Larsen, Springer International Publishing, Cham, 2021, ISBN 978-3-030-72013-1, pp. 3–23.

[22] M. Blondin, C. Haase and P. Offtermatt, «FastForward: A tool for reachability in Petri nets with infinite state spaces. Artifact for the TACAS21 Contribution "Directed Reachability for Infinite-State Systems"», Jan 2021, doi:10.6084/m9.figshare.13573592.v1.

[23] M. Blondin, T. Leys, F. Mazowiecki, P. Offtermatt and G. Pérez, «Continuous one-counter automata», *ACM Trans. Comput. Logic*, vol. 24(1), jan 2023, ISSN 1529-3785, doi:10.1145/3558549.

[24] M. Blondin, T. Leys, F. Mazowiecki, P. Offtermatt and G. A. Perez, «Continuous one-counter automata», in «Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science», LICS '21, Association

for Computing Machinery, New York, NY, USA, 2021, ISBN 9781665448956, doi:10.1109/LICS52264.2021.9470525.

[25] M. Blondin, F. Mazowiecki and P. Offtermatt, «The complexity of soundness in workflow nets», in «Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science», Association for Computing Machinery, New York, NY, USA, 2022, ISBN 9781450393515.

[26] M. Blondin, F. Mazowiecki and P. Offtermatt, «The complexity of soundness in workflow nets», *CoRR*, vol. abs/2201.05588, 2022. 2201.05588.

[27] M. Blondin, F. Mazowiecki and P. Offtermatt, «Verifying generalised and structural soundness of workflow nets via relaxations», in «Computer Aided Verification», edited by S. Shoham and Y. Vizel, Springer International Publishing, Cham, 2022, ISBN 978-3-031-13188-2, pp. 468–489.

[28] M. Blondin, F. Mazowiecki and P. Offtermatt, «Verifying generalised and structural soundness of workflow nets via relaxations», *CoRR*, vol. abs/2206.02606, 2022, doi:10.48550/arXiv.2206.02606. 2206.02606.

[29] A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, P. Moro and T. Vojnar, «Programs with lists are counter automata», in «Computer Aided Verification», edited by T. Ball and R. B. Jones, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, ISBN 978-3-540-37411-4, pp. 517–531.

[30] Z. Bouziane and A. Finkel, «Cyclic Petri net reachability sets are semi-linear effectively constructible», in «Second International Workshop on Verification of Infinite State Systems (INFINITY)», *Electronic Notes in Theoretical Computer Science*, vol. 9, 1997, pp. 15–24, doi:10.1016/S1571-0661(05)80423-2.

[31] L. Bozzelli and P. Ganty, «Complexity analysis of the backward coverability algorithm for VASS», in «Reachability Problems», edited by G. Delzanno and I. Potapov, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, ISBN 978-3-642-24288-5, pp. 96–109.

[32] T. Brázdil, K. Chatterjee, A. Kucera, P. Novotný, D. Velan and F. Zuleger, «Efficient algorithms for asymptotic bounds on termination time in VASS», in «Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018», edited by A. Dawar and E. Grädel, ACM, 2018, pp. 185–194, doi:10.1145/3209108.3209191.

[33] H. Bride, O. Kouchnarenko and F. Peureux, «Reduction of workflow nets for generalised soundness verification», in «Proc. 18th International Conference Verification, Model Checking, and Abstract Interpretation (VMCAI)», 2017, pp. 91–111, doi:10.1007/978-3-319-52234-0_6.

[34] E. Cardoza, R. Lipton and A. R. Meyer, «Exponential space complete problems for Petri nets and commutative semigroups (preliminary report)», in «Proceedings of the Eighth Annual ACM Symposium on Theory of Computing», STOC '76, Association for Computing Machinery, New York, NY, USA, 1976, ISBN 9781450374149, pp. 50–54, doi:10.1145/800113.803630.

[35] C. Chitic and D. Rosu, «On validation of xml streams using finite state machines», in «Proceedings of the 7th International Workshop on the Web and Databases: Colocated with ACM SIGMOD/PODS 2004», WebDB '04, Association for Computing Machinery, New York, NY, USA, 2004, ISBN 9781450377881, pp. 85–90, doi:10.1145/1017074.1017096.

[36] F. L. Ţiplea and D. C. Marinescu, «Structural soundness of workflow nets is decidable», *Information Processing Letters*, vol. 96(2), 2005, pp. 54–58, doi:10.1016/j.ipl.2005.06.002.

[37] W. Czerwiński and L. Orlikowski, «Reachability in vector addition systems is ackermann-complete», in «2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)», 2022, pp. 1229–1240, doi:10.1109/FOCS52979.2021.00120.

[38] R. David and H. Alla, «Continuous Petri nets», in «Proc. 8th European Workshop on Application and Theory of Petri nets», 1987, pp. 275–294.

[39] R. Dechter and J. Pearl, «Generalized best-first search strategies and the optimality of A*», *J. ACM*, vol. 32(3), July 1985, pp. 505–536, ISSN 0004-5411, doi:10.1145/3828.3830.

[40] G. Delzanno, J.-F. Raskin and L. V. Begin, «Towards the automated verification of multithreaded java programs», in «International Conference on Tools and Algorithms for the Construction and Analysis of Systems», Springer, 2002, pp. 173–187.

[41] M. Der Jeng and S. C. Chen, «A heuristic search approach using approximate solutions to petri net marking equations for scheduling flexible manufacturing systems», *International Journal of Flexible Manufacturing Systems*, vol. 10(2), 1998, pp. 139–162.

[42] J. Desel and J. Esparza, *Free Choice Petri Nets*, Cambridge University Press, 1995, doi:10.1017/CBO9780511526558.

[43] R. M. Dijkman, M. Dumas and C. Ouyang, «Formal semantics and analysis of bpmn process models using petri nets», *Queensland University of Technology, Tech. Rep*, 2007, pp. 1–30.

[44] R. M. Dijkman, M. Dumas and C. Ouyang, «Semantics and analysis of business process models in bpmn», *Information and Software technology*, vol. 50(12), 2008, pp. 1281–1294.

[45] E. W. Dijkstra *et al.*, «A note on two problems in connexion with graphs», *Numerische mathematik*, vol. 1(1), 1959, pp. 269–271.

[46] A. Dixon and R. Lazić, «Kreach: A tool for reachability in Petri nets», in «Proc. 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)», Springer, 2020, pp. 405–412, doi:10.1007/978-3-030-45190-5_22.

[47] E. D'Osualdo, J. Kochems and C. L. Ong, «Automatic verification of erlang-style concurrency», in «Proc. 20th International Symposium on Static Analysis (SAS)», Springer, 2013, pp. 454–476, doi:10.1007/978-3-642-38856-9_24.

[48] V. D'Silva, D. Kroening and G. Weissenbacher, «A survey of automated techniques for formal software verification», *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27(7), 2008, pp. 1165–1178, doi:10.1109/TCAD.2008.923410.

[49] M. Dumas and A. H. Ter Hofstede, «Uml activity diagrams as a workflow specification language», in «International conference on the unified modeling language», Springer, 2001, pp. 76–90.

[50] P. Emanuelsson and U. Nilsson, «A comparative study of industrial static analysis tools», *Electronic Notes in Theoretical Computer Science*, vol. 217, 2008, pp. 5–21, ISSN 1571-0661, doi:10.1016/j.entcs.2008.06.039. Proceedings of the 3rd International Workshop on Systems Software Verification (SSV 2008).

[51] J. Esparza, «Advances in quantitative analysis of free-choice workflow Petri nets (invited talk)», in «24th International Symposium on Temporal Representation and Reasoning (TIME 2017)», edited by S. Schewe, T. Schneider and J. Wijsen, *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 90, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017, ISBN 978-3-95977-052-1, ISSN 1868-8969, pp. 2:1–2:6, doi:10.4230/LIPIcs.TIME.2017.2.

[52] J. Esparza, R. Ledesma-Garza, R. Majumdar, P. Meyer and F. Niksic, «An smt-based approach to coverability analysis», in «Computer Aided Verification», edited by A. Biere and R. Bloem, Springer International Publishing, Cham, 2014, ISBN 978-3-319-08867-9, pp. 603–619.

[53] J. Esparza, R. Ledesma-Garza, R. Majumdar, P. J. Meyer and F. Nikšić, «An SMT-based approach to coverability analysis», in «Proc. 26th International Conference on Computer Aided Verification (CAV)», Springer, 2014, pp. 603–619, doi:10.1007/978-3-319-08867-9_40.

[54] D. Fahland, C. Favre, B. Jobstmann, J. Koehler, N. Lohmann, H. Völzer and K. Wolf, «Instantaneous soundness checking of industrial business process mod-

els», in «Proc. 7th International Conference on Business Process Management (BPM)», 2009, pp. 278–293, doi:10.1007/978-3-642-03848-8_19.

[55] C. Favre, H. Völzer and P. Müller, «Diagnostic information for control-flow analysis of workflow graphs (a.k.a. free-choice workflow nets)», in «Proc. 22nd Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)», 2016, pp. 463–479, doi:10.1007/978-3-662-49674-9_27.

[56] J. Fearnley and M. Jurdziński, «Reachability in two-clock timed automata is pspace-complete», in «Automata, Languages, and Programming», edited by F. V. Fomin, R. Freivalds, M. Kwiatkowska and D. Peleg, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, ISBN 978-3-642-39212-2, pp. 212–223.

[57] Y. Feng, R. Martins, Y. Wang, I. Dillig and T. W. Reps, «Component-based synthesis for complex apis», *SIGPLAN Not.*, vol. 52(1), January 2017, pp. 599–612, ISSN 0362-1340, doi:10.1145/3093333.3009851.

[58] E. Fraca and S. Haddad, «Complexity analysis of continuous Petri nets», *Fundamenta informaticae*, vol. 137(1), 2015, pp. 1–28.

[59] P. Ganty, *Algorithmes et structures de données efficaces pour la manipulation de contraintes sur les intervalles*, Master's thesis, Université Libre de Bruxelles, 2002. (In French).

[60] T. Geffroy, J. Leroux and G. Sutre, «Occam's razor applied to the Petri net coverability problem», *Theoretical Computer Science*, vol. 750, 2018, pp. 38–52, doi:10.1016/j.tcs.2018.04.014.

[61] S. M. German and A. P. Sistla, «Reasoning about systems with many processes», *J. ACM*, vol. 39(3), July 1992, pp. 675–735, ISSN 0004-5411, doi:10.1145/146637.146681.

[62] M. Hack, «The recursive equivalence of the reachability problem and the liveness problem for Petri nets and vector addition systems», in «15th Annual Symposium on Switching and Automata Theory (swat 1974)», 1974, pp. 156–164, doi:10.1109/SWAT.1974.28.

[63] P. E. Hart, N. J. Nilsson and B. Raphael, «A formal basis for the heuristic determination of minimum cost paths», *IEEE Transactions on Systems Science and Cybernetics*, vol. 4(2), 1968, pp. 100–107, doi:10.1109/TSSC.1968.300136.

[64] K. M. van Hee, N. Sidorova and M. Voorhoeve, «Soundness and separability of workflow nets in the stepwise refinement approach», in «Proc. 24th International Conference on Applications and Theory of Petri Nets 2003 (ICATPN)», vol. 2679, 2003, pp. 337–356, doi:10.1007/3-540-44919-1_22.

[65] K. M. van Hee, N. Sidorova and M. Voorhoeve, «Generalised soundness of workflow nets is decidable», in «Proc. 25th International Conference on Applications and Theory of Petri Nets (ICATPN)», 2004, pp. 197–215, doi:10.1007/978-3-540-27793-4_12.

[66] J. Hopcroft and J.-J. Pansiot, «On the reachability problem for 5-dimensional vector addition systems», *Theoretical Computer Science*, vol. 8(2), 1979, pp. 135–159.

[67] M. M. zu Hörste and E. Schnieder, «Modelling and simulation of train control systems using petri nets», in «FMrail workshop», vol. 3, 1999.

[68] J. Janák, *Issue Tracking Systems*, Master's thesis, Masaryk University, 2009.

[69] A. Kaiser, D. Kroening and T. Wahl, «A widening approach to multithreaded program verification», *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 36(4), 2014, pp. 14:1–14:29, doi:10.1145/2629608.

[70] I. Koch, W. Reisig and F. Schreiber, *Modeling in systems biology: the Petri net approach*, vol. 16, Springer science & business media, 2010.

[71] S. R. Kosaraju, «Decidability of reachability in vector addition systems (preliminary version)», in «Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing», STOC '82, Association for Computing Machinery, New York, NY, USA, 1982, ISBN 0897910702, pp. 267–281, doi:10.1145/800070.802201.

[72] D. Kozma, P. Varga and F. Larrinaga, «Data-driven workflow management by utilising bpmn and cpn in iiot systems with the arrowhead framework», in «2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)», IEEE, 2019, pp. 385–392.

[73] J. Lambert, «A structure to decide reachability in Petri nets», *Theoretical Computer Science*, vol. 99(1), 1992, pp. 79–104, ISSN 0304-3975, doi:10.1016/0304-3975(92)90173-D.

[74] J. Leroux, «Vector addition systems reachability problem (a simpler solution)», in «Turing-100. The Alan Turing Centenary», edited by A. Voronkov, *EPiC Series in Computing*, vol. 10, 2012, ISSN 2398-7340, pp. 214–228, doi:10.29007/bnx2.

[75] J. Leroux, «The reachability problem for petri nets is not primitive recursive», in «2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)», 2022, pp. 1241–1252, doi:10.1109/FOCS52979.2021.00121.

[76] J. Leroux and S. Schmitz, «Reachability in vector addition systems is primitive-recursive in fixed dimension», in «2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)», 2019, pp. 1–13, doi:10.1109/LICS.2019.8785796.

[77] R. Lipton, «The reachability problem requires exponential space», *Research Report 62. Department of Computer Science, Yale University*, 1976.

[78] Lu Tan and Neng Wang, «Future internet: The internet of things», in «2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)», vol. 5, 2010, pp. V5–376–V5–380, doi:10.1109/ICACTE.2010.5579543.

[79] C. A. Mack, «Fifty years of moore's law», *IEEE Transactions on Semiconductor Manufacturing*, vol. 24(2), 2011, pp. 202–207, doi:10.1109/TSM.2010.2096437.

[80] E. W. Mayr, «An algorithm for the general Petri net reachability problem», in «Proceedings of the Thirteenth Annual ACM Symposium on Theory of Com-

puting», STOC '81, Association for Computing Machinery, New York, NY, USA, 1981, ISBN 9781450373920, pp. 238–246, doi:10.1145/800076.802477.

[81] E. W. Mayr, «An algorithm for the general Petri net reachability problem», *SIAM Journal on Computing*, vol. 13(3), 1984, pp. 441–460, doi:10.1137/0213029.

[82] E. W. Mayr and A. R. Meyer, «The complexity of the word problems for commutative semigroups and polynomial ideals», *Advances in Mathematics*, vol. 46(3), 1982, pp. 305–329, doi:10.1016/0001-8708(82)90048-2.

[83] E. W. Mayr and J. Weihmann, «A framework for classical petri net problems: Conservative petri nets as an application», in «International Conference on Applications and Theory of Petri Nets and Concurrency», Springer, 2014, pp. 314–333.

[84] P. J. Meyer, J. Esparza and P. Offtermatt, «Computing the expected execution time of probabilistic workflow nets», in «Tools and Algorithms for the Construction and Analysis of Systems», edited by T. Vojnar and L. Zhang, Springer International Publishing, Cham, 2019, ISBN 978-3-030-17465-1, pp. 154–171.

[85] M. L. Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall, Inc., 1967, ISBN 0131655639.

[86] L. M. de Moura and N. Bjørner, «Z3: an efficient SMT solver», in «Proc. 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)», Springer, 2008, pp. 337–340, doi:10.1007/978-3-540-78800-3_24.

[87] T. Murata, «Petri nets: Properties, analysis and applications», *Proceedings of the IEEE*, vol. 77(4), 1989, pp. 541–580, doi:10.1109/5.24143.

[88] P. Offtermatt, M. Blondin and F. Mazowiecki, «FastForward for Soundness: Verifying generalised and structural soundness of workflow nets via relaxations», 5 2022, doi:10.6084/m9.figshare.19721674.v2.

[89] C. H. Papadimitriou, *Computational complexity*, Addison-Wesley, 1994, ISBN 978-0-201-53082-7.

[90] L. Ping, H. Hao and L. Jian, «On 1-soundness and soundness of workflow nets», in «Third Workshop on Modelling of Objects, Components, and Agents», 2004, p. 21.

[91] C. Rackoff, «The covering and boundedness problems for vector addition systems», *Theoretical Computer Science*, vol. 6(2), 1978, pp. 223–231, ISSN 0304-3975, doi:10.1016/0304-3975(78)90036-1.

[92] H. G. Rice, «Classes of recursively enumerable sets and their decision problems», *Transactions of the American Mathematical Society*, vol. 74(2), 1953, pp. 358–366, ISSN 00029947.

[93] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd edn., Prentice Hall Press, 2009.

[94] A.-W. Scheer, O. Thomas and O. Adam, «Process modeling using event-driven process chains.», *Process-aware information systems*, vol. 119, 2005.

[95] K. Schmidt, «Lola a low level analyser», in «Application and Theory of Petri Nets 2000», edited by M. Nielsen and D. Simpson, Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, ISBN 978-3-540-44988-1, pp. 465–474.

[96] M. Sipser, «Introduction to the theory of computation», *ACM Sigact News*, vol. 27(1), 1996, pp. 27–29.

[97] E. D. Sontag, «Real addition and the polynomial hierarchy», *Information Processing Letters*, vol. 20(3), 1985, pp. 115–120.

[98] E. Steinitz, «Bedingt konvergente Reihen und konvexe Systeme», 1913.

[99] T. Strazny, *An Algorithmic Framework for Checking Coverability in Well-Structured Transition Systems*, Ph.D. thesis, Universität Oldenburg, Fakultät II (Informatik, Wirtschafts- und Rechtswissenschaften), Department für Informatik, 2014.

[100] G. Uma and B. Prasad, «Reachability trees for Petri nets: a heuristic approach», *Knowledge-Based Systems*, vol. 6(3), 1993, pp. 174–177, ISSN 0950-7051, doi:10.1016/0950-7051(93)90042-R.

[101] W. Van Der Aalst, «Process mining», *Communications of the ACM*, vol. 55(8), 2012, pp. 76–83.

[102] W. M. Van Der Aalst, «Three good reasons for using a petri-net-based workflow management system», in «Information and Process Integration in Enterprises», Springer, 1998, pp. 161–182.

[103] W. M. Van Der Aalst and A. H. Ter Hofstede, «Yawl: yet another workflow language», *Information systems*, vol. 30(4), 2005, pp. 245–275.

[104] E. Verbeek and W. M. Van Der Aalst, «Woflan 2.0 a petri-net-based workflow diagnosis tool», in «International Conference on application and theory of Petri Nets», Springer, 2000, pp. 475–484.

[105] H. Verbeek, J. Buijs, B. Van Dongen and W. M. van der Aalst, «Prom 6: The process mining toolkit», *Proc. of BPM Demonstration Track*, vol. 615, 2010, pp. 34–39.

[106] K. Wolf, «Petri net model checking with LoLA 2», in «Application and Theory of Petri Nets and Concurrency», 2018, ISBN 978-3-319-91268-4, pp. 351–362.

[107] M. T. Wynn, W. M. van der Aalst, A. H. ter Hofstede and D. Edmond, «Verifying workflows with cancellation regions and or-joins: an approach based on reset nets and reachability analysis», in «International Conference on Business Process Management», Springer, 2006, pp. 389–394.

[108] M. T. Wynn, D. Edmond, W. M. van der Aalst and A. H. ter Hofstede, «Achieving a general, formal and decidable approach to the or-join in workflow using reset nets», in «International Conference on Application and Theory of Petri Nets», Springer, 2005, pp. 423–443.

[109] F. L. Ţiplea, C. Bocăneală and R. Chiroşcă, «On the complexity of deciding soundness of acyclic workflow nets», *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45(9), 2015, pp. 1292–1298, doi:10.1109/TSMC.2015.2394735.

Appendix A

Directed Reachability for Infinite-State Systems

This paper was published as a peer-reviewed conference article. A full version with an appendix containing missing proofs omitted from the conference paper due to space constraints was uploaded to arXiv at the URL https://arxiv.org/abs/2010.07912, see [20].

Cite as: M. Blondin, C. Haase, and P. Offtermatt. Directed reachability for infinite-state systems. In J. F. Groote and K. G. Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 3–23, Cham, 2021. Springer International Publishing.

Summary We propose a novel approach for semi-deciding reachability in Petri nets. It is based on using efficiently computable approximations as distance oracles in directed graph exploration algorithms such as A*and greedy best-first search. We show that relaxations of reachability in Petri net can be used to construct heuristic functions that satisfy certain properties which are useful for directed search algorithms. Thus, using A*or greedy best-first search with these heuristic functions gives guarantees on optimality and termination. We present a prototype implementation of the approach. On a set of about 200 instances from program synthesis and verification of concurrent programs, it is competitive against state-of-the-art tools for Petri nets.

Contribution of this author This author took on sole responsibility for the implementation and experimental evaluation of the approach. This includes the creation of the artifact for the conference submission. It also involved contributing to discussions regarding algorithmic design and the choice of which directed search algorithms and heuristics to consider. Further, the author contributed to the development of the manuscript, particularly taking a leading role in the writing of Section 5.

Directed Reachability for Infinite-State Systems*

Michael Blondin[1], Christoph Haase[2], and Philip Offtermatt[1,3]

[1] Université de Sherbrooke, Canada
[2] University of Oxford, United Kingdom
[3] Max Planck Institute for Software Systems, Saarbrücken, Germany

Abstract. Numerous tasks in program analysis and synthesis reduce to deciding reachability in possibly infinite graphs such as those induced by Petri nets. However, the Petri net reachability problem has recently been shown to require non-elementary time, which raises questions about the practical applicability of Petri nets as target models. In this paper, we introduce a novel approach for efficiently semi-deciding the reachability problem for Petri nets in practice. Our key insight is that computationally lightweight over-approximations of Petri nets can be used as distance oracles in classical graph exploration algorithms such as A* and greedy best-first search. We provide and evaluate a prototype implementation of our approach that outperforms existing state-of-the-art tools, sometimes by orders of magnitude, and which is also competitive with domain-specific tools on benchmarks coming from program synthesis and concurrent program analysis.

Keywords: Petri nets · reachability · shortest paths · model checking

1 Introduction

Many problems in program analysis, synthesis and verification reduce to deciding reachability of a vertex or a set of vertices in infinite graphs, *e.g.*, when reasoning about concurrent programs with an unbounded number of threads, or when arbitrarily many components can be used in a synthesis task. For automated reasoning tasks, those infinite graphs are finitely represented by some mathematical model. Finding the right such model requires a trade-off between the two conflicting goals of maximal expressive power and computational feasibility of the relevant decision problems. Petri nets are a ubiquitous mathematical model that provides a good compromise between those two goals. They are

* This work is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 852769, ARiAT). It is also supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC). Parts of this research were carried out while the second author was affiliated with the Department of Computer Science, University College London, UK.

expressive enough to find a plethora of applications in computer science, in particular in the analysis of concurrent processes, yet the reachability problem for Petri nets is decidable [50,43,44,46]. *Counter abstraction* has evolved as a generic abstraction paradigm that reduces a variety of program analysis tasks to problems in Petri nets or variants thereof such as well-structured transition systems, see *e.g.* [32,42,64,5]. Due to their generality and versatility, Petri nets and their extensions find numerous applications also in other areas, including the design and analysis of protocols [22], business processes [60], biological systems [36,10] and chemical systems [2]. The goal of this paper is to introduce and evaluate an efficient generic approach to deciding the Petri net reachability problem on instances arising from applications in program verification and synthesis.

A Petri net comprises a finite set of *places* with a finite number of *transitions*. Places carry a finite yet unbounded number of *tokens* and transitions can remove and add tokens to places. A *marking* specifies how many tokens each place carries. An example of a Petri net is given on the left-hand side of Figure 1, where the two places $\{p_1, p_2\}$ are depicted as circles and transitions $\{t_1, t_2, t_3\}$ as squares. Places carry tokens depicted as filled circles; thus p_1 carries one token and p_2 carries none. We write this as $[p_1 \colon 1, p_2 \colon 0]$, or $(1, 0)$ if there is a clear ordering on the places. Transition t_1 can add a single token to place p_1 at any moment. As soon as a token is present in p_1, it can be consumed by transition t_2, which then adds a token to place p_2 and puts back one token to place p_1. Finally, transition t_3 consumes tokens from p_1 without any adding token at all.

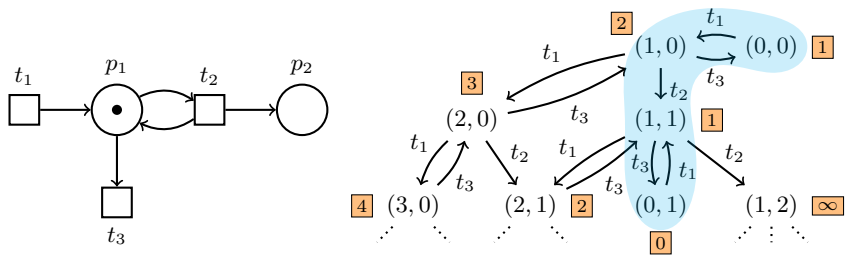


Fig. 1. *Left:* A Petri net $\mathcal{N}$. *Right:* Search of the forthcoming Algorithm 1 over the graph $G_{\mathbb{N}}(\mathcal{N})$ from $(0, 0)$ to $(0, 1)$, where (x, y) denotes $[p_1 \colon x, p_2 \colon y]$ and each number in a box next to a marking is its heuristic value. Only the blue region is expanded.

A Petri net induces a possibly infinite directed graph whose vertices are markings, and whose edges are determined by the transitions of the Petri net, *cf.* the right side of Figure 1. Given two markings, the *reachability problem* asks whether they are connected in this graph. In Figure 1, the marking $(0, 1)$ is reachable from $(0, 0)$, *e.g.*, via paths of lengths 3 and 5: $(0, 0) \xrightarrow{t_1} (1, 0) \xrightarrow{t_2} (1, 1) \xrightarrow{t_3} (0, 1)$ and $(0, 0) \xrightarrow{t_1} (1, 0) \xrightarrow{t_1} (2, 0) \xrightarrow{t_2} (2, 1) \xrightarrow{t_3} (1, 1) \xrightarrow{t_3} (0, 1)$.

In practice, the Petri net reachability problem is a challenging decision problem due to its horrendous worst-case complexity: an exponential-space lower bound was established in the 1970s [48], and a non-elementary time lower bound

has only recently been established [12]. One may thus question whether a problem with such high worst-case complexity is of any practical relevance, and whether reducing program analysis tasks to Petri net reachability is anything else than merely an intellectual exercise. We debunk those concerns and present a technique which decides most reachability instances appearing in the wild. When evaluated on large-scale instances involving Petri nets with thousands of places and tens of thousands of transitions, our prototype implementation is most of the time faster, even up to several orders of magnitude on large-scale instances, and solves more instances than existing state-of-the-art tools. Our implementation is also competitive with specialized domain-specific tools. One of the biggest advantages of our approach is that it is extremely simple to describe and implement, and it readily generalizes to many extensions of Petri nets. In fact, it was surprising to us that our approach has not yet been discovered. We now describe the main observations and techniques underlying our approach.

Ever since the early days of research in Petri nets, state-space over-approximations have been studied to attenuate the high computational complexity of their decision problems. One such over-approximation is, informally speaking, to allow places to carry a negative numbers of tokens. Deciding reachability then reduces to solving the so-called *state equation*, a system of linear equations associated to a Petri net. Another over-approximation are *continuous Petri nets*, a variant where places carry fractional tokens and "fractions of transitions" can be applied [13]. The benefit is that deciding reachability drops down to polynomial time [26]. While those approximations have been applied for pruning search spaces, see *e.g.* [23,4,8,31], we make the following simple key observation:

> *If a marking $\boldsymbol{m}$ is reachable from an initial marking in an over-approximation, then the length of a shortest witnessing path in the over-approximation lower bounds the length of a shortest path reaching $\boldsymbol{m}$.*

The availability of an oracle providing lower bounds on the length of shortest paths between markings enables us to appeal to classical graph traversal algorithms which have been highly successful in artificial intelligence and require such oracles, namely A^* and greedy best-first search, see *e.g.* [55]. In particular, determining the length of shortest paths in the over-approximations described above can be phrased as optimization problems in (integer) linear programming and optimization modulo theories, for which efficient off-the-shelf solvers are available [35,7]. Thus, oracle calls can be made at comparably modest computational cost, which is crucial for the applicability of those algorithms. As a result, a large class of existing state-space over-approximations can be applied to obtain a highly efficient forward-analysis semi-decision procedure for the reachability problem. For example, in Figure 1, using the state equation as distance oracle, A^* only explores the four vertices in the blue region and directly reaches the target vertex, whereas a breadth-first search may need to explore all vertices of the figure and a depth-first search may even not terminate.

In theory, our approach could be turned into a decision procedure by applying bounds on the length of shortest paths in Petri nets [47]. However, such

lengths can grow non-elementarily in the number of places [12], and just computing the cut-off length will already be infeasible for any Petri net of practical relevance. It is worth mentioning that, in practice, it has been observed that the over-approximations we employ also often witness non-reachability though, see *e.g.* [23]. Still, when dealing with finite state spaces, our procedure is complete.

A noteworthy benefit of our approach is that it enables finding *shortest* paths when A^* is used as the underlying algorithm. In program analysis, paths usually correspond to traces reaching an erroneous configuration. In this setting, shorter error traces are preferred as they help understanding why a certain error occurs. Furthermore, in program synthesis, paths correspond to synthesis plans. Again, shorter paths are preferred as they yield shorter synthesized programs. In fact, we develop our algorithmic framework for *weighted* Petri nets in which transitions are weighted with positive integers. Classical Petri nets correspond to the special instance where all weights are equal to one. Weighted Petri nets are useful to reflect cost or preferences in synthesis tasks. For example, there are program synthesis approaches where software projects are mined to determine how often API methods are called to guide a procedure by preferring more frequent methods [28,27,49]. Similarity metrics can also be used to obtain costs estimating the relevance of invoking methods [25]. It has further been argued that weighted Petri nets are a good model for synthesis tasks of chemical reactions as they can reflect costs of various chemical compounds [61]. Finally, weights can be viewed as representing an amount of time it takes to fire a transition, see *e.g.* [53].

Related work. Our approach falls under the umbrella term *directed model checking* coined in the early 2000s, which refers to a set of techniques to tackle the state-explosion problem via guided state-space exploration. It primarily targets disproving safety properties by quickly finding a path to an error state without the need to explicitly construct the whole state space. As such, directed model checking is useful for bug-finding since, in the words of Yang and Dill [63], *in practice, model checkers are most useful when they find bugs, not when they prove a property.* The survey paper [20] gives an overview over various directed model checking techniques for finite-state systems.

For Petri nets, directed reachability algorithms based on over-approximations as developed in this work have not been described. In [59], it is argued that exploration heuristics, like A^*, can be useful for Petri nets, but they do not consider over-approximations for the underlying heuristic functions. The authors of [39] use Petri nets for scheduling problems and employ the state equation, viewed as a system of linear equations over $\mathbb{Q}$, in order to explore and prune reachability graphs. This approach is, however, not guaranteed to discover shortest paths. There has been further work on using A^* for exploring the reachability graph of Petri nets for scheduling problems, see, *e.g.*, [45,51] and the references therein.

2 Preliminaries

Let $\mathbb{N} \coloneqq \{0, 1, \ldots\}$. For all $\mathbb{D} \subseteq \mathbb{Q}$ and $\succ \in \{\geq, >\}$, let $\mathbb{D}_{\succ 0} \coloneqq \{a \in \mathbb{D} : a \succ 0\}$, and for every set X, let $\mathbb{D}^X$ denote the set of vectors $\mathbb{D}^X \coloneqq \{\boldsymbol{v} \mid \boldsymbol{v} \colon X \to \mathbb{D}\}$.

We naturally extend operations componentwise. In particular, $(\boldsymbol{u} + \boldsymbol{v})(x) \coloneqq \boldsymbol{u}(x) + \boldsymbol{v}(x)$ for every $x \in X$, and $\boldsymbol{u} \geq \boldsymbol{v}$ iff $\boldsymbol{u}(x) \geq \boldsymbol{v}(x)$ for every $x \in X$.

Graphs. A *(labeled directed) graph* is a triple $G = (V, E, A)$, where V is a set of *nodes*, A is a finite set of elements called *actions*, and $E \subseteq V \times A \times V$ is the set of *edges* labeled by actions. We say that G has *finite out-degree* if the set of outgoing edges $\{(w, a, w') \in E : w = v\}$ is finite for every $v \in V$. Similarly, it has *finite in-degree* if the set of ingoing edges is finite for every $v \in V$. If G has both finite out- and in-degree, then we say that G is *locally finite*. A *path* π is a finite sequence of nodes $(v_i)_{1 \leq i \leq n}$ and actions $(a_i)_{1 \leq i < n}$ such that $(v_i, a_i, v_{i+1}) \in E$ for all $1 \leq i < n$. We say that π is a *path from v to w* (or a *v-w path*) if $v = v_1$ and $w = v_n$, and its *label* is $a_1 a_2 \cdots a_{n-1}$, where ε denotes the empty sequence.

A *weighted* graph is a tuple $G = (V, E, A, \mu)$ where (V, E, A) is a graph with a *weight function* $\mu \colon E \to \mathbb{Q}_{>0}$. The *weight* of path π is the weight of its edges, *i.e.* $\mu(\pi) \coloneqq \sum_{1 \leq i < n} \mu(v_i, a_i, v_{i+1})$. A *shortest path from v to w* is a v-w path π minimizing $\mu(\pi)$. We define $\mathrm{dist}_G \colon V \times V \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ as the *distance function* where $\mathrm{dist}_G(v, w)$ is the weight of a shortest path from v to w, with $\mathrm{dist}_G(v, w) \coloneqq \infty$ if there is none. We assume throughout the paper that weighted graphs have a *minimal weight*, *i.e.* that $\min\{\mu(e) : e \in E\}$ exists. For graphs with finite out-degree, this ensures that if a path exists between two nodes, then a shortest one exists.[4] This mild assumption always holds in our setting.

Petri nets. A *weighted Petri net* is a tuple $\mathcal{N} = (P, T, f, \lambda)$ where

- P is a finite set whose elements are called *places*,
- T is a finite set, disjoint from P, whose elements are called *transitions*,
- $f \colon (P \times T) \cup (T \times P) \to \mathbb{N}$ is the *flow function* assigning multiplicities to arcs connecting places and transitions, and
- $\lambda \colon T \to \mathbb{Q}_{>0}$ is the *weight function* assigning weights to transitions.

A *marking* is a vector $\boldsymbol{m} \in \mathbb{N}^P$ which indicates that place p holds $\boldsymbol{m}(p)$ *tokens*. A weighted Petri net with $\lambda(t) = 1$ for each $t \in T$ is called a *Petri net*. For example, Figure 1 depicts a Petri net $\mathcal{N}$ with $P = \{p_1, p_2\}$, $T = \{t_1, t_2, t_3\}$, $f(p_1, t_3) = f(p_1, t_2) = f(t_1, p_1) = f(t_2, p_1) = f(t_2, p_2) = 1$ (multiplicity omitted on arcs) and $f(-, -) = 0$ elsewhere (no arc). Moreover, $\mathcal{N}$ is marked with $[p_1 \colon 1, p_2 \colon 0]$.

The *guard* and *effect* of a transition $t \in T$ are vectors $\boldsymbol{g}_t \in \mathbb{N}^p$ and $\boldsymbol{\Delta}_t \in \mathbb{Z}^p$ where $\boldsymbol{g}_t(p) \coloneqq f(p, t)$ and $\boldsymbol{\Delta}_t(p) \coloneqq f(t, p) - f(p, t)$. We say that t is *firable* from marking $\boldsymbol{m}$ if $\boldsymbol{m} \geq \boldsymbol{g}_t$. If t is firable from $\boldsymbol{m}$, then it may be *fired*, which leads to marking $\boldsymbol{m}' \coloneqq \boldsymbol{m} + \boldsymbol{\Delta}_t$. We write this as $\boldsymbol{m} \xrightarrow{t}_{\mathbb{N}} \boldsymbol{m}'$. These notions naturally extend to sequences of transitions, *i.e.* $\xrightarrow{\varepsilon}_{\mathbb{N}}$ denotes the identity relation over $\mathbb{N}^P$, $\boldsymbol{\Delta}_\varepsilon \coloneqq \boldsymbol{0}$, $\lambda(\varepsilon) \coloneqq 0$, and for every $t_1, t_2, \ldots, t_k \in T$: $\boldsymbol{\Delta}_{t_1 t_2 \cdots t_k} \coloneqq \boldsymbol{\Delta}_{t_1} + \boldsymbol{\Delta}_{t_2} + \cdots + \boldsymbol{\Delta}_{t_k}$, $\lambda(t_1 t_2 \cdots t_k) \coloneqq \lambda(t_1) + \lambda(t_2) + \cdots + \lambda(t_k)$, and

$$\xrightarrow{t_1 t_2 \cdots t_k}_{\mathbb{N}} \coloneqq \xrightarrow{t_k}_{\mathbb{N}} \circ \cdots \circ \xrightarrow{t_2}_{\mathbb{N}} \circ \xrightarrow{t_1}_{\mathbb{N}},$$

[4] Otherwise, there could be increasingly better paths, *e.g.* of weights $1, 1/2, 1/4, \ldots$.

We say that $\rightarrow_{\mathbb{N}} := \cup_{t \in T} \xrightarrow{t}_{\mathbb{N}}$ and $\xrightarrow{*}_{\mathbb{N}} := \cup_{\sigma \in T^*} \xrightarrow{\sigma}_{\mathbb{N}}$ are the *step* and *reachability* relations. Note that the latter is the reflexive transitive closure of $\rightarrow_{\mathbb{N}}$.

For example, $\boldsymbol{m} \xrightarrow{t_2 t_3}_{\mathbb{N}} \boldsymbol{m}'$ and $\boldsymbol{m} \xrightarrow{t_1 t_2 t_3 t_3}_{\mathbb{N}} \boldsymbol{m}'$ in Figure 1, where $\boldsymbol{m} := [p_1 : 1, p_2 : 0]$ and $\boldsymbol{m}' := [p_1 : 0, p_2 : 1]$. Moreover, t_2 is not firable in $\boldsymbol{m}'$.

Given a sequence $\sigma \in T^*$, denote by $|\sigma|_t \in \mathbb{N}$ the number of times transition t occurs in T. The *Parikh image* of σ is the vector $\boldsymbol{\sigma} \in \mathbb{N}^T$ that captures the number of occurrences of transitions appearing in σ, *i.e.* $\boldsymbol{\sigma}(t) := |\sigma|_t$ for all $t \in T$.

Each weighted Petri net $\mathcal{N} = (P, T, f, \lambda)$ induces a locally finite weighted graph $G_{\mathbb{N}}(\mathcal{N}) := (V, E, T, \mu)$, called its *reachability graph*, where $V := \mathbb{N}^P$, $E := \{(\boldsymbol{m}, t, \boldsymbol{m}') : \boldsymbol{m} \xrightarrow{t}_{\mathbb{N}} \boldsymbol{m}'\}$ and $\mu(\boldsymbol{m}, t, \boldsymbol{m}') := \lambda(t)$ for each $(\boldsymbol{m}, t, \boldsymbol{m}') \in E$. An example of a reachability graph is given on the right of Figure 1. We write $\mathrm{dist}_{\mathcal{N}}$ to denote $\mathrm{dist}_{G_{\mathbb{N}}(\mathcal{N})}$. We have $\mathrm{dist}_{\mathcal{N}}(\boldsymbol{m}, \boldsymbol{m}') \neq \infty$ iff $\boldsymbol{m} \xrightarrow{\sigma}_{\mathbb{N}} \boldsymbol{m}'$ for some $\sigma \in T^*$, and if the latter holds, then $\mathrm{dist}_{\mathcal{N}}(\boldsymbol{m}, \boldsymbol{m}')$ is the minimal weight among such firing sequences σ. Moreover, for (unweighted) Petri nets, $\mathrm{dist}_{\mathcal{N}}(\boldsymbol{m}, \boldsymbol{m}')$ is the minimal number of transitions to fire to reach $\boldsymbol{m}'$ from $\boldsymbol{m}$.

3 Directed search algorithms

Our approach relies on classical pathfinding procedures guided by node selection strategies. Their generic scheme is described in Algorithm 1. Its termination with a value $d \neq \infty$ indicates that the weighted graph $G = (V, E, A, \mu)$ has a path from s to t of weight d, whereas termination with $d = \infty$ signals that $\mathrm{dist}_G(s, t) = \infty$.

```
1  g := [s ↦ 0, v ↦ ∞ : v ≠ s]
2  C := {s}
3  while C ≠ ∅ do
4      v := arg min_{v∈C} S(g, v)
5      if v = t then return g(t)
6      for (v, a, w) ∈ E do
7          if g(v) + μ(v, a, w) < g(w) then
8              g(w) := g(v) + μ(v, a, w)
9              C    := C ∪ {w}
10     C := C \ {v}
11 return ∞
```

Algorithm 1: Directed search algorithm.

Algorithm 1 maintains a set of *frontier nodes* C and a mapping $g \colon V \rightarrow \mathbb{Q}_{\geq 0} \cup \{\infty\}$ such that $g(w)$ is the weight of the best known path from s to w. In Line 4, a *selection strategy* S determines which node v to *expand* next. Starting from Line 6, a successor w of v is added to the frontier if its distance improves.

Let $h \colon V \rightarrow \mathbb{Q}_{\geq 0} \cup \{\infty\}$ estimate the distance from all nodes to a target $t \in V$. The selection strategies sending (g, v) respectively to $g(v)$, $g(v) + h(v)$ or $h(v)$ yield the classical Dijkstra's, A^* and greedy best-first search (*GBFS*) algorithms.

When instantiating S with Dijkstra's selection strategy, a return value $d \neq \infty$ is guaranteed to equal $\mathrm{dist}_G(s, t)$. This is not true for A^* and GBFS. However, if h fulfills the following *consistency* properties, then A^* also has this guarantee: $h(t) = 0$ and $h(v) \leq \mu(v, a, w) + h(w)$ for every $(v, a, w) \in E$ (see, *e.g.*, [55]).

In the setting of infinite graphs, unlike GBFS, A^* and Dijkstra's selection strategies guarantee termination if $\mathrm{dist}_G(s, t) \neq \infty$. Yet, we introduce *unbounded heuristics* for which termination is also guaranteed for GBFS. Note that these

guarantees would vanish in the presence of zero weights. An *infinite path* π is a sequence of nodes $(v_i)_{i\in\mathbb{N}}$ and actions $(a_i)_{i\in\mathbb{N}}$ such that $(v_i, a_i, v_{i+1}) \in E$ for all $i \in \mathbb{N}$. We say that π is *bounded* w.r.t. h if its nodes are pairwise distinct and there exists $b \in \mathbb{Q}_{\geq 0}$ with $h(v_i) \leq b$ for all $i \geq 0$. We say that h is *unbounded* if it admits no bounded sequence. The following technical lemma enables to prove termination of GFBS in the presence of unbounded heuristics.

Lemma 1. *If G is locally finite and h is unbounded, then the following holds:*

1. *The set of paths of weight at most $c \in \mathbb{Q}_{\geq 0}$ starting from node s is finite.*
2. *Let $W \subseteq V$. The set $\mathrm{dist}_G(W, t) \coloneqq \{\mathrm{dist}_G(w, t) : w \in W\}$ has a minimum.*
3. *No node is expanded infinitely often by Algorithm 1.*

Theorem 1. *Algorithm 1 with the greedy best-first search selection strategy always finds reachable targets for locally finite graphs and unbounded heuristics.*

Proof. First observe that Algorithm 1 satisfies this invariant:

$$\text{if } g(v) \neq \infty, \text{ then } g(v) \text{ is the weight of a path from } s \text{ to } v \text{ in } G \text{ whose nodes were all expanded, except possibly } v. \quad (*)$$

Assume $\mathrm{dist}_G(s, t) \neq \infty$. For the sake of contradiction, suppose t is never expanded. Let K_i be the subgraph of G induced by nodes expanded at least once within the first i iterations of the **while** loop. In particular, K_1 is the graph made only of node s. Let $K = K_1 \cup K_2 \cup \cdots$. By Lemma 1 (3), no node is expanded infinitely often, hence K is infinite. Moreover, K has finite out-degree, and each node of K is reachable from s in K by $(*)$. Thus, by König's lemma, K contains an infinite path $v_0, v_1, \ldots \in V$ of pairwise distinct nodes.

Let w be a node of K minimizing $\mathrm{dist}_G(w, t)$. It is well-defined by Lemma 1 (2). We have $\mathrm{dist}_G(w, t) \neq \infty$ as t is reachable from s and the latter belongs to $K_1 \subseteq K$. By minimality of $w \neq t$, there exists an edge (w, a, w') of G such that $\mathrm{dist}_G(w', t) < \mathrm{dist}_G(w, t)$ and w' does not appear in K. Note that w' is added to C at some point, but is never expanded as it would otherwise belong to K. Let i be the smallest index such that w belongs to K_i. Since h is unbounded, there exists j such that $h(v_j) > h(w')$ and v_j is expanded after iteration i of the while loop. This is a contradiction as w' would have been expanded instead of v_j. □

4 Directed reachability

In this section, we explain how to instantiate Algorithm 1 for finding short(est) firing sequences witnessing reachability in weighted Petri nets. Since Dijkstra's selection strategy does not require any heuristic, we focus on A^* and greedy best-first search which require consistent and unbounded heuristics. More precisely, we introduce distance under-approximations (Section 4.1); present relevant concrete distance under-approximations (Section 4.2); and put everything together into our framework (Section 4.3).

4.1 Distance under-approximations

A *distance under-approximation* of a weighted Petri net $\mathcal{N} = (P, T, f, \lambda)$ is a function $d\colon \mathbb{N}^P \times \mathbb{N}^P \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ such that for all $\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{m}'' \in \mathbb{N}^P$:

- $d(\boldsymbol{m}, \boldsymbol{m}') \leq \mathrm{dist}_{\mathcal{N}}(\boldsymbol{m}, \boldsymbol{m}')$,
- $d(\boldsymbol{m}, \boldsymbol{m}'') \leq d(\boldsymbol{m}, \boldsymbol{m}') + d(\boldsymbol{m}', \boldsymbol{m}'')$ (*triangle inequality*), and
- d is effective, *i.e.* there is an algorithm that evaluates d on all inputs.

We naturally obtain a heuristic from d for a directed search towards marking $\boldsymbol{m}_{\text{target}}$. Indeed, let $h\colon \mathbb{N}^P \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ be defined by $h(\boldsymbol{m}) \coloneqq d(\boldsymbol{m}, \boldsymbol{m}_{\text{target}})$. The following proposition shows that h is a suitable heuristic for A^*:

Proposition 1. *Mapping h is a consistent heuristic.*

Proof. Let $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{N}^P$ and $t \in T$ be such that $\boldsymbol{m} \xrightarrow{t}_{\mathbb{N}} \boldsymbol{m}'$. We have:

$$\begin{aligned}
h(\boldsymbol{m}) &= d(\boldsymbol{m}, \boldsymbol{m}_{\text{target}}) && \text{(by def. of } h\text{)}\\
&\leq d(\boldsymbol{m}, \boldsymbol{m}') + d(\boldsymbol{m}', \boldsymbol{m}_{\text{target}}) && \text{(by the triangle inequality)}\\
&\leq \mathrm{dist}_{\mathcal{N}}(\boldsymbol{m}, \boldsymbol{m}') + d(\boldsymbol{m}', \boldsymbol{m}_{\text{target}}) && \text{(by distance under-approximation)}\\
&\leq \lambda(t) + d(\boldsymbol{m}', \boldsymbol{m}_{\text{target}}) && \text{(since } \boldsymbol{m} \xrightarrow{t}_{\mathbb{N}} \boldsymbol{m}'\text{)}\\
&= \lambda(t) + h(\boldsymbol{m}') && \text{(by def. of } h\text{).}
\end{aligned}$$

Moreover, $h(\boldsymbol{m}_{\text{target}}) = d(\boldsymbol{m}_{\text{target}}, \boldsymbol{m}_{\text{target}}) \leq \mathrm{dist}_{\mathcal{N}}(\boldsymbol{m}_{\text{target}}, \boldsymbol{m}_{\text{target}}) = 0$, where the last equality follows from the fact that weights are positive. □

4.2 From Petri net relaxations to distance under-approximations

We now introduce classical relaxations of Petri nets which over-approximate reachability and consequently give rise to distance under-approximations. The main source of hardness of the reachability problem stems from the fact that places are required to hold a non-negative number of tokens. If we relax this requirement and allow negative numbers of tokens, we obtain a more tractable relation. More precisely, we write $\boldsymbol{m} \xrightarrow{t}_{\mathbb{Z}} \boldsymbol{m}'$ iff $\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{\Delta}_t$. Note that transitions are always firable under this semantics. Moreover, they may lead to "markings" with negative components.

Another source of hardness comes from the fact that markings are discrete. Hence, we can further relax $\to_{\mathbb{Z}}$ into $\to_{\mathbb{Q}}$ where transitions may be scaled down:

$$\boldsymbol{m} \xrightarrow{t}_{\mathbb{Q}} \boldsymbol{m}' \iff \boldsymbol{m}' = \boldsymbol{m} + \delta \cdot \boldsymbol{\Delta}_t \text{ for some } 0 < \delta \leq 1.$$

One gets a less crude relaxation from considering *nonnegative* "markings" only:

$$\boldsymbol{m} \xrightarrow{t}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}' \iff (\boldsymbol{m} \geq \delta \cdot \boldsymbol{g}_t) \text{ and } (\boldsymbol{m}' = \boldsymbol{m} + \delta \cdot \boldsymbol{\Delta}_t) \text{ for some } 0 < \delta \leq 1.$$

Under these, we obtain "markings" from $\mathbb{Q}^P$ and $\mathbb{Q}^P_{\geq 0}$ respectively. Petri nets equipped with relation $\to_{\mathbb{Q}_{\geq 0}}$ are known as *continuous Petri nets* [13,14].

To unify all three relaxations, we sometimes write $\boldsymbol{m} \xrightarrow{\delta t}_{\mathbb{G}} \boldsymbol{m}'$ to emphasize the scaling factor δ, where $\delta = 1$ whenever $\mathbb{G} = \mathbb{Z}$. Let $d_{\mathbb{G}} \colon \mathbb{N}^P \times \mathbb{N}^P \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ be defined as $d_{\mathbb{G}}(\boldsymbol{m}, \boldsymbol{m}') \coloneqq \infty$ if $\boldsymbol{m} \not\xrightarrow{*}_{\mathbb{G}} \boldsymbol{m}'$, and otherwise:

$$d_{\mathbb{G}}(\boldsymbol{m}, \boldsymbol{m}') \coloneqq \min \left\{ \sum_{i=1}^{n} \delta_i \cdot \lambda(t_i) : \boldsymbol{m} \xrightarrow{\delta_1 t_1 \cdots \delta_n t_n}_{\mathbb{G}} \boldsymbol{m}' \right\}.$$

In words, $d_{\mathbb{G}}(\boldsymbol{m}, \boldsymbol{m}')$ is the weight of a shortest path from $\boldsymbol{m}$ to $\boldsymbol{m}'$ in the graph induced by the relaxed step relation $\to_{\mathbb{G}}$, where weights are scaled accordingly.

We now show that any $d_{\mathbb{G}}$, which we call the $\mathbb{G}$-*distance*, is a distance under-approximation, and first show effectiveness of all $d_{\mathbb{G}}$. It is well-known and readily seen that reachability over $\mathbb{G} \in \{\mathbb{Z}, \mathbb{Q}\}$ is characterized by the following *state equation*, since transitions are always firable due to the absence of guards:

$$\boldsymbol{m} \xrightarrow{*}_{\mathbb{G}} \boldsymbol{m}' \iff \exists \boldsymbol{\sigma} \in \mathbb{G}^T_{\geq 0} : \boldsymbol{m}' = \boldsymbol{m} + \sum_{t \in T} \boldsymbol{\sigma}(t) \cdot \boldsymbol{\Delta}_t.$$

Here, $\boldsymbol{\sigma}$ can be seen as the Parikh image of a sequence σ leading from $\boldsymbol{m}$ to $\boldsymbol{m}'$.

Proposition 2. *The functions $d_{\mathbb{Z}}$, $d_{\mathbb{Q}}$, $d_{\mathbb{Q}_{\geq 0}}$ are effective.*

Proof. By the state equation, we have:

$$d_{\mathbb{G}}(\boldsymbol{m}, \boldsymbol{m}') = \min \left\{ \sum_{t \in T} \lambda(t) \cdot \boldsymbol{\sigma}(t) : \boldsymbol{\sigma} \in \mathbb{G}^T_{\geq 0}, \boldsymbol{m}' = \boldsymbol{m} + \sum_{t \in T} \boldsymbol{\sigma}(t) \cdot \boldsymbol{\Delta}_t \right\}.$$

Therefore, $d_{\mathbb{Q}}(\boldsymbol{m}, \boldsymbol{m}')$ (resp. $d_{\mathbb{Z}}(\boldsymbol{m}, \boldsymbol{m}')$) are computable by (resp. integer) linear programming, which is is complete for P (resp. NP), in its variant where one must check whether the minimal solution is at most some bound.

For $d_{\mathbb{Q}_{\geq 0}}$, note that the reachability relation of a continuous Petri net can be expressed in the existential fragment of linear real arithmetic [8]. Hence, effectiveness follows from the decidability of linear real arithmetic. □

Altogether, we conclude that $d_{\mathbb{G}}$ is a distance under-approximation. Furthermore, we can show that $d_{\mathbb{G}}$ yields *unbounded* heuristics, which, by Theorem 1, ensure termination of GBFS on reachable instances:

Theorem 2. *Let $\mathbb{G} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{Q}_{\geq 0}\}$, then $d_{\mathbb{G}}$ is a distance under-approximation. Moreover, the heuristics arising from it are unbounded.*

Proof. Let $\mathcal{N} = (P, T, f, \lambda)$ be a weighted Petri net. Effectiveness of $d_{\mathbb{G}}$ follows from Proposition 2. By definitions and a simple induction, $\xrightarrow{\sigma}_{\mathbb{N}} \subseteq \xrightarrow{\sigma}_{\mathbb{G}}$ for any sequence $\sigma \in T^*$, with weights left unchanged for unscaled transitions. This implies that $d_{\mathbb{G}}(\boldsymbol{m}, \boldsymbol{m}') \leq \mathrm{dist}_{\mathcal{N}}(\boldsymbol{m}, \boldsymbol{m}')$ for every $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{G}^P$. Moreover, the triangle inequality holds since for every $\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{m}'' \in \mathbb{G}^P$ and sequences σ, σ':

$$\boldsymbol{m} \xrightarrow{\sigma}_{\mathbb{G}} \boldsymbol{m}' \xrightarrow{\sigma'}_{\mathbb{G}} \boldsymbol{m}'' \text{ implies } \boldsymbol{m} \xrightarrow{\sigma\sigma'}_{\mathbb{G}} \boldsymbol{m}''.$$

Let us sketch the proof of the second part. Let $\boldsymbol{m}_{\text{target}}$ be a marking and let $h_{\mathbb{G}}$ be the heuristic obtained from $d_{\mathbb{G}}$ for $\boldsymbol{m}_{\text{target}}$. Since $h_{\mathbb{Q}}(\boldsymbol{m}) \leq h_{\mathbb{G}}(\boldsymbol{m})$ for all $\boldsymbol{m}$ and $\mathbb{G} \in \{\mathbb{Z}, \mathbb{Q}_{\geq 0}\}$, it suffices to prove that $d_{\mathbb{Q}}$ is unbounded. Suppose it is not. There exist $b \in \mathbb{Q}_{\geq 0}$ and pairwise distinct markings $\boldsymbol{m}_0, \boldsymbol{m}_1, \ldots$ each with $h_{\mathbb{Q}}(\boldsymbol{m}_i) \leq b$. Let $\boldsymbol{x}_i$ be a solution to the state equation that gives $h_{\mathbb{Q}}(\boldsymbol{m}_i)$. By well-quasi-ordering and pairwise distinctness, there is a subsequence such that $\boldsymbol{m}_{i_0}(p) < \boldsymbol{m}_{i_1}(p) < \cdots$ for some $p \in P$. Thus, $\lim_{j\to\infty} \boldsymbol{m}_{\text{target}}(p) - \boldsymbol{m}_{i_j}(p) = -\infty$, and hence $\lim_{j\to\infty} \boldsymbol{x}_{i_j}(s) = \infty$ for some $s \in T$ with $\boldsymbol{\Delta}_s(p) < 0$. This means that $b \geq h_{\mathbb{Q}}(\boldsymbol{m}_{i_j}) = \sum_{t\in T} \lambda(t) \cdot \boldsymbol{x}_{i_j}(t) > b$ for a sufficiently large j. □

4.3 Directed reachability based on distance under-approximations

We have all the ingredients to use Algorithm 1 for answering reachability queries.

A *distance under-approximation scheme* is a mapping $\mathcal{D}$ that associates a distance under-approximation $\mathcal{D}(\mathcal{N})$ to each weighted Petri net $\mathcal{N}$. Let $h_{\mathcal{D}(\mathcal{N}),\boldsymbol{m}_{\text{target}}}$ be the heuristic obtained from $\mathcal{D}(\mathcal{N})$ for marking $\boldsymbol{m}_{\text{target}}$. By instantiating Algorithm 1 with this heuristic, we can search for a short(est) firing sequence witnessing that $\boldsymbol{m}_{\text{target}}$ is reachable. Of course, constructing the reachability graph of $\mathcal{N}$ would be at least as difficult as answering this query, or impossible if it is infinite. Hence, we provide $G_{\mathbb{N}}(\mathcal{N})$ *symbolically* through $\mathcal{N}$ and let Algorithm 1 explore it on-the-fly by progressively firing its transitions.

For each $\mathbb{G} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{Q}_{\geq 0}\}$, the function $\mathcal{D}_{\mathbb{G}}$ mapping a weighted Petri net $\mathcal{N}$ to its $\mathbb{G}$-distance $d_{\mathbb{G}}$ is a distance under-approximation scheme with consistent and unbounded heuristics by Proposition 1, Theorem 1 and Theorem 2. Although Algorithm 1 is geared towards finding paths, it can prove *non*-reachability even for infinite reachability graphs. Indeed, at some point, every candidate marking $\boldsymbol{m} \in C$ may be such that $h_{\mathcal{D}(\mathcal{N}),\boldsymbol{m}_{\text{target}}}(\boldsymbol{m}) = \infty$, which halts with ∞. There is no guarantee that this happens, but, as reported *e.g.* by [23,8], the $\mathbb{G}$-distance for domains $\mathbb{G} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{Q}_{\geq 0}\}$ does well for witnessing non-reachability in practice, often from the very first marking $\boldsymbol{m}_{\text{init}}$.

An example. We illustrate our approach with a toy example and $\mathcal{D}_{\mathbb{Q}}$ (the scheme based on the state equation over $\mathbb{Q}_{\geq 0}^T$). Consider the Petri net $\mathcal{N}$ illustrated on the left of Figure 1, but marked with $\boldsymbol{m}_{\text{init}} \coloneqq [p_1 \colon 0, p_2 \colon 0]$. Suppose we wish to determine whether $\boldsymbol{m}_{\text{init}}$ can reach marking $\boldsymbol{m}_{\text{target}} \coloneqq [p_1 \colon 0, p_2 \colon 1]$ in $\mathcal{N}$.

We consider the case where Algorithm 1 follows a greedy best-first search, but the markings would be expanded in the same way with A^*. Let us abbreviate a marking $[p_1 \colon x, p_2 \colon y]$ as (x, y). Since $\boldsymbol{\Delta}_{t_2} = (0, 1)$, the heuristic considers that $\boldsymbol{m}_{\text{init}}$ can reach $\boldsymbol{m}_{\text{target}}$ in a single step using transition t_2 (it is unaware of the guard). Marking $(1, 0)$ is expanded and its heuristic value increases to 2 as the state equation considers that both t_2 and t_3 must be fired (in some unknown order). Markings $(2, 0)$ and $(1, 1)$ are both discovered with respective heuristic values 3 and 1. The latter is more promising, so it is expanded and target $(0, 1)$ is discovered. Since its heuristic value is 0, it is immediately expanded and the correct distance $\text{dist}_{\mathcal{N}}(\boldsymbol{m}_{\text{init}}, \boldsymbol{m}_{\text{target}}) = 3$ is returned. Note that, in this example, the only markings expanded are precisely those occurring on the shortest path.

Handling multiple targets. Algorithm 1 can be adapted to search for *some* marking from a given target set $X \subseteq \mathbb{N}^P$. The idea consists simply in using a heuristic $h_X \colon \mathbb{N}^P \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ estimating the weight of a shortest path to *any* target:

$$h_X(\boldsymbol{m}) \coloneqq \min\{h_{\mathcal{D}(\mathcal{N}),\boldsymbol{m}_{\text{target}}}(\boldsymbol{m}) : \boldsymbol{m}_{\text{target}} \in X\}.$$

This is convenient for partial reachability instances occurring in practice, *i.e.*

$$X \coloneqq \left\{\boldsymbol{m}_{\text{target}} \in \mathbb{N}^P \colon \boldsymbol{m}_{\text{target}}(p) \sim_p \boldsymbol{c}(p)\right\} \text{ where } \boldsymbol{c} \in \mathbb{N}^P \text{ and } \textit{each} \sim_p \in \{=, \geq\}.$$

5 Experimental results

We implemented Algorithm 1 in a prototype, called FastForward, which supports all selection strategies and distance under-approximations presented in the paper. We evaluate FastForward empirically with three main goals in mind. First, we show that our approach is competitive with established tools and can even vastly outperform them, and we also give insights on its performance w.r.t. its parameterizations. Second, we compare the length of the witnesses reported by the different tools. Third, we briefly discuss the quality of the heuristics.

Technical details. Our tool is written in C# and uses Gurobi [35], a state-of-the-art MILP solver, for distance under-approximations. We performed our benchmarks on a machine with an 8-Core Intel® Core™ i7-7700 CPU @ 3.60GHz running on Ubuntu 18.04 with memory constrained to ∼8GB. We used a timeout of 60 seconds per instance, and all tools were invoked from a Python script using the `time` module for time measurements.

A minor challenge arises from the fact that many instances specify an upward-closed set of initial markings rather than a single one. For example, $\boldsymbol{m}_{\text{init}}(p) \geq 1$ to specify, *e.g.*, an arbitrary number of threads. We handle this by setting $\boldsymbol{m}_{\text{init}}(p) = 1$ and adding a transition t_p producing a token into p.

As a preprocessing step, we implemented *sign analysis* [31]. It is a general pruning technique that has been shown beneficial for reducing the size of the state-space of Petri nets. Initially, places that carry tokens are viewed as marked. For each transition whose input places are marked, the output places also become marked. When a fixpoint is reached, places left unmarked cannot carry tokens in any reachable marking, so they are discarded.

Benchmarks. Due to the lack of tools handling reachability for *unbounded* state spaces, benchmarks arising in the literature are primarily *coverability* instances[5], *i.e.* reachability towards an upward closed set of target markings. We gathered 61 positive and 115 negative coverability instances originating from five suites [42,29,6,38,18] previously used for benchmarking [23,8,31]. They arise from the analysis of multi-threaded C programs with shared-memory; mutual

[5] The Model Checking Contest focuses on reachability for *finite* state spaces.

exclusion algorithms; communication protocols; provenance analysis in the context of a medical messaging and a bug-tracking system; and the verification of Erlang concurrent programs. We further extracted the *sypet* suite made of 30 positive (standard) reachability instances arising from queries encountered in type-directed program synthesis [25]. The overall goal of this work is to enable a vast range of untapped applications requiring reachability over unbounded state-spaces, rather than just coverability. To obtain further (positive) instances of the Petri net reachability problem, we performed random walks on the Petri nets from the aforementioned coverability benchmarks. To this end, we used the largest quarter of distinct Petri nets from each coverability suite, for a total of 33. We performed one random walk each of lengths 20, 25, 30, 35, 40, 50, 60, 75, 90 and 100, and we saved the resulting marking as the target. For nets with an upward-closed initial marking, we randomly chose to start with a number of tokens between 1 and 20% of the length of the walk. It is important to note that even with long random walks, instances can (and in fact tend to) have short witnesses. To remove trivial instances and only keep the most challenging ones, we removed those instances where FastForward or LoLA reported a witness of length at most 20, disregarding the transitions used to generate the initial marking. This leaves us with 127 challenging instances on which the shortest witness is either unknown or has length more than 20. Moreover, this yields real-world Petri nets with no bias towards any specific kind of targets.

This table summarizes the characteristics of the various benchmarks:

Suite	**Size**	**Number of places**				**Number of transitions**			
		min.	med.	mean	max.	min.	med.	mean	max.
coverability	61	16	87	226	2826	14	181	1519	27370
sypet	30	65	251	320	1199	537	2307	2646	8340
random walks	127	52	306	531	2826	60	3137	5885	27370

Tool comparison. To evaluate our approach on reachability instances, we compare FastForward to LoLA [56], a tool developed for two decades that wins several categories of the Model Checking Contest every year. LoLA is geared towards model checking of finite state spaces, but it implements semi-decision procedures for the unbounded case. We further compare the three selection strategies of Algorithm 1: A^*, GBFS and Dijkstra; the two first with the distance under-approximation scheme $\mathcal{D}_{\mathbb{Q}}$, which provides the best trade-off between estimate quality and efficiency. We also considered comparing with KReach [17], a tool showcased at TACAS'20 that implements an exact non-elementary algorithm. However, it timed out on all instances, even with larger time limits.

Figure 2 depicts the number of reachability instances decided by the tools within the time limit. As shown, all approaches outperform LoLA, with GBFS as the clear winner on the random-walk suite and A^* slightly better on the sypet suite. Note that Dijkstra's selection strategy sometimes competes due to its locally very cheap computational cost (no heuristic evaluation), but its performance generally decreases as the distance increases.

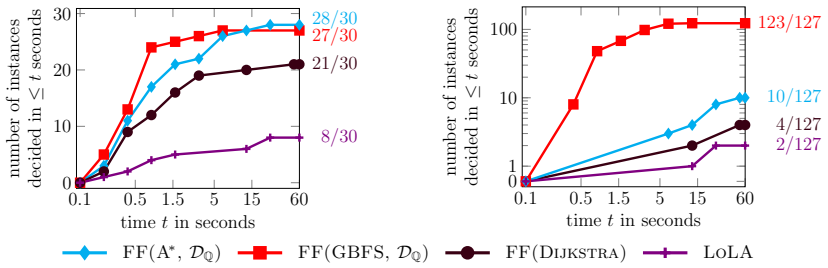


Fig. 2. Cumulative number of reachability instances decided over time. *Left*: SYPET suite (semi-log scale). *Right*: RANDOM-WALK suite (log scale).

To demonstrate the versatility of our approach, we also benchmarked FastForward on the original coverability instances. Recall that coverability is an EXPSPACE-complete problem that reduces to reachability in linear time [48,54]. While its complexity exceeds the PSPACE-completeness of reachability for finite state-spaces [41,21], it is much more tame than the non-elementary complexity of (unbounded) reachability. We compare FastForward to four tools implementing algorithms tailored specifically to the coverability problem: LoLA, Bfc [42], ICover [31] and the backward algorithm (based on [1]) of mist [29]. We did not test Petrinizer [23] since it only handles negative instances, while we focus on positive ones; likewise for QCover [8] since it is superseded by ICover.

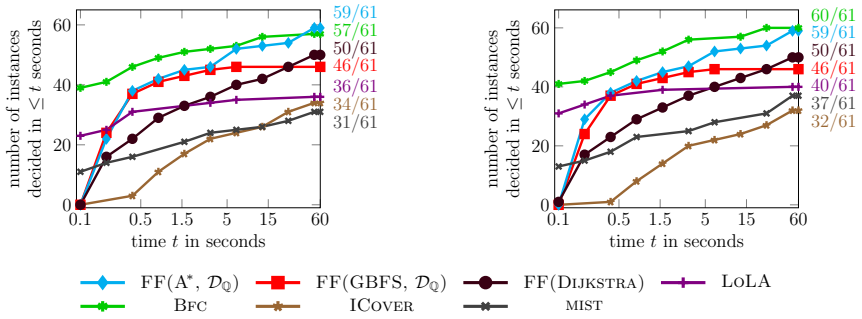


Fig. 3. Cumulative number of (positive) coverability instances decided over time. *Left*: Evaluation on the original instances. *Right*: Evaluation on the pre-pruned instances.

Figure 3 illustrates the number of coverability instances decided within the time limit. The left side corresponds to an evaluation on the original instances where FastForward performs pruning (included in its runtime). On the right hand right side the pruned instances are the input for all tools, and the time for this pruning is not included for any tool. As a caveat, ICover performs its

own preprocessing which includes pruning among techniques specific to coverability. This preprocessing is enabled (and its time is included) even when pruning is already done. Using FastForward(A^*, $\mathcal{D}_{\mathbb{Q}}$), we decide more instances than all tools on unpruned Petri nets, and one less than Bfc for pre-pruned instances. It is worth mentioning that with a time limit of 10 minutes per instance, FastForward(A^*, $\mathcal{D}_{\mathbb{Q}}$) is the only tool to decide all 61 instances.

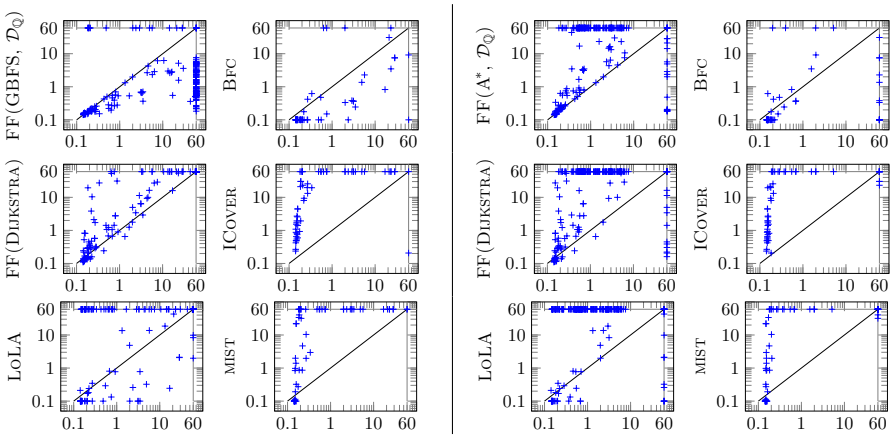


Fig. 4. Runtime comparison against FF(A^*, $\mathcal{D}_{\mathbb{Q}}$) (*left*) and FF(GBFS, $\mathcal{D}_{\mathbb{Q}}$) (*right*), in seconds, for individual instances without pre-pruning. Tools on the first column of each side include coverability and reachability instances, while those on the second column of each side include coverability only. Marks on the gray lines denote timeouts (60 s).

We also compared the running time of A^* and GBFS with $\mathcal{D}_{\mathbb{Q}}$ to the other tools and approaches. For each tool, we considered the type of instances it can handle: either reachability and coverability, or coverability only. Figure 4 depicts this comparison, where the base approach is faster for data points that lie in the upper-left half of the graph. The axes start at 0.1 second to avoid a comparison based on technical aspects such as the programming language. Yet, LoLA, Bfc and mist regularly solve instances faster than this, which speaks to their level of optimization. We can see that FastForward outperforms ICover, LoLA and mist overall. We cannot compete with Bfc in execution time as it is a highly optimized tool specifically tailored to only the coverability problem that can employ optimization techniques such as Karp-Miller trees that do not work for reachability queries.

Length of the witnesses. Since our approach is also geared towards the identification of short(est) reachability witnesses, we compared the different tools with respect to length of the reported one, depicted in Figure 5. Positive values on the y-axis mean the witness was not minimal, while $y = 0$ means it was.

Note that the points for BFC must be taken with a grain of salt: it uses a different file format, and its translation utility can introduce additional transitions. This means that even if BFC found a shortest witness, it could be longer than a shortest one of the original instance.

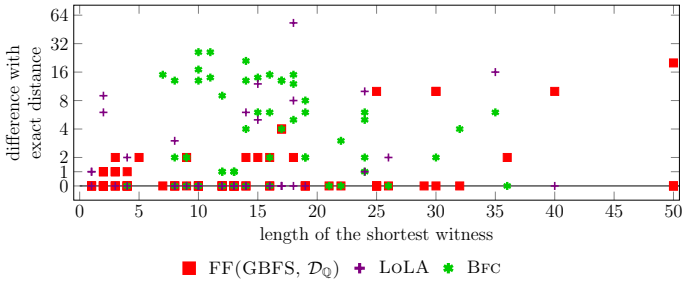


Fig. 5. Length of the returned witness, per tool, compared to the length of a shortest witness. ICOVER is left out as it does not return witnesses. FF(A*, $\mathcal{D}_{\mathbb{Q}}$), FF(DIJKSTRA) and MIST are left out as they are guaranteed to return shortest witnesses.

Still, the graph shows that reported witnesses can be far from minimal. For example, on one instance LoLA returns a witness that is 53 transitions longer than the one of FASTFORWARD(A*, $\mathcal{D}_{\mathbb{Q}}$). Still, LoLA returns a shortest witness on 28 out of 43 instances. Similarly, FASTFORWARD(GBFS, $\mathcal{D}_{\mathbb{Q}}$) finds a shortest path on 60 out of 83 instances[6]. In contrast, MIST finds a shortest witness on all instances since its backward algorithm is guaranteed to do so on unweighted Petri nets, which constitute all of our instances. Again, this approach is tailored to coverability and cannot be lifted to reachability.

Heuristics and pruning. We briefly discuss the quality of the heuristics and the impact of pruning. The left-hand side of Figure 6 compares the exact distance to the estimated distance from the initial marking.[7] It shows that it is incredibly accurate for all $\mathbb{G}$-distances, but even more so for $\mathbb{G} = \mathbb{Q}_{\geq 0}$. We experimented with this distance using the logical translation of [8] and Z3 [52] as the optimization modulo theories solver. At present, it appears that the gain in estimate quality does not compensate for the extra computational cost.

As depicted on the right-hand side of Figure 6, pruning can make some instances trivial, but in general, many challenging instances remain so. On average, around 50% of places and 40% of transitions were pruned.

[6] These numbers disregard instances where the tool did not finish or where a shortest witness is not known, *i.e.* no method guaranteeing one finished in time.

[7] Z3 reported two non optimal solutions which explains the two points above the line.

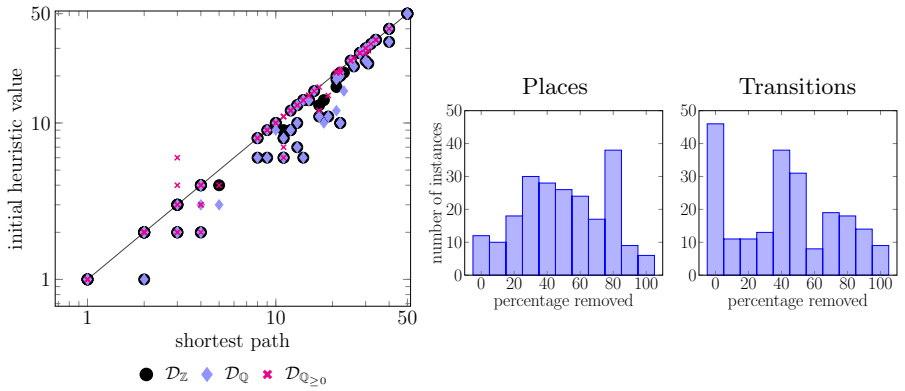


Fig. 6. *Left*: initial distance estimation compared to the exact distance (points closer to the diagonal are better). *Right*: number of instances per percentage of places (left) and transitions (right) removed by pruning (rounded to nearest multiple of 10).

6 Conclusion

We presented an efficient approach to the Petri net reachability problem that uses state-space over-approximations as distance oracles in the classical graph traversal algorithms A^* and greedy best-first search. Our experiments have shown that using the state equation over $\mathbb{Q}^T_{\geq 0}$ provides the best trade-off between computational feasibility and the accuracy of the oracle. However, we expect that further advances in optimization modulo theories solvers may enable employing stronger over-approximations such as continuous Petri nets in the future.

Moreover, non-algebraic distance under-approximations also fit naturally in our framework, *e.g.* the syntactic distance of [58] and "α-graphs" of [25]. These are crude approximations with low computational cost. Our preliminary tests show that, although they could not compete with our distances, they can provide early speed-ups on instances with large branching factors. An interesting line of research consists in identifying cheap approximations with better estimates.

We wish to emphasize that our approach to the reachability problem has the potential to also be naturally used for semi-deciding reachabiltiy in extensions of Petri nets with a recursively enumerable reachability problem, such as Petri nets with resets and transfers [3,19] as well as colored Petri nets [40]. These extensions have, for instance, been used for the generation of program loop invariants [57], the validation of business processes [62] and the verification of multi-threaded C and Java program skeletons with communication primitives [15,42]. Linear rational and integer arithmetic over-approximations for such extended Petri nets exist [11,9,37,34] and could smoothly be used inside our framework.

Acknowledgments

We thank Juliette Fournis d'Albiat for her help with extracting the SYPET suite.

References

1. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.: General decidability theorems for infinite-state systems. In: Proc. 11th Annual IEEE Symposium on Logic in Computer Science (LICS). pp. 313–321. IEEE Computer Society (1996). https://doi.org/10.1109/LICS.1996.561359
2. Angeli, D., De Leenheer, P., Sontag, E.D.: A Petri net approach to the study of persistence in chemical reaction networks. Mathematical Biosciences **210**(2), 598–618 (2007). https://doi.org/10.1016/j.mbs.2007.07.003
3. Araki, T., Kasami, T.: Some decision problems related to the reachability problem for Petri nets. Theoretical Computer Science **3**(1), 85–104 (1976). https://doi.org/10.1016/0304-3975(76)90067-0
4. Athanasiou, K., Liu, P., Wahl, T.: Unbounded-thread program verification using thread-state equations. In: Proc. 8th International Joint Conference on Automated Reasoning (IJCAR). pp. 516–531. Springer (2016). https://doi.org/10.1007/978-3-319-40229-1_35
5. Bansal, K., Koskinen, E., Wies, T., Zufferey, D.: Structural counter abstraction. In: Proc. 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 62–77. Springer (2013). https://doi.org/10.1007/978-3-642-36742-7_5
6. Barth, A., Mitchell, J.C., Datta, A., Sundaram, S.: Privacy and utility in business processes. In: Proc. 20th IEEE Computer Security Foundations Symposium (CSF). pp. 279–294. IEEE Computer Society (2007). https://doi.org/10.1109/CSF.2007.26
7. Bjørner, N., Phan, A., Fleckenstein, L.: νZ - an optimizing SMT solver. In: Proc. 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 194–199. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_14
8. Blondin, M., Finkel, A., Haase, C., Haddad, S.: The logical view on continuous Petri nets. ACM Transactions on Computational Logic (TOCL) **18**(3), 24:1–24:28 (2017). https://doi.org/10.1145/3105908
9. Blondin, M., Haase, C., Mazowiecki, F.: Affine extensions of integer vector addition systems with states. In: Proc. 29th International Conference on Concurrency Theory (CONCUR). pp. 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.CONCUR.2018.14
10. Chaouiya, C.: Petri net modelling of biological networks. Briefings in Bioinformatics **8**(4), 210–219 (2007). https://doi.org/10.1093/bib/bbm029
11. Chistikov, D., Haase, C., Halfon, S.: Context-free commutative grammars with integer counters and resets. Theoretical Computer Science **735**, 147–161 (2018). https://doi.org/10.1016/j.tcs.2016.06.017
12. Czerwiński, W., Lasota, S., Lazić, R., Leroux, J., Mazowiecki, F.: The reachability problem for Petri nets is not elementary. In: Proc. 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC). pp. 24–33. ACM (2019). https://doi.org/10.1145/3313276.3316369
13. David, R., Alla, H.: Continuous Petri nets. In: Proc. 8th European Workshop on Application and Theory of Petri nets. vol. 340, pp. 275–294 (1987)
14. David, R., Alla, H.: Discrete, Continuous, and Hybrid Petri nets. Springer, 2nd edn. (2010)
15. Delzanno, G., Raskin, J., Van Begin, L.: Towards the automated verification of multithreaded Java programs. In: Proc. 8th International Conference on Tools and

Algorithms for the Construction and Analysis of Systems (TACAS). pp. 173–187. Springer (2002). https://doi.org/10.1007/3-540-46002-0_13
16. Deutsch, A., Li, Y., Vianu, V.: Verification of hierarchical artifact systems. ACM Transactions on Database Systems (TODS) **44**(3), 12:1–12:68 (2019). https://doi.org/10.1145/3321487
17. Dixon, A., Lazić, R.: Kreach: A tool for reachability in Petri nets. In: Proc. 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 405–412. Springer (2020). https://doi.org/10.1007/978-3-030-45190-5_22
18. D'Osualdo, E., Kochems, J., Ong, C.L.: Automatic verification of erlang-style concurrency. In: Proc. 20th International Symposium on Static Analysis (SAS). pp. 454–476. Springer (2013). https://doi.org/10.1007/978-3-642-38856-9_24
19. Dufourd, C., Finkel, A., Schnoebelen, P.: Reset nets between decidability and undecidability. In: Proc. 25th International Colloquium on Automata, Languages and Programming (ICALP). pp. 103–115. Springer (1998). https://doi.org/10.1007/BFb0055044
20. Edelkamp, S., Schuppan, V., Bosnacki, D., Wijs, A., Fehnker, A., Aljazzar, H.: Survey on directed model checking. In: Proc. 5th International Workshop on Model Checking and Artificial Intelligence (MoChArt). pp. 65–89. Springer (2008). https://doi.org/10.1007/978-3-642-00431-5_5
21. Esparza, J.: Decidability and complexity of Petri net problems — An introduction, pp. 374–428. Springer (1998). https://doi.org/10.1007/3-540-65306-6_20
22. Esparza, J., Ganty, P., Leroux, J., Majumdar, R.: Verification of population protocols. Acta Informatica **54**(2), 191–215 (2017). https://doi.org/10.1007/s00236-016-0272-3
23. Esparza, J., Ledesma-Garza, R., Majumdar, R., Meyer, P.J., Nikšić, F.: An SMT-based approach to coverability analysis. In: Proc. 26th International Conference on Computer Aided Verification (CAV). pp. 603–619. Springer (2014). https://doi.org/10.1007/978-3-319-08867-9_40
24. Farzan, A., Kincaid, Z., Podelski, A.: Proofs that count. In: Proc. 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). pp. 151–164. ACM (2014). https://doi.org/10.1145/2535838.2535885
25. Feng, Y., Martins, R., Wang, Y., Dillig, I., Reps, T.W.: Component-based synthesis for complex APIs. In: Proc. 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL). pp. 599–612. ACM (2017). https://doi.org/10.1145/3009837.3009851
26. Fraca, E., Haddad, S.: Complexity analysis of continuous Petri nets. Fundamenta Informaticae **137**(1), 1–28 (2015). https://doi.org/10.3233/FI-2015-1168
27. Galenson, J.: Dynamic and Interactive Synthesis of Code Snippets. Ph.D. thesis, University of California (2014)
28. Galenson, J., Reames, P., Bodík, R., Hartmann, B., Sen, K.: Codehint: dynamic and interactive synthesis of code snippets. In: Proc. 36th International Conference on Software Engineering (ICSE). pp. 653–663. ACM (2014). https://doi.org/10.1145/2568225.2568250
29. Ganty, P.: Algorithmes et structures de données efficaces pour la manipulation de contraintes sur les intervalles. Master's thesis, Université Libre de Bruxelles (2002), (In French)
30. Ganty, P., Majumdar, R.: Algorithmic verification of asynchronous programs. ACM Transactions on Programming Languages and Systems (TOPLAS) **34**(1), 6:1–6:48 (2012). https://doi.org/10.1145/2160910.2160915

31. Geffroy, T., Leroux, J., Sutre, G.: Occam's razor applied to the Petri net coverability problem. Theoretical Computer Science **750**, 38–52 (2018). https://doi.org/10.1016/j.tcs.2018.04.014
32. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. Journal of the ACM **39**(3), 675–735 (1992). https://doi.org/10.1145/146637.146681
33. Guo, Z., James, M., Justo, D., Zhou, J., Wang, Z., Jhala, R., Polikarpova, N.: Program synthesis by type-guided abstraction refinement. Proc. ACM on Programming Languages (POPL) **4**(12), 1–28 (2020). https://doi.org/10.1145/3371080
34. Gupta, U., Shah, P., Akshay, S., Hofman, P.: Continuous reachability for unordered data Petri nets is in PTime. In: Proc. 22nd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS). pp. 260–276. Springer (2019). https://doi.org/10.1007/978-3-030-17127-8_15
35. Gurobi Optimization, L.: Gurobi optimizer reference manual (2020), `http://www.gurobi.com`
36. Heiner, M., Gilbert, D.R., Donaldson, R.: Petri nets for systems and synthetic biology. In: Formal Methods for Computational Systems Biology. pp. 215–264. Springer (2008). https://doi.org/10.1007/978-3-540-68894-5_7
37. Hofman, P., Leroux, J., Totzke, P.: Linear combinations of unordered data vectors. In: Proc. 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). pp. 1–11. IEEE Computer Society (2017). https://doi.org/10.1109/LICS.2017.8005065
38. Janák, J.: Issue Tracking Systems. Master's thesis, Masaryk University (2009)
39. Jeng, M.D., Chen, S.C.: A heuristic search approach using approximate solutions to Petri net state equations for scheduling flexible manufacturing systems. International Journal of Flexible Manufacturing Systems **10**(2), 139–162 (1998). https://doi.org/10.1023/A:1008097430956
40. Jensen, K.: Coloured Petri nets: basic concepts, analysis methods and practical use, vol. 1. Springer Science & Business Media (2013)
41. Jones, N.D., Landweber, L.H., Lien, Y.E.: Complexity of some problems in Petri nets. Theoretical Computer Science **4**(3), 277–299 (1977). https://doi.org/10.1016/0304-3975(77)90014-7
42. Kaiser, A., Kroening, D., Wahl, T.: A widening approach to multithreaded program verification. ACM Transactions on Programming Languages and Systems (TOPLAS) **36**(4), 14:1–14:29 (2014). https://doi.org/10.1145/2629608
43. Kosaraju, S.R.: Decidability of reachability in vector addition systems (preliminary version). In: Proc. 14th Symposium on Theory of Computing (STOC). pp. 267–281. ACM (1982). https://doi.org/10.1145/800070.802201
44. Lambert, J.: A structure to decide reachability in Petri nets. Theoretical Computer Science **99**(1), 79–104 (1992). https://doi.org/10.1016/0304-3975(92)90173-D
45. Lee, D.Y., DiCesare, F.: Scheduling flexible manufacturing systems using Petri nets and heuristic search. IEEE Transactions on robotics and automation **10**(2), 123–132 (1994). https://doi.org/10.1109/70.282537
46. Leroux, J.: Vector addition systems reachability problem (A simpler solution). In: Turing-100 – The Alan Turing Centenary. pp. 214–228. EasyChair (2012)
47. Leroux, J., Schmitz, S.: Demystifying reachability in vector addition systems. In: Proc. 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). pp. 56–67. IEEE Computer Society (2015). https://doi.org/10.1109/LICS.2015.16
48. Lipton, R.J.: The reachability problem requires exponential space. Tech. rep., Yale University (1976)

49. Liu, B., Dong, W., Zhang, Y.: Accelerating API-based program synthesis via API usage pattern mining. IEEE Access **7**, 159162–159176 (2019). https://doi.org/10.1109/ACCESS.2019.2950232
50. Mayr, E.W.: An algorithm for the general Petri net reachability problem. In: Proc. 13th Symposium on Theory of Computing (STOC). pp. 238–246. ACM (1981). https://doi.org/10.1145/800076.802477
51. Mejía, G., Odrey, N.G.: An approach using Petri nets and improved heuristic search for manufacturing system scheduling. Journal of Manufacturing Systems **24**(2), 79–92 (2005). https://doi.org/10.1016/S0278-6125(05)80009-3
52. de Moura, L.M., Bjørner, N.: Z3: an efficient SMT solver. In: Proc. 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 337–340. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_24, tool available at `https://github.com/Z3Prover/z3`.
53. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989). https://doi.org/10.1109/5.24143
54. Rackoff, C.: The covering and boundedness problems for vector addition systems. Theoretical Computer Science **6**, 223–231 (1978). https://doi.org/10.1016/0304-3975(78)90036-1
55. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall Press, 3rd edn. (2009)
56. Schmidt, K.: LoLA: A low level analyser. In: Proc. International Conference on Application and Theory of Petri Nets (ICATPN). pp. 465–474. Springer (2000). https://doi.org/10.1007/3-540-44988-4_27
57. Silverman, J., Kincaid, Z.: Loop summarization with rational vector addition systems. In: Proc. 31st International Conference on Computer Aided Verification (CAV). pp. 97–115. Springer (2019). https://doi.org/10.1007/978-3-030-25543-5_7
58. Strazny, T.: An algorithmic framework for checking coverability in well-structured transition systems. Ph.D. thesis, Universität Oldenburg (2014), `http://csd.informatik.uni-oldenburg.de/~skript/pub/diss/strazny-phdthesis-roterbericht.pdf`
59. Uma, G., Prasad, B.: Reachability trees for Petri nets: a heuristic approach. Knowledge-Based Systems **6**(3), 174 – 177 (1993). https://doi.org/10.1016/0950-7051(93)90042-R
60. van der Aalst, W.: The application of Petri nets to workflow management. Journal of Circuits, Systems, and Computers **8**(1), 21–66 (1998). https://doi.org/10.1142/S0218126698000043
61. Watel, D., Weisser, M., Barth, D.: Parameterized complexity and approximability of coverability problems in weighted Petri nets. In: Proc. 38th International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS). pp. 330–349. Springer (2017). https://doi.org/10.1007/978-3-319-57861-3_19
62. Wynn, M.T., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Synchronization and cancelation in workflows based on reset nets. International Journal of Cooperative Information Systems **18**(1), 63–114 (2009). https://doi.org/10.1142/S0218843009002002
63. Yang, C.H., Dill, D.L.: Validation with guided search of the state space. In: Proc. 35th Conference on Design Automation (DAC). pp. 599–604. ACM (1998). https://doi.org/10.1145/277044.277201
64. Zuck, L.D., Pnueli, A.: Model checking and abstraction to the aid of parameterized systems (a survey). Computer Languages, Systems & Structures **30**(3-4), 139–169 (2004). https://doi.org/10.1016/j.cl.2004.02.006

A A primer on applications of Petri net reachability

This section provides two representative examples from the literature that illustrate the important role of Petri net reachability. They allow us to underpin our claim that it is desirable to find shortest paths witnessing reachability. Our examples come from program synthesis and concurrent program analysis. We conclude this section with a brief discussion on further applications.

Program synthesis. The authors of [25] and [33] have recently employed the Petri net reachability problem for automated program synthesis. In their setting, one is given an API containing hundreds or thousands of functions, together with a type signature and a number of test cases. The goal is to automatically synthesize a loop-free program using functions from the API that respects the specified type signature and satisfies the given test cases.

```
                    java.awt.geom
new AffineTransformation()
Shape Shape.createTransformedShape(AffineTransformation)
String Point2D.ToString()
double Point2D.getX()
double Point2D.getY()
void AffineTransformation.setToRotation(double, double, double)
void AffineTransformation.invert()
Area Area.createTransformedArea(AffineTransformation)
```

Fig. 7. A small sample of methods from library `java.awt.geom`.

Let us illustrate the approach with an example from [25]. Suppose we have access to library `java.awt.geom`, and we wish to synthesize a function `rotate` with type signature

`Area rotate(Area object, Point2D point, double angle).`

Naturally, the function should rotate the supplied `Area` around `point` by `angle` degrees. We assume the `java.awt.geom` library is sufficient for this task in that it contains the functions needed to synthesize the method. Figure 7 presents an excerpt of functions contained in the API.

The authors of [25] suggest to view an API as a Petri net whose places correspond to types and transitions correspond to API functions which, informally speaking, consume input types and produce an output type. Figure 8 illustrates the Petri net corresponding to the excerpt of API functions listed in Figure 7. To synthesize the `rotate` function above, we start with tokens in the places corresponding to the input parameters of our function. Thus, in Figure 8 we have one token in each of the places corresponding to `Area`, `Point2D` and `double`. The goal is then to reach a marking with a single token in the place corresponding to the return type. In our example, we aim for one token in `Area`, and no token in

any other place. This corresponds to invoking a sequence of functions that "use up" all input parameters, and finally return the correct type. To allow reuse of variables, additional "copy" transitions are introduced for each place; they take one token from a place and put two tokens back. If the target marking is reachable, then the witnessing path corresponds to a partial sketch of a program.

For example, the path

$$\text{copy}_{\text{Point2D}} \rightarrow \text{GetY} \rightarrow \text{GetX} \rightarrow \text{new AffineTransformation} \rightarrow \\ \text{copy}_{\text{AffineTransformation}} \rightarrow \text{setToRotation} \rightarrow \text{createTransformedArea}$$

tells us which functions to apply, and in which order to apply them. Since Petri nets do not store information about the identity of tokens, when we have multiple objects of the same type, we do not know which to supply as an argument to which function. This can be figured out by a separate process involving SAT solving (see [25] for more details).

As discussed in [25], finding short paths of the Petri net is a natural goal. Indeed, since short programs are easier to test, there are fewer possibilities for the arguments of each function, and it is easier for humans to verify that the synthesized program has the desired functionality.

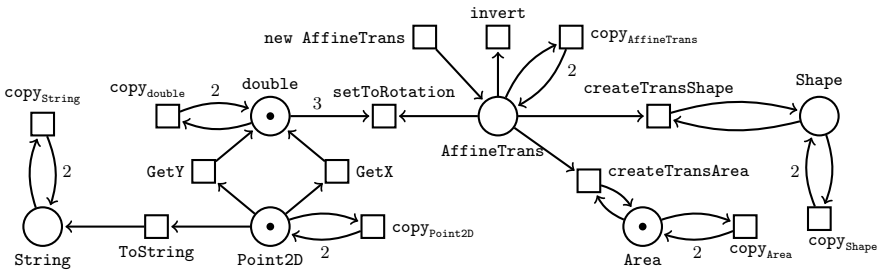


Fig. 8. A Petri net modelling the API of Figure 7.

Concurrent program analysis. Perhaps most prominently, Petri nets have been used in order to model and analyze concurrent processes. Let us begin with a simple example illustrating how the Petri net reachability problem can be used in order to detect race conditions in concurrent programs. Consider function `fun()` of Figure 9 in which `s` is a global shared Boolean variable. If there is a single thread running `fun()`, then the condition of the `if`-statement in Line 3 never evaluates to true and an error cannot occur. However, if there are two independently interleaved threads running `fun()`, it is possible that one thread reaches Line 3 whilst `s` is set to 1, which means an error could occur.

In more technical terms, we consider non-recursive Boolean programs in which an unbounded number of identical programs run in parallel. The authors

```
0 def fun():
1   s = 1
2   s = 0
3   if s == 1: raise Err()
```

Fig. 9. Simple program with a potential race condition.

of [32] showed that verifying safety properties of such concurrent programs can be reduced to the *coverability problem* for Petri nets using a technique called counter abstraction. The coverability problem is a weaker version of the reachability problem. Given a target marking, the coverability problem asks whether it is possible to reach a marking in which every place carries at least as many tokens as specified by the target marking. The Petri net obtained by applying the approach of [32] to the program from Figure 9 is depicted in Figure 10. The places on the top of the Petri net correspond to the program locations of Figure 9. Tokens in each of the places on the top count the number of threads which are currently at the respective program location, which is a form of counter abstraction. At any time, transition `fun()` can add tokens to $\texttt{loc}_1$, reflecting that a new thread executing `fun()` can be spawned at any point in time arbitrarily often. The two places on the bottom encode the state of the Boolean variable `s` which is updated whenever a transition moves tokens from $\texttt{loc}_1$ to $\texttt{loc}_2$, or from $\texttt{loc}_2$ to $\texttt{loc}_3$. Determining whether an error can occur then reduces to deciding whether the marking [`Err`: 1] is coverable, *i.e.*, whether there is an interleaving in which at least one thread produces an error.

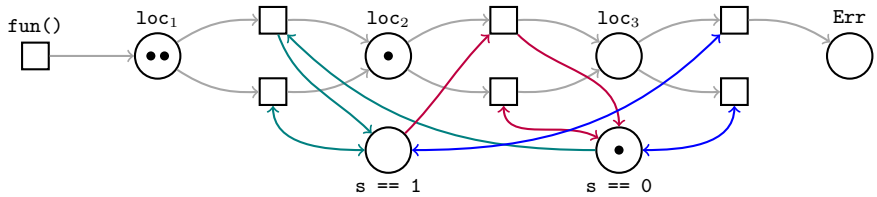


Fig. 10. The Petri net modeling the program of Figure 9. A token in place $\texttt{loc}_i$ represents a thread at program location i, and a token in place `s == b` indicates that variable `s` has value b. Transition `fun()` spawns threads. A bidirectional arc $p \leftrightarrow t$ abbreviates two arcs: $p \rightarrow t$ and $t \rightarrow p$. Colors are only meant to help readability.

In stark contrast to the reachability problem, it was shown in [54] that the coverability problem belongs to EXPSPACE. There is a natural reduction from the coverability problem to the reachability problem: by introducing additional transitions that can non-deterministically remove tokens from every place corresponding to program lines, a target marking in the original Petri net is coverable iff it is reachable in the Petri net with the additional transitions. Alternatively,

deciding coverability can be rephrased as the problem of determining whether an upward-closed set of markings is reachable in the directed graph induced by a given Petri net, which is the approach that we take.

Further applications The authors of [24] show how proofs involving counting arguments, which can, for instance, naturally prove properties of concurrent programs with recursive procedures, can automatically be synthesized by a reduction to the Petri net reachability problem. The authors of [30] propose a model for reasoning about finite-data asynchronous programs. They show that proving liveness properties of such programs in their model is inter-reducible with the Petri net reachability problem. In a broader context, it was shown that various verification problems for population protocols, a formal model of sensor networks, reduce to the Petri net reachability problem [22]. The authors of [16] develop a method that allows for verifying rich models of data-driven workflows by a reduction to the coverability problem for Petri nets. See also survey [53] for further classical application areas of Petri nets and their extensions.

B Missing proofs of Section 3

Recall the following invariant satisfied by Algorithm 1:

$$\text{if } g(v) \neq \infty, \text{ then } g(v) \text{ is the weight of a path from } s \text{ to } v \text{ in } G \text{ whose nodes were all expanded, except possibly } v. \quad (*)$$

We prove this lemma from the main text:

Lemma 1. *If G is locally finite and h is unbounded, then the following holds:*

1. *The set of paths of weight at most $c \in \mathbb{Q}_{\geq 0}$ starting from node s is finite.*
2. *Let $W \subseteq V$. The set $\mathrm{dist}_G(W, t) \coloneqq \{\mathrm{dist}_G(w, t) : w \in W\}$ has a minimum.*
3. *No node is expanded infinitely often by Algorithm 1.*

Proof. Let $d \coloneqq \min\{\mu(e) : e \in E\}$.

1. Any path of weight at most c traverses at most $k \coloneqq \lceil c/d \rceil$ edges. Since the graph has finite out-degree, the number of paths from s using at most k edges is finite.
2. Suppose the claim false. We have $\mathrm{dist}_G(v_0, t) > \mathrm{dist}_G(v_1, t) > \cdots$ for some $v_0, v_1, \ldots \in W$. Let $k \coloneqq \lceil \mathrm{dist}_G(v_0, t)/d \rceil$. Let $V_{\leq k}$ be the set of nodes that can reach t by traversing at most k edges. Since G has finite in-degree, $V_{\leq k}$ is finite. Moreover, any node $v \in V \setminus V_{\leq k}$ is such that $\mathrm{dist}_G(v, t) > k \cdot d \geq \mathrm{dist}_G(v_0, t)$. Hence, $\{v_0, v_1, \ldots\} \subseteq V_{\leq k}$ is finite, which is a contradiction.
3. For the sake of contradiction, assume a node v is expanded infinitely often. Each time node v is expanded, it is removed from C. Hence, it is reinserted infinitely often in C. Moreover, each time this happens, value $g(v)$ is decreased. Let $q_0, q_1, \ldots \in \mathbb{Q}_{\geq 0}$ denote these increasingly smaller values. By $(*)$, there is a path π_i from s to v of weight q_i in G. By (1), $\{\pi_i : i \in \mathbb{N}\}$ is finite as the weight of these paths is at most q_0. This contradicts $q_0 > q_1 > \cdots$. $\square$

C Missing proofs of Section 4.2

Proposition 2. *The functions $d_{\mathbb{Z}}$, $d_{\mathbb{Q}}$, $d_{\mathbb{Q}_{\geq 0}}$ are effective.*

Proof. Let us prove the case of $d_{\mathbb{Q}_{\geq 0}}$ which was only sketched in the main text. The reachability relation of a continuous Petri net can be expressed in the existential fragment of linear real arithmetic, *i.e.* $\mathrm{FO}\langle \mathbb{Q}, +, < \rangle$, the first-order theory of the rationals with addition and order [8]. More precisely, there exists a linear-time computable formula $\psi \in \exists\,\mathrm{FO}\langle \mathbb{Q}, +, < \rangle$ such that $\psi(\boldsymbol{m}, \boldsymbol{x}, \boldsymbol{m}')$ holds iff

$$\text{there exists a sequence } \sigma \in ((0,1] \times T)^* \text{ s.t. } \boldsymbol{m} \xrightarrow{\sigma}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}' \text{ and } \boldsymbol{\sigma} = \boldsymbol{x}.$$

Let $\Phi(\boldsymbol{m}, \boldsymbol{m}', \ell) \coloneqq \exists \boldsymbol{x} \in \mathbb{Q}_{\geq 0}^T : \psi(\boldsymbol{m}, \boldsymbol{x}, \boldsymbol{m}') \wedge \ell = \sum_{t \in T} \lambda(t) \cdot \boldsymbol{x}(t)$. Formula $\Phi \in \exists\,\mathrm{FO}\langle \mathbb{Q}, +, < \rangle$ can be constructed in linear time and is such that $\Phi(\boldsymbol{m}, \boldsymbol{m}', \ell)$ holds for $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{Q}_{\geq 0}^P$ and $\ell \in \mathbb{Q}_{\geq 0}$ iff $\ell = d_{\mathbb{Q}_{\geq 0}}(\boldsymbol{m}, \boldsymbol{m}')$. Thus, $d_{\mathbb{Q}_{\geq 0}}$ is computable as an instance of a decidable optimization modulo theories problem. □

Theorem 2. *Let $\mathbb{G} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{Q}_{\geq 0}\}$, then $d_{\mathbb{G}}$ is a distance under-approximation. Moreover, the heuristics arising from it are unbounded.*

Proof. The first was part of the statement was fully shown in the main text. Let us prove the second part more formally. Let $\mathcal{N} = (P, T, f, \lambda)$ be a weighted Petri net, let $\boldsymbol{m}_{\text{target}}$ be a target marking, and let $h_{\mathbb{G}}$ be the heuristic obtained from $d_{\mathbb{G}}$ for $\boldsymbol{m}_{\text{target}}$. Observe that $h_{\mathbb{Q}}(\boldsymbol{m}) \leq h_{\mathbb{G}}(\boldsymbol{m})$ for every marking $\boldsymbol{m}$ and every $\mathbb{G} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{Q}_{\geq 0}\}$. Hence, if $h_{\mathbb{Q}}$ is unbounded, so are all three heuristics. Thus, it suffices to prove the case $\mathbb{G} = \mathbb{Q}$.

For the sake of contradiction, suppose $h_{\mathbb{Q}}$ is not unbounded. There exists $b \in \mathbb{Q}_{\geq 0}$ and an infinite sequence of pairwise distinct markings $\boldsymbol{m}_0, \boldsymbol{m}_1, \ldots \in \mathbb{N}^P$ with $h_{\mathbb{Q}}(\boldsymbol{m}_i) \leq b$ for every $i \geq 0$. Let $\boldsymbol{x}_i \in \mathbb{Q}_{\geq 0}^T$ be a solution to the state equation over $\mathbb{Q}_{\geq 0}$ that yields $h_{\mathbb{Q}}(\boldsymbol{m}_i)$, *i.e.* such that $h_{\mathbb{Q}}(\boldsymbol{m}_i) = \sum_{t \in T} \lambda(t) \cdot \boldsymbol{x}_i(t)$ is minimized subject to

$$\boldsymbol{m}_{\text{target}} = \boldsymbol{m}_i + \sum_{t \in T} \boldsymbol{x}_i(t) \cdot \boldsymbol{\Delta}_t. \tag{1}$$

Since $\mathbb{N}^P$ is well-quasi-ordered, there exist indices $i_0 < i_1 < \cdots$ such that $\boldsymbol{m}_{i_0} \leq \boldsymbol{m}_{i_1} \leq \cdots$. Since these markings are pairwise distinct, we may assume w.l.o.g. the existence of a place $p \in P$ such that $\boldsymbol{m}_{i_0}(p) < \boldsymbol{m}_{i_1}(p) < \cdots$ (otherwise, we could extract such a subsequence).

Let us define the following constants:

$$c \coloneqq \min\{\lambda(t) : t \in T\} \text{ and } d \coloneqq \frac{b \cdot |T| \cdot \max\{|\boldsymbol{\Delta}_t(p)| : t \in T\}}{c}.$$

Let $j \geq 0$ be such that $\boldsymbol{m}_{\text{target}}(p) - \boldsymbol{m}_{i_j}(p) < -d$. Such an index j exists as p takes arbitrarily large values along our infinite sequence. By (1), we have:

$$\sum_{t \in T} \boldsymbol{x}_{i_j}(t) \cdot \boldsymbol{\Delta}_t(p) = \boldsymbol{m}_{\text{target}}(p) - \boldsymbol{m}_{i_j}(p) < -d.$$

Thus, there exists $s \in T$ such that $\boldsymbol{\Delta}_s(p) < 0$ and $\boldsymbol{x}_{i_j}(s) > b/c$. Indeed, if it was not the case, it would be impossible to obtain a negative value smaller than $-d$.

We are done since we obtain the following contradiction:

$$\begin{aligned}
h_{\mathbb{Q}}(\boldsymbol{m}_{i_j}) &= \sum_{t \in T} \lambda(t) \cdot \boldsymbol{x}_{i_j}(t) && \text{(by definition)} \\
&\geq \lambda(s) \cdot \boldsymbol{x}_{i_j}(s) && \text{(by } \lambda(t) > 0 \text{ and } \boldsymbol{x}_{i_j}(t) \geq 0 \text{ for each } t \in T\text{)} \\
&> \lambda(s) \cdot (b/c) && \text{(by } \lambda(s) > 0 \text{ and } \boldsymbol{x}_{i_j}(s) > b/c\text{)} \\
&\geq \lambda(s) \cdot (b/\lambda(s)) && \text{(by } \lambda(s) \geq c\text{)} \\
&= b \\
&\geq h_{\mathbb{Q}}(\boldsymbol{m}_{i_j}) && \text{(by boundedness).}
\end{aligned}$$

□

D Experimental results

Figure 11 depicts an evaluation on reachability instances where all tools were given the pruned Petri nets (preprocessing time not included for any tool). The results are essentially the same as those of Figure 2.

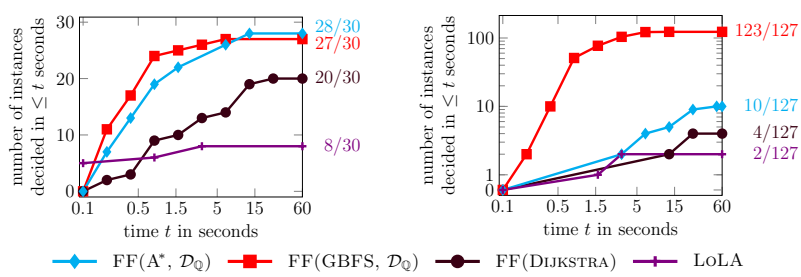


Fig. 11. Cumulative number of reachability instances decided over time (on pre-pruned instances). *Left*: SYPET suite (semi-log scale). *Right*: RANDOM-WALK suite (log scale).

E Structural distance

All three $\mathbb{G}$-distances presented in the main text have an algebraic flavor. While their complexity is significantly lower than the non-elementary time complexity of Petri net reachability, they involve solving optimization problems. An alternative avenue, mentioned in the conclusion, consists in constructing less precise but more efficient distance under-approximation based on structural properties.

We describe such a distance under-approximation adapted from the syntactic distance of [58] and related to the "α-graphs" used by [25]. Let $\mathcal{N} = (P, T, f, \lambda)$ be a weighted Petri net. The *structural abstraction* of $\mathcal{N}$ is a weighted graph

$G_{\text{struct}}(\mathcal{N})$ with places as nodes with an edge (p, t, q) iff transition t consumes tokens from p and produces tokens into q. Since some transitions may consume or produce no token, we imagine these as consuming from, or producing to, an artificial "sink place" $\perp$. Intuitively, if $\boldsymbol{m}$ can reach $\boldsymbol{m}'$, then each token of $\boldsymbol{m}$ must either make its way to $\boldsymbol{m}'$ or disappear. Of course, tokens cannot move independently and freely in $\mathcal{N}$. However, paths in $G_{\text{struct}}(\mathcal{N})$ yield a lower bound on an actual path from $\boldsymbol{m}$ to $\boldsymbol{m}'$. A structural abstraction is given in Figure 12.

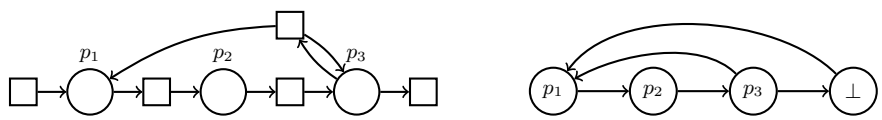


Fig. 12. *Left:* A Petri net $\mathcal{N}$. *Right:* Its structural abstraction $G_{\text{struct}}(\mathcal{N})$.

Formally, let $\text{in}(t) \coloneqq \{p \in P : f(p, t) > 0\}$ be the set of input places of t if it is nonempty, and $\text{in}(t) \coloneqq \{\perp\}$ otherwise; and let $\text{out}(t) \coloneqq \{p \in P : f(t, p) > 0\}$ be the set of output places of t if it is nonempty, and $\text{out}(t) \coloneqq \{\perp\}$ otherwise. We define $G_{\text{struct}}(\mathcal{N}) \coloneqq (V, E, T, \mu)$ with $V \coloneqq P \cup \{\perp\}$, $\mu(p, t, q) \coloneqq \lambda(t)$ and

$$E \coloneqq \{(p, t, q) : p \neq q, t \in T, p \in \text{in}(t) \text{ and } \text{out}(t) \ni q\}\,.$$

We obtain the *structural distance* $d_{\text{struct}} \colon \mathbb{N}^P \times \mathbb{N}^P \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ defined as follows. For every marking $\boldsymbol{m}$, let $[\![\boldsymbol{m}]\!] \coloneqq \{p \in P : \boldsymbol{m}(p) > 0\} \cup \{\perp\}$ be the places marked in $\boldsymbol{m}$ together with $\perp$ (considered permanently marked). Let:

$$\begin{aligned} d_{\text{struct}}(\boldsymbol{m}, \boldsymbol{m}') &\coloneqq \max\{\kappa_{\boldsymbol{m}'}(p) : p \in [\![\boldsymbol{m}]\!]\}\,, \text{ where} \\ \kappa_{\boldsymbol{m}'}(p) &\coloneqq \min\{\text{dist}_{G_{\text{struct}}}(p, q) : q \in [\![\boldsymbol{m}']\!]\}\,. \end{aligned}$$

Informally, $\kappa_{\boldsymbol{m}'}(p)$ is the distance required to freely move a token from place p to a place marked in $\boldsymbol{m}'$, or to destroy it. Since every token of $\boldsymbol{m}$ must achieve this task, d_{struct} maximizes $\kappa_{\boldsymbol{m}'}(p)$ among all places marked in $\boldsymbol{m}$. Consider the Petri net of Figure 12 with $\boldsymbol{m} \coloneqq [p_1 \colon 0, p_2 \colon 1, p_3 \colon 1]$ and $\boldsymbol{m}' \coloneqq [p_1 \colon 1, p_2 \colon 0, p_3 \colon 0]$. We have $d_{\text{struct}}(\boldsymbol{m}, \boldsymbol{m}') = 2$ since $\kappa_{\boldsymbol{m}'}(p_2) = 2$ and $\kappa_{\boldsymbol{m}'}(p_3) = 1$.

We show that d_{struct} is an under-approximation by first proving a lemma:

Lemma 2. *If $\boldsymbol{m} \xrightarrow{\sigma}_{\mathbb{N}} \boldsymbol{m}'$, then for every $p \in [\![\boldsymbol{m}]\!]$ there exists a path of weight at most $\lambda(\sigma)$ from p to some $q \in [\![\boldsymbol{m}']\!]$ in $G_{struct}(\mathcal{N})$.*

Proof. We proceed by induction on $|\sigma|$. If $|\sigma| = 0$, then the claim follows immediately with the empty path. Assume $\sigma = t\tau$ with $t \in T$ and $\tau \in T^*$. There is some marking $\boldsymbol{m}''$ such that $\boldsymbol{m} \xrightarrow{t}_{\mathbb{N}} \boldsymbol{m}'' \xrightarrow{\tau}_{\mathbb{N}} \boldsymbol{m}'$. By induction hypothesis, for every $r \in [\![\boldsymbol{m}'']\!]$, there exists a path π_r of weight at most $\lambda(\tau)$ from r to some $q \in [\![\boldsymbol{m}']\!]$ in $G_{\text{struct}}(\mathcal{N})$. Let $p \in [\![\boldsymbol{m}]\!]$. We must exhibit a path from p.

If $p \in [\![\boldsymbol{m}'']\!]$, then we are done as path π_p satisfies $\mu(\pi_p) \leq \lambda(\tau) \leq \lambda(\sigma)$. So, assume $p \notin [\![\boldsymbol{m}'']\!]$. By definition of E, we have $e \coloneqq (p, t, r) \in E$ for some

$r \in [\![\boldsymbol{m}'']\!]$. Thus, path $\pi := e\pi_r$ satisfies the claim since $\lambda(\pi) = \lambda(t) + \mu(\pi_r) \leq \lambda(t) + \lambda(\tau) = \lambda(\sigma)$. □

Proposition 3. *It is the case that d_{struct} is a distance under-approximation.*

Proof. Let $\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{m}'' \in \mathbb{N}^P$ be markings. We prove admissibility by establishing each property.

Distance under-approximation. We must show that $d_{\text{struct}}(\boldsymbol{m}, \boldsymbol{m}') \leq \text{dist}_{\mathcal{N}}(\boldsymbol{m}, \boldsymbol{m}')$. Assume the latter differs from ∞, as we are otherwise done. Let $\sigma \in T^*$ be a shortest firing sequence such that

$$\boldsymbol{m} \xrightarrow{\sigma}_{\mathbb{N}} \boldsymbol{m}'.$$

Let $p \in [\![\boldsymbol{m}]\!]$ maximize $\kappa_{\boldsymbol{m}'}(p)$. By Lemma 2, $G_{\text{struct}}(\mathcal{N})$ has a path π of weight at most $\lambda(\sigma)$ from p to some $q \in [\![\boldsymbol{m}']\!]$. Thus, $d_{\text{struct}}(\boldsymbol{m}, \boldsymbol{m}') = \kappa_{\boldsymbol{m}'}(p) \leq \text{dist}_{G_{\text{struct}}(\mathcal{N})}(p, q) \leq \mu(\pi) \leq \lambda(\sigma) = \text{dist}_{\mathcal{N}}(\boldsymbol{m}, \boldsymbol{m}')$.

Triangle inequality. We show $d_{\text{struct}}(\boldsymbol{m}, \boldsymbol{m}'') \leq d_{\text{struct}}(\boldsymbol{m}, \boldsymbol{m}') + d_{\text{struct}}(\boldsymbol{m}', \boldsymbol{m}'')$. Assume that the right-hand side does not equal ∞ as we are otherwise done. Let $p, p' \in [\![\boldsymbol{m}]\!]$ and $q \in [\![\boldsymbol{m}']\!]$ respectively maximize $\kappa_{\boldsymbol{m}'}(p)$, $\kappa_{\boldsymbol{m}''}(p')$ and $\kappa_{\boldsymbol{m}''}(q)$.

Let $q' \in [\![\boldsymbol{m}']\!]$ and $r \in [\![\boldsymbol{m}'']\!]$ be such that $\kappa_{\boldsymbol{m}'}(p') = \text{dist}_{G_{\text{struct}}}(p', q')$ and $\kappa_{\boldsymbol{m}''}(q') = \text{dist}_{G_{\text{struct}}}(q', r)$. Note that they are well-defined by $\kappa_{\boldsymbol{m}'}(p) \neq \infty$ and $\kappa_{\boldsymbol{m}''}(q) \neq \infty$.

We have:

$$\begin{aligned}
& d_{\text{struct}}(\boldsymbol{m}, \boldsymbol{m}'') \\
&= \kappa_{\boldsymbol{m}''}(p') && \text{(by def. of } d_{\text{struct}}) \\
&\leq \text{dist}_{G_{\text{struct}}}(p', r) && \text{(by } r \in [\![\boldsymbol{m}'']\!] \text{ and min. of } \kappa_{\boldsymbol{m}''}(p')) \\
&\leq \text{dist}_{G_{\text{struct}}}(p', q') + \text{dist}_{G_{\text{struct}}}(q', r) && \text{(by the triangle inequality)} \\
&= \kappa_{\boldsymbol{m}'}(p') + \kappa_{\boldsymbol{m}''}(q') \\
&\leq \kappa_{\boldsymbol{m}'}(p') + \kappa_{\boldsymbol{m}''}(q) && \text{(by } q' \in [\![\boldsymbol{m}']\!] \text{ and max. of } q) \\
&\leq \kappa_{\boldsymbol{m}'}(p) + \kappa_{\boldsymbol{m}''}(q) && \text{(by } p' \in [\![\boldsymbol{m}]\!] \text{ and max. of } p) \\
&= d_{\text{struct}}(\boldsymbol{m}, \boldsymbol{m}') + d_{\text{struct}}(\boldsymbol{m}', \boldsymbol{m}'') && \text{(by def. of } d_{\text{struct}}).
\end{aligned}$$

Effectiveness. The structural abstraction $G_{\text{struct}}(\mathcal{N})$ can be precomputed in linear time from $\mathcal{N}$, and $\text{dist}_{G_{\text{struct}}(\mathcal{N})}(p, q)$ can then be precomputed in polynomial time using *e.g.* Dijkstra's algorithm. After these steps, $d_{\text{struct}}(\boldsymbol{m}, \boldsymbol{m}')$ can be evaluated in time $\mathcal{O}(|[\![\boldsymbol{m}]\!]| \cdot |[\![\boldsymbol{m}']\!]|)$. □

Let us stress that $d_{\text{struct}}(\boldsymbol{m}, \boldsymbol{m}')$ yields a crude estimation of $\text{dist}_{\mathcal{N}}(\boldsymbol{m}, \boldsymbol{m}')$. Indeed, its value is always upper bounded by $|P| \cdot \max\{\lambda(t) : t \in T\}$, while the actual distance could be arbitrarily large in $\boldsymbol{m}$ and $\boldsymbol{m}'$. Nevertheless, it is lightweight since it enables pre-computations. This makes it useful in particular for reachability graphs with short paths but large branching factors.

For example, instances from the SYPET suite have a large branching factor. They have between 23 and 187 unguarded transitions. Most markings tend to

enable some guarded transitions as well, so the average branching factor is larger. In particular, the branching factor of initial markings ranges from 30 to 300.[8]

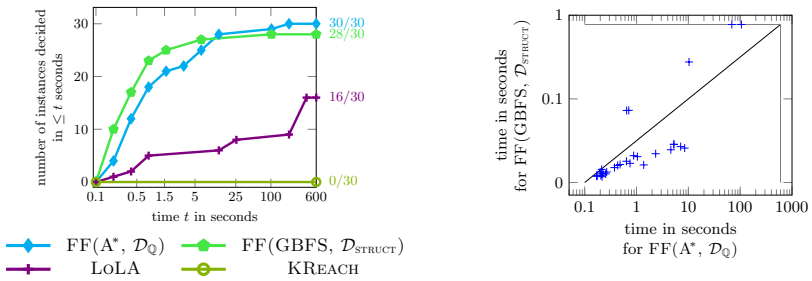


Fig. 13. Results on the SYPET suite with a time limit of 600 seconds. *Left:* Cumulative number of instances shown reachable. *Right:* Performance comparison per instance of FastFoward with two different schemes. Marks on the gray lines denote timeouts.

Let $\mathcal{D}_{\text{struct}}$ be distance under-approximation scheme obtained from the structural distance. This scheme is not unbounded, but can still be used with GBFS without termination guarantee. Figure 13 compares the performance of FastForward using A^* with $\mathcal{D}_{\mathbb{Q}}$ and using GBFS with $\mathcal{D}_{\text{struct}}$ on a time limit of 600 seconds. The former is faster on most instances, but it is vastly outperformed by A^* on a few instances. An explanation is provided by the large branching factor and short paths, and how these emphasize the characteristics of the different approaches. Note that the structural abstraction can be precomputed. On the other hand, A^* requires computing the heuristic on each successor before the next node is chosen for expansion. It thus is at a slight disadvantage on instances where a shortest witness is so short that it is found rather quickly even with the coarse structural distance. Its advantage is on the instances where the length of a shortest witness is at the upper end of the range. There, the large branching factor fully comes into play and a search algorithm must more aggressively discard parts of the search space.

[8] Chess and Go respectively have an average branching factor of ∼35 and ∼350 [55].

Appendix B

The Complexity of Soundness in Workflow Nets

This paper was published as a peer-reviewed conference article. A full version with an appendix containing missing proofs omitted from the conference paper due to space constraints was uploaded to arXiv at the URL https://arxiv.org/abs/2201.05588, see [26].

Cite as: M. Blondin, F. Mazowiecki, and P. Offtermatt. The Complexity of Soundness in Workflow Nets. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, New York, NY, USA, 2022.

Summary We investigate three related problems in workflow nets: Classical soundness, generalised soundness, and structural soundness. The complexity of all three problems has been investigated previously, yet for all three, only decidability was firmly known [3, 36, 65], though the literature also suggested that classical soundness is EXPSPACE-hard. We close the widely open gaps in the existing literature and show three main results, namely that 1) classical soundness is EXPSPACE-complete, 2) generalised soundness is PSPACE-complete, and 3) structural soundness is EXPSPACE-complete.

Contribution of this author The author was chiefly responsible for obtaining the results of the paper, contributing first drafts of all results and proofs. Notably, the

hardness results in Section 3.2, Section 5.5. and Section 6.3, and the characterization in Section 7 were the sole contribution of the author. The remaining results were obtained in joint discussions, in which the author took a leading role. The contribution further includes taking a leading role in the overall composition and development of the manuscript.

The complexity of soundness in workflow nets

Michael Blondin
Université de Sherbrooke, Canada
michael.blondin@usherbrooke.ca

Filip Mazowiecki
Max Planck Institute for Software Systems, Germany
filipm@mpi-sws.org

Philip Offtermatt
Max Planck Institute for Software Systems, Germany
Université de Sherbrooke, Canada
philip.offtermatt@usherbrooke.ca

Abstract

Workflow nets are a popular variant of Petri nets that allow for algorithmic formal analysis of business processes. The central decision problems concerning workflow nets deal with soundness, where the initial and final configurations are specified. Intuitively, soundness states that from every reachable configuration one can reach the final configuration. We settle the widely open complexity of the three main variants of soundness: classical, structural and generalised soundness. The first two are EXPSPACE-complete, and, surprisingly, the latter is PSPACE-complete, thus computationally simpler.

Keywords: Workflow nets, Petri nets, soundness, generalised soundness, structural soundness, complexity

ACM Reference Format:
Michael Blondin, Filip Mazowiecki, and Philip Offtermatt.

1 Introduction

Workflow nets are a formalism that allows for the modeling of business processes. Specifically, they allow to formally represent workflow procedures in Workflow Management Systems (WFMSs) (see *e.g.* [23, Section 4], where Figure 6 shows a workflow net for the processing of complaints; and [22, Section 3] for details on modeling procedures). Such a mathematical representation enables the algorithmic formal analysis of their behaviour. This is particularly relevant for large organisations that seek to manage the workflow of complex business processes. Such challenges have received, and continue to receive, intense academic attention, *e.g.* through the foundations track of the Business Process Management Conference (BPM), and via a discipline coined as *process mining* and pioneered prolifically by Wil van der Aalst[1]. In particular, many tools, such as those integrated in the ProM framework [27], can extract events from logs, *e.g.* of enterprise resource planning (ERP) systems, from which they synthesize workflow nets (and other models) to be formally analyzed (see [24] for a book on the topic).

More formally, workflow nets form a subset of (standard) Petri nets. They consist of *places* that can contain resources (called *tokens*) which can be consumed and produced via *transitions* in a nondeterministic and concurrent fashion. Two designated places, namely the *initial place* i and the *final place* f, respectively model the initialisation and termination of a business process. No token can be produced in the initial place, and no token can be consumed from the final place.

A central property studied since the inception of workflow nets is 1-*soundness* [22, 23]. Informally, quoting [22], it states that "*For any case, the procedure will terminate eventually [...]*". More formally, from the configuration with a single token in the initial place i, every reachable configuration can reach the configuration with a single token in the final place f. For readers familiar with computation temporal logic (CTL), 1-soundness can be loosely rephrased as $\mathsf{i} \models \forall\mathsf{G}\,\exists\mathsf{F}\,\mathsf{f}$. More generally, k-soundness states the same but for k tokens, *i.e.* $(k \cdot \mathsf{i}) \models \forall\mathsf{G}\,\exists\mathsf{F}\,(k \cdot \mathsf{f})$.

Classical soundness. Several variants of soundness have been considered in the literature (see [25] for a survey). The best-known is *classical soundness*. It states that a workflow net is 1-sound and that each transition is meaningful, *i.e.* each transition can be fired in at least one execution (often called *quasi-liveness*). It is well-known that deciding classical soundness amounts to checking boundedness and liveness of a slightly modified net. In particular, this means that classical soundness is decidable since boundedness and liveness are decidable problems. However, to the best of our knowledge, the (exact) complexity of classical soundness remains widely open. It has been suggested that classical soundness is EXPSPACE-hard. For example, the author of [9] mentions that "*IO-soundness is decidable but also EXPSPACE-hard ([26])*", yet [26] merely states the following:

> *[I]t may be intractable to decide soundness. (For arbitrary [workflow]-nets liveness and boundedness are decidable but also EXPSPACE-hard [...]).*

Furthermore, [23, p. 38] claims that EXPSPACE-hardness follows from the fact that "*deciding liveness and boundedness is EXPSPACE-hard*", which is attributed to [5]. However, [5] only mentions liveness to be EXPSPACE-hard (which was known prior to [5]).

The confusion arises from the fact that boundedness and liveness are *independently* EXPSPACE-hard problems, which suggests that classical soundness must naturally be at least as hard. However, this needs not be the case. For example, for a well-studied subclass of Petri nets, called free-choice nets,

[1]See http://www.processmining.org.

testing *simultaneously* boundedness and liveness has lower complexity than testing both properties independently[2] [10]. Moreover, since liveness is equivalent to the Petri net reachability problem [14], the only (implicitly) known upper bound is not even primitive recursive [16]. As a first contribution, we show that classical soundness and k-soundness are in fact both EXPSPACE-hard and in EXPSPACE, and hence EXPSPACE-complete. The upper bound is derived with a fortiori surprisingly little effort by invoking known results on coverability and so-called cyclicity. The hardness result is obtained by a careful reduction from the reachability problem for reversible Petri nets [4, 18]. There, we exploit subtle known results in a technically challenging way.

Generalised and structural soundness. Among the variants of soundness catalogued by the survey of van der Aalst et al. [25], *generalised soundness* [29, Def. 3] is the only fundamentally distinct property (in particular, see [25, Fig. 7]). It asks whether a given workflow net is k-sound *for all* $k \geq 1$. Generalised soundness, unlike classical soundness, preserves nice properties like composition [29]. The existential counterpart of generalised soundness, where "for all" is replaced by "for some", is known as *structural soundness* [1].

It is a priori not clear whether generalised and structural soundness are decidable, as the approach for deciding other types of soundness reasons about k-soundness for a given or fixed number k. Nonetheless, both problems have been shown decidable [7, 30]. The two algorithms, and a subsequent one [28], rely on Petri net reachability, which has very recently been shown Ackermann-complete [8, 15, 16].

As for classical soundness, the computational complexity of generalised and structural soundness remains open. In fact, we are not aware of any complexity result. In this work, we prove that generalised and structural soundness have much lower complexity than Petri net reachability: they are respectively PSPACE-complete and EXPSPACE-complete. In particular, the fact that generalised soundness is simpler than classical soundness is arguably surprising: positive instances of both problems require the given workflow net to be bounded, but for generalised soundness, one can avoid explicitly checking this EXPSPACE-complete property.

To derive the PSPACE membership, we introduce the notion of *strong soundness* which is (partly) defined in terms of a relaxed reachability relation (sometimes known as $\mathbb{Z}$-*reachability* or *pseudo-reachability*, *e.g.*, see [2]). Through results on integer linear programming and bounded vectors reordering, we prove that k-unsoundness of a workflow net must occur for a "small" number k. Furthermore, we show that it suffices to witness such a k for so-called $\mathbb{Z}$-bounded nonredundant nets, a more restrictive property than (standard) boundedness. By building upon these results, we establish the EXPSPACE membership of structural soundness, and, in fact, effectively characterise the set of sound numbers of workflow nets, which settles the open problem of [7].

The hardness for PSPACE and EXPSPACE are respectively obtained via reductions from the reachability problem for conservative Petri nets [19], and from 1-soundness.

Contribution and organisation. In summary, we settle, after around two decades, the exact computational complexity of the central decision problems for workflow nets. This is achieved in the rest of this work, organised as follows. In Section 2, we introduce general notation, Petri nets, workflow nets and soundness. In Section 3, we prove that classical soundness is EXPSPACE-complete. In Section 4, we provide bounds on vector reachability, which in turn allows us to prove PSPACE-completeness of generalised soundness (Section 5), and EXPSPACE-completeness of structural soundness (Section 6). In Section 7, we leverage the previous results to give a characterisation of numbers k for which a workflow net is k-sound. Finally, we conclude in Section 8. Due to space constraints, some proofs are deferred to an appendix.

2 Preliminaries

We denote naturals and integers with the usual font: $n \in \mathbb{N}$ and $z \in \mathbb{Z}$. Given $i, j \in \mathbb{Z}$, we write $[i..j]$ for $\{i, i+1, \ldots, j\}$. We use the bold font for vectors and matrices, *e.g.* $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{Z}^n$ and $\mathrm{A} \in \mathbb{Z}^{m \times n}$. Given $n \in \mathbb{N}$, we write $\boldsymbol{n}^d = (n, \ldots, n) \in \mathbb{N}^d$. We omit the dimension d when it is clear from the context, *e.g.* 0 denotes the null vector. We write $\boldsymbol{a}[i] = a_i$ and $\mathrm{A}[i, j]$ for matrix entries where $i \in [1..m]$ and $j \in [1..n]$. We write $\boldsymbol{x} \leq \boldsymbol{y}$ if $\boldsymbol{x}[i] \leq \boldsymbol{y}[i]$ holds for all $i \in [1..n]$. We write $\boldsymbol{x} < \boldsymbol{y}$ if at least one inequality is strict. Given a vector $\boldsymbol{a} \in \mathbb{Z}^n$ or a matrix $\mathrm{A} \in \mathbb{Z}^{m \times n}$, we define the norms $\|\boldsymbol{a}\| := \max_{1 \leq i \leq n} |\boldsymbol{a}[i]|$ and $\|\mathrm{A}\| := \max_{1 \leq j \leq m, 1 \leq i \leq n} |\mathrm{A}[j, i]|$.

2.1 Petri nets

A *Petri net* is a triple $\mathcal{N} = (P, T, F)$ such that:

- P and T are disjoint finite sets whose elements are respectively called *places* and *transitions*,
- $F \colon ((P \times T) \cup (T \times P)) \to \mathbb{N}$ is the *flow function*.

A *marking* is a vector $\boldsymbol{m} \colon P \to \mathbb{N}$ where $\boldsymbol{m}[p]$ indicates how many *tokens* are contained in place p. We say that a transition $t \in T$ is *enabled* in $\boldsymbol{m}$ if $\boldsymbol{m}[p] \geq F[p, t]$. Informally, $F[p, t]$ and $F[t, p]$ respectively correspond to the amount of tokens to be consumed from and produced in place p. Let ${}^\bullet t, t^\bullet \in \mathbb{N}^p$ respectively denote the vectors such that ${}^\bullet t[p] := F[p, t]$ and $t^\bullet[p] := F[t, p]$. Let $\Delta(t) := t^\bullet - {}^\bullet t$ denote the *effect* of t. If transition t is enabled in $\boldsymbol{m}$, then t may be *fired*, which leads to the marking $\boldsymbol{m}' := \boldsymbol{m} + \Delta(t)$. The latter is denoted by $\boldsymbol{m} \to^t \boldsymbol{m}'$, or simply by $\boldsymbol{m} \to \boldsymbol{m}'$ whenever we do not care about the transition that led to $\boldsymbol{m}'$.

[2]For free-choice nets: Boundedness is EXPSPACE-complete since any Petri net can trivially be made free-choice while preserving its reachability set up to projection; liveness is coNP-complete [10, Thm. 4.28]; and testing liveness *and* boundedness can be done in polynomial time [10, Cor. 6.18].

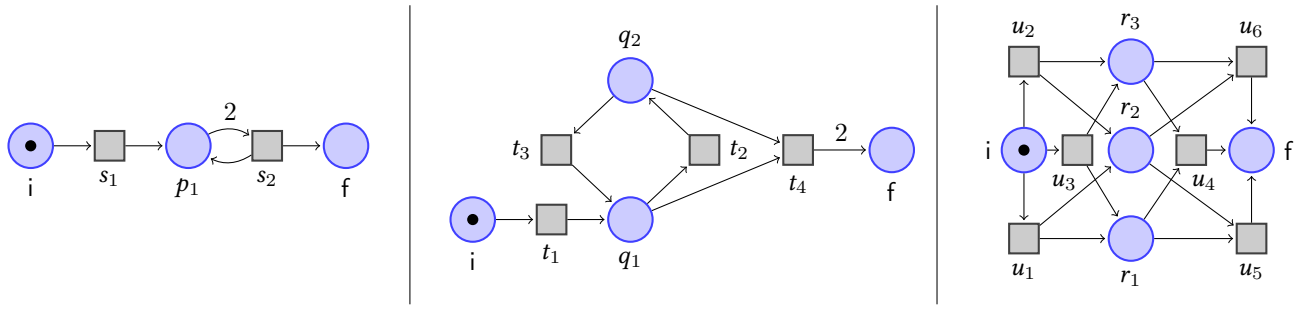


Figure 1. Three workflow nets, each marked with $\{\mathsf{i}\colon 1\}$.

We use a standard notation for markings, listing only nonzero values, *e.g.* if $P = \{p_1, p_2\}$, $\boldsymbol{m}[p_1] = 2$ and $\boldsymbol{m}[p_2] = 0$, then $\boldsymbol{m} = \{p_1 : 2\}$.

A *run* is a sequence of transitions $\rho = t_1 \cdots t_n \in T^*$. A run is enabled in $\boldsymbol{m}_0$ if there is a sequence of markings $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n$ such that $\boldsymbol{m}_i \to^{t_i} \boldsymbol{m}_{i+1}$ for all $0 \leq i < n$. If it is the case, then we denote this by $\boldsymbol{m}_0 \to^{\rho} \boldsymbol{m}_n$, or $\boldsymbol{m}_0 \to^* \boldsymbol{m}_n$ if ρ is not important. Given $\ell \in \mathbb{N}$, we say that ρ is *ℓ-bounded* if $\|\boldsymbol{m}_i\| \leq \ell$ for all $0 \leq i \leq n$. The support of a run is the set of transitions occurring in it, denoted $\mathrm{supp}(\rho) \coloneqq \{t_1, \ldots, t_n\}$.

We introduce a semantics where transitions can always be fired, and hence where markings may become negative. Formally, a *$\mathbb{Z}$-marking* is a vector $\boldsymbol{m}\colon P \to \mathbb{Z}$. We write $\boldsymbol{m} \to^t_{\mathbb{Z}} \boldsymbol{m}'$ (or simply $\boldsymbol{m} \to_{\mathbb{Z}} \boldsymbol{m}'$) if $\boldsymbol{m}' = \boldsymbol{m} + \Delta(t)$. Given a run ρ, we define in the obvious way $\to^{\rho}_{\mathbb{Z}}$ and $\to^*_{\mathbb{Z}}$. Note that markings are $\mathbb{Z}$-markings (with the domain restricted to $\mathbb{N}$). The definition of $\mathbb{Z}$-markings is mostly needed to use $\to^*_{\mathbb{Z}}$.

We define the *absolute value* and *norm* of a Petri net $\mathcal{N} = (P, T, F)$ by $|\mathcal{N}| \coloneqq |P| + |T|$ and $\|\mathcal{N}\| \coloneqq \|F\| + 1$, where F is seen as a vector over $(P \times T) \cup (T \times P)$. The *size* of a Petri net is defined as $\mathrm{size}(\mathcal{N}) \coloneqq |\mathcal{N}| \cdot (1 + \log\|\mathcal{N}\|)$. For some complexity problems, we will be given a Petri net and some markings, *e.g.* $\boldsymbol{m}$ and $\boldsymbol{m}'$. By the size of the input, we understand $\mathrm{size}(\mathcal{N}, \boldsymbol{m}, \boldsymbol{m}') \coloneqq \mathrm{size}(\mathcal{N}) + \log(\|\boldsymbol{m}\| + 1) + \log(\|\boldsymbol{m}'\| + 1)$.

A transition t is said to be *quasi-live* from marking $\boldsymbol{m}$ if there exists a marking $\boldsymbol{m}'$ such that $\boldsymbol{m} \to^* \boldsymbol{m}'$ and t is enabled in $\boldsymbol{m}'$. A transition t is said to be *live* from $\boldsymbol{m}$ if t is quasi-live from all $\boldsymbol{m}'$ such that $\boldsymbol{m} \to^* \boldsymbol{m}'$. We say that a Petri net $\mathcal{N}$ is *quasi-live* (resp. *live*) from $\boldsymbol{m}$ if each transition t of $\mathcal{N}$ is quasi-live (resp. live) from $\boldsymbol{m}$. Informally, quasi-liveness states that no transition is useless, and liveness states that transitions can always eventually be fired.

Example 2.1. Consider the Petri net $\mathcal{N}_{\text{middle}} = (P, T, F)$ illustrated in the middle of Figure 1. Places $P = \{\mathsf{i}, q_1, q_2, \mathsf{f}\}$ and transitions $T = \{t_1, t_2, t_3, t_4\}$ are depicted respectively as circles and squares. The flow function F is depicted by arcs, where unit weights are omitted, and where arcs with weight zero are not drawn, *e.g.* $F(\mathsf{i}, t_1) = 1$, $F(t_1, \mathsf{i}) = 0$, $F(t_4, \mathsf{f}) = 2$ and $F(\mathsf{f}, t_4) = 0$. In particular, transitions t_1, t_2 and t_3 are quasi-live from marking $\{\mathsf{i}\colon 1\}$ since

$$\{\mathsf{i}\colon 1\} \to^{t_1} \{q_1 : 1\} \to^{t_2} \{q_2 : 1\} \to^{t_3} \{q_1 : 1\}.$$

However, as no other marking is reachable, transition t_4 is not quasi-live. Note that t_2 and t_3 are both live from $\{\mathsf{i}\colon 1\}$, while t_1 is not live since it can only be fired once.

2.2 Workflow nets and soundness

A *workflow net* $\mathcal{N}$ is a Petri net that satisfies the following:

- there is a dedicated *initial* place i with ${}^\bullet t[\mathsf{i}] = 0$ for every transition t (cannot produce tokens in i);
- there is a dedicated *final* place $\mathsf{f} \neq \mathsf{i}$ with $t^\bullet[\mathsf{f}] = 0$ for every transition t (cannot consume tokens from f);
- each place and transition lies on at least one path from i to f in the underlying graph of $\mathcal{N}$, *i.e.* the graph (V, E) where $V \coloneqq P \cup T$ and $(u, v) \in E$ iff $F[u, v] > 0$.

Given $k \in \mathbb{N}$, we say that $\mathcal{N}$ is k-sound iff $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m}$ implies $\boldsymbol{m} \to^* \{\mathsf{f}\colon k\}$, i.e. starting from k tokens in the initial place, it is always possible to move the k tokens into the final place. We say that $\mathcal{N}$ is:

- *classically sound* iff $\mathcal{N}$ is 1-sound and quasi-live from $\{\mathsf{i} : 1\}$;
- *generalised sound* iff $\mathcal{N}$ is k-sound for all $k > 0$;
- *structurally sound* iff $\mathcal{N}$ is k-sound for some $k > 0$.

Example 2.2. Consider the workflow nets $\mathcal{N}_{\text{left}}$, $\mathcal{N}_{\text{middle}}$ and $\mathcal{N}_{\text{right}}$ depicted respectively in Figure 1.

Workflow nets $\mathcal{N}_{\text{left}}$ and $\mathcal{N}_{\text{middle}}$ are not 1-sound since their only transition that can mark place f is not quasi-live from $\{\mathsf{i}\colon 1\}$, namely s_2 and t_4. In particular, this means that both workflow nets are neither classically sound, nor generalized sound. Workflow net $\mathcal{N}_{\text{right}}$ is 1-sound, and in fact classically sound, as shown by the reachability graph of Figure 2.

In particular, this means that $\mathcal{N}_{\text{right}}$ is structurally sound. Workflow net $\mathcal{N}_{\text{left}}$ is not structurally sound as no matter the marking $\{\mathsf{i}\colon k\}$ from which it starts, there is no way to empty place p_1 once it is marked. Workflow net $\mathcal{N}_{\text{middle}}$ is 2-sound, and hence structurally sound. Indeed, from $\{\mathsf{i}\colon 2\}$, the two tokens must enter $\{q_1, q_2\}$ from which they can escape via $\{q_1 : 1, q_2 : 1\}$ by firing t_4, reaching marking $\{\mathsf{f}\colon 2\}$.

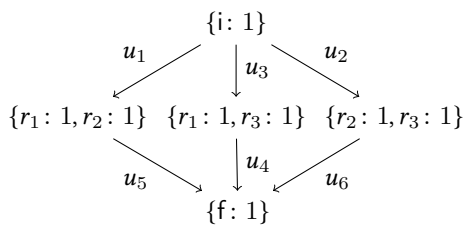


Figure 2. Markings reachable from $\{\mathsf{i}\colon 1\}$ in $\mathcal{N}_{\text{right}}$.

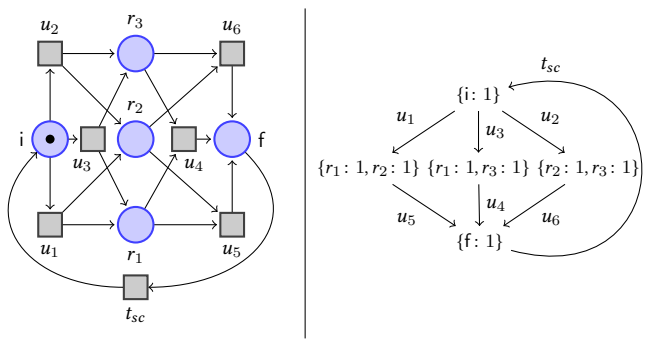


Figure 3. *Left*: Short-circuit net of the rightmost workflow net from Figure 1. *Right*: Its markings reachable from $\{\mathsf{i}\colon 1\}$.

Workflow net $\mathcal{N}_{\text{right}}$ is not 2-sound, and hence not generalised sound. Indeed, we have $\{\mathsf{i}\colon 2\} \to^{u_1 u_2 u_4} \{r_2\colon 2, \mathsf{f}\colon 1\}$ and no transition is enabled in the latter marking.

3 Classical soundness

As mentioned in the introduction, classical soundness is decidable, but its complexity has not yet been established. Let us recall why decidability holds. We say that a Petri net $\mathcal{N}$ is *bounded* from marking $\boldsymbol{m}$ if there exists $b \in \mathbb{N}$ such that $\boldsymbol{m} \to^* \boldsymbol{m}'$ implies $\boldsymbol{m}' \leq \boldsymbol{b}$. Otherwise, $\mathcal{N}$ is *unbounded* from $\boldsymbol{m}$. It is well-known that unboundedness holds iff there exist markings $\boldsymbol{m}' < \boldsymbol{m}''$ such that $\boldsymbol{m} \to^* \boldsymbol{m}' \to^* \boldsymbol{m}''$. The *short-circuit net* $\mathcal{N}_{sc}$ of a workflow net $\mathcal{N}$ is $\mathcal{N}$ extended with a transition t_{sc} such that $F[\mathsf{f}, t_{sc}] = F[t_{sc}, \mathsf{i}] = 1$ (and 0 for other entries relating to t_{sc}). Informally, the short-circuit net allows to restore the system upon completion, *i.e.* by moving a token from f to i.

For example, the left side of Figure 3 illustrates a short-circuit net $\mathcal{N}_{sc}$. By inspecting the graph of markings reachable from $\{\mathsf{i}\colon 1\}$ in $\mathcal{N}_{sc}$, we see that $\mathcal{N}_{sc}$ is live and bounded, *i.e.* it is always possible to (re)fire any transition, and each place is bounded by $b := 1$ token. It turns out that liveness and boundedness characterize classical soundness:

Proposition 3.1 ([22, Lemma 8])**.** *A workflow net $\mathcal{N}$ is classically sound iff $\mathcal{N}_{sc}$ is live and bounded from $\{\mathsf{i}\colon 1\}$.*

Decidability of classical soundness follows from Proposition 3.1. Indeed, boundedness can be tested in EXPSPACE [20], and liveness is decidable since it reduces to reachability [14, Thm 5.1] which is decidable [17]. However, the liveness problem is hard for the reachability problem [14, Thm 5.2], which was recently shown Ackermann-complete [8, 15, 16]. In this section, we first give a slightly different characterization not involving liveness which yields EXPSPACE membership. Then, we show that classical soundness is EXPSPACE-hard, and hence EXPSPACE-complete, via a reduction from the reachability problem for so-called reversible Petri nets.

3.1 EXPSPACE membership

Let us reformulate the characterization of Proposition 3.1 so that it deals with another property than liveness, namely "cyclicity". We say that a Petri net is *cyclic* from a marking $\boldsymbol{m}$ if $\boldsymbol{m} \to^* \boldsymbol{m}'$ implies $\boldsymbol{m}' \to^* \boldsymbol{m}$, i.e. it is always possible to go back to $\boldsymbol{m}$. For example, the short-circuit net $\mathcal{N}_{sc}$, illustrated on the left of Figure 3, is cyclic since each marking reachable from $\{\mathsf{i}\colon 1\}$ can reach $\{\mathsf{f}\colon 1\}$, which in turn can reach $\{\mathsf{i}\colon 1\}$.

Rather than directly considering classical soundness, we first consider 1-soundness. The characterization of Proposition 3.1 can be adapted to this problem as follows:

Proposition 3.2. *A workflow net $\mathcal{N}$ is 1-sound iff $\mathcal{N}_{sc}$ is bounded and transition t_{sc} is live from $\{\mathsf{i}\colon 1\}$.*

From the previous proposition, we prove the following.

Lemma 3.3. *A workflow net $\mathcal{N} = (P, T, F)$ is 1-sound iff $\mathcal{N}_{sc}$ is bounded and cyclic from $\{\mathsf{i}\colon 1\}$, and some transition $t \in T$ satisfies ${}^\bullet t = \{\mathsf{i}\colon 1\}$.*

Proof. $\Rightarrow$) Let $\mathcal{N}$ be 1-sound. Since $\{\mathsf{i}\colon 1\} \to^* \{\mathsf{f}\colon 1\}$ and $\mathsf{i} \neq \mathsf{f}$, some $t \in T$ satisfies ${}^\bullet t = \{\mathsf{i}\colon 1\}$. By Proposition 3.2, from $\{\mathsf{i}\colon 1\}$, $\mathcal{N}_{sc}$ is bounded and t_{sc} is live. It remains to show that $\mathcal{N}_{sc}$ is cyclic. Let $\{\mathsf{i}\colon 1\} \to^* \boldsymbol{m}$. By liveness of t_{sc}, there is a marking $\boldsymbol{m}'$ such that $\boldsymbol{m} \to^* \boldsymbol{m}'$ and $\boldsymbol{m}'$ enables t_{sc}. Note that ${}^\bullet t_{sc} = \{\mathsf{f}\colon 1\}$. If $\boldsymbol{m}' > \{\mathsf{f}\colon 1\}$, that is, $\boldsymbol{m}' = \{\mathsf{f}\colon 1\} + \boldsymbol{n}$ with $\boldsymbol{n} > 0$, then we obtain $\{\mathsf{i}\colon 1\} \to^* \{\mathsf{f}\colon 1\} + \boldsymbol{n} \to^{t_{sc}} \{\mathsf{i}\colon 1\} + \boldsymbol{n}$, and hence boundedness is violated. Thus, by boundedness and liveness of t_{sc}, $\boldsymbol{m} \to^* \boldsymbol{m}' = \{\mathsf{f}\colon 1\} \to^{t_{sc}} \{\mathsf{i}\colon 1\}$, which proves cyclicity.

$\Leftarrow$) Assume $\mathcal{N}_{sc}$ is bounded and cyclic from $\{\mathsf{i}\colon 1\}$, and that some $t \in T$ is as described. By Proposition 3.2, it suffices to show that t_{sc} is live from $\{\mathsf{i}\colon 1\}$. Let $\boldsymbol{m} \in \mathbb{N}^P$ be such that $\{\mathsf{i}\colon 1\} \to^* \boldsymbol{m}$ in $\mathcal{N}_{sc}$. We either have $\boldsymbol{m} = \{\mathsf{i}\colon 1\}$ or $\boldsymbol{m}[\mathsf{i}] = 0$, as otherwise $\mathcal{N}_{sc}$ is unbounded. If $\boldsymbol{m} = \{\mathsf{i}\colon 1\}$, we can fire t and obtain a marking where i is empty. Thus, assume w.l.o.g. that $\boldsymbol{m}[\mathsf{i}] = 0$. By cyclicity, we have $\boldsymbol{m} \to^\pi \{\mathsf{i}\colon 1\}$ for some π. Since t_{sc} is the only transition that produces tokens in place i, transition t_{sc} must appear in π. Hence, t_{sc} is live. □

Since classical soundness amounts to quasi-liveness and 1-soundness, we obtain the following corollary.

Corollary 3.4. *A workflow net $\mathcal{N}$ is classically sound iff $\mathcal{N}_{sc}$ is quasi-live, bounded and cyclic from $\{\mathsf{i}\colon 1\}$.*

Theorem 3.5. *Both 1-soundness and classical soundness are in EXPSPACE.*

Proof. Checking whether a transition t satisfies ${}^\bullet t = \{\mathsf{i}\colon 1\}$ can be carried in polynomial time. The other properties of Lemma 3.3 for 1-soundness, namely boundedness and cyclicity, belong to EXPSPACE [3, 20].

For quasi-liveness, we proceed as follows. The *coverability problem* asks whether given a Petri net and two markings $\boldsymbol{m}, \boldsymbol{m}'$, there exists a marking $\boldsymbol{m}'' \geq \boldsymbol{m}'$ such that $\boldsymbol{m} \to^* \boldsymbol{m}''$. This problem belongs to EXPSPACE [20]. Recall that quasi-liveness asks whether for each transition $t \in T \cup \{t_{sc}\}$, it is the case that $\{\mathsf{i}\colon 1\} \to^* \boldsymbol{m}$ for some some marking $\boldsymbol{m}$ that enables t, i.e. such that $\boldsymbol{m} \geq {}^\bullet t$. The latter is a coverability question. Hence, quasi-liveness amounts to $|T|+1$ coverability queries, which can be checked in EXPSPACE. □

We further show that the previous result can be extended to k-soundness through the following lemma.

Lemma 3.6. *Given a workflow net $\mathcal{N}$ and $k > 0$, one can compute, in polynomial time, a workflow net $\mathcal{N}'$ with $\|\mathcal{N}'\| = \|\mathcal{N}\| + \log(k)$ such that, for all $c > 0$, $\mathcal{N}$ is ck-sound iff $\mathcal{N}'$ is c-sound.*

Proof. Let $\mathcal{N} = (P, T, F)$. We define $\mathcal{N}' := (P', T', F')$ that rescales everything by k. Formally, we add two new places that are the new initial and final places $P' := P \cup \{\mathsf{i}', \mathsf{f}'\}$. We denote by i and f the previous initial and final places. We add two new transitions t_i and t_f defined by:

$$
\begin{aligned}
{}^\bullet t_\mathsf{i}[\mathsf{i}'] &= 1 \text{ and } {}^\bullet t_\mathsf{i}[p] = 0 \text{ for } p \neq \mathsf{i}',\\
t_\mathsf{i}^\bullet[\mathsf{i}] &= k \text{ and } t_\mathsf{i}^\bullet[p] = 0 \text{ for } p \neq \mathsf{i},\\
{}^\bullet t_\mathsf{f}[\mathsf{f}] &= k \text{ and } {}^\bullet t_\mathsf{f}[p] = 0 \text{ for } p \neq \mathsf{f},\\
t_\mathsf{f}^\bullet[\mathsf{f}'] &= 1 \text{ and } t_\mathsf{f}^\bullet[p] = 0 \text{ for } p \neq \mathsf{f}'.
\end{aligned}
$$

It is straightforward that $\mathcal{N}'$ satisfies the lemma. □

Corollary 3.7. *The k-soundness problem is in EXPSPACE.*

Proof. It suffices to invoke Lemma 3.6 with $c = 1$, and test 1-soundness of the resulting workflow net via Theorem 3.5. □

3.2 EXPSPACE-hardness

Let us now establish EXPSPACE-hardness of classical soundness. We will need the forthcoming lemma that essentially states that so-called reversible Petri nets can count up to (or down from) a doubly exponential number. Formally, we say that a Petri net $\mathcal{N} = (P, T, F)$ is *reversible* if each transition of $\mathcal{N}$ has an inverse, *i.e.* for every $t \in T$, there exists $t^{-1} \in T$ such that ${}^\bullet(t^{-1}) = t^\bullet$ and $(t^{-1})^\bullet = {}^\bullet t$. Note that for reversible Petri nets, it is the case that $\boldsymbol{m} \to^* \boldsymbol{m}'$ if and only if $\boldsymbol{m}' \to^* \boldsymbol{m}$. To emphasise this, we will sometimes write $\boldsymbol{m} \leftrightarrow^* \boldsymbol{m}'$.

Lemma 3.8 ([18, Lemma 3])**.** *Let $\mathcal{N}$ be a reversible Petri net and let $\boldsymbol{m}$ and $\boldsymbol{m}'$ be two markings. Let $n := \mathrm{size}(\mathcal{N}, \boldsymbol{m}, \boldsymbol{m}')$. There exists $c_n \in 2^{2^{O(n)}}$ such that if $\boldsymbol{m} \to^* \boldsymbol{m}'$ then $\boldsymbol{m} \to^\rho \boldsymbol{m}'$ for a c_n-bounded run ρ.*

Lemma 3.9 ([18, reformulation of Lemma 6 and Lemma 8])**.** *Let $n \in \mathbb{N}$ and $c_n \in 2^{2^{O(n)}}$. There exists a reversible Petri net $\mathcal{N}_n = (P_n, T_n, F_n)$ with four distinguished places $s, c, f, b \in P_n$. Given the two markings $\boldsymbol{m}_n := \{s\colon 1, c\colon 1\}$ and $\boldsymbol{m}'_n := \{f\colon 1, c\colon 1, b\colon c_n\}$, the following holds for all $\boldsymbol{m}$:*

1. *$\boldsymbol{m}_n \leftrightarrow^* \boldsymbol{m}'_n$;*
2. *$\boldsymbol{m}_n \leftrightarrow^* \boldsymbol{m}$ and $\boldsymbol{m}[f] > 0$ implies $\boldsymbol{m} = \boldsymbol{m}'_n$;*
3. *$\boldsymbol{m} \leftrightarrow^* \boldsymbol{m}'_n$ and $\boldsymbol{m}[s] > 0$ implies $\boldsymbol{m} = \boldsymbol{m}_n$;*
4. *if $\boldsymbol{m} < \boldsymbol{m}'_n$ and $\boldsymbol{m}[f] = 0$ then no transition can be fired from $\boldsymbol{m}$;*
5. *for all $p \in P_n$ there exists $\boldsymbol{m}_n \leftrightarrow^* \boldsymbol{m}$ s.t. $\boldsymbol{m}[p] > 0$.*

Furthermore, $\mathcal{N}_n$ is: of polynomial size in n; constructible in polynomial time in n; and quasi-live both from $\boldsymbol{m}_n$ and $\boldsymbol{m}'_n$.

Theorem 3.10. *The classical soundness and 1-soundness problems are EXPSPACE-hard.*

Proof. We give a reduction from the reachability problem for reversible Petri nets. This problem is known to be EXPSPACE-complete [4, 18]. Let $\mathcal{N} = (P, T, F)$ be a reversible Petri net, and let $\boldsymbol{m}, \boldsymbol{m}'$ be two markings for which we would like to know whether $\boldsymbol{m} \to^* \boldsymbol{m}'$ in $\mathcal{N}$.

Let $n := \mathrm{size}(\mathcal{N}, \boldsymbol{m}, \boldsymbol{m}')$. Let c_n be the value given by Lemma 3.8 for n. Let $\mathcal{N}_n = (P_n, T_n, F_n)$ be the Petri net given by Lemma 3.9 for c_n.

We construct a workflow net $\mathcal{N}' = (P', T', F')$ such that $\mathcal{N}'$ is classically sound if and only if $\boldsymbol{m} \to^* \boldsymbol{m}'$ in $\mathcal{N}$. To avoid any confusion, we will denote markings in $\mathcal{N}'$ by $\boldsymbol{n}, \boldsymbol{n}'$, etc.

The construction will ensure that

$$\boldsymbol{m} \to^* \boldsymbol{m}' \text{ in } \mathcal{N} \text{ iff } \mathcal{N}' \text{ is classically sound.} \tag{1}$$

Moreover, 1-soundness of $\mathcal{N}'$ will imply $\boldsymbol{m} \to^* \boldsymbol{m}'$, which will prove that both classical soundness and 1-soundness are EXPSPACE-hard.

Formally, the set of places P' consists of: P; its disjoint copy $\overline{P} := \{\overline{p} \mid p \in P\}$; seven extra places

$$\{\mathsf{i}, \mathsf{f}, p_{\mathrm{start}}, p_{\mathrm{inProgress}}, p_{\mathrm{cover}}, p_{\mathrm{simple}}, p_{\mathrm{canFire}}\};$$

two disjoint copies of P_n (from Lemma 3.9), with one copy of b removed. One of the copies will be marked with $\heartsuit$ to avoid any confusion, thus we write *e.g.* $p^\heartsuit \in P_n^\heartsuit$. The two places b and $b^\heartsuit$ are merged into a single place denoted b.

Before presenting the transitions, we would like to emphasise that, intuitively, place $\overline{p} \in \overline{P}$ will contain a "budget" of tokens that is an upper bound on how many more tokens can be present in p. Most of the time, for every marking $\boldsymbol{m}$ and place $p \in P$, we will keep $\boldsymbol{m}[p] + \boldsymbol{m}[\overline{p}] = c_n$ as an invariant.

In Figure 4, we present the most relevant parts of $\mathcal{N}'$. Formally, the set of transitions is divided into four subsets $T' = T_1 \cup T_2 \cup T_3 \cup T_4$. Transitions will be defined by giving ${}^\bullet t'[p]$ and $t'^\bullet[p]$. The values are zero on unmentioned places.

First, for every transition $t \in T$, we define $t' \in T_1$ by:

- ${}^\bullet t'[p] := {}^\bullet t[p]$ and $t'^\bullet[p] := t^\bullet[p]$ for all $p \in P$;
- ${}^\bullet t'[\overline{p}] := t^\bullet[p]$ and $t'^\bullet[\overline{p}] := {}^\bullet t[p]$ for all $p \in P$;
- ${}^\bullet t'[p_{\mathrm{canFire}}] = t'^\bullet[p_{\mathrm{canFire}}] := 1$.

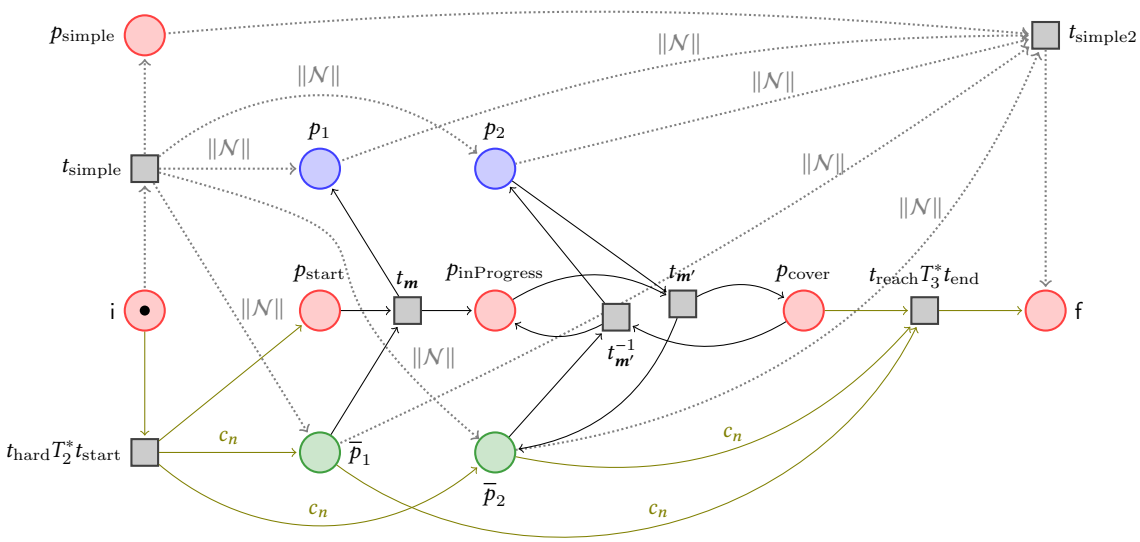


Figure 4. A workflow net $\mathcal{N}'$ which is classically sound iff $\boldsymbol{m} \to^* \boldsymbol{m}'$ in the reversible Petri net $\mathcal{N} = (P, T)$. In the example, $P = \{p_1, p_2\}$, $\boldsymbol{m} = (1, 0)$ and $\boldsymbol{m}' = (0, 1)$. The original places are blue, their copies are green, and other new places are red. We omit the transitions in T_1 that originated from T (recall that these transitions are modified to consume and produce tokens also in green places), and we omit the place p_{canFire} (used only to allow transitions in T_1 to fire). We only sketch transitions in T_2 and T_3 (and some other transitions), by writing the intuitive meaning of the gadgets that add/remove c_n tokens (these "transitions" are marked with a different color). The transition t_{hard} initiates the bottom part of $\mathcal{N}'$ (by filling the green places with c_n tokens) that checks $\boldsymbol{m} \to^* \boldsymbol{m}'$. The transition t_{simple} initiates the top part of $\mathcal{N}'$. We denote transitions in the top part with dotted gray color. This part is rather trivial and its only purpose is to ensure quasi-liveness of transitions in T_1 (by filling blue and green places with $\|\mathcal{N}\|$ tokens).

It is easy to see that since $\mathcal{N}$ is a reversible Petri net, for every transition in T_1, its reverse is also in T_1. We will say that T_1 is reversible. Notice that, for all $t' \in T_1$ and $p \in P$, the sum of tokens in p and $\overline{p}$ does not change under t'.

Second, for every $t \in T_n$, we add $t' \in T_2$ such that:

- ${}^\bullet t'[p] := {}^\bullet t[p]$ and $t'^\bullet[p] := t^\bullet[p]$ for all $p \in P_n$;
- ${}^\bullet t'[\overline{p}] := {}^\bullet t[b]$ and $t'^\bullet[\overline{p}] := t^\bullet[b]$ for all $\overline{p} \in \overline{P}$.

Intuitively, places in $\overline{P}$ behave as b to initialise the budget of c_n tokens. Similarly, for every $t^\heartsuit \in T_n^\heartsuit$, we add $t' \in T_3$ such that:

- ${}^\bullet t'[p^\heartsuit] := {}^\bullet t[p^\heartsuit]$ and $t'^\bullet[p^\heartsuit] := t^\bullet[p^\heartsuit]$ for all $p^\heartsuit \in P_n^\heartsuit$;
- ${}^\bullet t'[\overline{p}] := {}^\bullet t[b]$ and $t'^\bullet[\overline{p}] := t^\bullet[b]$ for all $\overline{p} \in \overline{P}$.

Note that since $\mathcal{N}_n$ is reversible, both T_2 and T_3 are reversible.

The set T_4 consists of the ten remaining transitions

$$\{t_{\text{hard}}, t_{\text{start}}, t_{\boldsymbol{m}}, t_{\boldsymbol{m}'}, t_{\boldsymbol{m}'}^{-1}, t_{\text{isEmpty}}, t_{\text{reach}}, t_{\text{reach}}^{-1}, t_{\text{simple}}, t_{\text{simple2}}\}.$$

Intuitively, the first two are needed to initialise places in $\overline{P}$ with c_n tokens; the next three transitions respectively add $\boldsymbol{m}$, $\boldsymbol{m}'$ and $-\boldsymbol{m}'$ to P; the next three transitions transfer tokens towards the final places; and the last two transitions are needed for quasi-liveness. Formally,

- ${}^\bullet t_{\text{hard}}[\mathsf{i}] = t_{\text{hard}}^\bullet[s] = t_{\text{hard}}^\bullet[c] := 1$;
- ${}^\bullet t_{\text{start}}[f] = {}^\bullet t_{\text{start}}[c] = t_{\text{start}}^\bullet[p_{\text{start}}] := 1$;
- $t_{\boldsymbol{m}}^\bullet[p] = {}^\bullet t_{\boldsymbol{m}}[\overline{p}] := \boldsymbol{m}[p]$ for all $p \in P$; and ${}^\bullet t_{\boldsymbol{m}}[p_{\text{start}}] = t_{\boldsymbol{m}}^\bullet[p_{\text{inProgress}}] = t_{\boldsymbol{m}}^\bullet[p_{\text{canFire}}] := 1$;
- ${}^\bullet t_{\boldsymbol{m}'}[p] = t_{\boldsymbol{m}'}^\bullet[\overline{p}] := \boldsymbol{m}'[p]$ for all $p \in P$; ${}^\bullet t_{\boldsymbol{m}'}[p_{\text{inProgress}}] = {}^\bullet t_{\boldsymbol{m}}[p_{\text{canFire}}] = t_{\boldsymbol{m}'}^\bullet[p_{\text{cover}}] := 1$; and $t_{\boldsymbol{m}'}^{-1}$ is its reverse transition;
- ${}^\bullet t_{\text{reach}}[p_{\text{cover}}] = t_{\text{reach}}^\bullet[f^\heartsuit] = t_{\text{reach}}^\bullet[c^\heartsuit] := 1$; and t_{reach}^{-1} is its reverse transition;
- ${}^\bullet t_{\text{end}}[s^\heartsuit] = {}^\bullet t_{\text{end}}[c^\heartsuit] = t_{\text{end}}^\bullet[\mathsf{f}] := 1$;
- ${}^\bullet t_{\text{simple}}[\mathsf{i}] = t_{\text{simple}}^\bullet[p_{\text{simple}}] = t_{\text{simple}}^\bullet[p_{\text{canFire}}] := 1$; and $t_{\text{simple}}^\bullet[p] = t_{\text{simple}}^\bullet[\overline{p}] := \|\mathcal{N}\|$ for all $p \in P$;
- ${}^\bullet t_{\text{simple2}}[p] = {}^\bullet t_{\text{simple2}}[\overline{p}] := \|\mathcal{N}\|$ for all $p \in P$; and ${}^\bullet t_{\text{simple2}}[p_{\text{simple}}] = {}^\bullet t_{\text{simple2}}[p_{\text{canFire}}] = t_{\text{simple2}}^\bullet[\mathsf{f}] := 1$.

Recall that $P \subseteq P'$ and that $\boldsymbol{m}$ is a marking on P. To ease the notation, we will assume that $\boldsymbol{m}$ is a marking on P' (with 0 tokens in places from $P' \setminus P$).

We are ready to prove Equation (1). Notice that for every reachable configuration $\{\mathsf{i} : 1\} \to^\rho \boldsymbol{n}$ the value $\boldsymbol{n}[p_{\text{canFire}}]$ is always equal to $\boldsymbol{n}[p_{\text{simple}}]$ or $\boldsymbol{n}[p_{\text{inProgress}}]$ (depending on whether the first transitions of ρ is t_{simple} or t_{hard}). For

readability, we omit the value of p_{canFire} in the markings of $\mathcal{N}'$.

$\Leftarrow$) Suppose that $\mathcal{N}'$ is 1-sound (we will not rely on $\mathcal{N}'$ being quasi-live). By Lemma 3.9 (1), we know that

$$\{\mathsf{i}\colon 1\} \to^{t_{\text{hard}}} \{s\colon 1, c\colon 1\} \to^{*} \{f\colon 1, c\colon 1, b\colon c_n\} + \sum_{\overline{p}\in\overline{P}} \{\overline{p}\colon c_n\}.$$

Let us denote the last marking by $\boldsymbol{n}$. Notice that

$$\boldsymbol{n} \to^{t_{\text{start}} t_{\boldsymbol{m}}} \{p_{\text{inProgress}}\colon 1, b\colon c_n\} + \boldsymbol{m} + \sum_{\overline{p}\in\overline{P}} \{\overline{p}\colon c_n - \boldsymbol{m}[p]\}.$$

We denote the latter marking by $\boldsymbol{n}'$. Since $\mathcal{N}'$ is 1-sound, $\boldsymbol{n}' \to^{\rho} \{\mathsf{f}\colon 1\}$ for some run ρ. This is possible if t_{reach} was fired at least once in ρ. Let $\boldsymbol{n}_1 \to^{t_{\text{reach}}} \boldsymbol{n}_2$ be the last time t_{reach} was fired in ρ. We claim that $\boldsymbol{n}_2 = \{f^{\heartsuit}\colon 1, c^{\heartsuit}\colon 1, b\colon c_n\} + \sum_{\overline{p}} \{\overline{p}\colon c_n\}$. Indeed, it has to be that

$$\boldsymbol{n}_2 \to^{\rho'} \{s^{\heartsuit}\colon 1, c^{\heartsuit}\colon 1\} \to^{t_{\text{end}}} \{\mathsf{f}\colon 1\},$$

where ρ' uses transition only from T_3. By Lemma 3.9 (4), this is possible only if $\boldsymbol{n}_2$ is as claimed. Let ρ'' be the prefix of the run ρ from $\boldsymbol{n}'$ such that it ends in $\boldsymbol{n}_1$. Finally, ρ'', when restricted to P, witnesses reachability for $\boldsymbol{m} \to^{*} \boldsymbol{m}'$.

$\Rightarrow$) Suppose that $\boldsymbol{m} \to^{*} \boldsymbol{m}'$. The proof of 1-soundness is very technical and can be found in the appendix. In a nutshell, recall that T_1, T_2 and T_3 are reversible, and for $t_{\boldsymbol{m}'}, t_{\text{reach}} \in T_4$ we include their reverse transitions. This allows us to revert any configuration to a configuration from which it is easy to define a run to $\{\mathsf{f}\colon 1\}$.

To conclude this implication, we need to prove that $\mathcal{N}'$ is quasi-live. Indeed, from the proof of 1-soundness it is easy to see that $\boldsymbol{m} \to^{*} \boldsymbol{m}'$ implies that all transitions are fireable, with the possible exception of transitions from T_1. However,

$$\{\mathsf{i}\colon 1\} \to^{t_{\text{simple}}} \{p_{\text{simple}}\colon 1\} + \sum_{p\in P} \{p\colon \|\mathcal{N}\|, \overline{p}\colon \|\mathcal{N}\|\}.$$

From the latter configuration, any transition of T_1 is fireable.

Finally, observe that $\mathcal{N}'$ is a workflow net. Indeed, by taking t_{simple} we put tokens in P, and by taking t_{simple2} we can put tokens in $\overline{P}$. Each place from copies in $\mathcal{N}_n$ is on a path from i to f by Lemma 3.9 (5). The remaining places are clearly on such a path by definition (see Figure 4). □

4 Bounds on vector reachability

In this section, we present technical results that will be helpful to establish complexity bounds in the forthcoming sections. It is well-known that Petri nets are complex due to their nonnegativity constraints. Namely, markings are over $\mathbb{N}$ (not $\mathbb{Z}$), which blocks transitions from being fired whenever the amount of tokens would drop below zero. By lifting this restriction, *i.e.* allowing markings over $\mathbb{Z}$, transitions cannot be blocked and we obtain a provably simpler model (*e.g.* see [13]). We recall known results that provide bounds on reachability problems for vectors over $\mathbb{Z}$. Based on these results, we will derive useful bounds for the next sections.

4.1 Integer linear programs

Given positive natural numbers $n, m > 0$, let $\mathrm{A} \in \mathbb{Z}^{m\times n}$ be an integer matrix, $\boldsymbol{b} \in \mathbb{Z}^m$ an integer vector and $\boldsymbol{x} = (x_1, \ldots, x_n)^{\mathsf{T}}$ a vector of variables. We say that $G \coloneqq \mathrm{A}\cdot\boldsymbol{x} \geq \boldsymbol{b}$ is an $(m \times n)$-*ILP*, that is, an integer linear program (ILP) with m inequalities and n variables. The set of solutions of G is

$$[\![G]\!] \coloneqq \{\boldsymbol{\mu} \in \mathbb{Z}^n \mid \mathrm{A}\cdot\boldsymbol{\mu} \geq \boldsymbol{b}\},$$

and the set of natural solutions is $[\![G]\!]_{\geq 0} \coloneqq [\![G]\!] \cap \mathbb{N}^n$. We will only be interested in the natural solutions $[\![G]\!]_{\geq 0}$ but sometimes we will need to refer to $[\![G]\!]$. We shall assume that these sets are equal, by implicitly adding a new inequality for each variable specifying that it is greater or equal to 0.

Often it is convenient to write an equality constraint, *e.g.* $x - y = 0$. This can be simulated by two inequalities, so we will allow to define G both with equalities and inequalities.

We introduce some notation about *semi-linear* sets from [6] to obtain bounds on the sizes of solutions to ILPs. A set of vectors is called *linear* if it is of the form $L(\boldsymbol{b}, P) = \{\boldsymbol{b} + \lambda_1\boldsymbol{p}_1 + \ldots + \lambda_k\boldsymbol{p}_k \mid \lambda_1, \ldots, \lambda_k \in \mathbb{N}\}$, where $\boldsymbol{b} \in \mathbb{Z}^n$ is a vector and $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k\} \subseteq \mathbb{Z}^n$ is a finite set of vectors. A set is called *hybrid linear* if it is of the form $L(B, P) = \bigcup_{\boldsymbol{b}\in B} L(\boldsymbol{b}, P)$ for a finite set of vectors $B = \{\boldsymbol{b}_1 \ldots, \boldsymbol{b}_\ell\} \subseteq \mathbb{Z}^n$.

The *size* of a finite set of vectors B and of an $(m \times n)$-ILP G are defined respectively as $\|B\| \coloneqq \max_{\boldsymbol{b}\in B}\|\boldsymbol{b}\|$ and $\|G\| \coloneqq \|\mathrm{A}\| + \|\boldsymbol{b}\| + m + n$.

Lemma 4.1 ([31], presentation adapted from [6, Prop. 3])**.** *Let G be an $(m\times n)$-ILP. It is the case that $[\![G]\!] = \bigcup_{i\in I} L(B_i, P_i)$, where $\max_{i\in I}\|B_i\| \leq \|G\|^{O(n\log n)}$.*

For the forthcoming lemmas, recall that $\boldsymbol{c} = (c, \ldots, c)$.

Lemma 4.2. *Let G be an $(m \times n)$-ILP. There exists a number $c \leq \|G\|^{O(n\log n)}$ such that for all $\boldsymbol{\mu} \in [\![G]\!]_{\geq 0}$, there is some $\boldsymbol{\mu}' \in [\![G]\!]_{\geq 0}$ such that $\boldsymbol{\mu}' \leq \boldsymbol{\mu}$ and $\boldsymbol{\mu}' \leq \boldsymbol{c}$.*

Proof. Recall that we can assume $[\![G]\!] = [\![G]\!]_{\geq 0}$. By Lemma 4.1, $[\![G]\!] = \bigcup_{i\in I} L(B_i, P_i)$. We set $c \coloneqq \max_{i\in I}\|B_i\|$. Let $\boldsymbol{\mu} \in [\![G]\!]_{\geq 0}$. There exist $i \in I$ and $\boldsymbol{b} \in B_i$ such that $\boldsymbol{\mu} \in L(\boldsymbol{b}, P_i)$. Note that $\boldsymbol{p} \geq 0$ for all $\boldsymbol{p} \in P_i$. Hence, we have $\boldsymbol{b} \in [\![G]\!]_{\geq 0}$, $\boldsymbol{b} \leq \boldsymbol{\mu}$ and $\boldsymbol{b} \leq \boldsymbol{c}$. Thus, we can set $\boldsymbol{\mu}' \coloneqq \boldsymbol{b}$. □

Lemma 4.3. *Let $G = \mathrm{A}\cdot\boldsymbol{x} \geq \boldsymbol{b}$ be an $(m \times n)$-ILP, where $\boldsymbol{b} \geq 0$. There exists $c \leq \|G\|^{O((m+n)\log(m+n))}$ such that the following holds. For every $\boldsymbol{\mu} \in [\![G]\!]_{\geq 0}$, there exists $\boldsymbol{\nu} \in [\![G]\!]_{\geq 0}$ such that $\boldsymbol{\nu} \leq \boldsymbol{\mu}$, $\boldsymbol{\nu} \leq \boldsymbol{c}$, and $\mathrm{A}\cdot\boldsymbol{\nu} \leq \mathrm{A}\cdot\boldsymbol{\mu}$.*

4.2 Steinitz Lemma

Let us recall the Steinitz Lemma [21] based on the presentation of [11].

Lemma 4.4. *Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ be such that $\sum_{i=1}^n \boldsymbol{x}_i = 0$ and $\|\boldsymbol{x}_i\| \leq 1$ for all i. There exists a permutation π on $[1..n]$*

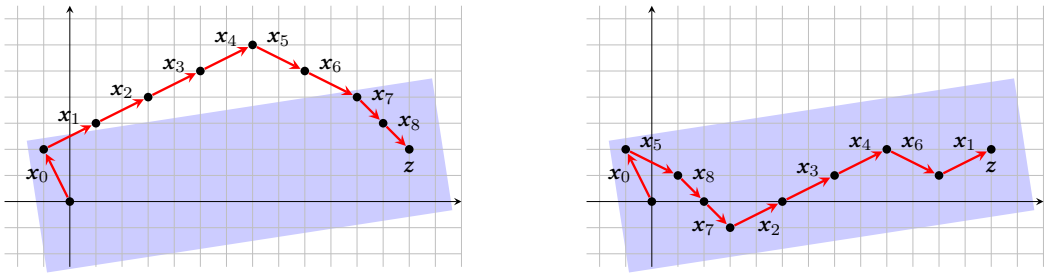


Figure 5. An example of Lemma 4.5 in dimension $d = 2$. The vectors $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_n$ form a path from 0 to $\boldsymbol{z}$. The colored background highlights points that are within some bounded distance from the line 0 to $\boldsymbol{z}$ (the bound depends on d and $\boldsymbol{x}_i$, but not on $\boldsymbol{z}$). In the right picture, the vectors are reordered so that they all fit within the bound. The additional constraints are that: the first vector $\boldsymbol{x}_0$ remains first ($\pi(0) = 0$); and, intuitively, that the points are getting closer to $\boldsymbol{z}$ ($0 \leq c_0 \leq c_1 \leq \ldots \leq c_n$).

such that

$$\left\| \sum_{j=1}^{i} \boldsymbol{x}_{\pi(j)} \right\| \leq d \qquad \textit{for all } i \in [1..n].$$

The following formulation of the lemma, which is depicted graphically in Figure 5, will be more convenient for us.

Lemma 4.5. *Let $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{Z}^d$, $b \coloneqq \max_{j=0}^{n} \|\boldsymbol{x}_j\|$, and $\boldsymbol{z} \coloneqq \sum_{j=0}^{n} \boldsymbol{x}_j$. There exists a permutation π of $[0..n]$ such that: $\pi(0) = 0$; and there exist $0 \leq c_0 \leq c_1 \leq \ldots \leq c_n$, where*

$$\left\| \sum_{j=0}^{i} \boldsymbol{x}_{\pi(j)} - c_i \cdot \boldsymbol{z} \right\| \leq b(d+2) \qquad \textit{for all } i \in [0..n].$$

5 Generalised soundness

A Petri net $\mathcal{N}$ is *$\mathbb{Z}$-bounded* from a marking $\boldsymbol{m}$ if there exists $b \in \mathbb{N}$ such that $\boldsymbol{m} \to_{\mathbb{Z}}^{*} \boldsymbol{m}' \geq 0$ implies $\boldsymbol{m}' \leq \boldsymbol{b}$ (*i.e.* we replace $\to^{*}$ with $\to_{\mathbb{Z}}^{*}$ in the definition of boudedness). Otherwise, we say that $\mathcal{N}$ is *$\mathbb{Z}$-unbounded*. Observe that being $\mathbb{Z}$-bounded does not mean that the set of reachable markings is bounded by below, but only from above.

Let $k \geq 0$. We say that $\mathcal{N}$ is *strongly k-sound* if for every $\boldsymbol{m} \in \mathbb{N}^P$ such that $\{\mathsf{i} \colon k\} \to_{\mathbb{Z}}^{*} \boldsymbol{m}$, it holds that $\boldsymbol{m} \to^{*} \{\mathsf{f} \colon k\}$. Note that every strongly k-sound net is also k-sound.

The aim of the next three subsections is to prove the following theorem.

Theorem 5.1. *Generalised soundness is in PSPACE.*

The proof has two parts. First, we prove that if there is a k for which the net is not k-sound, then there is also such a k bounded exponentially. Second, we prove that k-soundness for exponentially bounded k can be verified in PSPACE.

5.1 Nonredundant workflow nets

Fix a workflow net $\mathcal{N} = (P, T, F)$. We say that a place $p \in P$ is *nonredundant* if there exists $k \in \mathbb{N}$ such that $\{\mathsf{i} \colon k\} \to^{*} \boldsymbol{m}$ and $\boldsymbol{m}[p] > 0$. By removing a redundant place p from $\mathcal{N}$, we mean removing p from P and all transitions $t \in T$ such that $({}^{\bullet}t)[p] > 0$. With the remaining transitions restricted to the domain $P \setminus \{p\}$, we obtain a new workflow net $\mathcal{N}' \coloneqq (P \setminus \{p\}, T')$. It is clear that $\mathcal{N}$ is k-sound if and only if $\mathcal{N}'$ is k-sound for all $k \in \mathbb{N}$. Thus, in particular, this procedure preserves generalised soundness.

It will be convenient to assume that all places in the studied workflow nets are nonredundant. At first, it might seem that this requires coverability checks for every place. However, since the number of initial tokens is arbitrary, finding redundant places amounts to a simple polynomial-time saturation procedure. More details can be found in [30, Thm. 8, Def. 10, Sect. 3.2] (and in the appendix). We will call workflow nets without redundant places *nonredundant workflow nets*[3]. To summarise we conclude the following.

Proposition 5.2. *Given a workflow net $\mathcal{N}$, one can identify and remove all redundant places from it in polynomial time. The resulting workflow net $\mathcal{N}'$ is nonredundant. Moreover, $\mathcal{N}$ is k-sound if and only if $\mathcal{N}'$ is k-sound for all $k \in \mathbb{N}$.*

In the following lemma, intuitively, we show that the initial budget is small for nonredundant workflow nets.

Lemma 5.3. *Let $\mathcal{N} = (P, T, F)$ be a nonredundant workflow net and let $p \in P$ be a place. There exists $k < (\|T\| + 2)^{|T|}$ such that $\{\mathsf{i} \colon k\} \to^{*} \boldsymbol{m}$ and $\boldsymbol{m}[p] > 0$.*

Proof. A transition t *increases* a place p' if $\Delta(t)[p'] > 0$. We say that a run ρ *increases* p' if there exists $t \in \mathrm{supp}(\rho)$ that increases p'. For the proof of the lemma, we assume that $p \neq \mathsf{i}$, as otherwise it suffices to define $k = 1$.

We prove that for all run $\{\mathsf{i} \colon k'\} \to^{\rho} \boldsymbol{m}'$, there is a run π such that: $\mathrm{supp}(\pi) = \mathrm{supp}(\rho)$, and $\{\mathsf{i} \colon k\} \to^{\pi} \boldsymbol{m}$ for some $k < (\|T\| + 2)^n$ and $\boldsymbol{m}$, where $\boldsymbol{m}[p'] \geq 1$ for all places p' increased by ρ. Note that, since $\mathcal{N}$ is a nonredundant workflow net, if we exhibit such a run then we are done as there exists ρ that increases p.

Let $\{\mathsf{i} \colon k'\} \to^{\rho} \boldsymbol{m}'$. The proof is by induction on n, where $\mathrm{supp}(\rho) = \{t_1, \ldots, t_n\}$. Assume $n = 1$. The only transition

[3]The results in [30] deal with batch workflow nets, which are in particular nonredundant workflow nets.

used by ρ is t_1, which increases p. Recall that $\|T\|$ is the maximal number occurring on any arc of $\mathcal{N}$. Since workflow nets start with tokens only in place i, we must have $\{\mathsf{i}\colon \|T\|\} \geq {}^\bullet\pi$. It suffices to define $\pi := t_1$ and $k := \|T\| < (\|T\| + 2)$.

For the induction step, assume $n > 1$ and that the lemma holds for $n-1$. Let ρ_{n-1} be the longest prefix of ρ such that $\mathrm{supp}(\rho_{n-1}) = \{t_1, \dots, t_{n-1}\}$. The induction hypothesis for ρ_{n-1} yields $k_{n-1} < (\|T\|+2)^{n-1}$, and π_{n-1} with $\mathrm{supp}(\pi_{n-1}) = \{t_1, \dots, t_{n-1}\}$. Let $\{\mathsf{i}\colon k_{n-1}\} \to^{\pi_{n-1}} \boldsymbol{m}_{n-1}$. Note that $\mathrm{supp}({}^\bullet t_n) \subseteq \mathrm{supp}(\pi_{n-1}^\bullet) \cup \{i\}$ since ρ is a run, where t_n is fired. By repeating $\|T\| + 1$ times the run π_{n-1}, we get

$$\{\mathsf{i}\colon (k_{n-1}+1)\cdot(\|T\|+1)\} \to^* \{\mathsf{i}\colon \|T\|+1\} + (\|T\|+1)\cdot \boldsymbol{m}_{n-1}.$$

To ease the notation, let $\boldsymbol{n} := \{\mathsf{i}\colon \|T\| + 1\} + (\|T\| + 1) \cdot \boldsymbol{m}_n$. By definition of $\boldsymbol{m}_{n-1}$, it holds that $\boldsymbol{n}[p'] \geq \|T\| + 1$ for all $p' \in \pi^\bullet$. Furthermore, we can fire t_n from $\boldsymbol{n}$. Let $\boldsymbol{n} \to^{t_n} \boldsymbol{m}$. To conclude, consider a place p' increased by ρ. If it is increased by one of the transitions $t_1, \dots, t_{n-1}$, then after firing t_n at least one token was left in p'. Otherwise, p' is increased by t_n. In both cases, we have $\boldsymbol{m}[p] \geq 1$. It remains to observe that $k = (k_{n-1} + 1) \cdot (\|T\| + 1) < (\|T\| + 2)^n$. □

5.2 Unsoundness occurs for small numbers

Recall a result by van Hee et al. that establishes a connection between reachability relations $\to^*_{\mathbb{Z}}$ and $\to^*$.

Lemma 5.4 (adaptation of [30, Lemma 12])**.** *Let $\mathcal{N}$ be a nonredundant workflow net, and let $\boldsymbol{m}$ be a marking for which there exists $k \geq 0$ satisfying $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m}$. There exists $\ell \geq 0$ such that $\{\mathsf{i}\colon k + \ell\} \to^* \boldsymbol{m} + \{\mathsf{f}\colon \ell\}$.*

Note that Lemma 5.4 is an easy consequence of the definition of nonredundancy. Namely, it suffices to put "enough budget" in each place so that the run under $\to^*_{\mathbb{Z}}$ becomes a run under $\to^*$. We restate the result to give a bound on ℓ.

Lemma 5.5. *Let $\mathcal{N} = (P, T, F)$ be a nonredundant workflow net. Let k and $\boldsymbol{m} \in \mathbb{N}^P$ be such that $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m}$. There exist $\ell \leq (\|T\| + 2)^{|T|} \cdot \max(\|T\|, k) \cdot |P|(|P| + 2)$ and $\boldsymbol{m}' \in \mathbb{N}^P$ such that $\{\mathsf{i}\colon \ell\} \to^* \boldsymbol{m}'$ and $\{\mathsf{i}\colon \ell + k\} \to^* \boldsymbol{m} + \boldsymbol{m}'$.*

Proof. Let $\rho = t_1 t_2 \cdots t_n$ be such that $\{\mathsf{i}\colon k\} \to^\rho_{\mathbb{Z}} \boldsymbol{m}$. Let us define $\boldsymbol{x}_0 := \{\mathsf{i}\colon k\}$ and $\boldsymbol{x}_j := \Delta(t_j)$ for all $j \in [1..n]$. By Lemma 4.5, we can assume that the transitions t_j are ordered so that there exist $c_0, \dots, c_n \geq 0$ where

$$\left\| \{\mathsf{i}\colon k\} + \sum_{j=1}^{i} \Delta(t_j) - c_i \boldsymbol{m} \right\| \leq \max(\|T\|, k) \cdot (|P| + 2),$$

for all $i \in [0..n]$. Since $\boldsymbol{m} \geq 0$, we get for all $p \in P$:

$$\left(\{\mathsf{i}\colon k\} + \sum_{j=1}^{i} \Delta(t_j) \right)[p] \geq -\max(\|T\|, k) \cdot (|P| + 2). \quad (2)$$

By Lemma 5.3, there exists $\ell \leq (\|T\| + 2)^{|T|}$ such that for every place p there is a run $\{\mathsf{i}\colon \ell\} \to^{\pi_p} \boldsymbol{m}_p$ with $\boldsymbol{m}_p[p] > 0$. Thus, to put $\max(\|T\|, k) \cdot (|P| + 2)$ tokens in all places, it suffices to repeat $\max(\|T\|, k) \cdot (|P| + 2)$ times the run π_p for every $p \in P$. This requires $\ell \leq (\|T\| + 2)^{|T|} \cdot \max(\|T\|, k) \cdot |P|(|P| + 2)$ tokens. Let $\boldsymbol{m}'$ be the marking obtained afterwards. By (2), $\boldsymbol{m}'$ allows to fire ρ. Therefore, we obtain $\{\mathsf{i}\colon \ell\} \to^* \boldsymbol{m}'$ and $\{\mathsf{i}\colon \ell + k\} \to^* \boldsymbol{m} + \boldsymbol{m}'$ as required. □

This lemma allows us to focus on $\to^*_{\mathbb{Z}}$ instead of $\to^*$.

Lemma 5.6. *Let $\mathcal{N} = (P, T, F)$ be a nonredundant workflow net. It is the case that $\mathcal{N}$ is generalised sound iff it is strongly k-sound for all $k \geq 0$. Moreover, if $\mathcal{N}$ is not strongly k-sound, then there exists $k' \leq k + (\|T\|+2)^{|T|} \cdot \max(\|T\|, k) \cdot |P|(|P|+2)$ such that $\mathcal{N}$ is not k'-sound.*

Proof. The "if" implication is trivial. Indeed, if $\mathcal{N}$ is not k-sound then it cannot be strongly k-sound.

To prove the "only if" implication, assume that $\mathcal{N}$ is not strongly k-sound. We show that there exists k' such that $\mathcal{N}$ is not k'-sound. We will also prove the promised bound on k'. Since $\mathcal{N}$ is not strongly k-sound, there must be some $\boldsymbol{m} \in \mathbb{N}^P$ and π such that $\{\mathsf{i}\colon k\} \to^\pi_{\mathbb{Z}} \boldsymbol{m}$ and $\boldsymbol{m} \not\to^* \{\mathsf{f}\colon k\}$. By Lemma 5.5, there exists $\ell \leq (\|T\| + 2)^{|T|} \cdot \max(\|T\|, k) \cdot |P|(|P| + 2)$ and $\boldsymbol{m}'$ such that $\{\mathsf{i}\colon \ell\} \to^* \boldsymbol{m}'$ and $\{\mathsf{i}\colon \ell + k\} \to^* \boldsymbol{m} + \boldsymbol{m}'$. If $\mathcal{N}$ is not ℓ-sound, then we are done. Otherwise, if $\mathcal{N}$ is ℓ-sound, then it must hold that $\boldsymbol{m}' \to^* \{\mathsf{f}\colon \ell\}$. So, $\{\mathsf{i}\colon \ell + k\} \to^* \boldsymbol{m} + \boldsymbol{m}' \to^* \boldsymbol{m} + \{\mathsf{f}\colon \ell\}$. Recall that $\boldsymbol{m} \not\to^* \{\mathsf{f}\colon k\}$. Thus, $\boldsymbol{m} + \{\mathsf{f}\colon \ell\} \not\to^* \{\mathsf{f}\colon \ell + k\}$. We are done since this means that $\mathcal{N}$ is not $(\ell + k)$-sound. □

In the remainder of this section, we will show that if there exists some k such that $\mathcal{N}$ is not strongly k-sound, then k is at most exponential in $|\mathcal{N}|$. We define an ILP which is closely related to the markings reachable from at least one initial number of tokens in $\mathcal{N}$. Essentially, the ILP will encode that there exists $k > 0$ and $\boldsymbol{m} \geq 0$ such that $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m}$. This can be done since only "firing counts" matter, *i.e.* $\boldsymbol{m} \to^\pi_{\mathbb{Z}} \boldsymbol{m}'$ implies $\boldsymbol{m} \to^{\pi'}_{\mathbb{Z}} \boldsymbol{m}'$ for any permutation π' of π.

Let $\mathcal{N} = (P, T, F)$ be a workflow net. We define $\mathrm{ILP}_{\mathcal{N}} := \mathrm{A} \cdot \boldsymbol{x} \geq 0$ as an ILP with $|P| + |T| + 1$ inequalities and $|T| + 1$ variables. The variables of $\mathrm{ILP}_{\mathcal{N}}$ are $\boldsymbol{x} := (\kappa, \tau_1, \dots, \tau_{|T|})$. For ease of notation, we write $\boldsymbol{\tau} = (\tau_1, \dots, \tau_{|T|})$. We assume an implicit bijection between T and $[1..|T|]$, *i.e.* for every $t \in T$ there is a unique i such that: $\boldsymbol{\tau}[t] = \tau_i$. The matrix A is defined by the following inequalities:

1. $\kappa + \sum_{t \in T} \boldsymbol{\tau}[t] \cdot \Delta(t)[\mathsf{i}] \geq 0$,
2. $\kappa \geq 1$,
3. $\sum_{t \in T} \boldsymbol{\tau}[t] \cdot \Delta(t)[p] \geq 0$ for all $p \in P \setminus \{\mathsf{i}\}$,
4. $\tau_i \geq 0$ for all $i \in [1..|T|]$.

The first two inequalities concern the initial "budget" k of tokens in i which is represented by κ. Intuitively, $\kappa \geq 1$ has to be at least as much as $\boldsymbol{\tau}$ consumes from the initial place. The last two inequalities guarantee that we obtain a marking over $\mathbb{N}^P$ and that the "firing count" is over $\mathbb{N}^T$.

Let $\boldsymbol{\mu} \colon \boldsymbol{x} \to \mathbb{N}$ be a solution to $\mathrm{ILP}_{\mathcal{N}}$. We define

$$\mathrm{marking}(\boldsymbol{\mu}) \coloneqq \{\mathsf{i} \colon \mu(\kappa)\} + \sum_{t \in T}^{|T|} \mu(\tau_j) \cdot \Delta(t_j).$$

The following claim follows by definition of $\mathrm{ILP}_{\mathcal{N}}$ and $\to^*_{\mathbb{Z}}$.

Claim 5.7. *Let $\boldsymbol{m} \in \mathbb{N}^P$ and $k > 0$. It holds that $\{\mathsf{i} \colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m}$ iff there exists a solution $\boldsymbol{\mu}$ to $\mathrm{ILP}_{\mathcal{N}}$ such that $\mathrm{marking}(\boldsymbol{\mu}) = \boldsymbol{m}$ and $\boldsymbol{\mu}[\kappa] = k$.*

We conclude this part with the following bound.

Lemma 5.8. *Let $\mathcal{N}$ be a nonredundant workflow net. If $\mathcal{N}$ is strongly i-sound for all $1 \leq i < k$, and not strongly k-sound, then $k \leq c$, where c is the bound from Lemma 4.3 for $\mathrm{ILP}_{\mathcal{N}}$.*

Proof. For the sake of contradiction, assume that $k > c$ is as in the statement. Since $\mathcal{N}$ is not strongly k-sound, there exists a marking $\boldsymbol{m} \in \mathbb{N}^P$ such that $\{\mathsf{i} \colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m}$ and $\boldsymbol{m} \not\to^* \{\mathsf{f} \colon k\}$. By Claim 5.7, there exists a solution $\boldsymbol{\mu}$ to $\mathrm{ILP}_{\mathcal{N}}$ such that $\mathrm{marking}(\boldsymbol{\mu}) = \boldsymbol{m}$ and $\boldsymbol{\mu}[\kappa] = k$. By Lemma 4.3, there exists a solution $\boldsymbol{\mu}' \leq \boldsymbol{\mu}$ to $\mathrm{ILP}_{\mathcal{N}}$ such that $\boldsymbol{\mu}'[\kappa] \leq c < k = \boldsymbol{\mu}[\kappa]$ and $\mathrm{A}\boldsymbol{\mu}' \leq \mathrm{A}\boldsymbol{\mu}$, where A is the underlying matrix of $\mathrm{ILP}_{\mathcal{N}}$. The latter inequality implies $\mathrm{marking}(\boldsymbol{\mu}') \leq \mathrm{marking}(\boldsymbol{\mu})$.

Consider the vector $\boldsymbol{\pi} \coloneqq \boldsymbol{\mu} - \boldsymbol{\mu}'$. We prove that $\boldsymbol{\pi}$ is a solution to $\mathrm{ILP}_{\mathcal{N}}$. Since $\boldsymbol{\mu}' \leq \boldsymbol{\mu}$ we know that $\boldsymbol{\pi}$ is nonnegative. The inequalities of A are satisfied since $\mathrm{A}\boldsymbol{\pi} \geq 0 \equiv \mathrm{A}\boldsymbol{\mu} \geq \mathrm{A}\boldsymbol{\mu}'$ and $\boldsymbol{\mu}'[\kappa] \leq c < \boldsymbol{\mu}[\kappa]$. Thus, $\boldsymbol{\pi}$ is a solution to $\mathrm{ILP}_{\mathcal{N}}$.

By Claim 5.7, $\{\mathsf{i} \colon \boldsymbol{\mu}'[\kappa]\} \to^*_{\mathbb{Z}} \mathrm{marking}(\boldsymbol{\mu}')$ and $\{\mathsf{i} \colon \boldsymbol{\pi}[\kappa]\} \to^*_{\mathbb{Z}} \mathrm{marking}(\boldsymbol{\pi})$. Recall that $\boldsymbol{\mu}'[\kappa], \boldsymbol{\pi}[\kappa] < \boldsymbol{\mu}[\kappa] = k$. By assumption, $\mathcal{N}$ is strongly $\boldsymbol{\mu}'[\kappa]$-sound and strongly $\boldsymbol{\pi}[\kappa]$-sound. Therefore, $\mathrm{marking}(\boldsymbol{\mu}') \to^* \{\mathsf{f} \colon \boldsymbol{\mu}'[\kappa]\}$ and $\mathrm{marking}(\boldsymbol{\pi}) \to^* \{\mathsf{f} \colon \boldsymbol{\pi}[\kappa]\}$. Since the function $\mathrm{marking}(\cdot)$ is linear, we get

$$\boldsymbol{m} = \mathrm{marking}(\boldsymbol{\mu}) = \mathrm{marking}(\boldsymbol{\mu}') + \mathrm{marking}(\boldsymbol{\pi}).$$

This implies $\boldsymbol{m} \to^* \{\mathsf{f} \colon \boldsymbol{\mu}'[\kappa]\} + \{\mathsf{f} \colon \boldsymbol{\pi}[\kappa]\} = \{\mathsf{f} \colon k\}$, which is a contradiction. □

5.3 Reachability in $\mathbb{Z}$-bounded nets is in PSPACE

Note that $\{\mathsf{i} \colon 0\} = \{\mathsf{f} \colon 0\} = 0$. We will use these notations interchangeably depending on the emphasis.

Lemma 5.9. *Let $\mathcal{N} = (P, T, F)$ be a nonredundant workflow net and $k > 0$. If $\mathcal{N}$ is $\mathbb{Z}$-unbounded from $\{\mathsf{i} \colon k\}$, then $\mathcal{N}$ is not generalised sound.*

Proof. Since $\mathcal{N}$ is $\mathbb{Z}$-unbounded from $\{\mathsf{i} \colon k\}$, there exist $\boldsymbol{m}, \boldsymbol{m}'$ and π such that $\boldsymbol{m} < \boldsymbol{m}'$ and $\{\mathsf{i} \colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m} \to^{\pi}_{\mathbb{Z}} \boldsymbol{m}'$. Thus, $\{\mathsf{i} \colon 0\} \to^{\pi}_{\mathbb{Z}} \boldsymbol{m}' - \boldsymbol{m} > 0$. For the sake of contradiction, assume that $\mathcal{N}$ is generalised sound. It is strongly k-sound in particular for $k = 0$ by Lemma 5.6, so we have $\boldsymbol{m}' - \boldsymbol{m} \to^* \{\mathsf{f} \colon 0\}$, which contradicts the fact that $t^\bullet \neq 0$ for all $t \in T$. □

Lemma 5.10. *Let $\mathcal{N} = (P, T, F)$ be a workflow net. Let $\boldsymbol{m} \in \mathbb{N}^P$ be a marking such that $\|\boldsymbol{m}\| > \max(\|T\|, k)^2 \cdot (|P|+2) \cdot |P|$. If $\{\mathsf{i} \colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m}$ then $\mathcal{N}$ is $\mathbb{Z}$-unbounded.*

Proof. Let $\{\mathsf{i} \colon k\} \to^{\sigma}_{\mathbb{Z}} \boldsymbol{m}$ for some $\sigma = t_1 t_2 \cdots t_n$. We use the notation $\wr \cdot \int$ for multisets, *e.g.* $\wr a, a, b \int$ contains two occurrences of a and one of b. Without loss of generality, assume that no submultiset of $\wr t_1, t_2, \ldots, t_n \int$ sums to 0. Otherwise, we can shorten σ by removing such a submultiset.

By Lemma 4.5, we can assume that $t_1, t_2, \ldots, t_n$ are ordered so that there exist $0 \leq c_0 \leq c_1 \leq \ldots \leq c_n$, where

$$\left\| \{\mathsf{i} \colon k\} + \sum_{j=1}^{i} \Delta(t_j) - c_i \boldsymbol{m} \right\| \leq \max(\|T\|, k) \cdot (|P| + 2),$$

for all $i \in [0..n]$. Since $\|\boldsymbol{m}\| > \max(\|T\|, k)^2 \cdot (|P|+2) \cdot |P|$, we know that $n > \max(\|T\|, k) \cdot (|P|+2) \cdot |P|$. By the pigeonhole principle, there must be $0 \leq i_1 < i_2 \leq n$ such that

$$\{\mathsf{i} \colon k\} + \sum_{j=1}^{i_1} \Delta(t_j) - c_{i_1} \boldsymbol{m} = \{\mathsf{i} \colon k\} + \sum_{j=1}^{i_2} \Delta(t_j) - c_{i_2} \boldsymbol{m}.$$

This is equivalent to

$$\sum_{j=i_1+1}^{i_2} \Delta(t_j) = (c_{i_2} - c_{i_1}) \boldsymbol{m}.$$

We have $(c_{i_2} - c_{i_1}) \boldsymbol{m} \geq 0$ and, since no subset of $\wr t_1, t_2, \ldots, t_n \int$ sums to 0, we have a strict inequality. Let $\boldsymbol{z} \coloneqq \sum_{j=i_1+1}^{i_2} \Delta(t_j)$. We proved that $\{\mathsf{i} \colon 0\} \to^*_{\mathbb{Z}} \boldsymbol{z} > 0$, so $\mathcal{N}$ is $\mathbb{Z}$-unbounded. □

We are ready to prove the PSPACE membership of generalised soundness.

Proof of Theorem 5.1. Consider a workflow net $\mathcal{N} = (P, T, F)$. By Proposition 5.2, we can assume that $\mathcal{N}$ is a nonredundant workflow net. By Lemma 5.6 and Lemma 5.8, to prove generalised soundness it suffices to prove that it is k-sound for all $k \leq \|\mathcal{N}\|^{\mathrm{poly}(|\mathcal{N}|)}$.

By Lemma 5.9 and Lemma 5.10, if $\{\mathsf{i} \colon k\} \to^* \boldsymbol{m}$ and $\|\boldsymbol{m}\| \geq C_k$ for some $C_k = (\|\mathcal{N}\| + k)^{\mathrm{poly}(|\mathcal{N}|)}$, then the net is unsound. Since we need to consider only $k \leq \|\mathcal{N}\|^{\mathrm{poly}(|\mathcal{N}|)}$, all constants C_k are bounded exponentially and can be written in polynomial space.

Thus, to verify k-soundness we proceed as follows. First, we check if a configuration $\boldsymbol{m}$ such that $\|\boldsymbol{m}\| \geq C_k$ can be reached. This can be easily performed in NPSPACE = PSPACE as such a run would be witnessed by a sequence of configurations, such that each configuration can be stored in polynomial space. If such a configuration can be reached, then the algorithm outputs no. Otherwise, for every $\boldsymbol{m} \in \mathbb{N}^P$ such that $\|\boldsymbol{m}\| < C_k$ one needs to verify whether $\{\mathsf{i} \colon k\} \to^* \boldsymbol{m}$ implies $\boldsymbol{m} \to^* \{\mathsf{f} \colon k\}$. This can be done in coNPSPACE = coPSPACE = PSPACE. □

5.4 PSPACE-hardness

A *conservative Petri net* is a Petri net $\mathcal{N} = (P, T, F)$ such that transitions preserve the number of tokens. That is, for all $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{N}^P$, it is the case that $\boldsymbol{m} \to \boldsymbol{m}'$ implies $\sum_{p \in P} \boldsymbol{m}[p] = \sum_{p \in P} \boldsymbol{m}'[p]$. The *reachability problem* for conservatrice Petri

nets asks whether $\boldsymbol{m} \to^* \boldsymbol{m}'$, given $\mathcal{N}$, a source marking $\boldsymbol{m}$ and a target marking $\boldsymbol{m}'$.

Theorem 5.11. *Generalised soundness is PSPACE-hard.*

Proof. We give a reduction from reachability in conservative Petri nets, which is known to be PSPACE-complete [19].

Let $\mathcal{N} = (P, T, F)$ be a conservative Petri net, and let $\boldsymbol{m}, \boldsymbol{m}'$ be the source and target markings. We define the constant $c \coloneqq \sum_{p \in P} \boldsymbol{m}[p] = \sum_{p \in P} \boldsymbol{m}'[p]$.

We construct a workflow net $\mathcal{N}' = (P', T', F')$ such that $\mathcal{N}'$ is generalised sound if and only if $\boldsymbol{m} \to^* \boldsymbol{m}'$ in $\mathcal{N}$. To do so, we extend $\mathcal{N}$ with three new places $P' \coloneqq P \cup \{\mathsf{i}, \mathsf{f}, r\}$. Places i and f serve as decidated initial and final places, respectively. Place r will be used to reset configurations. It could be merged with i, if not for the restriction that, in a workflow net, place i cannot have any incoming arc.

We define $T' \supseteq T$ by keeping the existing transitions and adding $3 + |P|$ new transitions. Namely:

1. transition t_{i} defined by ${}^\bullet t_{\mathsf{i}} \coloneqq \{\mathsf{i} \colon 1\}$, and $t_{\mathsf{i}}^\bullet \coloneqq \{r \colon c\}$,
2. transition $t_{\boldsymbol{m}}$ defined by ${}^\bullet t_{\boldsymbol{m}} \coloneqq \{r \colon c\}$, and $t_{\boldsymbol{m}}^\bullet \coloneqq \boldsymbol{m}$,
3. transition $t_{\boldsymbol{m}'}$ defined by ${}^\bullet t_{\boldsymbol{m}'} \coloneqq \boldsymbol{m}'$, and $t_{\boldsymbol{m}'}^\bullet \coloneqq \{\mathsf{f} \colon 1\}$,
4. transition t_p defined by ${}^\bullet t_p \coloneqq \{p \colon 1\}$, and $t_p^\bullet \coloneqq \{r \colon 1\}$.

The first two transitions move a token from i and create the marking $\boldsymbol{m}$. The third transition consumes $\boldsymbol{m}'$ and puts a token into f. Transitions from the fourth group allow to move tokens from any place in the original Petri net P to r. See Figure 6 for a graphical presentation.

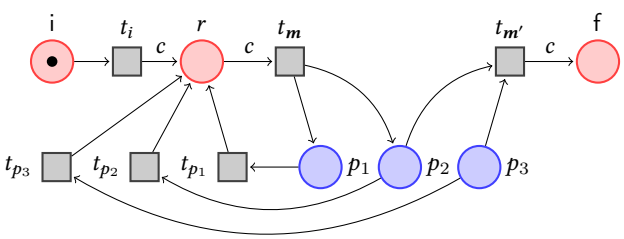


Figure 6. A workflow net $\mathcal{N}'$ which is generalised sound iff $m \to^* m'$ in the conservative Petri net $\mathcal{N} = (P, T, F)$. Here, $P = \{p_1, p_2, p_3\}$, $\boldsymbol{m} = \{p_1 \colon 1, p_2 \colon 1\}$, $\boldsymbol{m}' = \{p_2 \colon 1, p_3 \colon 1\}$ and $c = 2$. The original places are blue and the new places are red. We omit the original transitions (from T) in the picture.

It remains to show that $\mathcal{N}'$ is correct. Suppose $\mathcal{N}'$ is generalised sound. It must also be 1-sound and in particular $\{\mathsf{i} \colon 1\} \to^* \{\mathsf{f} \colon 1\}$. Since $\mathcal{N}$ is conservative, it is easy to see that $t_{\boldsymbol{m}}$ can be fired only if there are no tokens in P. Moreover, a token can be transferred to f only using $t_{\boldsymbol{m}'}$, which consumes $\boldsymbol{m}'$. Thus, we have $\boldsymbol{m} \to^* \boldsymbol{m}'$ in $\mathcal{N}$.

The converse implication is shown in the appendix. □

6 Structural soundness

In this section, we establish the EXPSPACE-completeness of structural soundness. Recall that the latter asks whether, given a workflow net, k-soundness holds for some $k \geq 1$.

6.1 EXPSPACE membership

Theorem 6.1. *Structural soundness is in EXPSPACE.*

Let $\mathcal{N} = (P, T, F)$ be a workflow net. We define an $(|T| + 2|P| + 1) \times (|T| + 1)$-ILP, called $\mathrm{ILP}^s_{\mathcal{N}}$. The variables are the same as for $\mathrm{ILP}_{\mathcal{N}}$ in Section 5.2: $(\kappa, \tau_1, \ldots, \tau_n)$, with the intuition that κ denotes the number of initial tokens and τ_i the number of times the transitions are used. We will keep the notation $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_n)$ and the notation $\boldsymbol{\tau}[t]$ for $t \in T$. The inequalities are defined as follows:

1. $\{\mathsf{i} \colon \kappa\} + \sum_{t \in T} \boldsymbol{\tau}[t] \cdot \Delta(t) = \{\mathsf{f} \colon \kappa\}$ (expressed with $2|P|$ inequalities);
2. $\boldsymbol{\tau} \geq 0$ ($|T|$ inequalities);
3. and $\kappa > 0$.

The first set of inequalities expresses that the effect of the transitions yields the final marking. The second type ensures that each transition is fired a nonnegative number of times. Finally the last one ensures that the initial marking has at least one token. The following is immediate.

Claim 6.2. *There exists $k > 0$ such that $\{\mathsf{i} \colon k\} \to^*_{\mathbb{Z}} \{\mathsf{f} \colon k\}$ if and only if there exists a solution $\boldsymbol{\mu}$ to $\mathrm{ILP}^s_{\mathcal{N}}$ such that $\boldsymbol{\mu}[\kappa] = k$.*

Lemma 6.3. *Let $\mathcal{N} = (P, T, F)$ be a nonredundant workflow net that is k-sound, and i-unsound for all $1 \leq i < k$. It is the case that $k \leq c + (\|T\| + 2)^{|T|} \cdot \max(\|T\|, c) \cdot |P|(|P| + 2)$, where c is the bound given by Lemma 4.3 for $\mathrm{ILP}^s_{\mathcal{N}}$.*

Proof. Towards a contradiction, suppose that $k > c + (\|T\| + 2)^{|T|} \cdot \max(\|T\|, c) \cdot |P|(|P| + 2)$. Consider $\mathrm{ILP}^s_{\mathcal{N}}$. Since $\mathcal{N}$ is k-sound, there is a run $\{\mathsf{i} \colon k\} \to^*_{\mathbb{Z}} \{\mathsf{f} \colon k\}$ and thus ILP_s has a solution $\boldsymbol{\mu}$. By Lemma 4.3, we can assume that $\boldsymbol{\mu} \leq \boldsymbol{c}$.

By Claim 6.2, $\{\mathsf{i} \colon \boldsymbol{\mu}[\kappa]\} \to^*_{\mathbb{Z}} \{\mathsf{f} \colon \boldsymbol{\mu}[\kappa]\}$. By Lemma 5.5, there exist $\ell \leq (\|T\| + 2)^{|T|} \cdot \max(\|T\|, \boldsymbol{\mu}[\kappa]) \cdot |P|(|P| + 2)$ and $\boldsymbol{m} \in \mathbb{N}^P$ such that $\{\mathsf{i} \colon \ell\} \to^* \boldsymbol{m}$ and $\{\mathsf{i} \colon \ell + \boldsymbol{\mu}[\kappa]\} \to^* \boldsymbol{m} + \{\mathsf{f} \colon \boldsymbol{\mu}[\kappa]\}$. Note that $\ell + \boldsymbol{\mu}[\kappa] < k$. Let $g \coloneqq k - (\ell + \boldsymbol{\mu}[\kappa]) > 0$. We have $\{\mathsf{i} \colon k\} = \{\mathsf{i} \colon \ell + \boldsymbol{\mu}[\kappa] + g\} \to^* \{\mathsf{i} \colon g\} + \boldsymbol{m} + \{\mathsf{f} \colon \boldsymbol{\mu}[\kappa]\}$. Since $\mathcal{N}$ is k-sound, we have

$$\{\mathsf{i} \colon g\} + \boldsymbol{m} + \{\mathsf{f} \colon \boldsymbol{\mu}[\kappa]\} \to^* \{\mathsf{f} \colon \ell + \boldsymbol{\mu}[\kappa] + g\}.$$

Thus, $\{\mathsf{i} \colon g\} + \boldsymbol{m} \to^* \{\mathsf{f} \colon \ell + g\}$. We obtain

$$\begin{aligned}\{\mathsf{i} \colon \ell + \boldsymbol{\mu}[\kappa] + g\} &\to^* \{\mathsf{i} \colon \boldsymbol{\mu}[\kappa] + g\} + \boldsymbol{m} \\ &\to^* \{\mathsf{i} \colon \boldsymbol{\mu}[\kappa]\} + \{\mathsf{f} \colon \ell + g\}.\end{aligned}$$

Therefore, since $\mathcal{N}$ is k-sound, it must be $\boldsymbol{\mu}[\kappa]$-sound (recall that tokens in f are never consumed). This contradicts the fact that $\mathcal{N}$ is i-unsound for all $1 \leq i < k$. □

We may now prove Theorem 6.1.

Proof of Theorem 6.1. By Proposition 5.2, we can assume that the input $\mathcal{N}$ is a nonredundant workflow net. By Lemma 6.3, it suffices to check if $\mathcal{N}$ is k-sound for some value k bounded exponentially in $\|\mathcal{N}\|$. First, we guess k, which can be written with polynomially many bits. Then, we test k-soundness in EXPSPACE via Corollary 3.7. □

6.2 EXPSPACE-hardness

Theorem 6.4. *Structural soundness is EXPSPACE-hard.*

Proof. Let $\mathcal{N}$ be a workflow net. We construct a workflow net $\mathcal{N}'$ which is structurally sound iff $\mathcal{N}$ is 1-sound. We simply add a single new transition t to $\mathcal{N}$ with ${}^\bullet t := \{\mathsf{i}\colon 2\}$ and $t^\bullet := \{\mathsf{f}\colon 1\}$. We show that $\mathcal{N}'$ is k-unsound for every $k \geq 2$. Towards a contradiction, suppose it is k-sound for some $k \geq 2$.

Notice that k cannot be even because $\{\mathsf{i}\colon k\} \to^{t^{k/2}} \{\mathsf{f}\colon k/2\}$ and f has no outgoing arcs, and hence $\{\mathsf{f}\colon k/2\} \not\to^* \{\mathsf{f}\colon k\}$. Thus, it is the case that $k \geq 3$ is odd and $\{\mathsf{i}\colon k\} \to^{t^*} \{\mathsf{i}\colon 1\} + \{\mathsf{f}\colon \lfloor k/2 \rfloor\}$. Since $\mathcal{N}$ is k-sound, $\{\mathsf{i}\colon 1\} \to^* \{\mathsf{f}\colon \lceil k/2 \rceil\}$. But that implies $\{\mathsf{i}\colon k\} \to^* \{\mathsf{f}\colon k \cdot \lceil k/2 \rceil\}$. Note that $k \cdot \lceil k/2 \rceil > k$ as $k \geq 3$, which yields a contradiction since f has no outgoing arcs to get rid of the extra tokens.

To conclude, we observe that if the initial configuration in $\mathcal{N}'$ is $\{\mathsf{i}\colon 1\}$, then it behaves like $\mathcal{N}$ would, since t will never be enabled, *i.e.* it is not quasi-live. Thus, $\mathcal{N}'$ is structurally sound if and only if $\mathcal{N}$ is 1-sound, and EXPSPACE-hardness follows from Theorem 3.10. □

7 Characterizing the set of sound numbers

Given a workflow net $\mathcal{N}$, we define the set $\mathrm{Sound}(\mathcal{N}) := \{k \geq 1 \mid \mathcal{N} \text{ is } k\text{-sound}\}$. That is, $\mathrm{Sound}(\mathcal{N})$ contains all the numbers for which $\mathcal{N}$ is sound (except 0 which is trivial as any workflow net is 0-sound). This section is dedicated to providing and computing a representation of $\mathrm{Sound}(\mathcal{N})$.

First, let us state a simple fact about $\mathrm{Sound}(\mathcal{N})$.[4]

Lemma 7.1. *The set $\mathrm{Sound}(\mathcal{N})$ is closed under subtraction with positive results.*

Proof. Let $g, k \in \mathrm{Sound}(\mathcal{N})$ be such that $g > k$. We show that $g - k \in \mathrm{Sound}(\mathcal{N})$. Since $k \in \mathrm{Sound}(\mathcal{N})$, we have $\{\mathsf{i}\colon g\} = \{\mathsf{i}\colon k + (g-k)\} \to^* \{\mathsf{f}\colon k\} + \{\mathsf{i}\colon g-k\}$. Since N is g-sound, it must also be $(g-k)$-sound. So, $g - k \in \mathrm{Sound}(\mathcal{N})$. □

Corollary 7.2. *There exist $p > 0$ and $k \in \mathbb{N} \cup \{+\infty\}$ such that $\mathrm{Sound}(\mathcal{N}) = \{i \cdot p \mid 1 \leq i < k\}$.*

By the above, $\mathrm{Sound}(\mathcal{N})$ is characterized by p and k. We thus say that a net is (k,p)-*sound* if and only if $\mathrm{Sound}(\mathcal{N}) = \{i \cdot p \mid 1 \leq i < k\}$. Note that $k = 0$ implies $\mathrm{Sound}(\mathcal{N}) = \emptyset$. Further, $k = +\infty$ if and only if $\mathrm{Sound}(\mathcal{N})$ is infinite. Finally, a workflow net is generalised sound iff it is $(1, +\infty)$-sound; and it is structurally sound iff there exist $p, k \geq 1$ such that it is (k,p)-sound. We show that k and p can be computed. This will rely on insights from the prior sections about the smallest numbers for which a net is unsound or sound.

Theorem 7.3. *Given a workflow net $\mathcal{N}$, the numbers p and k that characterize $\mathrm{Sound}(\mathcal{N})$ are bounded by $\|\mathcal{N}\|^{\mathrm{poly}\, O(|\mathcal{N}|)}$, and hence can be represented with polynomially many bits. Given $\mathcal{N}$, p' and k', the problem of deciding whether $\mathcal{N}$ is (k', p')-sound is in EXPSPACE. Moreover, the algorithm computes p and k such that $\mathcal{N}$ is (k,p)-sound.*

Proof. Consider a workflow net $\mathcal{N}$. By Proposition 5.2, we can assume that $\mathcal{N}$ is nonredundant. We will compute for which p and k the net $\mathcal{N}$ is (k,p)-sound. By Lemma 6.3, if $\mathrm{Sound}(\mathcal{N}) \neq \emptyset$, then there exists $G \leq \|\mathcal{N}\|^{\mathrm{poly}\, O(|\mathcal{N}|)}$ such that $\mathcal{N}$ is ℓ-sound for some $\ell \leq G$. By Corollary 3.7, it is possible to check 1-soundness, 2-soundness, …, G-soundness in EXPSPACE. Thus, in EXPSPACE, we can identify the smallest p such that $\mathcal{N}$ is p-sound.

It remains to compute k. Using Lemma 3.6, we construct a net $\mathcal{N}'$ which is c-sound if and only if $\mathcal{N}$ is cp-sound for all $c > 0$. Thus, the smallest number c for which $\mathcal{N}'$ is not c-sound is the smallest c such that $\mathcal{N}$ is not cp-sound. By Lemma 5.8, if $\mathrm{Sound}(\mathcal{N}') \neq \mathbb{N} \setminus \{0\}$ then there exists $G' \leq \|\mathcal{N}\|^{\mathrm{poly}\, O(|\mathcal{N}|)}$ such that $\mathcal{N}'$ is c-unsound for some $c \leq G'$. Thus, it suffices to check 1-soundness, 2-soundness, …, G'-soundness to identify whether $k = +\infty$, or to compute the largest $k \in \mathbb{N}$ such that $\mathcal{N}$ is pk-sound. By Corollary 3.7, k can be computed in EXPSPACE. □

8 Conclusion

In this work, we settled, after around two decades, the complexity of the main decision problems concerning workflow nets: k-soundness, classical soundness, generalised soundness and structural soundness. The first three are EXPSPACE-complete, while the latter is PSPACE-complete and hence surprisingly simpler. We have further characterised the set of sound numbers of workflow nets: they have a specific shape that can be computed with exponential space.

As further work, we intend to study extensions of these problems in the context of Petri nets. For example, a natural extension of generalised soundness asks, given markings $\boldsymbol{m}$ and $\boldsymbol{m}'$, whether for every $k \in \mathbb{N}$, every marking reachable from $k \cdot \boldsymbol{m}$ may lead to $k \cdot \boldsymbol{m}'$. Contrary to workflow nets, a Petri net that satisfies this property needs not to be bounded.

References

[1] Kamel Barkaoui and Laure Petrucci. Structural analysis of workflow nets with shared resources. In *Proc. Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM)*, volume 98/7, pages 82–95, 1998.

[2] Michael Blondin. The ABCs of Petri net reachability relaxations. *ACM SIGLOG News*, 7(3), 2020. doi:10.1145/3436980.3436984.

[3] Zakaria Bouziane and Alain Finkel. Cyclic petri net reachability sets are semi-linear effectively constructible. In *Second International Workshop on Verification of Infinite State Systems (INFINITY)*, volume 9 of

[4] A similar observation was made, but not explicitly stated, in [7, Lemma 2.2 and 2.3].

Electronic Notes in Theoretical Computer Science, pages 15–24, 1997. doi:10.1016/S1571-0661(05)80423-2.

[4] E. Cardoza, Richard J. Lipton, and Albert R. Meyer. Exponential space complete problems for Petri nets and commutative semigroups: Preliminary report. In *Proc. 8th Annual ACM Symposium on Theory of Computing (STOC)*, pages 50–54, 1976. doi:10.1145/800113.803630.

[5] Allan Cheng, Javier Esparza, and Jens Palsberg. Complexity results for 1-safe nets. In *Proc. 13th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 761, pages 326–337, 1993. doi:10.1007/3-540-57529-4_66.

[6] Dmitry Chistikov and Christoph Haase. The Taming of the Semi-Linear Set. In *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 55, pages 128:1–128:13, 2016. doi:10.4230/LIPIcs.ICALP.2016.128.

[7] Ferucio Laurenţiu Ţiplea and Dan Cristian Marinescu. Structural soundness of workflow nets is decidable. *Information Processing Letters*, 96(2):54–58, 2005. doi:10.1016/j.ipl.2005.06.002.

[8] Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2021. to appear.

[9] Wil Van der Aalst. Interorganizational workflows: An approach based on message sequence charts and Petri nets. *Systems Analysis, Modelling, Simulation*, 34(3):335–367, 1999.

[10] Jörg Desel and Javier Esparza. *Free Choice Petri Nets*. Cambridge University Press, 1995. doi:10.1017/CBO9780511526558.

[11] Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for integer programming using the Steinitz lemma. *ACM Transactions on Algorithms (TALG)*, 16(5):5:1–5:14, 2019. doi:10.1145/3340322.

[12] Estíbaliz Fraca and Serge Haddad. Complexity analysis of continuous Petri nets. *Fundamenta Informaticae*, 137(1):1–28, 2015. doi:10.3233/FI-2015-1168.

[13] Christoph Haase and Simon Halfon. Integer vector addition systems with states. In *Proc. 8th International Workshop on Reachability Problems (RP) 2014*, volume 8762, pages 112–124, 2014. doi:10.1007/978-3-319-11439-2_9.

[14] Michel Henri Théodore Hack. *Decidability questions for Petri Nets*. PhD thesis, Massachusetts Institute of Technology, 1976.

[15] Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2021. to appear.

[16] Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *Proc. 34th Symposium on Logic in Computer Science (LICS)*, 2019.

[17] Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *Proc. 13th Symposium on Theory of Computing (STOC)*, pages 238–246, 1981. doi:10.1145/800076.802477.

[18] Ernst W Mayr and Albert R Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46(3):305–329, 1982. doi:10.1016/0001-8708(82)90048-2.

[19] Ernst W. Mayr and Jeremias Weihmann. A framework for classical Petri net problems: Conservative Petri nets as an application. In *Proc. 35th International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS)*, pages 314–333, 2014. doi:10.1007/978-3-319-07734-5_17.

[20] Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6:223–231, 1978. doi:10.1016/0304-3975(78)90036-1.

[21] Ernst Steinitz. Bedingt konvergente Reihen und konvexe Systeme. 1913.

[22] Wil M. P. van der Aalst. Verification of workflow nets. In *Proc. 18th International Conference on Application and Theory of Petri Nets (ICATPN)*, volume 1248, pages 407–426, 1997. doi:10.1007/3-540-63139-9_48.

[23] Wil M. P. van der Aalst. The application of Petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998. doi:10.1142/S0218126698000043.

[24] Wil M. P. van der Aalst and Christian Stahl. *Modeling Business Processes - A Petri Net-Oriented Approach*. Cooperative Information Systems series. MIT Press, 2011.

[25] Wil M. P. van der Aalst, Kees M. van Hee, Arthur H. M. ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011. doi:10.1007/s00165-010-0161-4.

[26] Wil MP Van der Aalst. Structural characterizations of sound workflow nets. *Computing science reports*, 96(23):18–22, 1996.

[27] Boudewijn F. van Dongen, Ana Karla A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and Wil M. P. van der Aalst. The prom framework: A new era in process mining tool support. In *Proc. 26th International Conference on Applications and Theory of Petri Nets (ICATPN)*, volume 3536, pages 444–454, 2005. doi:10.1007/11494744_25.

[28] Kees M. van Hee, Olivia Oanea, Natalia Sidorova, and Marc Voorhoeve. Verifying generalized soundness of workflow nets. In *Proc. 6th International Andrei Ershov Memorial Conference on Perspectives of Systems Informatics (PSI)*, pages 235–247, 2006. doi:10.1007/978-3-540-70881-0_21.

[29] Kees M. van Hee, Natalia Sidorova, and Marc Voorhoeve. Soundness and separability of workflow nets in the stepwise refinement approach. In *Proc. 24th International Conference on Applications and Theory of Petri Nets 2003 (ICATPN)*, volume 2679, pages 337–356, 2003. doi:10.1007/3-540-44919-1_22.

[30] Kees M. van Hee, Natalia Sidorova, and Marc Voorhoeve. Generalised soundness of workflow nets is decidable. In *Proc. 25th International Conference on Applications and Theory of Petri Nets (ICATPN)*, pages 197–215, 2004. doi:10.1007/978-3-540-27793-4_12.

[31] Joachim von zur Gathen and Malte Sieveking. A bound on solutions of linear integer equalities and inequalities. *Proceedings of the American Mathematical Society*, 72(1):155–158, 1978.

Appendix

Missing proofs of Section 3

Proposition 3.2. *A workflow net $\mathcal{N}$ is 1-sound iff $\mathcal{N}_{sc}$ is bounded and transition t_{sc} is live from $\{\mathsf{i}\colon 1\}$.*

Proof. Let $\mathcal{N} = (P, T, F)$.

$\Rightarrow$) For the sake of contradiction, suppose that $\mathcal{N}_{sc}$ is unbounded. There exist markings $\boldsymbol{m} < \boldsymbol{m}'$ such that $\{\mathsf{i}\colon i\} \to^{\pi} \boldsymbol{m} \to^{\pi'} \boldsymbol{m}'$ in $\mathcal{N}_{sc}$. Let us assume, without loss of generality, that no marking repeats along the run. There are two cases to consider: either $\pi\pi'$ contains t_{sc}, or not.

Let us argue that the first case cannot hold. For the sake of contradiction, assume it does. Let σt_{sc} be the shortest prefix of $\pi\pi'$ such that $\{\mathsf{i}\colon 1\} \to^{\sigma} \{\mathsf{f}\colon 1\} + \boldsymbol{n} \to^{t_{sc}} \{\mathsf{i}\colon 1\} + \boldsymbol{n}$ in $\mathcal{N}$. If $\boldsymbol{n} = 0$, then we obtain a contradiction as no marking repeats. Otherwise, by 1-soundness, we must have $\boldsymbol{n} \to^* 0$, which is a contradiction as $t^\bullet \neq 0$ for every $t \in T$.

Thus, $\pi\pi'$ only contains transitions from T, which means we can reason about $\mathcal{N}$ (rather than $\mathcal{N}_{sc}$). By 1-soundness, we have $\{\mathsf{i}\colon 1\} \to^* \boldsymbol{m} \to^* \{\mathsf{f}\colon 1\}$ in $\mathcal{N}$. Since $\{\mathsf{i}\colon 1\} \to^* \boldsymbol{m}'$ in $\mathcal{N}$, altogether this yields

$$\{\mathsf{i}\colon 1\} \to^* \boldsymbol{m}' = \boldsymbol{m} + (\boldsymbol{m}' - \boldsymbol{m}) \to^* \{\mathsf{f}\colon 1\} + (\boldsymbol{m}' - \boldsymbol{m}).$$

By 1-soundness, this means that $(\boldsymbol{m}' - \boldsymbol{m}) \to^* 0$, which is impossible. Consequently, $\mathcal{N}_{sc}$ is bounded from $\{\mathsf{i}\colon 1\}$.

It remains to argue that t_{sc} is live from $\{\mathsf{i}\colon 1\}$. Let $\{\mathsf{i}\colon 1\} \to^{\rho} \boldsymbol{m}$ in $\mathcal{N}_{sc}$, where no marking repeats. We can assume that t_{sc} does not appear in ρ as it would mean that $\{\mathsf{i}\colon 1\}$ is repeated. Hence, $\{\mathsf{i}\colon 1\} \to^{\rho} \boldsymbol{m}$ in $\mathcal{N}$. By 1-soundness, we have $\boldsymbol{m} \to^* \{\mathsf{f}\colon 1\}$, from which t_{sc} is enabled as desired.

$\Leftarrow$) Let $\{\mathsf{i}\colon 1\} \to^* \boldsymbol{m}$ in $\mathcal{N}$ (and so in $\mathcal{N}_{sc}$). Since t_{sc} is live from $\{\mathsf{i}\colon 1\}$, we have $\boldsymbol{m} \to^* \{\mathsf{f}\colon 1\} + \boldsymbol{n}$ for some $\boldsymbol{n} \in \mathbb{N}^P$. If $\boldsymbol{n} > 0$, then we obtain $\{\mathsf{i}\colon 1\} \to^* \{\mathsf{f}\colon 1\} + \boldsymbol{n} \to^{t_{sc}} \{\mathsf{i}\colon 1\} + \boldsymbol{n}$ which violates boundedness. Thus, $\boldsymbol{n} = 0$, and hence $\boldsymbol{m} \to^* \{\mathsf{f}\colon 1\}$ as desired. □

Theorem 3.10. *The classical soundness and 1-soundness problems are EXPSPACE-hard.*

Proof. Recall that in the proof within the main text, we have shown the implication from right to left of Equation (1). It remains to prove the other direction.

$\Rightarrow$) Suppose that $\boldsymbol{m} \to^* \boldsymbol{m}'$. First, we prove that $\mathcal{N}'$ is 1-sound. Consider a configuration $\{\mathsf{i}\colon 1\} \to^{\rho} \boldsymbol{n}$ for some run ρ. We consider cases depending on ρ and $\boldsymbol{n}$.

Case 0: suppose that t_{hard} does not occur in ρ. If $\boldsymbol{n} = \{\mathsf{i}\colon 1\}$ then $\boldsymbol{n} \to^{t_{\text{simple}} t_{\text{simple2}}} \{\mathsf{f}\colon 1\}$. Otherwise, either $\boldsymbol{n} = \{\mathsf{f}\colon 1\}$ or

$$\boldsymbol{n}' = \{p_{\text{simple}}\colon 1\} + \sum_{p \in P} \{p\colon \|\mathcal{N}\|, \overline{p}\colon \|\mathcal{N}\|\} \to^{\rho'} \boldsymbol{n},$$

where $\rho' \in T_1^*$. Since T_1 is reversible, we also have $\boldsymbol{n} \to^* \boldsymbol{n}'$. This concludes this case as $\boldsymbol{n}' \to^{t_{\text{simple2}}} \{\mathsf{f}\colon 1\}$. Notice that in the remaining cases the transitions t_{simple} and t_{simple2} can never be fired.

Case 1: suppose that t_{hard} occurs in ρ but t_{start} does not. It is easy to see that $\rho \in t_{\text{hard}} T_2^*$. Since T_2 is reversible and by Lemma 3.9 (1) , it is the case that

$$\boldsymbol{n} \to^* \{s\colon 1, c\colon 1\} \to^* \{f\colon 1, c\colon 1, b\colon c_n\} + \sum_{\overline{p} \in \overline{P}} \{\overline{p}\colon c_n\}.$$

Note that t_{start} is fireable from the latter configuration, and thus we can extend ρ with that transition. We will discuss how to proceed from there in the next case.

Case 2: suppose that t_{start} occurs in ρ, but t_{end} does not. We have $\rho \in t_{\text{hard}} T_2^* t_{\text{start}} (T_1 \cup \{t_{\boldsymbol{m}}, t_{\boldsymbol{m}'}, t_{\boldsymbol{m}'}^{-1}, t_{\text{reach}}, t_{\text{reach}}^{-1}\})^*$. By Lemma 3.9 (2), the transition t_{start} in ρ is fired between the configurations

$$\{f\colon 1, c\colon 1, b\colon c_n\} + \sum_{\overline{p} \in \overline{P}} \{\overline{p}\colon c_n\} \to^{t_{\text{start}}} \{p_{\text{start}}\colon 1, b\colon c_n\} + \sum_{\overline{p} \in \overline{P}} \{\overline{p}\colon c_n\}.$$

If $\boldsymbol{n}$ is the latter configuration, then only $t_{\boldsymbol{m}}$ can be fired. Otherwise, the transition $t_{\boldsymbol{m}}$ is never available again. In any case, since the transitions $T_1 \cup \{t_{\boldsymbol{m}'}, t_{\boldsymbol{m}'}^{-1}, t_{\text{reach}}, t_{\text{reach}}^{-1}\}$ are reversible, $\boldsymbol{n} \to^* \{p_{\text{inProgress}}\colon 1, b\colon c_n\} + \boldsymbol{m} + \sum_{\overline{p} \in \overline{P}} \{\overline{p}\colon c_n - \boldsymbol{m}[p]\}$.

Since $\boldsymbol{m} \to^* \boldsymbol{m}'$, and by Lemma 3.8, we know that

$$\{p_{\text{inProgress}}\colon 1, b\colon c_n\} + \boldsymbol{m} + \sum_{\overline{p} \in \overline{P}} \{\overline{p}\colon c_n - \boldsymbol{m}[p]\} \to^*$$

$$\{p_{\text{inProgress}}\colon 1, b\colon c_n\} + \boldsymbol{m}' + \sum_{\overline{p} \in \overline{P}} \{\overline{p}\colon c_n - \boldsymbol{m}'[p]\} \to^{t_{\boldsymbol{m}'}}$$

$$\{p_{\text{cover}}\colon 1, b\colon c_n\} + \sum_{\overline{p} \in \overline{P}} \{\overline{p}\colon c_n\} \to^{t_{\text{reach}}}$$

$$\{f^\heartsuit\colon 1, c^\heartsuit\colon 1, b\colon c_n\} + \sum_{\overline{p} \in \overline{P}} \{\overline{p}\colon c_n\}.$$

By Lemma 3.9 (1), we get

$$\{f^\heartsuit\colon 1, c^\heartsuit\colon 1, b\colon c_n\} + \sum_{\overline{p} \in \overline{P}} \{\overline{p}\colon c_n\} \to^* \{s^\heartsuit\colon 1, c^\heartsuit\colon 1\}.$$

Then, by firing t_{end}, we reach $\{\mathsf{f}\colon 1\}$ as required.

Case 3: suppose that t_{end} occurs in ρ. We divide the run into the fragments where t_{start} and t_{end} were used for the first time. It is the case that $\rho = \rho_1 t_{\text{start}} t_{\boldsymbol{m}} \rho_2 t_{\text{end}} \rho_3$, where

$$\{\mathsf{i}\colon 1\} \to^{\rho_1 t_{\text{start}} t_{\boldsymbol{m}}} \{p_{\text{inProgress}}\colon 1, b\colon c_n\} + \boldsymbol{m} + \sum_{\overline{p} \in \overline{P}} \{\overline{p}\colon c_n - \boldsymbol{m}[p]\} = \boldsymbol{n}_1,$$

and ρ_2 consists only of transitions in

$$T_1 \cup T_3 \cup \{t_{\boldsymbol{m}'}, t_{\boldsymbol{m}'}^{-1}, t_{\text{reach}}, t_{\text{reach}}^{-1}\}.$$

From Lemma 3.9 (2) and definition of transitions in T' for every reachable configuration $\{\mathsf{i}\colon 1\} \to^* \boldsymbol{n}'$ in $\mathcal{N}'$

$$\boldsymbol{n}'[f] + \boldsymbol{n}'[p_{\text{inProgress}}] + \boldsymbol{n}'[p_{\text{cover}}] = 1. \tag{3}$$

In other words, there can be a token only in one of the three places. Recall that transitions in $T_1 \cup \{t_{\boldsymbol{m}}\}$ can be fired only if $\boldsymbol{n}'[p_{\text{inProgress}}] = 1$ (as $\boldsymbol{n}'[p_{\text{inProgress}}] = \boldsymbol{n}'[p_{\text{canFire}}]$); and transitions in $\{t_{\boldsymbol{m}'}^{-1}, t_{\text{reach}}\}$ only if $\boldsymbol{n}'[p_{\text{cover}}] = 1$.

Consider the Petri net $\mathcal{N}'' = (P'', T'', F'')$, which is as (P', T', F') but with places reduced to $P'' \coloneqq P^{\heartsuit}$ (recall that $b = b^{\heartsuit}$) and transitions T'' the same as T' projected onto P''. Notice that among transitions used in ρ_2 only transitions in $T_3 \cup \{t_{\text{reach}}, t_{\text{reach}}^{-1}\}$ have impact on P''. Slightly abusing the notation, we keep the names of transitions in $\mathcal{N}''$ (from $\mathcal{N}'$); and similarly for configurations. Since $\boldsymbol{n}_1 \to^{\rho_2} \boldsymbol{v}$ such that $\boldsymbol{v} \geq \{s^{\heartsuit}\colon 1, c^{\heartsuit}\colon 1\}$ (as t_{end} can be fired afterwards), we get in $\mathcal{N}''$:

$$\{b\colon c_n\} \to^{\rho_2} \boldsymbol{v}.$$

Thus by Lemma 3.9 (4) and Eq. (3) we get that

$$\rho_2 \in T_{1+}^* t_{\text{reach}} T_3^* \left(t_{\text{reach}}^{-1} T_{1+}^* t_{\text{reach}} T_3^*\right)^*.$$

where $T_{1+} = T_1 \cup \{t_{\boldsymbol{m}} \cup t_{\boldsymbol{m}}^{-1}\}$. Consider the configurations in $\mathcal{N}''$ after firing transitions in ρ_2 starting from $\boldsymbol{n}_1$. We claim that every time after t_{reach} was fired the configuration is $\boldsymbol{v}' = \{b\colon c_n, f^{\heartsuit}\colon 1, c^{\heartsuit}\colon 1\}$. Indeed, after the first time this is because $\{b\colon c_n\} \to^{t_{\text{reach}}} \boldsymbol{v}'$ (recall that transitions in T_{1+} have no impact on $\mathcal{N}''$). For the remaining cases, notice that between t_{reach} and t_{reach}^{-1} only transitions from T_3 are fired. By Lemma 3.9 (1 and 2) after firing t_{reach}^{-1} the configuration has to be $\{b\colon c_n\}$. Since transitions in T_{1+} have no impact on $\mathcal{N}''$ we are ready to conclude the proof.

Let $\rho_2 = \rho_{pre} t_{\text{reach}} \rho_{suf}$ such that ρ_{suf} does not contain t_{reach}. We know that

$$\boldsymbol{n}_1 \to^{\rho_{pre}} \{b\colon c_n\} \to^{t_{\text{reach}}} \boldsymbol{v}' \to^{\rho_{suf}} \boldsymbol{v}$$

in $\mathcal{N}''$, and recall that $\boldsymbol{v} \geq \{s^{\heartsuit}\colon 1, c^{\heartsuit}\colon 1\}$. By Lemma 3.9 (3), we get $\boldsymbol{v} = \{s^{\heartsuit}\colon 1, c^{\heartsuit}\colon 1\}$ in $\mathcal{N}''$.

Let us analyse the vector $\boldsymbol{v}$ in $\mathcal{N}'$. By Eq. (3) we know that $\boldsymbol{v}[p_{\text{inProgress}}] = \boldsymbol{v}[p_{\text{cover}}] = 0$. Note that

$$\boldsymbol{v}[b] = \boldsymbol{v}[p] + \boldsymbol{v}[\overline{p}], \tag{4}$$

for all $p \in P$. This is easy to see since every transition preserves this equality for all reachable configuration. Thus $\boldsymbol{v}[p] = \boldsymbol{v}[\overline{p}] = 0$ for all $p \in P$. Notice that this concludes the proof as $\boldsymbol{v} \to^{t_{\text{end}}} \{\mathsf{f}\colon 1\}$ and thus ρ_3 is an empty run and $\boldsymbol{n} = \{\mathsf{f}\colon 1\}$. □

Missing proofs of Section 4

Lemma 4.3. *Let $G = \mathrm{A} \cdot \boldsymbol{x} \geq \boldsymbol{b}$ be an $(m \times n)$-ILP, where $\boldsymbol{b} \geq 0$. There exists $c \leq \|G\|^{O((m+n)\log(m+n))}$ such that the following holds. For every $\boldsymbol{\mu} \in [\![G]\!]_{\geq 0}$, there exists $\boldsymbol{\nu} \in [\![G]\!]_{\geq 0}$ such that $\boldsymbol{\nu} \leq \boldsymbol{\mu}$, $\boldsymbol{\nu} \leq \boldsymbol{c}$, and $\mathrm{A} \cdot \boldsymbol{\nu} \leq \mathrm{A} \cdot \boldsymbol{\mu}$.*

Proof. Let $x_1, \ldots, x_n$ be the variables of G. We define a $(3m \times (m+n))$-ILP G' by slightly modifying G. For every inequality in the original ILP G, we add one fresh variable. We denote them $y_1, \ldots, y_m$. Now, recall that the inequalities in G are of the form: $\sum_{i=1}^{n} \mathrm{A}[j,i] \cdot x_i \geq \boldsymbol{b}[j]$ for $j \in [1..m]$. The ILP G' is defined with the same inequalities, plus m new equalities (recall that this requires $2m$ inequalities): $\sum_{i=1}^{n} \mathrm{A}[j,i] \cdot x_i - y_j = 0$ for $j \in [1..m]$.

Notice that, in solutions for G', the variables y_j are uniquely determined by the valuation of $x_1, \ldots, x_n$. For convenience, we will write $\boldsymbol{\mu}[x_i], \boldsymbol{\mu}'[y_j]$ when referring to the components of solutions. For every $\boldsymbol{\mu} \in [\![G]\!]_{\geq 0}$, there is a unique $\boldsymbol{\mu}' \in [\![G']\!]$ such that $\boldsymbol{\mu}'[x_i] = \boldsymbol{\mu}[x_i]$ for all $i \in [1..n]$. Thus, since $\boldsymbol{b} \geq 0$, we have $[\![G']\!]_{\geq 0} = \{\boldsymbol{\mu}' \mid \boldsymbol{\mu} \in [\![G]\!]\}$.

We define c as the constant from Lemma 4.2 for G'. Now, let $\boldsymbol{\mu} \in [\![G]\!]_{\geq 0}$ and let $\boldsymbol{\mu}' \in [\![G']\!]_{\geq 0}$ be its corresponding solution. By Lemma 4.2, there exists $\boldsymbol{\nu}' \in [\![G']\!]_{\geq 0}$ such that $\boldsymbol{\nu}' \leq \boldsymbol{\mu}'$ and $\boldsymbol{\nu}' \leq \boldsymbol{c}$. We define $\boldsymbol{\nu} \in [\![G]\!]_{\geq 0}$ as the solution corresponding to $\boldsymbol{\nu}'$. It is clear that $\boldsymbol{\nu} \leq \boldsymbol{\mu}$ and $\boldsymbol{\nu} \leq \boldsymbol{c}$. For the remaining part, fix $j \in [1..m]$. Recall that $\boldsymbol{\nu}'[y_j] = \sum_{i=1}^{n} \mathrm{A}[j,i] \cdot \boldsymbol{\nu}'[x_i]$ and $\boldsymbol{\mu}'[y_j] = \sum_{i=1}^{n} \mathrm{A}[j,i] \cdot \boldsymbol{\mu}'[x_i]$. Thus,

$$\sum_{i=1}^{n} \mathrm{A}[j,i] \cdot \boldsymbol{\nu}[x_i] \leq \sum_{i=1}^{n} \mathrm{A}[j,i] \cdot \boldsymbol{\mu}[x_i],$$

which concludes the proof. □

Lemma 4.5. *Let $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{Z}^d$, $b \coloneqq \max_{j=0}^{n} \|\boldsymbol{x}_j\|$, and $\boldsymbol{z} \coloneqq \sum_{j=0}^{n} \boldsymbol{x}_j$. There exists a permutation π of $[0..n]$ such that: $\pi(0) = 0$; and there exist $0 \leq c_0 \leq c_1 \leq \ldots \leq c_n$, where*

$$\left\| \sum_{j=0}^{i} \boldsymbol{x}_{\pi(j)} - c_i \cdot \boldsymbol{z} \right\| \leq b(d+2) \qquad \textit{for all } i \in [0..n].$$

Proof. Let $c \coloneqq \|\boldsymbol{z}\|$. We can assume that $b, c > 0$, as otherwise the proof follows immediately from Lemma 4.4. We use the notation $\wr \cdot \int$ for multisets, *e.g.* $\wr a, a, b \int$ contains two occurrences of a and one occurrence of b. Let $V \coloneqq \wr \boldsymbol{x}_0, \ldots, \boldsymbol{x}_n \int$ and let $W \coloneqq \wr -\boldsymbol{z}/c, \ldots, -\boldsymbol{z}/c \int$ be c copies of the same vector. Clearly, $\sum_{\boldsymbol{x} \in V \cup W} \boldsymbol{x} = 0$ and $\|\boldsymbol{x}\| \leq b$ for all $\boldsymbol{x} \in V'$.

Consider the multiset of vectors $X \coloneqq \wr \boldsymbol{x}/b \mid \boldsymbol{x} \in V \cup W \int$, *i.e.* rescaled vectors from $V \cup W$. By Lemma 4.4 we can order the vectors in X: $\boldsymbol{a}'_0, \ldots, \boldsymbol{a}'_{n+c}$, so that

$$\left\| \sum_{j=0}^{i} \boldsymbol{a}'_j \right\| \leq d \text{ for all } i \in [0..n+c].$$

By scaling the vectors back, we get an order $\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{n+c}$ on vectors in $V \cup W$ such that

$$\left\| \sum_{j=0}^{i} \boldsymbol{a}_j \right\| \leq bd \text{ for all } i \in [0..n+c]. \tag{5}$$

Let $0 \leq s_0 < s_1 < \ldots < s_n \leq n+c$ be indices such that $V = \wr \boldsymbol{a}_{s_0}, \ldots, \boldsymbol{a}_{s_n} \int$. Fix some $i \in [0..n]$. Note that the number of remaining vectors is

$$\left| \wr \boldsymbol{a}_j \mid j \notin \{s_0, \ldots, s_n\}, j \leq i \int \right| = s_i - i$$

for all $i \in [0..n]$. By (5),

$$\left\| \sum_{j=0}^{i} \boldsymbol{a}_{s_j} - \frac{s_i - i}{c} \boldsymbol{z} \right\| = \left\| \sum_{j=0}^{s_i} \boldsymbol{a}_j \right\| \leq bd. \tag{6}$$

Thus, by defining $c_i := \frac{s_i - i}{c}$, we get $0 \leq c_0 \leq c_1 \leq \ldots \leq c_n$. To conclude the lemma, it remains to show that we can reorder $\boldsymbol{a}_{s_0}, \ldots, \boldsymbol{a}_{s_n}$ so that $\boldsymbol{a}_{s_0} = \boldsymbol{x}_0$. Indeed, suppose that $\ell \in [0..n]$ is an index such that $\boldsymbol{a}_{s_\ell} = \boldsymbol{x}_0$. By (6) and the triangle inequality

$$\left\| \sum_{j=0}^{i} \boldsymbol{a}_{s_j} - \frac{s_i - i}{c} \boldsymbol{z} + \boldsymbol{a}_{s_\ell} - \boldsymbol{a}_{s_0} \right\| \leq b(d+2).$$

Therefore, by swapping $\boldsymbol{a}_{s_\ell}$ and $\boldsymbol{a}_{s_0}$ we obtain the desired permutation. □

Missing proofs of Section 5

Proposition 5.2. *Given a workflow net $\mathcal{N}$, one can identify and remove all redundant places from it in polynomial time. The resulting workflow net $\mathcal{N}'$ is nonredundant. Moreover, $\mathcal{N}$ is k-sound if and only if $\mathcal{N}'$ is k-sound for all $k \in \mathbb{N}$.*

Proof. Redundancy can be checked with a saturation algorithm. We give a short proof that relies on the decidability of continuous Petri nets (which internally uses such a procedure).

Let $\mathcal{N} = (P, T, F)$. The continuous semantics of $\mathcal{N}$ allows to scale transitions by nonnegative coefficients, and for markings to hold nonnegative rational values. More formally, in this context, a *marking* is a vector $\boldsymbol{m} \in \mathbb{Q}_{\geq 0}^P$. Given $\lambda \in \mathbb{Q}_{\geq 0}$ and $t \in T$, we say that λt is *enabled* in $\boldsymbol{m}$ if $\boldsymbol{m} - \lambda^\bullet t \geq 0$. If λt is enabled, then firing it leads to $\boldsymbol{m}' := \boldsymbol{m} - \lambda^\bullet t + \lambda t^\bullet = \boldsymbol{m} + \lambda \Delta(t)$, which is denoted by $\boldsymbol{m} \xrightarrow{\lambda t}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$, or simply $\boldsymbol{m} \rightarrow_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$.

We write $\boldsymbol{m} \rightarrow^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$ if $\boldsymbol{m}'$ can be reached in zero, one or several such steps from $\boldsymbol{m}$.

Deciding continuous coverability (and, in fact, continuous reachability) can be done in polynomial time [12]. Thus, one can test the following in polynomial time, where $p \in P$:

$$\exists \boldsymbol{m}' \in \mathbb{Q}_{\geq 0}^P : \boldsymbol{m} \rightarrow^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}' \wedge \boldsymbol{m}'[p] > 0. \tag{7}$$

As $\boldsymbol{m} \rightarrow^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$ holds iff $k\boldsymbol{m} \rightarrow^* k\boldsymbol{m}'$ holds for some $k > 0$, (7) is equivalent to

$$\exists k > 0, \boldsymbol{m}' \in \mathbb{N}^P : k\boldsymbol{m} \rightarrow^* \boldsymbol{m}' \wedge \boldsymbol{m}'[p] > 0. \tag{8}$$

Hence, to test whether a place p is nonredundant, it suffices to check (8) in polynomial time with $\boldsymbol{m} := \{\mathsf{i} : 1\}$. □

Theorem 5.11. *Generalised soundness is PSPACE-hard.*

Proof. Recall the workflow net $\mathcal{N}'$ constructed in the proof within the main text. We must show that $\mathcal{N}'$ is generalised sound if and only if $\boldsymbol{m} \rightarrow^* \boldsymbol{m}'$ in $\mathcal{N}$. The implication from left to right has already been proven in the main text.

For the converse implication, suppose that $\boldsymbol{m} \rightarrow^* \boldsymbol{m}'$. Fix some k and suppose $\{\mathsf{i} : k\} \rightarrow^* \boldsymbol{v}$. Notice that the transitions are defined in such a way that for every reachable configuration $\boldsymbol{v}$, the invariant $ck = \boldsymbol{v}[\mathsf{i}] \cdot c + \sum_{p \in P \cup \{r\}} \boldsymbol{v}[p] + \boldsymbol{v}[\mathsf{f}] \cdot c$ holds. Thus, by repeatedly firing transitions t_i and t_p, all tokens but those in f can be moved to r, i.e.

$$\boldsymbol{v} \rightarrow^* \{r : (k - \boldsymbol{v}[\mathsf{f}]) \cdot c\} + \{\mathsf{f} : \boldsymbol{v}[\mathsf{f}]\}.$$

From there, to reach $\{\mathsf{f} : k\}$, it suffices to repeat $(k - \boldsymbol{v}[\mathsf{f}])$ times the following: fire $t_{\boldsymbol{m}}$; fire the run that witnesses $\boldsymbol{m} \rightarrow^* \boldsymbol{m}'$; and fire $t_{\boldsymbol{m}'}$. □

Appendix C

Verifying Generalised and Structural Soundness of Workflow Nets via Relaxations

This paper was published as a peer-reviewed conference article. A full version with an appendix containing missing proofs omitted from the conference paper due to space constraints was uploaded to arXiv at the URL https://arxiv.org/abs/2206.02606, see [28].

Cite as: M. Blondin, F. Mazowiecki, and P. Offtermatt. Verifying generalised and structural soundness of workflow nets via relaxations. In S. Shoham and Y. Vizel, editors, *Computer Aided Verification (CAV)*, pages 468–489, Cham, 2022. Springer International Publishing.

Summary In [25], we show that classical soundness, generalised soundness and structural soundness have high theoretical complexity. Even so, there exist tools for deciding soundness, though most focus on classical soundness. In this paper, we propose novel scalable algorithms for semi-deciding structural and generalised soundness, based on reachability relaxations of Petri nets. We further show that on the important case of free-choice workflow nets, the three notions of soundness are equivalent, and our semi-decision algorithm for generalised soundness is complete, thus decides the problem.

Contribution of this author This author was chiefly responsible for the development of the results in the manuscript, notably solely contributing Theorem 2, Proposition 7 and the results in Section 6, and leading joint discussions in which the remaining results were obtained. The author further took on sole responsibility for developing and evaluating the prototype implementation, including choosing and obtaining benchmarks, making decisions on algorithmic design, and the preparation of the software artifact for the conference submission. Additionally, the author took a leading role in the development of the manuscript, making major contributions to the composition of all sections.

Verifying generalised and structural soundness of workflow nets via relaxations

Michael Blondin[1[0000−0003−2914−2734]], Filip Mazowiecki[2[0000−0002−4535−6508]], and Philip Offtermatt[1,2[0000−0001−8477−2849]]

[1] Université de Sherbrooke, Sherbrooke, Canada
[2] Max Planck Institute for Software Systems, Saarbrücken, Germany

Abstract. Workflow nets are a well-established mathematical formalism for the analysis of business processes arising from either modeling tools or process mining. The central decision problems for workflow nets are k-soundness, generalised soundness and structural soundness. Most existing tools focus on k-soundness. In this work, we propose novel scalable semi-procedures for generalised and structural soundness. This is achieved via integral and continuous Petri net reachability relaxations. We show that our approach is competitive against state-of-the-art tools.

1 Introduction

Workflow nets are a well-established mathematical formalism for the description of business processes arising from software modelers and process mining (*e.g.*, see [2,3]), and further notations such as UML activity diagrams [4]. More precisely, a workflow net consists of *places* that contain resources, and *transitions* that can consume, create and move resources concurrently. Two designated places, denoted i and f, respectively model the initialization and completion of a process. Workflow nets, which form a subclass of Petri nets, enable the automatic formal verification of business processes. For example, 1-*soundness* states that from the initial configuration $\{\mathsf{i}\colon 1\}$, every reachable configuration can reach the final configuration $\{\mathsf{f}\colon 1\}$. Informally, this means that given any partial execution of a business process, it is possible to complete it properly.

Soundness. The main decision problems concerning workflow nets revolve around soundness properties. The generalisation of 1-soundness to several resources is k-*soundness*. It asks whether from $\{\mathsf{i}\colon k\}$, every reachable configuration can reach $\{\mathsf{f}\colon k\}$ (here, $\{p\colon k\}$ indicates that place p contains k resources). *Generalised soundness* asks whether k-soundness holds for all $k \geq 1$. Unlike k-soundness, generalised soundness preserves desirable properties like composition [23]. *Structural soundness* is the existential counterpart of generalised soundness, *i.e.* it asks whether k-soundness holds for some $k \geq 1$. These problems are all decidable [1,24,36], but with high complexity: either PSPACE- or EXPSPACE-complete [10]. Most of the (software) tools focus on k-soundness, with an emphasis on $k = 1$. Existing algorithms for generalised and structural soundness

rely on Petri net reachability [24,36,22], which was recently shown Ackermann-complete [28,27,14], so not primitive recursive. In this work, we describe *novel scalable semi-procedures for generalised and structural soundness.*

We focus on "negative instances", *i.e.* where soundness does *not* hold. Let us motivate this. It is known that given a workflow net $\mathcal{N}$, one can iteratively apply simple reduction rules to $\mathcal{N}$. The resulting workflow net $\mathcal{N}'$ is sound iff $\mathcal{N}$ is as well [11,25]. In practice, one infers that $\mathcal{N}$ is sound from the fact that $\mathcal{N}'$ has been reduced to a trivial workflow net where only i and f remain. However, if $\mathcal{N}$ is *not* sound, one obtains some nontrivial $\mathcal{N}'$ that must be verified via some other approach such as model checking. In this work, we provide algorithmic building blocks for this case, where state-space exploration is prohibitive.

Relaxations. This is achieved by considering two reachability relaxations, namely integer reachability and continuous reachability. As their name suggests, these two notions relax some forbidden behaviour of workflow nets. Informally, integer reachability allows for the amount of resources to become temporarily negative, while continuous reachability allows the fragmentation of resources into pieces. Such relaxations possibly introduce spurious behaviour, but enjoy significantly better algorithmic properties (*e.g.*, see [7]). For example, they have been successfully employed for the verification of multi-threaded program skeletons [17,5,8].

Generalised soundness. Based on these relaxations, we provide two necessary conditions for generalised soundness: *integer boundedness* and *continuous soundness*. The former states that the state-space of a given workflow net is bounded (from above) even under integer reachability. The latter states that a given workflow net is 1-sound under continuous reachability. We show the following for integer boundedness and continuous soundness:

- Well-established classical reduction rules preserve both properties;
- Integer boundedness is testable in polynomial time, and continuous soundness is coNP-complete;
- From a practical viewpoint, they are respectively translatable into instances of linear programming and linear arithmetic (which can be solved efficiently by dedicated tools such as SMT solvers);
- Under a mild computational assumption, continuous soundness implies integer boundedness.

Thus, altogether, in order to check whether a workflow net $\mathcal{N}$ is generalised *unsound*, one may first use classical reduction rules to obtain a smaller workflow net $\mathcal{N}'$; test integer *unboundedness* in polynomial time; and, if needed, move onto testing continuous *unsoundness*.

The fact that continuous reachability can be used to semi-decide generalised soundness is arguably surprising. Using the notation of computation temporal logic (CTL), k-soundness can be rephrased as $\{\mathsf{i}\colon k\} \models \forall\mathsf{G}\,\exists\mathsf{F}\,\{\mathsf{f}\colon k\}$. Some other well-studied properties have a similar structure, *e.g.* liveness and home-stateness amount to "$\boldsymbol{m}_{\text{init}} \models \bigwedge_{t\in T} \forall\mathsf{G}\,\exists\mathsf{F}\,(t \text{ is enabled})$" and "$\boldsymbol{m}_{\text{init}} \models \forall\mathsf{G}\,\exists\mathsf{F}\,\boldsymbol{m}_{\text{home}}$". It

is known that liveness, home-stateness, and other properties such as boundedness and inclusion, *cannot* be approximated continuously [9, Sect. 4]. Yet, generalised soundness quantifies k-soundness universally, and this enables a continuous over-approximation. Consequently, we provide a novel application of continuous relaxations for the efficient verification of properties beyond reachability.

Structural soundness. The authors of [36] have observed that a property called structural quasi-soundness is a necessary condition for structural soundness. The former states that $\{\mathsf{i}\colon k\}$ can reach $\{\mathsf{f}\colon k\}$ for some $k \geq 1$. In [36], structural quasi-soundness is reduced to Petri net reachability, which has non primitive recursive complexity. In this work, we show that structural quasi-soundness can be rephrased as continuous reachability. Since the latter can be tested in polynomial time [20], or alternatively via SMT solving [8], this vastly improves the practicability of structural quasi-soundness. We further show that this approach can be adapted so that it provides a lower bound on the first k such that $\{\mathsf{i}\colon k\}$ can reach $\{\mathsf{i}\colon f\}$. From a practical point of view, this is useful as it can vastly reduce the number of reachability queries to decide structural soundness.

Free-choice nets. Many real-world workflow nets have a specific structure where concurrency is restricted. Such nets are known as *free-choice* workflow nets (*e.g.*, see [15] for a book). In particular, free-choice workflow nets allow for the modeling of many features present in common workflow management systems [2]. Generalised soundness is equivalent to 1-soundness for free-choice workflow nets [32]. In this work, we prove that continuous soundness is equivalent to generalised soundness. As a byproduct of our proof, we show that structural soundness is also equivalent to continuous soundness. Altogether, the notions of $\{1\text{-}, \text{generalised}, \text{structural}, \text{continuous}\}$ soundness *all coincide* for free-choice nets. In particular, this means that the continuous relaxation is *exact* and can serve as an efficient addition to the existing algorithmic toolkit.

Experimental results. To demonstrate the viability of our approach, we have implemented and experimentally evaluated a prototype. As part of our evaluation, we propose several new synthetic instances for generalised and structural soundness, which are hard to decide with naive approaches. Some of these instances involve the composition of workflow nets arising from the modeling of business processes in the IBM WebSphere Business Modeler. Our prototype is competitive against both a state-of-the-art Petri net model checker, and a workflow net analyzer. In particular, our approach exhibits better signs of scalability.

Organization. The paper follows the structure of this introduction. Section 2 introduces notation, workflow nets and some properties. Section 3 defines integer and continuous relaxations, and further shows that they are preserved under reduction rules. Sections 4 to 6 present the aforementioned results on generalised soundness, structural soundness and free-choice nets. Section 7 provides experimental results. Section 8 concludes. Some proofs are deferred to an appendix.

2 Preliminaries

We use $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{Q}$ and $\mathbb{Q}_{\geq 0}$ to respectively denote the integers, the naturals (including 0), the rationals and the nonnegative rationals (including 0). Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Q}^S$ be vectors over a finite set S. We write $\boldsymbol{x} \leq \boldsymbol{y}$ if $\boldsymbol{x}[s] \leq \boldsymbol{y}[s]$ for all $s \in S$. We write $\boldsymbol{x} < \boldsymbol{y}$ if $\boldsymbol{x} \leq \boldsymbol{y}$ and $\boldsymbol{x}[s] < \boldsymbol{y}[s]$ for some $s \in S$. We extend addition and subtraction to vectors, *i.e.* $(\boldsymbol{x}+\boldsymbol{y})[s] \coloneqq \boldsymbol{x}[s]+\boldsymbol{y}[s]$ and $(\boldsymbol{x}-\boldsymbol{y})[s] \coloneqq \boldsymbol{x}[s]-\boldsymbol{y}[s]$ for all $s \in S$. We define $\mathrm{supp}(\boldsymbol{x}) = \{s \in S \mid \boldsymbol{x}[s] \neq 0\}$. Given $c \in \mathbb{Q}$, $\boldsymbol{c} \in \mathbb{Q}^S$ denotes the vector such that $\boldsymbol{c}[s] = c$ for all $s \in S$.

2.1 Petri nets

A *Petri net* $\mathcal{N}$ is a triple (P, T, F), where P is a finite set of *places*; T is a finite set of *transitions*, such that $T \cap P = \emptyset$; and $F\colon ((P \times T) \cup (T \times P)) \to \{0,1\}$ is a set of *arcs*. For readers familiar with Petri nets, note that arc weights are not allowed, *i.e.* the weights are always 1. A *marking* is a vector $\boldsymbol{m} \in \mathbb{N}^P$ such that $\boldsymbol{m}[p]$ denotes the number of *tokens* in place p. We denote markings listing nonzero values, *e.g.* $\boldsymbol{m} = \{p_1\colon 1\}$ means $\boldsymbol{m}[p_1] = 1$ and $\boldsymbol{m}[p] = 0$ for $p \neq p_1$.

Let $t \in T$. We define the *pre-vector* of t as ${}^\bullet t \in \mathbb{N}^P$, where ${}^\bullet t[p] \coloneqq F(p,t)$. We define its *post-vector* symmetrically with $t^\bullet[p] \coloneqq F(t,p)$. The *effect* of t is denoted as $\Delta(t) \coloneqq t^\bullet - {}^\bullet t$. We say that a transition t is *enabled* at a marking $\boldsymbol{m}$ if $\boldsymbol{m} \geq {}^\bullet t$. If this is the case, then t can be *fired* at $\boldsymbol{m}$, which results in a marking $\boldsymbol{m}'$ such that $\boldsymbol{m}' \coloneqq \boldsymbol{m} + \Delta(t)$. We write $\boldsymbol{m} \to^t$ to denote that t is *enabled* at $\boldsymbol{m}$, and we write $\boldsymbol{m} \to^t \boldsymbol{m}'$ whenever we care about the marking $\boldsymbol{m}'$ resulting from the firing. We further write $\boldsymbol{m} \to \boldsymbol{m}'$ to denote that $\boldsymbol{m} \to^t \boldsymbol{m}'$ for some $t \in T$.

We say that a sequence of transitions $\pi = t_1 \cdots t_n$ is a *run*. We extend the notion of effect, enabledness and firing from transitions to runs in a straightforward way. The *effect* of a run is defined as the sum of the effects of its transitions, that is, $\Delta(\pi) \coloneqq \Delta(t_1) + \ldots + \Delta(t_n)$. The run π is enabled at $\boldsymbol{m}$, denoted as $\boldsymbol{m} \to^\pi$, if $\boldsymbol{m} \to^{t_1} \boldsymbol{m}_1 \to^{t_2} \boldsymbol{m}_2 \cdots \to^{t_{n-1}} \boldsymbol{m}_{n-1} \to^{t_n}$ for some markings $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_{n-1}$. Furthermore, firing π from $\boldsymbol{m}$ leads to $\boldsymbol{m}'$, denoted as $\boldsymbol{m} \to^\pi \boldsymbol{m}'$, if $\boldsymbol{m} \to^\pi$ and $\boldsymbol{m}' = \boldsymbol{m} + \Delta(\pi)$. We denote the reflexive and transitive closure of $\to$ by $\to^*$.

A pair $(\mathcal{N}, \boldsymbol{m})$, where $\mathcal{N}$ is a Petri net and $\boldsymbol{m}$ is a marking of $\mathcal{N}$, is called a *marked Petri net*. We write $\mathrm{Reach}(\mathcal{N}, \boldsymbol{m}) \coloneqq \{\boldsymbol{m}' \mid \boldsymbol{m} \to^* \boldsymbol{m}'\}$ to denote the set of markings reachable from $\boldsymbol{m}$ in $\mathcal{N}$.

A marked Petri net $(\mathcal{N}, \boldsymbol{m})$ is *bounded* if there exists $b \in \mathbb{N}$ such that $\boldsymbol{m}' \in \mathrm{Reach}(\mathcal{N}, \boldsymbol{m})$ implies $\boldsymbol{m}'[p] \leq b$ for all $p \in P$. It is further *safe* if $b = 1$. We say *unbounded* and *unsafe* for "not bounded" and "not safe".

Sometimes, we argue about transformations on Petri nets which take as an input a Petri net $\mathcal{N}$ and output a Petri net $\mathcal{N}'$. We say that such a transformation *preserves* some property if $\mathcal{N}$ satisfies that property iff $\mathcal{N}'$ satisfies it.

Example 1. The left-hand side of Figure 1 illustrates a Petri net $\mathcal{N}_{\text{left}} = (P, T, F)$ where $P \coloneqq \{\mathsf{i}, p_1, p_2, q_1, q_2, \mathsf{f}\}$, $T \coloneqq \{s, t_1, t_2, u\}$, and F is depicted by arcs, *e.g.* $F[\mathsf{i}, s] = 1$ and $F[s, \mathsf{i}] = 0$. The Petri net is marked by $\{\mathsf{i}\colon 1\}$, *i.e.* with one token in place i. We have $\{\mathsf{i}\colon 1\} \to^s \{p_1\colon 1, p_2\colon 1\} \to^{t_1 t_2} \{q_1\colon 1, q_2\colon 1\} \to^u \{\mathsf{f}\colon 1\}$. $\lhd$

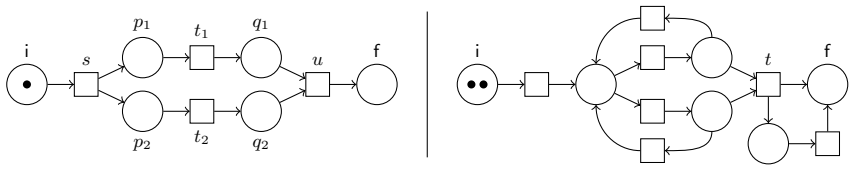


Fig. 1. Example of two Petri nets: respectively $\mathcal{N}_{\text{left}}$ and $\mathcal{N}_{\text{right}}$.

2.2 Workflow nets

A workflow net $\mathcal{N}$ is a Petri net [1] such that:

- there is a designated *initial place* i such that $t^\bullet[\mathsf{i}] = 0$ for all $t \in T$;
- there is a designated *final place* $\mathsf{f} \neq \mathsf{i}$ such that ${}^\bullet t[\mathsf{f}] = 0$ for all $t \in T$; and
- each place and transition lies on at least one path from i to f in the underlying graph of $\mathcal{N}$, *i.e.* (V, E) where $V \coloneqq P \cup T$ and $(u, v) \in E$ iff $F(u, v) \neq 0$.

We say that $\mathcal{N}$ is:

- *k-sound* if for all $\boldsymbol{m} \in \text{Reach}(\mathcal{N}, \{\mathsf{i}\colon k\})$ it is the case that $\boldsymbol{m} \to^* \{\mathsf{f}\colon k\}$ [1];
- *generalised sound* if $\mathcal{N}$ is k-sound for all $k \in \mathbb{N}_{\geq 1}$ [23, Def. 3],
- *structurally sound* if $\mathcal{N}$ is k-sound for some $k \in \mathbb{N}_{\geq 1}$ [6].

Example 2. Figure 1 depicts two workflow nets: $\mathcal{N}_{\text{left}}$ and $\mathcal{N}_{\text{right}}$. The former is generalised sound, but the latter is not. Indeed, from $\{\mathsf{i}\colon 1\}$, transition t cannot be enabled (as transitions preserve the sum of all tokens). Both workflow nets are structurally sound. Indeed, $\mathcal{N}_{\text{right}}$ is 2-sound as it is always possible to redistribute the two tokens so that t can be fired in order to reach $\{\mathsf{f}\colon 2\}$. $\lhd$

3 Reachability relaxations

Fix a Petri net $\mathcal{N} = (P, T, F)$. We describe the two aforementioned relaxations.

Integer reachability. An *integral marking* is a vector $\boldsymbol{m} \in \mathbb{Z}^P$. Any transition $t \in T$ is *enabled* in $\boldsymbol{m} \in \mathbb{Z}^P$, and *firing* t leads to $\boldsymbol{m}' \coloneqq \boldsymbol{m} + \Delta(t)$, denoted $\boldsymbol{m} \to^t_{\mathbb{Z}} \boldsymbol{m}'$. We define $\boldsymbol{m} \to_{\mathbb{Z}} \boldsymbol{m}'$ and $\boldsymbol{m} \to^*_{\mathbb{Z}} \boldsymbol{m}'$ analogously to the standard setting but w.r.t. $\to^t_{\mathbb{Z}}$ rather than $\to^t$. Similarly, $\mathbb{Z}$-Reach$(\mathcal{N}, \boldsymbol{m}) \coloneqq \{\boldsymbol{m}' \in \mathbb{Z}^P \mid \boldsymbol{m} \to^*_{\mathbb{Z}} \boldsymbol{m}'\}$. As transitions are always enabled, the order of a firing sequence is irrelevant. In particular, $\boldsymbol{m} \to^*_{\mathbb{Z}} \boldsymbol{m}'$ iff there exists $\boldsymbol{x} \in \mathbb{N}^T$ such that $\boldsymbol{m}' = \boldsymbol{m} + \sum_{t \in T} \boldsymbol{x}[t] \cdot \Delta(t)$. Thus, integer reachability amounts to integer linear programming. Moreover, it is NP-complete [21,13].

Continuous reachability. A *continuous marking* is a vector $\boldsymbol{m} \in \mathbb{Q}^P_{\geq 0}$. Let $\lambda \in (0, 1]$. We say that λt is *enabled* in $\boldsymbol{m}$, denoted $\boldsymbol{m} \to^{\lambda t}_{\mathbb{Q}_{\geq 0}}$, if $\boldsymbol{m} \geq \lambda \cdot {}^\bullet t$. In this context, λ is called the *scaling factor*. Furthermore, we denote by $\boldsymbol{m} \to^{\lambda t}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$

that λt is enabled in $\boldsymbol{m}$, and that its *firing* results in $\boldsymbol{m}' \coloneqq \boldsymbol{m} + \lambda \cdot \Delta(t)$. A sequence of pairs of scaling factors and transitions is called a *continuous run*.

The notations $\boldsymbol{m} \to_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$ and $\boldsymbol{m} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$ are defined analogously to the discrete case but with respect to $\to^{\lambda t}_{\mathbb{Q}_{\geq 0}}$ rather than $\to^t$ (the internal factors λ can differ). Similarly, $\mathbb{Q}_{\geq 0}$-Reach$(\mathcal{N}, \boldsymbol{m}) \coloneqq \{\boldsymbol{m}' \mid \boldsymbol{m} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'\}$ denotes the markings continuously reachable from $\boldsymbol{m}$. For example, for $\mathcal{N}_{\text{left}}$ from Figure 1 and $\pi \coloneqq \frac{1}{2} s \frac{1}{4} t_1$, we have $\{\mathsf{i}\colon 1\} \to^{\pi}_{\mathbb{Q}_{\geq 0}} \{\mathsf{i}\colon 1/2, p_1\colon 1/4, p_2\colon 1/2, q_1\colon 1/4\}$. It is known that continuous reachability, namely determining whether $\boldsymbol{m} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$, given $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{Q}^P_{\geq 0}$, can be checked in polynomial time [20].

Let us establish the following helpful lemma similar to [20, Lemma 12(1)].

Lemma 1. *Let $\boldsymbol{m}$, $\boldsymbol{m}'$ be continuous markings. It is the case that $\boldsymbol{m} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$ iff there exists $b \in \mathbb{N}_{\geq 1}$ such that $b \cdot \boldsymbol{m} \to^* b \cdot \boldsymbol{m}'$.*

3.1 Preservation under reduction rules

In [11], the authors present six reduction rules, denoted $R_1, \ldots, R_6$, that generalize the existing reduction rules of [31]. In the following, we show that these reduction rules preserve natural properties for the two reachability relaxations. This means we will be able to check these properties on a reduced workflow net and get the same results as on the original one.

Formally, the rules simplify a given workflow net $\mathcal{N} = (P, T, F)$. In particular, the places of the resulting workflow net $\mathcal{N}' = (P', T, F')$ form a subset of P. Let us fix a domain $\mathbb{D} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}_{\geq 0}\}$ and let $P' \subseteq P$. For ease of notation, we we write $P'' = P \setminus P'$ to denote the (possibly empty) set of removed places. Rules never remove the initial and output places, *i.e.* $\mathsf{i}, \mathsf{f} \in P'$. We denote by $\pi\colon \mathbb{D}^P \to \mathbb{D}^{P'}$ the obvious projection function, and by $\pi_0\colon \mathbb{D}^{P'} \to \mathbb{D}^P$ the "reverse projection" which fills new places with 0. Formally, $\pi_0(\boldsymbol{m})[p'] \coloneqq \boldsymbol{m}[p']$ for all $p' \in P'$ and $\pi_0(\boldsymbol{m})[p''] \coloneqq 0$ for all $p'' \in P''$.

In [11], the authors prove that the rules preserve generalised soundness. This of course implies that they preserve k-soundness for all k. The technical proposition below will be helpful in the forthcoming sections to show the preservation of useful properties based on reachability relaxations.

Proposition 1. *Let $\mathcal{N} = (P, T, F)$ be a workflow net, and let $\mathbb{D} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}_{\geq 0}\}$. Let $\mathcal{N}' = (P', T', F')$ be a workflow net obtained by applying a reduction rule R_i to $\mathcal{N}$, where $P = P' \cup P''$. The following holds.*

- Rule R_1. *We have $P'' = \{p\}$. There exists a nonempty set $R' \subseteq P'$ such that if $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{D}} \boldsymbol{m}$ in $\mathcal{N}$, then $\boldsymbol{m}[p] = \sum_{r \in R'} \boldsymbol{m}[r']$. Moreover, $\boldsymbol{m} \to^*_{\mathbb{D}} \boldsymbol{n}$ in $\mathcal{N}$ iff $\pi(\boldsymbol{m}) \to^*_{\mathbb{D}} \pi(\boldsymbol{n})$ in $\mathcal{N}'$.*
- Rules R_2 and R_3. *We have $P'' = \emptyset$ and $\boldsymbol{m} \to^*_{\mathbb{D}} \boldsymbol{n}$ in $\mathcal{N}$ iff $\boldsymbol{m} \to^*_{\mathbb{D}} \boldsymbol{n}$ in $\mathcal{N}'$.*
- Rules R_4 and R_5. *We have $P'' = \{p\}$. For all $\boldsymbol{m}'$ and $\boldsymbol{n}'$, $\boldsymbol{m}' \to^*_{\mathbb{D}} \boldsymbol{n}'$ in $\mathcal{N}'$ iff $\pi_0(\boldsymbol{m}') \to^*_{\mathbb{D}} \pi_0(\boldsymbol{n}')$ in $\mathcal{N}$. Further, for all $t \in T$ and $p' \in P'$: either ${}^\bullet t[p] = 1$ implies ${}^\bullet t[p'] = 0$; or $t^\bullet[p] = 1$ implies $t^\bullet[p'] = 0$. Also, for $\mathbb{D} \neq \mathbb{Z}$, if $\exists \boldsymbol{m} : \{\mathsf{i}\colon 1\} \to^*_{\mathbb{D}} \boldsymbol{m} \not\to^*_{\mathbb{D}} \{\mathsf{f}\colon 1\}$ holds in $\mathcal{N}$, then $\exists \boldsymbol{m}' : \{\mathsf{i}\colon 1\} \to^*_{\mathbb{D}} \boldsymbol{m}' \not\to^*_{\mathbb{D}} \{\mathsf{f}\colon 1\}$ holds in $\mathcal{N}'$.*

– Rule R_6. *We have* $P'' = \{p_2, \ldots, p_k\}$. *There exists* $p_1 \in P'$ *such that for all* $\boldsymbol{n} \in P^{\mathbb{D}}$, *if* $\sum_{i=1}^{k} \boldsymbol{m}[p_i] = \sum_{i=1}^{k} \boldsymbol{n}[p_i]$ *and* $\boldsymbol{n}[p'] = \boldsymbol{m}[p']$ *for* $p' \in P' \setminus \{p_1\}$, *then* $\boldsymbol{m} \to_{\mathbb{D}}^{*} \boldsymbol{n}$. *Moreover, if* $\boldsymbol{m}[p_i] = \boldsymbol{n}[p_i] = 0$ *for* $i > 1$, *then* $\boldsymbol{m} \to_{\mathbb{D}}^{*} \boldsymbol{n}$ *in* $\mathcal{N}$ *iff* $\pi(\boldsymbol{m}) \to_{\mathbb{D}}^{*} \pi(\boldsymbol{n})$ *in* $\mathcal{N}'$.

4 Using relaxations for generalised soundness

In this section, we explain how reachability relaxations can be leveraged in order to semi-decide generalised soundness of workflow nets. More precisely, we state two necessary conditions for a workflow net to be generalised sound: one phrased in terms of integer reachability, and one in terms of continuous reachability. Furthermore, for each condition we: (1) show that it is preserved under reduction rules, and (2) establish its computational complexity. Overall, this means that to conclude that a given workflow net $\mathcal{N}$ is *not* generalised sound, one may first reduce $\mathcal{N}$, and *then* efficiently test for one of these two necessary conditions.

For integer boundedness, we need the mild assumption of nonredundancy. Let $\mathcal{N} = (P, T, F)$ be a workflow net. We say that a place $p \in P$ is *nonredundant*[3] if there exist $k \in \mathbb{N}_{\geq 1}$ and $\boldsymbol{m} \in \mathbb{N}^P$ such that $\{\mathsf{i}\colon k\} \to^{*} \boldsymbol{m}$ and $\boldsymbol{m}[p] \geq 1$. It is known (and simple to see) that redundant places can be removed from a workflow net without changing whether it is generalised sound. Moreover, testing whether a place is nonredundant can be done in polynomial time. Indeed, by Lemma 1, it amounts to testing for the existence of some $\boldsymbol{m} \in \mathbb{Q}_{\geq 0}^{P}$ such that $\{\mathsf{i}\colon 1\} \to_{\mathbb{Q}_{\geq 0}}^{*} \boldsymbol{m}$ and $\boldsymbol{m}[p] > 0$. The latter is known as a *coverability* query and it can be checked in polynomial time [20]. Thus, in order to test whether a given workflow net is generalised sound, one can first remove its redundant places. We call a workflow net without redundant places a *nonredundant workflow net*.

4.1 Integer unboundedness

Recall that a marked Petri net $(\mathcal{N}, \boldsymbol{m})$ is *bounded* if there exists $b \in \mathbb{N}$ such that $\boldsymbol{m}' \in \mathrm{Reach}(\mathcal{N}, \boldsymbol{m})$ implies $\boldsymbol{m}' \leq \boldsymbol{b}$. It is well-known that any 1-sound workflow net must be bounded from $\{\mathsf{i}\colon 1\}$ [1]. In particular, this means that boundedness is a necessary condition for generalised soundness. However, testing boundedness has extensive computational cost as it is EXPSPACE-complete [12,33]. Consider the relaxed property of *integer boundedness*. It is defined as boundedness, but where "$\boldsymbol{m}' \in \mathrm{Reach}(\mathcal{N}, \boldsymbol{m})$" is replaced with "$\boldsymbol{m}' \in \mathbb{Z}\text{-}\mathrm{Reach}(\mathcal{N}, \boldsymbol{m}) \cap \mathbb{N}^P$".

Proposition 2 ([10, Lemma 5.9]). *Let* $\mathcal{N}$ *be a nonredundant workflow net. If* $(\mathcal{N}, \{\mathsf{i}\colon 1\})$ *is integer unbounded, then* $\mathcal{N}$ *is not generalised sound.*

Proposition 3. *The reduction rules from [11] preserve integer unboundedness.*

[3] This notion is adapted from batch workflow nets considered in [24].

Next, we establish the complexity of integer unboundedness in two steps. The first step, in the next proposition, shows that testing integer boundedness amounts to a simple condition, independent of the initial marking. The second step shows the condition can be translated into a linear program over $\mathbb{Q}$, rather than $\mathbb{N}$. As a corollary, integer unboundedness is testable in polynomial time.

Proposition 4. *A marked Petri net $(\mathcal{N}, \boldsymbol{m})$ is integer unbounded iff there exists a marking $\boldsymbol{m}' > \mathbf{0}$ such that $\mathbf{0} \to_{\mathbb{Z}}^{*} \boldsymbol{m}'$ (independent of $\boldsymbol{m}$).*

Proof. Let $\mathcal{N} = (P, F, T)$ be a Petri net and let $\boldsymbol{m} \in \mathbb{N}^P$.

$\Rightarrow$) By assumption, there exist $\boldsymbol{m}_0, \boldsymbol{m}_1, \ldots \in \mathbb{Z}\text{-Reach}(\mathcal{N}, \boldsymbol{m}) \cap \mathbb{N}^P$ such that, for every $i \in \mathbb{N}$, it is the case that $m_i \not\leq \boldsymbol{i}$. Since $(\mathbb{N}^P, \leq)$ is well-quasi-ordered, there exist indices $i_0, i_1, \ldots$ such that $\boldsymbol{m}_{i_j} \leq \boldsymbol{m}_{i_k}$ for all $j < k$. Without loss of generality, we can assume that $\boldsymbol{m}_{i_j} < \boldsymbol{m}_{i_k}$ for all $j < k$, as we could otherwise extract such a subsequence. Recall that each $\boldsymbol{m}_{i_\ell} \in \mathbb{Z}\text{-Reach}(\mathcal{N}, \boldsymbol{m})$. Let $\pi_\ell \in T^*$ be such that $\boldsymbol{m} \to_{\mathbb{Z}}^{\pi_\ell} \boldsymbol{m}_{i_\ell}$. Let $\boldsymbol{x}_\ell \in \mathbb{N}^T$ be the vector such that $\boldsymbol{x}_\ell(t)$ indicates the number of occurrences of transition t in π_ℓ. Since $(\mathbb{N}^T, \leq)$ is well-quasi-ordered, there exist $j < k$ such that $\boldsymbol{x}_j \leq \boldsymbol{x}_k$. Let $\boldsymbol{m}' \coloneqq \boldsymbol{m}_{i_k} - \boldsymbol{m}_{i_j}$ and $\pi \coloneqq \prod_{t \in T} t^{(\boldsymbol{x}_k[t] - \boldsymbol{x}_\ell[t])}$. We have $\mathbf{0} \to_{\mathbb{Z}}^{\pi} \boldsymbol{m}' > \mathbf{0}$ as desired since:

$$\begin{aligned}\boldsymbol{m}' = \boldsymbol{m}_{i_k} - \boldsymbol{m}_{i_j} &= (\boldsymbol{m} + \Delta(\pi_k)) - (\boldsymbol{m} + \Delta(\pi_\ell)) = \Delta(\pi_k) - \Delta(\pi_\ell) \\ &= \sum_{t \in T} \boldsymbol{x}_k[t] \cdot \Delta(t) - \sum_{t \in T} \boldsymbol{x}_\ell[t] \cdot \Delta(t) = \sum_{t \in T} (\boldsymbol{x}_k - \boldsymbol{x}_\ell)[t] \cdot \Delta(t) = \Delta(\pi).\end{aligned}$$

$\Leftarrow$) By assumption $\mathbf{0} \to_{\mathbb{Z}}^{\pi} \boldsymbol{m}' > \mathbf{0}$. In particular, this means that $\boldsymbol{m} \to_{\mathbb{Z}}^{\pi} \boldsymbol{m} + \boldsymbol{m}' \to_{\mathbb{Z}}^{\pi} \boldsymbol{m} + 2\boldsymbol{m}' \to_{\mathbb{Z}} \cdots$. Therefore, $(\mathcal{N}, \boldsymbol{m})$ is not integer bounded. □

Proposition 5. *A marked Petri net $(\mathcal{N}, \boldsymbol{m})$, where $\mathcal{N} = (P, T, F)$, is integer unbounded iff this system has a solution: $\exists \boldsymbol{x} \in \mathbb{Q}_{\geq 0}^T : \sum_{t \in T} \boldsymbol{x}[t] \cdot \Delta(t) > \mathbf{0}$. In particular, given a workflow net $\mathcal{N}$, testing integer boundedness of $(\mathcal{N}, \{\mathsf{i}\colon 1\})$ can be done in polynomial time.*

4.2 Continuous soundness

Let us now introduce a continuous variant of 1-soundness based on continuous reachability. We prove that this variant, which we call *continuous soundness*, is a necessary condition for generalised soundness, and preserved by reduction rules. Moreover, we show that continuous soundness is coNP-complete, and relates to integer boundedness.

We say that a workflow net $\mathcal{N}$ is *continuously sound* if for all continuous markings $\boldsymbol{m} \in \mathbb{Q}_{\geq 0}\text{-Reach}(\mathcal{N}, \{\mathsf{i}\colon 1\})$ it is the case that $\boldsymbol{m} \to_{\mathbb{Q}_{\geq 0}}^{*} \{\mathsf{f}\colon 1\}$.

Theorem 1. *Continuous unsoundness implies generalised unsoundness.*

Proof. Let $\mathcal{N} = (P, T, F)$ be a workflow net that is not continuously sound. By definition of continuous soundness, there exists some continuous marking $\boldsymbol{m} \in \mathbb{Q}^P_{\geq 0}$ such that $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}$ and $\boldsymbol{m} \not\to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$. By Lemma 1, there exists $b \in \mathbb{N}_{\geq 1}$ such that $\{\mathsf{i}\colon b\} \to^* b \cdot \boldsymbol{m}$. Furthermore, by Lemma 1, $b \cdot \boldsymbol{m} \not\to^* \{\mathsf{f}\colon b\}$. This means that $\mathcal{N}$ is not b-sound, and consequently not generalised sound. □

Proposition 6. *The reduction rules from [11] preserve continuous soundness.*

Theorem 2. *Continuous soundness is coNP-complete. Moreover, coNP-hardness holds even if the underlying graph of the given workflow net is acyclic.*

Proof (of membership in coNP). The *inclusion problem* consists in determining whether, given Petri nets $\mathcal{N}$ and $\mathcal{N}'$ over a common set of places, and markings $\boldsymbol{m}$ and $\boldsymbol{m}'$, it is the case that $\mathbb{Q}_{\geq 0}\text{-Reach}(\mathcal{N}, \boldsymbol{m}) \subseteq \mathbb{Q}_{\geq 0}\text{-Reach}(\mathcal{N}', \boldsymbol{m}')$. The inclusion problem is known to be coNP-complete [8, Prop. 4.6].

Let $\mathcal{N} = (P, T)$ be a workflow net. Let $\mathcal{N}^{-1} = (P, T^{-1})$ be defined as $\mathcal{N}$ but with its transitions reversed, *i.e.* where $T^{-1} \coloneqq \{t^{-1} \mid t \in T\}$ with ${}^\bullet(t^{-1}) \coloneqq t^\bullet$ and $(t^{-1})^\bullet \coloneqq {}^\bullet t$. It is the case that $\boldsymbol{m} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$ in $\mathcal{N}$ iff $\boldsymbol{m}' \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}$ in $\mathcal{N}^{-1}$. Observe that $\mathcal{N}$ is continuously sound iff the following holds for all $\boldsymbol{m}$:

$$\boldsymbol{m} \in \mathbb{Q}_{\geq 0}\text{-Reach}(\mathcal{N}, \{\mathsf{i}\colon 1\}) \implies \{\mathsf{f}\colon 1\} \in \mathbb{Q}_{\geq 0}\text{-Reach}(\mathcal{N}, \boldsymbol{m}).$$

So, as $\{\mathsf{f}\colon 1\} \in \mathbb{Q}_{\geq 0}\text{-Reach}(\mathcal{N}, \boldsymbol{m})$ is equivalent to $\boldsymbol{m} \in \mathbb{Q}_{\geq 0}\text{-Reach}(\mathcal{N}^{-1}, \{\mathsf{f}\colon 1\})$, continuous soundness holds iff $\mathbb{Q}_{\geq 0}\text{-Reach}(\mathcal{N}, \{\mathsf{i}\colon 1\}) \subseteq \mathbb{Q}_{\geq 0}\text{-Reach}(\mathcal{N}^{-1}, \{\mathsf{f}\colon 1\})$. As inclusion can be tested in coNP, membership follows. □

Proof (of coNP-hardness). We give a reduction from the problem of determining whether a Boolean formula in disjunctive normal form (DNF) is a tautology. We adapt a construction from [35] used to show that soundness in acyclic workflow nets is coNP-hard. The proof is more challenging under the continuous semantics as several variable valuations and clauses can be simultaneously used.

The reduction is depicted in Figure 2 for $\varphi = (x_1 \wedge x_2 \wedge \neg x_4) \vee (\neg x_1 \wedge x_3 \wedge x_4)$. In general, let $\varphi = \bigvee_{j \in [1..k]} C_j$ be a Boolean formula in DNF with k clauses over variables $x_1, \ldots, x_m$. We define a workflow net $\mathcal{N}_\varphi = (P, T, F)$.

Definition. The places are defined as $P \coloneqq \{\mathsf{i}, p_{\text{cl}}, \mathsf{f}\} \cup P_{\text{var}} \cup P_{\text{clean}}$, where $P_{\text{var}} \coloneqq \bigcup_{i \in [1..m]} \{p_{i,?}, p_{i,1}, p_{i,0}\}$ and $P_{\text{clean}} \coloneqq \bigcup_{i \in [1..m]} \{q_i, r_i\}$. The transitions are defined as $T \coloneqq \{t_{\text{init}}, t_{\text{fin}}\} \cup T_{\text{var}} \cup T_{\text{clauses}} \cup T_{\overline{\text{var}}}$, where

$$T_{\text{var}} \coloneqq \bigcup_{i \in [1..m]} \{v_{i,1}, v_{i,0}\}, T_{\text{clauses}} \coloneqq \{c_i \mid i \in [1..k]\} \text{ and } T_{\overline{\text{var}}} \coloneqq \bigcup_{i \in [1..m]} \{\overline{v}_{i,1}, \overline{v}_{i,0}\}.$$

Let us explain how $\mathcal{N}_\varphi$ is *intended* to work. Transition t_{init} enables the initialization of variables and the selection of a clause that satisfies φ, i.e. ${}^\bullet t_{\text{init}} \coloneqq \{\mathsf{i}\colon 1\}$ and $t^\bullet_{\text{init}} \coloneqq \{p_{i,?}\colon 1 \mid i \in [1..m]\} + \{p_{\text{cl}}\colon 1\}$. A token in place $p_{i,b}$ indicates that variable x_i has been assigned value b (where "?" indicates "none"). Consequently, we have ${}^\bullet v_{i,b} \coloneqq p_{i,?}$ and $v^\bullet_{i,b} \coloneqq p_{i,b}$ for each $i \in [1..m]$ and $b \in \{0, 1\}$.

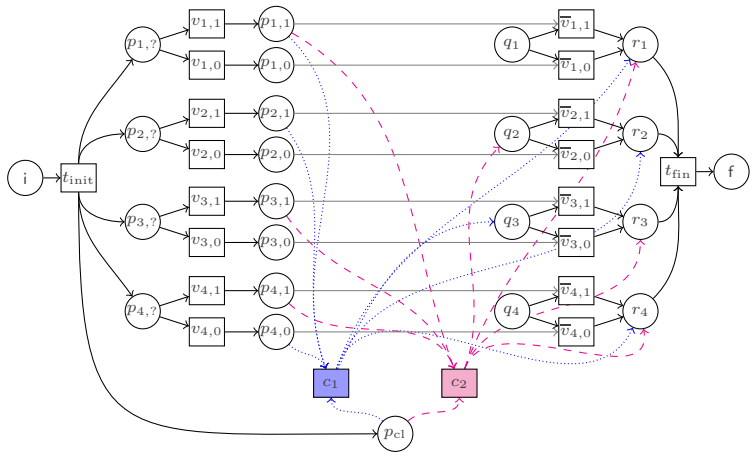


Fig. 2. A workflow net $\mathcal{N}_\varphi$ such that $\mathcal{N}_\varphi$ is continuously sound iff $\varphi = (x_1 \wedge x_2 \wedge \neg x_4) \vee (x_1 \wedge x_3 \wedge x_4)$ is a tautology. Places and transitions contain their names (not values). Arcs corresponding to the first and second clauses are respectively dotted and dashed.

Transition c_j consumes a token associated to each literal of clause C_j, *i.e.* ${}^\bullet c_j \coloneqq \{v_{i,1} \mid x_i \in C_j\} + \{v_{i,0} \mid \neg x_i \in C_j\}$. A token in place q_i indicates that variable x_i is not needed anymore (due to some satisfied clause). A token in place r_i indicates that variable x_i has been discarded. Therefore, transition c_j produces these tokens: $c_j^\bullet \coloneqq \{q_i \mid x_i \notin C_j \wedge \neg x_i \notin C_j\} + \{r_i \mid x_i \in C_j \vee \neg x_i \in C_j\}$.

Transition $\overline{v}_{i,b}$ discards variable x_i, i.e. ${}^\bullet\overline{v}_{i,b} \coloneqq \{p_{i,b}, q_i\}$ and ${}^\bullet\overline{v}_{i,b} \coloneqq \{q_i\}$. Once each variable is discarded, transition t_{fin} terminates the execution, i.e. ${}^\bullet t_{\text{fin}} \coloneqq \{r_i \mid i \in [1..m]\}$ and $t_{\text{fin}}^\bullet \coloneqq \{\mathsf{f}\colon 1\}$.

Correctness. Note that under $\rightarrow^*_{\mathbb{Q}_{\geq 0}}$, the workflow net needs not to proceed as described. Indeed, it could, e.g., assign half a token to $p_{i,0}$ and half a token to $p_{i,1}$. Similarly, several clauses can be used, with distinct scaling factors. Nonetheless, $\mathcal{N}_\varphi$ is continuously sound iff φ is a tautology.

$\Rightarrow$) Let $b_1, \ldots, b_m \in \{0, 1\}$. Let $\pi \coloneqq t_{\text{init}} v_{1,b_1} \cdots v_{m,b_m}$. We have: $\{\mathsf{i}\colon 1\} \rightarrow^\pi_{\mathbb{Q}_{\geq 0}} \{v_{i,b_i}\colon 1 \mid i \in [1..m]\} + \{p_{\text{cl}}\colon 1\}$. Since $\mathcal{N}_\varphi$ is continuously sound by assumption, there must exists some $j \in [1..k]$ such that c_j is enabled. This implies that clause C_j is satisfied by the assignment. Hence, φ is a tautology.

$\Leftarrow$) The proof is technical and involves several invariants (see appendix). $\square$

We may now prove that any nonredundant workflow net that is integer unbounded is also continuously unsound (the reverse is not necessarily true). Therefore, integer unboundedness relates to continuous soundness much like continuous unsoundness relates to generalised soundness.

Proposition 7. *Let $\mathcal{N}$ be a nonredundant workflow net and $\boldsymbol{m} \in \mathbb{N}^P$. If $(\mathcal{N}, \boldsymbol{m})$ is integer unbounded, then $\mathcal{N}$ is not continuously sound.*

Proof. Let $\mathcal{N} = (P, T, F)$ and $\boldsymbol{m} \in \mathbb{N}^P$ be such that $(\mathcal{N}, \boldsymbol{m})$ is not integer bounded. By Proposition 4, there exists $\boldsymbol{m}' > \boldsymbol{0}$ such that $\boldsymbol{0} \to^*_{\mathbb{Z}} \boldsymbol{m}'$. By nonredundancy, there exist $\lambda \in \mathbb{N}_{\geq 1}$ and $\boldsymbol{m}'' \in \mathbb{N}^P$ such that $\{\mathsf{i}\colon \lambda\} \to^* \{\mathsf{f}\colon 1\} + \boldsymbol{m}''$.

In [24, Lemma 12], it is shown that $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Z}} \boldsymbol{n}$ implies the existence of some $\ell \in \mathbb{N}$ such that $\{\mathsf{i}\colon k + \ell\} \to^* \{\mathsf{f}\colon \ell\} + \boldsymbol{n}$. By invoking this lemma with $k \coloneqq 0$ and $\boldsymbol{n} \coloneqq \boldsymbol{m}'$, we obtain $\{\mathsf{i}\colon \ell\} \to^* \{\mathsf{f}\colon \ell\} + \boldsymbol{m}'$ for some $\ell \in \mathbb{N}$.

Altogether, $\{\mathsf{i}\colon \lambda + \ell\} \to^* \{\mathsf{f}\colon \lambda + \ell\} + \boldsymbol{m}' + \boldsymbol{m}''$. Since $\lambda + \ell \geq 1$, Lemma 1 yields $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\} + \boldsymbol{m}'''$ where $\boldsymbol{m}''' \coloneqq (1/(\lambda+\ell))\boldsymbol{m}'$. As every transition of a workflow net produces at least one token, this contradicts the fact that $\mathcal{N}$ is continuously sound. Indeed, it is impossible to fully get rid of $\boldsymbol{m}''' > \boldsymbol{0}$. □

5 Using relaxations for structural soundness

A workflow net $\mathcal{N}$ is *k-quasi-sound* if $\{\mathsf{i}\colon k\} \to^* \{\mathsf{f}\colon k\}$. Furthermore, $\mathcal{N}$ is *structurally quasi-sound* if it is k-quasi-sound for some $k \in \mathbb{N}_{\geq 1}$.

As observed in [36], structural quasi-soundness is a necessary condition for structural soundness. The notion of structural quasi-soundness is naturally generalised to an arbitrary Petri net $\mathcal{N} = (P, T, F)$. Given markings $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{N}^P$, we say that $\boldsymbol{m}$ *structurally reaches* $\boldsymbol{m}'$ in $\mathcal{N}$ if $k \cdot \boldsymbol{m} \to^* k \cdot \boldsymbol{m}'$ for some $k \in \mathbb{N}_{\geq 1}$. A workflow net is structurally quasi-sound iff $\boldsymbol{m} \coloneqq \{\mathsf{i}\colon 1\}$ structurally reaches $\boldsymbol{m}' \coloneqq \{\mathsf{f}\colon 1\}$. So, the observation of [36] can be rephrased as follows.

Proposition 8. *Let $\mathcal{N}$ be a workflow net. If $\{\mathsf{i}\colon 1\}$ does not structurally reach $\{\mathsf{f}\colon 1\}$ in $\mathcal{N}$, then $\mathcal{N}$ is not structurally sound.*

The problem of structural quasi-soundness can be reduced to an instance of the Petri net reachability problem [36, Lemma 2.1]. Intuitively, the reduction produces a Petri net that nondeterministically chooses multiples of $\{\mathsf{i}\colon 1\}$ and $\{\mathsf{f}\colon 1\}$ for which to check reachability. Such an approach has a prohibitive computational cost as Petri net reachability is Ackermann-complete. However, we observe that structural reachability, and hence structural quasi-soundness, is equivalent to continuous reachability by Lemma 1.

Proposition 9. *Let $\mathcal{N} = (P, T, F)$ be a Petri net, and let $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{N}^P$ be markings. It is the case that $\boldsymbol{m}$ structurally reaches $\boldsymbol{m}'$ iff $\boldsymbol{m} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$.*

For a workflow net $\mathcal{N} = (P, T, F)$, let $k_{\mathcal{N}} \in \mathbb{N}_{\geq 1} \cup \{\infty\}$ be the smallest number for which $\mathcal{N}$ is $k_{\mathcal{N}}$-quasi-sound. Then $\mathcal{N}$ is structurally sound iff $k_{\mathcal{N}} \neq \infty$ and $\mathcal{N}$ is $k_{\mathcal{N}}$-sound [36, Thm 2.1]. By Proposition 9, $k_{\mathcal{N}} \neq \infty$ can be checked in polynomial time via a continuous reachability query. Moreover, a lower bound on $k_{\mathcal{N}}$ can be obtained by computing $k_{\mathcal{N},\mathbb{Z}} \in \mathbb{N}_{\geq 1} \cup \{\infty\}$, defined as the smallest value such that $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Z}} \{\mathsf{f}\colon k\}$. We obtain a better bound by defining $k_{\mathcal{N},\mathbb{Q}_{\geq 0}} \in \mathbb{N}_{\geq 1} \cup \{\infty\}$ as the smallest value for which there is a continuous run $\pi = \lambda_1 t_1 \cdots \lambda_n t_n$ such that $\{\mathsf{i}\colon k\} \to^{\pi}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon k\}$ and $\boldsymbol{\pi} \in \mathbb{N}^T$, where $\boldsymbol{\pi}[t] \coloneqq \sum_{i \in [1..n]: t_i = t} \lambda_i$. Values $k_{\mathcal{N},\mathbb{Z}}$ and $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$ can respectively be computed by a translation to integer linear programming, and a decidable optimization modulo theory.

Proposition 10. *Let $\mathcal{N}$ be a workflow net. It is the case that $k_{\mathcal{N},\mathbb{Z}} \leq k_{\mathcal{N},\mathbb{Q}_{\geq 0}} \leq k_{\mathcal{N}}$. Moreover, $k_{\mathcal{N},\mathbb{Z}}$ can be computed from an integer linear program $\mathcal{P}$; $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$ can be obtained by computing $\min k \in \mathbb{N}_{\geq 1} : \varphi(k)$ where φ is a formula from the existential fragment of mixed linear arithmetic φ, i.e. $\exists\mathsf{FO}(\mathbb{Q}, \mathbb{Z}, <, +)$; and both $\mathcal{P}$ and φ are constructible in polynomial time from $\mathcal{N}$.*

6 Free-choice workflow nets

Let $\mathcal{N} = (P, T, F)$ be a Petri net. We say that $\mathcal{N}$ is *free-choice* if for any $s, t \in T$, it is the case that either $\mathrm{supp}({}^\bullet s) \cap \mathrm{supp}({}^\bullet t) = \emptyset$ or ${}^\bullet s = {}^\bullet t$. For example, the nets $\mathcal{N}_{\text{left}}$ and $\mathcal{N}_{\text{right}}$ from Figure 1 are respectively free-choice and not free-choice.

It is known that generalised soundness is equivalent to 1-soundness in free-choice workflow nets [32]. We will show that the same holds for structural soundness, and that, surprisingly, for continuous soundness as well. This means that notions of soundness collapse for free-choice nets. This is proven in the forthcoming Lemma 2 and Theorem 3, which form one of the main theoretical contributions of this work.

Let $(\mathcal{N}, \boldsymbol{m})$ be a marked Petri net. We say that a transition t is *quasi-live* in $(\mathcal{N}, \boldsymbol{m})$ if there exists $\boldsymbol{m}'$ such that $\boldsymbol{m} \to^* \boldsymbol{m}' \to^t$. Similarly, we say that a transition t is *live* in $(\mathcal{N}, \boldsymbol{m})$ if for all $\boldsymbol{m}'$ such that $\boldsymbol{m} \to^* \boldsymbol{m}'$, t is quasi-live in $(\mathcal{N}, \boldsymbol{m}')$. In words, quasi-liveness states that there is at least one way to enable t, and liveness states that t can always be re-enabled. The set of *quasi-live* and *live* transitions of $(\mathcal{N}, \boldsymbol{m})$ are defined respectively as $F(m) \coloneqq \{t \in T \mid t \text{ is quasi-live in } (\mathcal{N}, \boldsymbol{m})\}$ and $L(m) \coloneqq \{t \in T \mid t \text{ is live in } (\mathcal{N}, \boldsymbol{m})\}$.

Lemma 2. *Let $\mathcal{N} = (P, T, F)$ be a free-choice Petri net, let $c \in \mathbb{N}_{\geq 1}$, and let $\boldsymbol{m} \in \mathbb{N}^P$. The following statements hold.*

1. *There exists a marking $\boldsymbol{m}'$ such that $\boldsymbol{m} \to^* \boldsymbol{m}'$ and $L(\boldsymbol{m}') = F(\boldsymbol{m}')$.*
2. *If $L(\boldsymbol{m}) = F(\boldsymbol{m})$, then $L(c \cdot \boldsymbol{m}) = F(c \cdot \boldsymbol{m}) = F(\boldsymbol{m})$.*
3. *If $L(c \cdot \boldsymbol{m}) = F(c \cdot \boldsymbol{m})$, $c \cdot \boldsymbol{m} \to^* \{\mathsf{f} \colon c\}$ and $(\mathcal{N}, c \cdot \boldsymbol{m})$ is bounded, then $\boldsymbol{m} = \{\mathsf{f} \colon 1\}$.*

Lemma 3. *Let $\mathcal{N}$ be a workflow net. If $\mathcal{N}$ is continuously sound, then $(\mathcal{N}, \{\mathsf{i} \colon k\})$ is bounded for all $k \in \mathbb{N}_{\geq 1}$.*

Theorem 3. *Let $\mathcal{N}$ be a free-choice workflow net. These statements are equivalent: (1) $\mathcal{N}$ is 1-sound, (2) $\mathcal{N}$ is generalised sound, (3) $\mathcal{N}$ is structurally sound, and (4) $\mathcal{N}$ is continuously sound.*

Proof. (1) $\Rightarrow$ (2). This was shown in [32].

(2) $\Rightarrow$ (3). By definition, if $\mathcal{N}$ is k-sound for all k, then it is for some k.

(2) $\Rightarrow$ (4). By Theorem 1.

(3) $\Rightarrow$ (1). Let $k \in \mathbb{N}_{\geq 1}$ be such that $\mathcal{N}$ is k-sound. Let $\boldsymbol{m} \in \mathbb{N}^P$ be such that $\{\mathsf{i} \colon 1\} \to^* \boldsymbol{m}$. By Lemma 2(1), there is a marking $\boldsymbol{m}' \in \mathbb{N}^P$ such that $\boldsymbol{m} \to^* \boldsymbol{m}'$ and $F(\boldsymbol{m}') = L(\boldsymbol{m}')$. By Lemma 2(2), we have $L(k \cdot \boldsymbol{m}') = F(k \cdot \boldsymbol{m}') = F(\boldsymbol{m}')$.

By k-soundness, $(\mathcal{N}, \{\mathsf{i}\colon k\})$ must be bounded [10, Proposition 3.2 and Lemma 3.6]. Thus, since $\{\mathsf{i}\colon k\} \to^* k \cdot \boldsymbol{m} \to^* k \cdot \boldsymbol{m}'$, it is also the case that $(\mathcal{N}, k \cdot \boldsymbol{m}')$ is bounded. By k-soundness, $k \cdot \boldsymbol{m}' \to^* \{\mathsf{f}\colon k\}$. By invoking Lemma 2(3) with $c \coloneqq k$, we conclude that $\boldsymbol{m}' = \{\mathsf{f}\colon 1\}$. So, $\mathcal{N}$ is 1-sound as $\{\mathsf{i}\colon 1\} \to^* \boldsymbol{m} \to^* \boldsymbol{m}' = \{\mathsf{f}\colon 1\}$.

(4) $\Rightarrow$ (1). Assume that $\mathcal{N}$ is continuously sound. Let $\boldsymbol{m} \in \mathbb{N}^P$ be a marking such that $\{\mathsf{i}\colon 1\} \to^* \boldsymbol{m}$. By Lemma 2(1), there exists $\boldsymbol{m}' \in \mathbb{N}^P$ such that $\boldsymbol{m} \to^* \boldsymbol{m}'$ and $L(m') = F(m')$. Clearly, $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$ and by continuous soundness $\boldsymbol{m}' \to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$. By Lemma 1, there exists $b \in \mathbb{N}_{\geq 1}$ such that $b \cdot \boldsymbol{m}' \to^* \{\mathsf{f}\colon b\}$.

By Lemma 3, continuous soundness of $\mathcal{N}$ implies that $(\mathcal{N}, b \cdot \boldsymbol{m}')$ is bounded, as $\{\mathsf{i}\colon b\} \to^* b \cdot \boldsymbol{m}'$. Since $L(m') = F(m')$, it follows from Lemma 2(2) that $L(b \cdot \boldsymbol{m}') = F(b \cdot \boldsymbol{m}')$. By invoking Lemma 2(3) with $c \coloneqq b$, we derive $\boldsymbol{m}' = \{\mathsf{f}\colon 1\}$. Therefore, $\mathcal{N}$ is 1-sound as $\{\mathsf{i}\colon 1\} \to^* \boldsymbol{m} \to^* \boldsymbol{m}' = \{\mathsf{f}\colon 1\}$. □

7 Experimental evaluation

We implemented our approaches for generalised and structural soundness in C#.[4] We test continuous soundness via SMT solving. More precisely, we use an existential $\psi_{\mathcal{N}}$ formula of linear arithmetic, i.e. $\mathsf{FO}(\mathbb{Q}, <, +)$, from [8]. This formula is such that $\psi(\boldsymbol{m}, \boldsymbol{m}')$ holds iff $\boldsymbol{m} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$ in $\mathcal{N}$. Continuous soundness amounts to the $\exists\forall$-formula $\psi_{\mathcal{N}}(\{\mathsf{i}\colon 1\}, \boldsymbol{m}) \wedge \neg\psi_{\mathcal{N}}(\boldsymbol{m}, \{\mathsf{f}\colon 1\})$. To solve such formulas, we use Z3 [30]. We further use Z3 to decide structural quasi-soundness and compute $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$ (see Proposition 10), again via the formulas of [8].

We evaluated our prototype implementation on a standard benchmark suite used regularly in the literature, and a novel suite of synthetic instances where generalised or structural soundness are hard to decide with a naive approach.

We compared with two established tools for soundness: LoLA (v2.0) [40], and Woflan [38].[5] The latter can only decide *classical* soundness (1-soundness + quasi-liveness). Nonetheless, we use quasi-live instances, so for which 1-soundness and classical soundness are equivalent. We further use a transformation to reduce the verification of k-soundness to the one of 1-soundness [10, Lemma 3.6]. On the other hand, LoLA can directly decide k-soundness. To do so, we start from $\{\mathsf{i}\colon k\}$ and check a CTL formula of the form $\forall\mathsf{G}\,\exists\mathsf{F}\,((\boldsymbol{m}[\mathsf{f}] = k) \wedge \bigwedge_{p \neq \mathsf{f}} \boldsymbol{m}[p] = 0)$.

Experiments were run on an 8-Core Intel® Core™ i7-7700 CPU @ 3.60GHz with Ubuntu 18.04. We limited memory to ∼8GB, and time to 120s for each instance. Tools were called from a Python script. For LoLA and our implementation, we used the *time* module to measure time. Running Woflan involves some overhead, so we instead take the total verification time reported by Woflan itself.

7.1 Free-choice benchmark suite

The benchmark suite encompasses 1386 free-choice Petri nets that represent business processes modeled in the IBM WebSphere Business Modeler. It was origi-

[4] In the case of acceptance, we will submit an artifact to the artifact evaluation.

[5] A version of Woflan suitable for running without user interaction was provided, via personal communication, by its maintainer.

nally presented in [18], and has been studied frequently in the literature [11,19]. These nets are not workflow nets by our definition, but can be transformed using a known procedure [26]. Intuitively, the nets are workflow nets with multiple final places, and the procedure adds a dedicated output place and ensures that the resulting workflow net represents the desired behaviour. However, roughly 1% of the nets are not workflow nets by our definition even after the procedure, as they contain nodes that are not on a path from i to f. We removed these nets.

We further checked each net for safety using LoLA and dropped unsafe nets. Recall that $(\mathcal{N}, \{\mathsf{i}\colon 1\})$ is sound if each reachable marking has at most one token per place. Unsafe instances can be dropped as unsafety implies 1-unsoundness in free-choice nets [39, Thm. 4.2 and 4.4], and as existing methods for checking safety, *e.g.* via state-space exploration with partial order reductions, are very efficient (here needing a mean of 3ms). Thus, we considered safe instances only. Among the 1386 instances, 1382 are workflow nets, and 977 are further safe.

We also invoked an implementation of the reduction rules of [11] to reduce the size of all instances.[6] As discussed in the introduction, the rules can reduce some instances to trivially sound nets. However, even the size of nontrivial reduced instances tends to be small, with an average number of places and transitions of roughly 14, while three quarters of nets have at most 18 places and transitions. This is small enough that a complete state-splace enumeration is often feasible, in particular as the nets are safe and especially LoLA utilizes powerful partial order reductions for such nets. As we want to focus on scalability, we chained instances to produce challenging synthetic nets based on real-world instances. This is a natural way of constructing workflow nets, intuitively, the final process can be composed of many subtasks. It can be seen as a special case of refinement operations, studied in the context of generalised soundness [23].

The chaining procedure merges two workflow nets $\mathcal{N} = (P, T, F)$ and $\mathcal{N}' = (P', T', F')$ into $\mathcal{N}'' := (P'', T'', F'')$ where $P'' := P \cup P'$, $T'' := T \cup T' \cup \{t_{aux}\}$ with F'' as $F' + F''$ extended with ${}^\bullet t_{\text{aux}}[\mathsf{f}] := 1$, $t^\bullet_{\text{aux}}[\mathsf{i}'] := 1$, and ${}^\bullet t_{\text{aux}}[p] = t^\bullet_{\text{aux}}[p'] := 0$ for other entries. It is readily seen that this construction (1) produces a free-choice net if both $\mathcal{N}$ and $\mathcal{N}'$ are free-choice; and (2) preserves safety.

This way, we generated large instances by using $\ell \in \{1, 21, 41, \ldots, 401\}$ randomly chosen unreduced safe instances from the benchmark suite as inputs to be chained into one instance, then reduced that instance. For each number ℓ, we produced 20 combined nets, with a fresh random choice each time, in order to have a more representative collection of nets for ℓ. This resulted in 420 instances, of which 405 are nontrivial after applying reduction rules.

A caveat is that such large nets may seem unlikely to arise in practice. It seems a human designer would avoid designing highly complex processes corresponding to Petri nets with thousands of places. However, process models are not only explicitly written by humans, but also machine-generated, *e.g.* by mining event logs (see [37] for a book on the topic). In particular, being free-choice is

[6] At time of writing, an implementation is available at https://github.com/LoW12/Hadara-AdSimul.

preserved by chaining, so a large free-choice net may "hide" and combine several less complex processes, which might necessitate analyzing large workflow nets.

Results. We checked the safe free-choice instances obtained as explained above for 1-soundness using LoLA, Woflan and our implementation of continuous soundness. The results are shown on the left of Figure 3. The right-hand side of the figure provides an overview over the sizes of the nets. In each case, N refers to the number of original instances that were chained to create each instance.

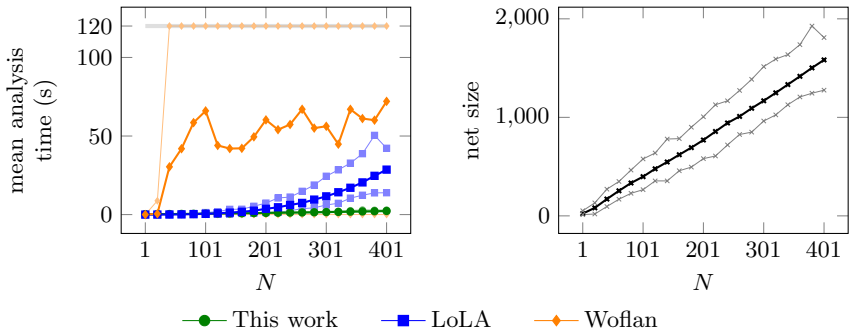


Fig. 3. Experiments on chained free-choice instances. The x-value denotes the number N of chained nets. Dark thick lines denote the mean, and light thin lines of the same color denote the minimum and maximum, respectively. For Woflan, the minimum line is slightly below the line of this work. For this work, the minimum and maximum lines are very close to the mean. *Left:* The y-value denotes time for checking soundness of the 20 nets for each N. Marks on the gray line at $120s$ denote timeouts. *Right:* The y-value denotes the size of generated nets.

The results show that state-space exploration via LoLA is very fast for moderate sizes, but does not scale as well. Continuous soundness is in fact outperformed by LoLA for $N \leq 100$, but scales much better, showing essentially linear growth in the given data range. For instance, continuous soundness takes a mean of $0.25s$ for $N = 1$, a mean of $1.07s$ for $N = 201$, and a mean of $2.28s$ for $N = 401$.

Woflan performs very well on the original instances, but times out frequently for larger instances. Woflan checks so-called S-coverability [39]. This is fast on many instances, even large ones, but starts running into the exponential-time worst case when instances get larger. For $N = 1$ and $N = 21$, Woflan does not ever time out, while it times out for roughly half of the instances in the range from $N = 201$ to $N = 401$. Overall, we infer that for large free-choice workflow nets, deciding soundness by checking continuous soundness can outperform existing techniques, while the procedure is still competitive on moderate instances.

7.2 Synthetic instances

In the previously discussed benchmark suite, nets are free-choice. So structural and generalised soundness are equivalent by Theorem 3. We considered including a second suite of 590 non-free-choice Petri nets that represent processes of the SAP reference model [29]. However, all of them turn out to be 1-quasi-sound but not 1-sound, so they represent trivial cases for generalised and structural soundness: simply checking 1-soundness, or 1-quasi-soundness and then 1-soundness, decides all instances. In order to have a wider variety of challenging instances, we introduce several families of synthetic workflow nets. The nets are simple to understand, but have large numbers of reachable marking, so are challenging for approaches relying on state-space exploration, *e.g.* model checking.

Encoding arc weights. To simplify the presentation, we describe synthetic instances utilizing arcs with weights. For benchmarking, we removed the arc weights and instead input equivalent weightless nets. To do so, we used an encoding that simulates exponentially large weights by polynomially many transitions and places (the encoding is explained in Appendix A.5). It preserves (quasi-)soundness, but significantly increases the number of reachable markings. Indeed, our synthetic instances are mostly trivial to solve by enumerating reachable markings when arcs have weights, but become much harder to decide when the encoding is used.[7] While much of the literature on workflow nets does not consider nets with arc weights, implicit structural encodings can occur in practice.

Generalised soundness

Benchmark instances. We introduce a synthetic family of nets where generalised soundness appears to be challenging. The family $\{\mathcal{N}_c\}_{c\in\mathbb{N}_{\geq 1}}$ is defined at the top of Figure 4. Parameter $c \in \mathbb{N}_{\geq 1}$ is the smallest value for which $\mathcal{N}_c$ is c-unsound. From $\{\mathsf{i}\colon c\}$, the sequence $t_{\mathsf{i}}^c t_r^{c+1}$ can be fired, which leads to the deadlock $\{r\colon c+1\}$. Yet, when starting with $k < c$ tokens in i, and firing t_{i}^k, transitions t_r and t_{f} can only be fired exactly k times, and $\{\mathsf{f}\colon k\}$ will be reached.

The naive approach to decide generalised soundness is to check k-soundness for all k until a counterexample is found or a bound is exceeded. It is known that if a counterexample exists, then there also is one of size at most exponential [10, Lemma 5.6 and 5.8]. The approach we chose for semi-deciding generalised soundness is to check continuous soundness. Recall that continuous soundness is a necessary (albeit not sufficient) condition, as shown in Theorem 1.

In our evaluation, we used Woflan and LoLA to check generalised soundness of the family for different c by checking 1-sound, ..., c-soundness, and compared the result to the time needed for testing continuous soundness. Our main goal is to evaluate whether checking continuous soundness is efficient enough to serve as an inexpensive way to witness generalised unsoundness for nontrivial instances.

[7] It is deliberately used to make instances challenging, not to ensure compatibility with LoLA or Woflan, as both support arc weights.

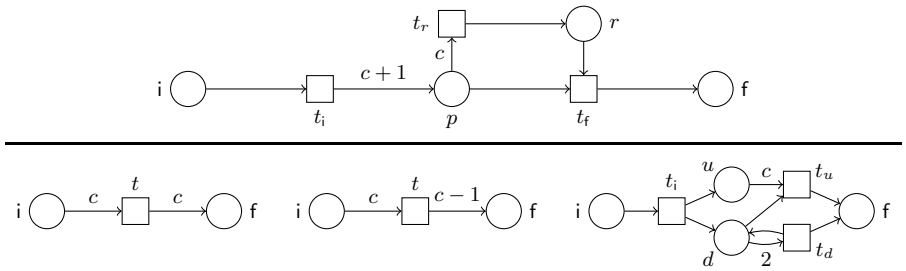


Fig. 4. *Top*: A workflow net $\mathcal{N}_c$ that is c-unsound and k-sound for all $k \in [1..c-1]$. *Bottom*: Three families of instances. *Bottom left:* $\mathcal{N}_{\text{sound-}c}$ is quasi-sound and ℓc-sound for all $\ell \in \mathbb{N}_{\geq 1}$. *Bottom center:* $\mathcal{N}_{\neg\text{quasi-}c}$ is not structurally quasi-sound. *Bottom right:* $\mathcal{N}_{\neg\text{sound-}c}$ is ℓc-quasi-sound for all $\ell \in \mathbb{N}_{\geq 1}$, but not structurally sound.

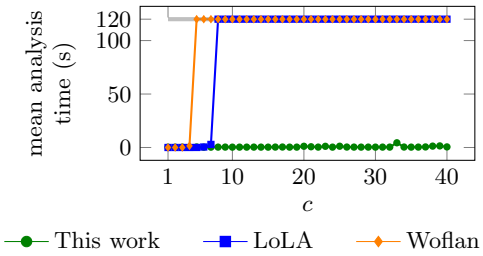


Fig. 5. Time to check generalised soundness of $\mathcal{N}_c$ for different values of c. Marks on the gray line at $120s$ denote timeouts.

Results. Figure 5 depicts the results. Woflan and LoLA show good performance for small values of c, but do not scale well to larger values. They respectively time out for $c \geq 5$ and $c \geq 8$. The instances are not free-choice, so LoLA and Woflan need to explore the state-space for each $k \leq c$, which becomes infeasible. For $c \geq 14$, Woflan cannot even check 1-soundness within the time limit. LoLA can check 1- and 2-soundness for $c \leq 28$, but cannot handle 2-soundness for larger c. Continuous soundness is efficiently verifiable even for $c = 40$. In particular, we need less than $5s$ on all instances. The greatest time is at $c = 33$. Further, at most $1s$ is needed on 34 out of 40 instances (mean of $0.6s$).

Structural soundness

Benchmark instances. For structural soundness, recall that our decision procedure is based on checking structural quasi-soundness and obtaining some lower bound for the smallest number for which the net is quasi-sound. Thus, we want to test on both benchmark instances that are structurally quasi-sound and those that are not. We introduce three families of non-free-choice nets for which struc-

tural soundness appears challenging. These instances are defined at the bottom of Figure 4. We respectively denote them $\mathcal{N}_{\text{sound-}c}$ (left), $\mathcal{N}_{\neg\text{quasi-}c}$ (center) and $\mathcal{N}_{\neg\text{sound-}c}$ (right). We claim that: $\mathcal{N}_{\text{sound-}c}$ is ℓc-sound for all $\ell \in \mathbb{N}_{\geq 1}$; $\mathcal{N}_{\neg\text{quasi-}c}$ is not structurally quasi-sound; $\mathcal{N}_{\neg\text{sound-}c}$ is ℓc-quasi-sound for all $\ell \in \mathbb{N}_{\geq 1}$, not k-quasi-sound for any other number $k \in \mathbb{N}_{\geq 1}$, and not structurally sound.

For the experiments, our goal is twofold. First, we want to evaluate whether utilizing continuous reachability to decide structural quasi-soundness is more efficient than using the known reduction to reachability described in [36, Lemma 2.1]. Woflan does not directly support checking reachability, so we only compare with LoLA. Second, we want to evaluate whether the lower bound for the smallest number for which the net is quasi-sound, which we dubbed $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$ towards the end of Section 5, is close to the actual smallest number, dubbed $k_{\mathcal{N}}$.

A caveat of this evaluation is that we evaluate only on our synthetic instances, and that computing $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$ is only one step in deciding structural soundness. However, we think that the evaluation on these hard synthetic instances can give insights into the applicability on nontrivial real-world instances.

Results. Figure 6 compares the time needed to verify structural reachability for LoLA and our prototype. For small instances, LoLA sometimes performs very well, but we scale better for large values. Of particular note is that in the absence of quasi-soundness, LoLA will generate an infinite state-space, so will generally run out of time or memory. In particular, LoLA times out for all c on $\mathbb{N}_{\neg\text{quasi-}c}$. It also times out for $c \geq 32$ on $\mathbb{N}_{\neg\text{sound-}c}$. On the other hand, continuous soundness never times out for the given values of c. In fact, when we tested continuous soundness for much larger values of c, we found that our implementation of continuous reachability decides structural quasi-soundness for $N_{\neg\text{quasi-}c}$ in under $2s$ for $c = 20\ 000\ 000$.

We further found that for all instances, $k_{\mathcal{N},\mathbb{Q}_{\geq 0}} = k_{\mathcal{N}}$, that is, our lower bound exactly matches the smallest number for which the net is quasi-sound. Thus, it only remains to decide $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$-quasi-soundness and $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$-soundness in order to decide structural soundness. This is in contrast to the naive approach, which starts at $k = 1$ and checks k-quasi-soundness for each value up to $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$.

8 Conclusion

In this work, we have shown how reachability relaxations allow to efficiently semi-decide generalised and structural soundness. Our approach combines nicely with reduction rules, as they all preserve relaxations. In particular, we have introduced continuous soundness as an approximation of generalised soundness, and shown that it coincides with other types of soundness for free-choice nets.

As part of future work, we plan to migrate our prototype into the process mining framework ProM, to make the algorithms available to practitioners.

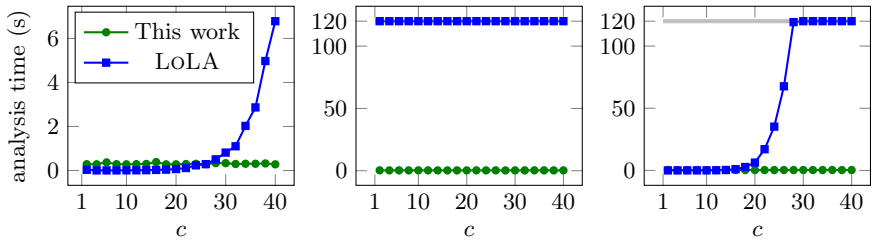


Fig. 6. Time taken vs parameter c for checking structural quasi-soundness using the reduction to reachability, and utilizing our approach to compute $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$, for each of the three families at the bottom of Figure 4: $\mathcal{N}_{\text{sound-}c}$ (*left*),$\mathcal{N}_{\neg\text{quasi-}c}$ (*center*), $\mathcal{N}_{\neg\text{sound-}c}$ (*right*). Note that the axis ranges differ. Marks on the gray line at $120s$ denote timeouts.

Acknowledgements

We thank Dirk Fahland and Eric Verbeek for their help with Woflan. Michael Blondin was supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), and by the Fonds de recherche du Québec – Nature et technologies (FRQNT).

References

1. van der Aalst, W.M.P.: Verification of workflow nets. In: Proc. 18th International Conference on Application and Theory of Petri Nets (ICATPN). vol. 1248, pp. 407–426 (1997). https://doi.org/10.1007/3-540-63139-9_48
2. van der Aalst, W.M.P.: The application of Petri nets to workflow management. Journal of Circuits, Systems, and Computers **8**(1), 21–66 (1998). https://doi.org/10.1142/S0218126698000043
3. van der Aalst, W.M.P.: Workflow verification: Finding control-flow errors using Petri-net-based techniques. In: Business Process Management, Models, Techniques, and Empirical Studies. pp. 161–183 (2000). https://doi.org/10.1007/3-540-45594-9_11
4. van der Aalst, W.M.P., van Hee, K.M.: Workflow Management: Models, Methods, and Systems. Cooperative information systems, MIT Press (2002)
5. Athanasiou, K., Liu, P., Wahl, T.: Unbounded-thread program verification using thread-state equations. In: Proc. 8th International Joint Conference on Automated Reasoning (IJCAR). pp. 516–531 (2016). https://doi.org/10.1007/978-3-319-40229-1_35
6. Barkaoui, K., Petrucci, L.: Structural analysis of workflow nets with shared resources. In: Proc. Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM). vol. 98/7, pp. 82–95 (1998)
7. Blondin, M.: The ABCs of Petri net reachability relaxations. ACM SIGLOG News **7**(3) (2020). https://doi.org/10.1145/3436980.3436984
8. Blondin, M., Finkel, A., Haase, C., Haddad, S.: The logical view on continuous Petri nets. ACM Transactions on Computational Logic (TOCL) **18**(3), 24:1–24:28 (2017). https://doi.org/10.1145/3105908

9. Blondin, M., Haase, C.: Logics for continuous reachability in Petri nets and vector addition systems with states. In: Proc. 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). pp. 1–12 (2017). https://doi.org/10.1109/LICS.2017.8005068
10. Blondin, M., Mazowiecki, F., Offtermatt, P.: The complexity of soundness in workflow nets. In: Proc. 37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (2022)
11. Bride, H., Kouchnarenko, O., Peureux, F.: Reduction of workflow nets for generalised soundness verification. In: Proc. 18th International Conference Verification, Model Checking, and Abstract Interpretation (VMCAI). pp. 91–111 (2017). https://doi.org/10.1007/978-3-319-52234-0_6
12. Cardoza, E., Lipton, R.J., Meyer, A.R.: Exponential space complete problems for Petri nets and commutative semigroups: Preliminary report. In: Proc. 8th Annual ACM Symposium on Theory of Computing (STOC). pp. 50–54 (1976). https://doi.org/10.1145/800113.803630
13. Chistikov, D., Haase, C., Halfon, S.: Context-free commutative grammars with integer counters and resets. Theoretical Computer Science **735**, 147–161 (2018). https://doi.org/10.1016/j.tcs.2016.06.017, https://doi.org/10.1016/j.tcs.2016.06.017
14. Czerwinski, W., Orlikowski, L.: Reachability in vector addition systems is Ackermann-complete. In: Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS) (2021), to appear
15. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge University Press (1995). https://doi.org/10.1017/CBO9780511526558
16. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge University Press (1995). https://doi.org/10.1017/CBO9780511526558
17. Esparza, J., Ledesma-Garza, R., Majumdar, R., Meyer, P.J., Nikšić, F.: An SMT-based approach to coverability analysis. In: Proc. 26th International Conference on Computer Aided Verification (CAV). pp. 603–619 (2014). https://doi.org/10.1007/978-3-319-08867-9_40
18. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Instantaneous soundness checking of industrial business process models. In: Proc. 7th International Conference on Business Process Management (BPM). pp. 278–293 (2009). https://doi.org/10.1007/978-3-642-03848-8_19
19. Favre, C., Völzer, H., Müller, P.: Diagnostic information for control-flow analysis of workflow graphs (a.k.a. free-choice workflow nets). In: Proc. 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). vol. 9636, pp. 463–479 (2016). https://doi.org/10.1007/978-3-662-49674-9_27
20. Fraca, E., Haddad, S.: Complexity analysis of continuous Petri nets. Fundamenta Informaticae **137**(1), 1–28 (2015). https://doi.org/10.3233/FI-2015-1168
21. Haase, C., Halfon, S.: Integer vector addition systems with states. In: Proc. 8th International Workshop on Reachability Problems (RP). pp. 112–124 (2014). https://doi.org/10.1007/978-3-319-11439-2_9
22. van Hee, K.M., Oanea, O., Sidorova, N., Voorhoeve, M.: Verifying generalized soundness of workflow nets. In: Proc. 6th International Andrei Ershov Memorial Conference on Perspectives of Systems Informatics (PSI). pp. 235–247 (2006). https://doi.org/10.1007/978-3-540-70881-0_21
23. van Hee, K.M., Sidorova, N., Voorhoeve, M.: Soundness and separability of workflow nets in the stepwise refinement approach. In: Proc. 24th International Con-

ference on Applications and Theory of Petri Nets 2003 (ICATPN). vol. 2679, pp. 337–356 (2003). https://doi.org/10.1007/3-540-44919-1_22

24. van Hee, K.M., Sidorova, N., Voorhoeve, M.: Generalised soundness of workflow nets is decidable. In: Proc. 25th International Conference on Applications and Theory of Petri Nets (ICATPN). pp. 197–215 (2004). https://doi.org/10.1007/978-3-540-27793-4_12
25. Hoffmann, P.E.: Workflow Nets: Reduction Rules and Games. Ph.D. thesis, Technische Universität München (2017)
26. Kiepuszewski, B., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Fundamentals of control flow in workflows. Acta Informatica **39**(3), 143–209 (2003). https://doi.org/10.1007/s00236-002-0105-4
27. Leroux, J.: The reachability problem for Petri nets is not primitive recursive. In: Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS) (2021), to appear
28. Leroux, J., Schmitz, S.: Reachability in vector addition systems is primitive-recursive in fixed dimension. In: Proc. 34th Symposium on Logic in Computer Science (LICS) (2019). https://doi.org/10.1109/LICS.2019.8785796
29. Mendling, J., Moser, M., Neumann, G., Verbeek, H.M.W., van Dongen, B.F., van der Aalst, W.M.P.: Faulty EPCs in the SAP reference model. In: Proc. 4th International Conference on Business Process Management (BPM). pp. 451–457 (2006). https://doi.org/10.1007/11841760_38
30. de Moura, L.M., Bjørner, N.: Z3: an efficient SMT solver. In: Proc. 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 337–340. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_24, tool available at `https://github.com/Z3Prover/z3`.
31. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989). https://doi.org/10.1109/5.24143
32. Ping, L., Hao, H., Jian, L.: On 1-soundness and soundness of workflow nets. In: Third Workshop on Modelling of Objects, Components, and Agents. p. 21 (2004)
33. Rackoff, C.: The covering and boundedness problems for vector addition systems. Theoretical Computer Science **6**, 223–231 (1978). https://doi.org/10.1016/0304-3975(78)90036-1
34. Schrijver, A.: Theory of Linear and Integer Programming. John Wiley & Sons (1986)
35. Ţiplea, F.L., Bocăneală, C., Chiroşcă, R.: On the complexity of deciding soundness of acyclic workflow nets. IEEE Transactions on Systems, Man, and Cybernetics: Systems **45**(9), 1292–1298 (2015). https://doi.org/10.1109/TSMC.2015.2394735
36. Ţiplea, F.L., Marinescu, D.C.: Structural soundness of workflow nets is decidable. Information Processing Letters IPL **96**(2), 54–58 (2005). https://doi.org/10.1016/j.ipl.2005.06.002
37. Van Der Aalst, W.: Data science in action. In: Process mining, pp. 3–23. Springer, 2 edn. (2016)
38. Verbeek, E., van der Aalst, W.M.P.: Woflan 2.0: A Petri-net-based workflow diagnosis tool. In: Proc. 21st International Conference on Application and Theory of Petri Nets 2000, , (ICATPN). pp. 475–484 (2000). https://doi.org/10.1007/3-540-44988-4_28
39. Verbeek, H.M.W., Basten, T., van der Aalst, W.M.P.: Diagnosing workflow processes using woflan. The Computer Journal **44**(4), 246–279 (2001). https://doi.org/10.1093/comjnl/44.4.246

40. Wolf, K.: Petri net model checking with LoLA 2. In: Application and Theory of Petri Nets and Concurrency. pp. 351–362 (2018)

A Appendix

A.1 Missing proofs of Section 3

Lemma 1. *Let $\boldsymbol{m}$, $\boldsymbol{m}'$ be continuous markings. It is the case that $\boldsymbol{m} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$ iff there exists $b \in \mathbb{N}_{\geq 1}$ such that $b \cdot \boldsymbol{m} \to^* b \cdot \boldsymbol{m}'$.*

Proof. $\Leftarrow$) Let $b \cdot m \to^{\pi} b \cdot \boldsymbol{m}'$. Let $\beta \coloneqq 1/b$. Let us prove that $\boldsymbol{m} \to^{\beta \cdot \pi} \boldsymbol{m}'$. To do so, we show that $b \cdot \boldsymbol{m} \to^{\pi[1:n]} \boldsymbol{m}_n$ implies $\boldsymbol{m} \to^{\pi[1:n]}_{\mathbb{Q}_{\geq 0}} \beta \cdot \boldsymbol{m}_n$. Let us proceed by induction on n. Assume that

$$b \cdot m \to^{\pi[1:n]} \boldsymbol{m}_n \to^{t_{n+1}} \boldsymbol{m}_{n+1} \text{ where } t_{n+1} \coloneqq \pi[n+1].$$

By induction hypothesis, we have $\boldsymbol{m} \to^{\pi[1:n]}_{\mathbb{Q}_{\geq 0}} \beta \cdot \boldsymbol{m}_n$. Note that $\boldsymbol{m}_{n+1} = \boldsymbol{m}_n + \Delta(t_{n+1})$. So, $\beta \cdot \boldsymbol{m}_n + \beta \cdot \Delta(t_{n+1}) = \beta \cdot \boldsymbol{m}_{n+1}$. Thus, $\beta \cdot t_{n+1}$ has the right effect to lead from $\boldsymbol{m}_n$ to $\boldsymbol{m}_{n+1}$. It only remains to show that $\beta \cdot t_{n+1}$ is enabled at $\boldsymbol{m}_n$. Note that t_{n+1} is enabled at $\boldsymbol{m}_n$, hence by definition, $\boldsymbol{m}_n[p] \geq {}^{\bullet}t_{n+1}[p]$ for all $p \in P$. It follows that $\beta \cdot \boldsymbol{m}_n[p] > \beta \cdot {}^{\bullet}t_{n+1}[p]$, so $\beta \cdot t_{n+1}$ is enabled in $\boldsymbol{m}_n$.

$\Rightarrow$) Let $\boldsymbol{m} \to^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$. Let β be the product of the scaling factors denominators along π. Let us show that $b \cdot \boldsymbol{m} \to^{\pi'} b \cdot \boldsymbol{m}'$. We establish the following for all n:

$$\text{if } \boldsymbol{m} \to^{\pi[1:n]}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}_n, \text{ then there exists } \pi'_n \text{ such that } b \cdot \boldsymbol{m} \to^{\pi'_n} b \cdot \boldsymbol{m}_n.$$

Assume this holds for some n. Let $\alpha \cdot t_{n+1} = \pi[n]$. We show the following:

$$\text{if } \boldsymbol{m}_n \to^{\alpha t_{n+1}}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}_{n+1}, \text{then } b \cdot \boldsymbol{m}_n \to^{(t_{n+1})^{b \cdot \alpha}} b \cdot \boldsymbol{m}_{n+1}.$$

First, let us argue that $b \cdot \alpha$ is an integer. Note that by the fact that scaling factors are chosen from $(0, 1]$, it follows that α can be written as u/d for some $u, d \in \mathbb{N}$ where $d \neq 0$. Further, note that b was chosen as the product of all denominators of scaling factors along π. In particular, d is a factor of b, so we have $b = d \cdot b'$ for some $b' \in \mathbb{N}$, and thus $b \cdot \alpha = d \cdot b' \cdot u/d = b' \cdot u$. Next, let us argue that $(t_{n+1})^{b \cdot \alpha}$ has the right effect to lead from $b \cdot \boldsymbol{m}_n$ to $b \cdot \boldsymbol{m}_{n+1}$. Note that $\boldsymbol{m}_{n+1} = \boldsymbol{m}_n + \alpha \cdot \Delta(t_{n+1})$. So, $b \cdot \boldsymbol{m}_{n+1} = b \cdot \boldsymbol{m}_n + b \cdot \alpha \Delta(t_{n+1}) = b \cdot \boldsymbol{m}_n + \Delta((t_{n+1})^{b \cdot \alpha})$. It remains to argue that $(t_{n+1})^{b \cdot \alpha}$ is fireable from $b \cdot \boldsymbol{m}_n$. By $\boldsymbol{m}_n \to^{\alpha t_{n+1}}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}_{n+1}$, it follows that $\boldsymbol{m}_n[p] \geq \alpha {}^{\bullet}t_{n+1}[p]$ for all $p \in P$. Since $b \in \mathbb{N}$, it is the case that $b \cdot \boldsymbol{m}_n[p] \geq b \cdot \alpha {}^{\bullet}t_{n+1}[p]$, and hence we are done. □

Proposition 1. *Let $\mathcal{N} = (P, T, F)$ be a workflow net, and let $\mathbb{D} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}_{\geq 0}\}$. Let $\mathcal{N}' = (P', T', F')$ be a workflow net obtained by applying a reduction rule R_i to $\mathcal{N}$, where $P = P' \cup P''$. The following holds.*

- Rule R_1. *We have $P'' = \{p\}$. There exists a nonempty set $R' \subseteq P'$ such that if $\{\mathsf{i} \colon 1\} \to^*_{\mathbb{D}} \boldsymbol{m}$ in $\mathcal{N}$, then $\boldsymbol{m}[p] = \sum_{r \in R'} \boldsymbol{m}[r']$. Moreover, $\boldsymbol{m} \to^*_{\mathbb{D}} \boldsymbol{n}$ in $\mathcal{N}$ iff $\pi(\boldsymbol{m}) \to^*_{\mathbb{D}} \pi(\boldsymbol{n})$ in $\mathcal{N}'$.*
- Rules R_2 and R_3. *We have $P'' = \emptyset$ and $\boldsymbol{m} \to^*_{\mathbb{D}} \boldsymbol{n}$ in $\mathcal{N}$ iff $\boldsymbol{m} \to^*_{\mathbb{D}} \boldsymbol{n}$ in $\mathcal{N}'$.*

- Rules R_4 and R_5. *We have* $P'' = \{p\}$. *For all* $\boldsymbol{m}'$ *and* $\boldsymbol{n}'$, $\boldsymbol{m}' \to^*_{\mathbb{D}} \boldsymbol{n}'$ *in* $\mathcal{N}'$ *iff* $\pi_0(\boldsymbol{m}') \to^*_{\mathbb{D}} \pi_0(\boldsymbol{n}')$ *in* $\mathcal{N}$. *Further, for all* $t \in T$ *and* $p' \in P'$: *either* ${}^\bullet t[p] = 1$ *implies* ${}^\bullet t[p'] = 0$; *or* $t^\bullet[p] = 1$ *implies* $t^\bullet[p'] = 0$. *Also, for* $\mathbb{D} \neq \mathbb{Z}$, *if* $\exists \boldsymbol{m} : \{\mathsf{i}\colon 1\} \to^*_{\mathbb{D}} \boldsymbol{m} \not\to^*_{\mathbb{D}} \{\mathsf{f}\colon 1\}$ *holds in* $\mathcal{N}$, *then* $\exists \boldsymbol{m}' : \{\mathsf{i}\colon 1\} \to^*_{\mathbb{D}} \boldsymbol{m}' \not\to^*_{\mathbb{D}} \{\mathsf{f}\colon 1\}$ *holds in* $\mathcal{N}'$.
- Rule R_6. *We have* $P'' = \{p_2, \ldots, p_k\}$. *There exists* $p_1 \in P'$ *such that for all* $\boldsymbol{n} \in P^{\mathbb{D}}$, *if* $\sum_{i=1}^k \boldsymbol{m}[p_i] = \sum_{i=1}^k \boldsymbol{n}[p_i]$ *and* $\boldsymbol{n}[p'] = \boldsymbol{m}[p']$ *for* $p' \in P' \setminus \{p_1\}$, *then* $\boldsymbol{m} \to^*_{\mathbb{D}} \boldsymbol{n}$. *Moreover, if* $\boldsymbol{m}[p_i] = \boldsymbol{n}[p_i] = 0$ *for* $i > 1$, *then* $\boldsymbol{m} \to^*_{\mathbb{D}} \boldsymbol{n}$ *in* $\mathcal{N}$ *iff* $\pi(\boldsymbol{m}) \to^*_{\mathbb{D}} \pi(\boldsymbol{n})$ *in* $\mathcal{N}'$.

Proof. We will informally present the rules by the properties they preserve. For a formal definition of the rules, we refer to [11, Sect. 4.2]. Most arguments apply to all $\mathbb{D} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}_{\geq 0}\}$ in the same way, thus usually we will not make a cumbersome case analysis.

Rule R_1 (place removal). This rule removes a place $p \in P$. Thus, $P' = P \setminus \{p\}$. It is guaranteed that there exist places $\{g_1, \ldots, g_n\} \subseteq P'$ such that the number of tokens in p is the sum of tokens in those places. Hence, it suffices to define $R' \coloneqq \{g_1, \ldots, g_n\}$.

Rules R_2 (transition removal) and R_3 (loop removal). For these rules, no place is removed and the reachability relation is preserved.

Rules R_4 (transition-place removal) and R_5 (place-transition removal). These rules remove a place p and its only input (for R_4) or output (for R_5) transition t. Transition t is merged with the output (for R_4) or input (for R_5) transitions. Thus, intuitively, the new transitions in $\mathcal{N}'$ immediately consume a token whenever it was put in p. This clearly proves that $\boldsymbol{m}' \to^*_{\mathbb{D}} \boldsymbol{n}'$ in $\mathcal{N}'$ iff $\pi_0(\boldsymbol{m}') \to^*_{\mathbb{D}} \pi_0(\boldsymbol{n}')$ in $\mathcal{N}$. Moreover, the requirements on when the rule can be applied imply either ${}^\bullet t[p] = 1 \implies {}^\bullet t[p'] = 0$; or $t^\bullet[p] = 1 \implies t^\bullet[p'] = 0$.

It remains to prove the final part when $\mathbb{D} \neq \mathbb{Z}$. Suppose there exists $\boldsymbol{m}$ such that $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{D}} \boldsymbol{m} \not\to^*_{\mathbb{D}} \{\mathsf{f}\colon 1\}$ in $\mathcal{N}$. Suppose first, that there exists $\boldsymbol{n}$ such that $\boldsymbol{m} \to^*_{\mathbb{D}} \boldsymbol{n}$ and $\boldsymbol{n}[p] = 0$. By the previous case, we have $\pi(\boldsymbol{n}) \not\to^*_{\mathbb{D}} \{\mathsf{f}\colon 1\}$, as otherwise we reach the contradiction $\boldsymbol{m} \to^*_{\mathbb{D}} \boldsymbol{n} \to^*_{\mathbb{D}} \{\mathsf{f}\colon 1\}$. We define $\boldsymbol{m}' \coloneqq \pi(\boldsymbol{n})$. In the second case, we can assume that for all $\boldsymbol{n}$, $\boldsymbol{m} \to^*_{\mathbb{D}} \boldsymbol{n}$ implies $\boldsymbol{n}[p] > 0$ (here we use $\mathbb{D} \neq \mathbb{Z}$). In particular, $\boldsymbol{m}[p] > 0$. Let $T_p \coloneqq \{t \in T \mid t^\bullet[p] = 1\}$. We conclude from the additional constraints on R_4 and R_5 (see [11]). These imply that in our case for every $t \in T_p$ and for all $r \in P$:

1. if $r \neq p$, then $t^\bullet[r] = 0$;
2. if ${}^\bullet t[r] = 1$ and $t' \neq t$, then ${}^\bullet t'[r] = 0$.

Let ρ be the run witnessing $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{D}} \boldsymbol{m}$. Let ρ' be the subrun of transitions in T_p that occur in ρ (it is nonempty since $\boldsymbol{m}[p] > 0$). By Item 1 we can remove (or downscale if $\mathbb{D} = \mathbb{Q}_{\geq 0}$) a suffix of $\boldsymbol{m}[p]$ transitions in ρ' (because it removes tokens only from p). We obtain a marking $\boldsymbol{m}_1$ such that: $\boldsymbol{m}_1[p] = 0$; the tokens in $\boldsymbol{m}_1[r]$ for all removed $t \in T_p$ such that ${}^\bullet t[r] = 1$ have increased accordingly; and $\boldsymbol{m}_1[p'] = \boldsymbol{m}[p]$ otherwise. We claim that $\boldsymbol{m}' = \pi(\boldsymbol{m}_1)$ satisfies the proposition.

Indeed, if there is a run $\pi(\boldsymbol{m}_1) \to^*_{\mathbb{D}} \{\mathtt{f}\colon 1\}$ in $\mathcal{N}'$ then by Item 2 we can extract from this a run $\boldsymbol{m} \to^*_{\mathbb{D}} \{\mathtt{f}\colon 1\}$ in $\mathcal{N}$, which would be a contradiction.

R_6 *(ring removal).* This rule merges a set of places $\{p_1, \ldots, p_k\} \subseteq P$ into a single place p_1[8]. Thus, $P' = P \setminus \{p_2, \ldots, p_k\}$. The conditions are that the tokens can be transferred arbitrarily between the places $p_1, \ldots, p_k$, which is enough to prove the proposition. □

A.2 Missing proofs of Section 4

Proposition 3. *The reduction rules from [11] preserve integer unboundedness.*

Proof. We will need to invoke Proposition 4 which is stated after Proposition 3 in the main text. Note that this ordering is simply for the sake of presentation, there is no circular dependency, the proof of Proposition 4 is self-contained.

By Proposition 4, being integer unbounded is equivalent to the existence of $\boldsymbol{v} > \boldsymbol{0}$ such that $\boldsymbol{0} \to^*_{\mathbb{Z}} \boldsymbol{v}$. Let $\mathcal{N}$ and $\mathcal{N}'$ be the workflow nets before and after the reduction. We invoke Proposition 1 depending on the applied reduction rule, and show that $\mathcal{N}$ is integer unbounded iff $\mathcal{N}'$ is integer unbounded.

- *Rule* R_1. Suppose $\boldsymbol{0} \to^*_{\mathbb{Z}} \boldsymbol{v} > \boldsymbol{0}$ in $\mathcal{N}$. We have $\pi(\boldsymbol{v}) > \boldsymbol{0}$, since $\boldsymbol{v}[p] = \sum_{r \in R'} \boldsymbol{v}[r]$. Thus, $\pi(\boldsymbol{v})[r] > \boldsymbol{0}$ for at least one $r \in R'$. The converse implication is trivial.
- *Rules* R_2 *and* R_3. This is trivial because $\to^*_{\mathbb{Z}}$ is preserved.
- *Rules* R_4 *and* R_5. We have $\boldsymbol{m}' \to^*_{\mathbb{Z}} \boldsymbol{n}'$ in $\mathcal{N}'$ iff $\pi_0(\boldsymbol{m}') \to^*_{\mathbb{Z}} \pi_0(\boldsymbol{n}')$ in $\mathcal{N}$. Thus, if $\boldsymbol{0} \to^*_{\mathbb{Z}} \boldsymbol{v}' > \boldsymbol{0}$ in $\mathcal{N}'$, then $\boldsymbol{0} \to^*_{\mathbb{Z}} \pi_0(\boldsymbol{v}') > \boldsymbol{0}$ in $\mathcal{N}$. Conversely, suppose that $\boldsymbol{0} \to^{\rho}_{\mathbb{Z}} \boldsymbol{v} > \boldsymbol{0}$ in $\mathcal{N}$. If $\boldsymbol{v}[p] = 0$, then we are done. Otherwise, by Proposition 1 for all $t \in T$ and $p' \in P'$: either ${}^\bullet t[p] = 1 \implies {}^\bullet t[p'] = 0$; or $t^\bullet[p] = 1 \implies t^\bullet[p'] = 0$. Let us assume the former and let $T_p \coloneqq \{t \in T \mid {}^\bullet t[p] = 1\}$. By removing $\boldsymbol{v}[p]$ transitions from T_p in ρ, we get $\boldsymbol{0} \to^*_{\mathbb{Z}} \boldsymbol{v}' > \boldsymbol{0}$ and $\boldsymbol{v}'[p] = 0$. Thus, $\boldsymbol{0} \to^*_{\mathbb{Z}} \pi(\boldsymbol{v}') > \boldsymbol{0}$ in $\mathcal{N}'$ as required. In the latter case, we proceed similarly, but one need to add some transitions to ρ that will move the tokens from p to other places.
- *Rule* R_6. In this case, if $\boldsymbol{m}[p_i] = \boldsymbol{n}[p_i] = 0$ for $i > 1$, then $\boldsymbol{m} \to^*_{\mathbb{Z}} \boldsymbol{n}$ in $\mathcal{N}$ iff $\pi(\boldsymbol{m}) \to^*_{\mathbb{Z}} \pi(\boldsymbol{n})$ in $\mathcal{N}'$. Thus, $\boldsymbol{0} \to^*_{\mathbb{Z}} \boldsymbol{v}' > \boldsymbol{0}$ in $\mathcal{N}'$ clearly implies $\boldsymbol{0} \to^*_{\mathbb{Z}} \boldsymbol{v} > \boldsymbol{0}$ in $\mathcal{N}$. Conversely, if $\boldsymbol{0} \to^*_{\mathbb{Z}} \boldsymbol{v} > \boldsymbol{0}$ in $\mathcal{N}$, then we know that $\boldsymbol{v} \to^*_{\mathbb{Z}} \boldsymbol{v}_1$ where $\boldsymbol{v}_1[p_1] = \sum_{i=1}^{k} \boldsymbol{v}_1[p_i]$ and $\boldsymbol{v}_1[p_i] = 0$ for $i > 1$. So, $\boldsymbol{0} \to^*_{\mathbb{Z}} \pi(\boldsymbol{v}_1) > \boldsymbol{0}$ in $\mathcal{N}'$. □

Proposition 5. *A marked Petri net* $(\mathcal{N}, \boldsymbol{m})$*, where* $\mathcal{N} = (P, T, F)$*, is integer unbounded iff this system has a solution:* $\exists \boldsymbol{x} \in \mathbb{Q}^T_{\geq 0} : \sum_{t \in T} \boldsymbol{x}[t] \cdot \Delta(t) > \boldsymbol{0}$*. In particular, given a workflow net* $\mathcal{N}$*, testing integer boundedness of* $(\mathcal{N}, \{\mathtt{i}\colon 1\})$ *can be done in polynomial time.*

Proof. Let $\mathcal{N} = (P, F, T)$ be a Petri net. By Proposition 4, $(\mathcal{N}, \boldsymbol{m})$ is integer bounded iff there exists $\boldsymbol{m}' > \boldsymbol{0}$ such that $\boldsymbol{0} \to^*_{\mathbb{Z}} \boldsymbol{m}'$. The latter amounts to the

[8] In [11], p_1 is also removed and a new place p is added, but this is trivially equivalent.

existence of $\pi \in T^*$ such that $\Delta(\pi) > \mathbf{0}$. So, this is equivalent to this system: $\exists \boldsymbol{x} \in \mathbb{N}^T : \sum_{t \in T} \boldsymbol{x}[t] \cdot \Delta(t) > \mathbf{0}$. It is readily seen that this system is equivalent to the one where $\boldsymbol{x} \in \mathbb{Q}^T_{\geq 0}$. Indeed, by homogeneity ($\mathbf{0}$ on the right-hand side), a rational solution can be scaled so that it becomes an integral solution.

The polynomial time decidability of integer boundedness follows immediately from the fact that linear programming can be solved in polynomial time (*e.g.*, see [34]). □

Proposition 6. *The reduction rules from [11] preserve continuous soundness.*

Proof. Let $\mathcal{N}$ and $\mathcal{N}'$ be the workflow nets before and after the reduction. We invoke Proposition 1 depending on the applied reduction rule and show that $\mathcal{N}$ is continuous sound iff $\mathcal{N}'$ is continuous sound.

- *Rule R_1.* Suppose $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}' \not\to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$ in $\mathcal{N}'$. Let $\boldsymbol{m}$ be such that $\pi(\boldsymbol{m}) = \boldsymbol{m}'$ and $\boldsymbol{m}[p] = \sum_{r \in R'} \boldsymbol{m}[r']$. Then $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}$ and $\boldsymbol{m} \to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$ would imply $\boldsymbol{m}' \to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$, which is a contradiction. The converse implication is trivial.
- *Rules R_2 and R_3.* This is trivial because $\to^*_{\mathbb{Q}_{\geq 0}}$ is preserved.
- *Rules R_4 and R_5.* Suppose $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}' \not\to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$ in $\mathcal{N}'$. We have $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{Q}_{\geq 0}} \pi_0(\boldsymbol{m}')$. If $\pi_0(\boldsymbol{m}') \to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$ in $\mathcal{N}$ then, since $\{\mathsf{f}\colon 1\}[p] = 0$, we obtain $\boldsymbol{m}' \to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$ in $\mathcal{N}'$, which is a contradiction. The other implication is explicitly written in Proposition 1.
- *Rule R_6.* Suppose $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m} \not\to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$ in $\mathcal{N}$. We have $\boldsymbol{m} \to^*_{\mathbb{Z}} \boldsymbol{m}_1$ where $\boldsymbol{m}_1[p_1] = \sum_{i=1}^k \boldsymbol{v}_1[p_i]$ and $\boldsymbol{m}_1[p_i] = 0$ for $i > 1$. If $\pi(\boldsymbol{m}_1) \to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$, then $\boldsymbol{m}_1 \to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$, which is a contradiction. The other implication is trivial. □

Theorem 2. *Continuous soundness is coNP-complete. Moreover, coNP-hardness holds even if the underlying graph of the given workflow net is acyclic.*

Proof (of coNP-hardness). Recall that, in the main text, we have defined a workflow net $\mathcal{N}_\varphi$ from a formula in DNF, and claimed that $\mathcal{N}_\varphi$ is continuously sound iff φ is a tautology. It remains to show the implication from right to left.

$\Rightarrow$) Suppose φ is a tautology. Let us first make an observation. Consider some sequence $b_1, \ldots, b_m \in \{0, 1\}$, and the marking $\boldsymbol{m} = \{p_{i,b_i} : 1 \mid i \in [1..m]\}$. Since φ is a tautology, there exists a clause C_j that satisfies the assignment $x_i \coloneqq b_i$. Let $i_1, \ldots, i_\ell$ be the indices of variables not occurring in C_j. It is easy to see that

$$\{\mathsf{i}\colon i\} \to^{t_{\text{init}} v_{1,b_1} \cdots v_{m,b_m}} \boldsymbol{m} \to^{c_j \overline{v}_{i_1,b_{i_1}} \cdots \overline{v}_{i_\ell,b_{i_\ell}}} \{r_i : 1 \mid i \in [1..m]\} \to^{t_{\text{fin}}} \{\mathsf{f}\colon 1\}.$$

By [20, Lemma 12(1)], we rescale the continuous run, *i.e.* for all $\alpha \in (0, 1]$:

$$\{\mathsf{i}\colon \alpha\} \to^*_{\mathbb{Q}_{\geq 0}} \alpha \boldsymbol{m} \to^*_{\mathbb{Q}_{\geq 0}} \sum_{i=1}^m \{r_i\colon \alpha\} \to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon \alpha\}. \tag{1}$$

Let us establish some invariants. Let $A_i \coloneqq \{\mathsf{i}, p_{i,?}, p_{i,1}, p_{i,0}, r_i, \mathsf{f}\}$ and $B_i \coloneqq \{\mathsf{i}, p_{\mathrm{cl}}, q_i, r_i, \mathsf{f}\}$. First, for all transition $t \in T$ and all index $i \in [1..m]$, we have

$$\sum_{p \in A_i} {}^\bullet t[p] = \sum_{A_i} t^\bullet[p], \text{ and } \sum_{p \in B_i} {}^\bullet t[p] = \sum_{p \in B_i} t^\bullet[p].$$

We say that a marking $\boldsymbol{n}$ is *reachable* if $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{n}$. From the above invariants, it follows that every reachable marking $\boldsymbol{n}$ satisfies

$$\sum_{p \in A_i} \boldsymbol{n}[p] = 1 \text{ and } \sum_{p \in B_i} \boldsymbol{n}[p] = 1. \tag{2}$$

Note that, from Equation (2), every reachable marking $\boldsymbol{n}$ satisfies

$$\boldsymbol{n}[p_{i,?}] + \boldsymbol{n}[p_{i,1}] + \boldsymbol{n}[p_{i,0}] = \boldsymbol{n}[p_{\mathrm{cl}}] + \boldsymbol{n}[q_i]. \tag{3}$$

We further have this remaining invariant for all $t \in T$ and $i, j \in [1..m]$:

$${}^\bullet t[q_i] + {}^\bullet t[r_i] + t^\bullet[q_i] + t^\bullet[r_i] = {}^\bullet t[q_j] + {}^\bullet t[r_j] + t^\bullet[q_j] + t^\bullet[r_j].$$

Since all places q_i and r_i are empty in $\{\mathsf{i}\colon 1\}$, every reachable marking $\boldsymbol{n}$ satisfies:

$$\boldsymbol{n}[q_i] + \boldsymbol{n}[r_i] = \boldsymbol{n}[q_j] + \boldsymbol{n}[r_j]. \tag{4}$$

We are ready to prove continuous soundness. Let $\boldsymbol{n}$ be a reachable marking. By Equation (1), we can assume w.l.o.g. that $\boldsymbol{n}[\mathsf{i}] = 0$, as we can move α remaining token to f. Similarly, we can assume w.l.o.g. that $\boldsymbol{n}[p_{i,?}] = 0$ for all $i \in [1..m]$ as otherwise we can fire transition $v_{i,1}$ or $v_{i,0}$ properly scaled (the choice is irrelevant). Consequently, by Equation (3), we have $\boldsymbol{n}[p_{i,1}] + \boldsymbol{n}[p_{i,0}] \geq \boldsymbol{n}[q_i]$. Therefore, by firing transitions $\overline{v}_{i,0}$ and $\overline{v}_{i,1}$, scaled appropriately, we obtain $\boldsymbol{n} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{n}'$ with $\boldsymbol{n}'[q_i] = 0$ for all $i \in [1..m]$. By Equation (4), $\boldsymbol{n}'[r_i] = \boldsymbol{n}'[r_j]$ for all $i, j \in [1..m]$. Hence, by firing t_{fin} scaled by $\boldsymbol{n}'[r_1]$, we get $\boldsymbol{n}' \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{n}''$ where $\boldsymbol{n}''$ has zero token in each place, except possibly places $P'_{\mathrm{var}} \coloneqq \{p_{i,b} \mid i \in [1..m], b \in \{0,1\}\}$, place p_{cl} and place f.

Let us explain how to empty $P'_{\mathrm{var}} \cup \{p_{\mathrm{cl}}\}$, if this is not already the case. For each $i \in [1..m]$, among places $p_{i,1}$ and $p_{i,0}$, we write $p_{i,\max}$ and $p_{i,\min}$ so that $\boldsymbol{n}''[p_{i,\max}] \geq \boldsymbol{n}''[p_{i,\min}]$ (if they are equal, then the choice is not important). Let $S \coloneqq \{p_{i,\min} \mid i \in [1..m], \boldsymbol{n}''[p_{i,\min}] > 0\}$. We consider two cases.

Case 1: $S = \emptyset$. By the left part of Equation (2), and by Equation (3), the following holds for all $i, j \in [1..m]$:

$$\boldsymbol{n}''[p_{i,1}] + \boldsymbol{n}''[p_{i,0}] = \boldsymbol{n}''[p_{j,1}] + \boldsymbol{n}''[p_{j,0}] = \boldsymbol{n}''[p_{\mathrm{cl}}]. \tag{5}$$

Thus, there exist $\alpha \in (0,1]$ and $b_1, \ldots, b_m \in \{0,1\}$ such that $\boldsymbol{n}''[p_{\mathrm{cl}}] = \boldsymbol{n}''[p_{i,b_i}] = \alpha$ and $\boldsymbol{n}''[p_{i,\neg b_i}] = 0$ for all $i \in [1..m]$. Since φ is a tautology, we can fire some transition c_j scaled by α, which empties place p_{cl}, and consequently P'_{var} as well by Equation (5).

Case 2: $S \neq \emptyset$. Let $i \in [1..m]$ be such that $\boldsymbol{n}''[p_{i,\min}] > 0$ is minimal, and let $\alpha \coloneqq \boldsymbol{n}''[p_{i,\min}]$. Let $\boldsymbol{n}''' \coloneqq \{p_{\mathrm{cl}} \colon \alpha, p_{i,\min} \colon \alpha\} + \{p_{j,\max} \colon \alpha \mid j \neq i\}$. Note that $\boldsymbol{n}''' \leq \boldsymbol{n}''$. We can apply Equation (1) and obtain

$$\boldsymbol{n}'' = (\boldsymbol{n}'' - \boldsymbol{n}''') + \boldsymbol{n}''' \to^*_{\mathbb{Q}_{\geq 0}} (\boldsymbol{n}'' - \boldsymbol{n}''') + \{\mathsf{f} \colon \alpha\}.$$

Performing this operation decreases the size of S. Hence, it can be repeated at most m times until S becomes empty, which has been handled in case 1. □

A.3 Missing proofs of Section 5

Proposition 10. *Let $\mathcal{N}$ be a workflow net. It is the case that $k_{\mathcal{N},\mathbb{Z}} \leq k_{\mathcal{N},\mathbb{Q}_{\geq 0}} \leq k_{\mathcal{N}}$. Moreover, $k_{\mathcal{N},\mathbb{Z}}$ can be computed from an integer linear program $\mathcal{P}$; $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$ can be obtained by computing $\min k \in \mathbb{N}_{\geq 1} : \varphi(k)$ where φ is a formula from the existential fragment of mixed linear arithmetic φ, i.e. $\exists\mathsf{FO}(\mathbb{Q}, \mathbb{Z}, <, +)$; and both $\mathcal{P}$ and φ are constructible in polynomial time from $\mathcal{N}$.*

Proof. Let $\mathcal{N} = (P, T, F)$ be a workflow net. Let us first establish $k_{\mathcal{N},\mathbb{Z}} \leq k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$. Let $\pi = \lambda_1 t_1 \cdots \lambda_n t_n$ be a continuous run such that $\{\mathsf{i} \colon k\} \to^{\pi}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f} \colon k\}$ and $\boldsymbol{\pi} \in \mathbb{N}^T$. In particular, we have

$$\begin{aligned}
\{\mathsf{f} \colon k\} &= \{\mathsf{i} \colon k\} + \sum_{i \in [1..n]} \lambda_i \cdot \Delta(t_i) \\
&= \{\mathsf{i} \colon k\} + \sum_{t \in T} \sum_{i \in [1..n] : t_i = t} \lambda_i \cdot \Delta(t) \\
&= \{\mathsf{i} \colon k\} + \sum_{t \in T} \boldsymbol{\pi}[t] \cdot \Delta(t).
\end{aligned}$$

As $\boldsymbol{\pi} \in \mathbb{N}^T$, we obtain $\{\mathsf{i} \colon k\} \to^{\pi}_{\mathbb{Z}} \{\mathsf{f} \colon k\}$. Consequently, $k_{\mathcal{N},\mathbb{Z}} \leq k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$.

The inequality $k_{\mathcal{N},\mathbb{Q}_{\geq 0}} \leq k_{\mathcal{N}}$ follows immediately from the fact that $\{\mathsf{i} \colon k\} \to^{\pi} \{\mathsf{f} \colon k\}$ implies $\{\mathsf{i} \colon k\} \to^{\pi}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f} \colon k\}$ (with all scaling factors set to 1).

It remains to argue that $k_{\mathcal{N},\mathbb{Z}}$ and $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$ can be obtained as described. By definition of integer reachability, $k_{\mathcal{N},\mathbb{Z}}$ is the value obtained from this program:

$$\min k \text{ subject to } k \in \mathbb{N}_{\geq 1}, \boldsymbol{x} \in \mathbb{N}^T \text{ and } \{\mathsf{i} \colon k\} + \sum_{t \in T} \boldsymbol{x}[t] \cdot \Delta(t) = \{\mathsf{f} \colon k\}.$$

For $k_{\mathcal{N},\mathbb{Q}_{\geq 0}}$, we use the fact that there is polynomial-time constructible formula $\psi_{\mathcal{N}}$ from existential linear arithmetic such that $\psi(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{x})$ holds iff there is a continuous run π that satisfies $\boldsymbol{m} \to^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$ and $\boldsymbol{x} = \boldsymbol{\pi}$ [8]. So, it suffices to take

$$\varphi(k) \coloneqq \exists \boldsymbol{x} \in \mathbb{N}^T : \psi(\{\mathsf{i} \colon k\}, \{\mathsf{f} \colon k\}, \boldsymbol{x}).$$ □

A.4 Missing proofs of Section 6

Recall the following unproven lemma from the main text.

Lemma 2. *Let $\mathcal{N} = (P, T, F)$ be a free-choice Petri net, let $c \in \mathbb{N}_{\geq 1}$, and let $\boldsymbol{m} \in \mathbb{N}^P$. The following statements hold.*

1. *There exists a marking $\boldsymbol{m}'$ such that $\boldsymbol{m} \to^* \boldsymbol{m}'$ and $L(\boldsymbol{m}') = F(\boldsymbol{m}')$.*
2. *If $L(\boldsymbol{m}) = F(\boldsymbol{m})$, then $L(c \cdot \boldsymbol{m}) = F(c \cdot \boldsymbol{m}) = F(\boldsymbol{m})$.*
3. *If $L(c \cdot \boldsymbol{m}) = F(c \cdot \boldsymbol{m})$, $c \cdot \boldsymbol{m} \to^* \{\mathsf{f}\colon c\}$ and $(\mathcal{N}, c \cdot \boldsymbol{m})$ is bounded, then $\boldsymbol{m} = \{\mathsf{f}\colon 1\}$.*

For the sake of readability, we prove each item of Lemma 2 as its own lemma.

Lemma 4. *Let $\mathcal{N} = (P, T, F)$ be a free-choice Petri net, and let $\boldsymbol{m} \in \mathbb{N}^P$. It is the case that $\boldsymbol{m} \to^* \boldsymbol{m}'$ for some marking $\boldsymbol{m}'$ such that $L(\boldsymbol{m}') = F(\boldsymbol{m}')$.*

Proof. If $F(\boldsymbol{m}) = L(\boldsymbol{m})$ holds, then we are done by taking $\boldsymbol{m}' \coloneqq \boldsymbol{m}$. Otherwise, let $t \in F(\boldsymbol{m}) \setminus L(\boldsymbol{m})$. Since t is not live in $(\mathcal{N}, \boldsymbol{m})$, there exists a marking $\boldsymbol{m}' \in \mathbb{N}^P$ that satisfies $\boldsymbol{m} \to^* \boldsymbol{m}'$ and $t \notin F(\boldsymbol{m}')$. Therefore, we have $F(\boldsymbol{m}') \subseteq F(\boldsymbol{m}) \setminus \{t\}$. This means that $|F(\boldsymbol{m}')| < |F(\boldsymbol{m})$. Since $L(\boldsymbol{m}') \subseteq F(\boldsymbol{m}')$, we can repeat this argument (up to $|T|$ times) until obtaining $L(\boldsymbol{m}') = F(\boldsymbol{m}')$. □

For a run σ, let us define $\boldsymbol{\sigma}\colon T \to \mathbb{N}$, where for each $t \in T$, $\boldsymbol{\sigma}[t]$ is the number of times t occurs in σ.

Lemma 5. *Let $\mathcal{N}$ be a free-choice workflow net, let $c \in \mathbb{N}_{\geq 1}$, and let $\boldsymbol{m} \in \mathbb{N}^P$ be such that $L(\boldsymbol{m}) = F(\boldsymbol{m})$. It is the case that $L(c \cdot \boldsymbol{m}) = F(c \cdot \boldsymbol{m}) = F(\boldsymbol{m})$.*

Proof. We first show that $F(c \cdot \boldsymbol{m}) = F(\boldsymbol{m})$, and then that $L(c \cdot \boldsymbol{m}) = F(c \cdot \boldsymbol{m})$.

We trivially have $F(c \cdot \boldsymbol{m}) \supseteq F(\boldsymbol{m})$. For the sake of contradiction, suppose there exists a transition $t \in F(c \cdot \boldsymbol{m})$ such that $t \notin F(\boldsymbol{m})$. Let σ_1 be a run such that $c \cdot \boldsymbol{m} \to^{\sigma_1 t}$. Without loss of generality, we may assume that $\boldsymbol{\sigma_1} \subseteq F(\boldsymbol{m})$. Indeed, if there is some $t' \in \boldsymbol{\sigma_1}$ such that $t' \notin F(\boldsymbol{m})$, then we can shorten σ_1 and take the shortened run which enables t' instead.

Let $\sigma_1 = t_1 t_2 \cdots t_n$. Recall that $t_i \in L(\boldsymbol{m}) = F(\boldsymbol{m})$ for each t_i, that is, from any marking reachable from $\boldsymbol{m}$, we can reach a marking that enables t_i. Therefore, we can define a run $\sigma_2 \coloneqq \phi_1 t_1 \phi_2 t_2 \cdots \phi_n t_n$, where ϕ_i is a run from $\boldsymbol{m} + \Delta(\phi_1 t_1 \cdots \phi_{i-1} t_{i-1})$ that enables t_i.

If there exists a transition s in the run σ_2 such that $\mathrm{supp}({}^\bullet s) \cap \mathrm{supp}({}^\bullet t) \neq \emptyset$, then ${}^\bullet s = {}^\bullet t$ as $\mathcal{N}$ is free-choice. Hence, since $s \in F(\boldsymbol{m})$, we obtain $t \in F(\boldsymbol{m})$, which is a contradiction. Thus no transition in σ_2 can consume tokens from places in $\mathrm{supp}({}^\bullet t)$. Since $c \cdot \boldsymbol{m} \to^{\sigma_1 t}$, we know that

$$\mathrm{supp}({}^\bullet t) \subseteq \mathrm{supp}(c \cdot \boldsymbol{m}) \cup \bigcup_{i=1}^{n} \mathrm{supp}(t_i^\bullet) = \mathrm{supp}(\boldsymbol{m}) \cup \bigcup_{i=1}^{n} \mathrm{supp}(t_i^\bullet).$$

Altogether, this means that the transitions t_i put enough tokens such that all places in $\mathrm{supp}({}^\bullet t)$ are marked, and that σ_2 cannot consume any of these tokens. Therefore, $\boldsymbol{m} \to^{\sigma_2 t}$, which is a contradiction.

It remains to prove that $L(c \cdot \boldsymbol{m}) = F(c \cdot \boldsymbol{m})$. We have $L(\boldsymbol{m}) \subseteq L(c \cdot \boldsymbol{m}) \subseteq F(c \cdot \boldsymbol{m})$. Since $L(\boldsymbol{m}) = F(\boldsymbol{m}) = F(c \cdot \boldsymbol{m})$, these inclusions are in fact equalities, and we are done. □

Lemma 6. *Let $\mathcal{N} = (P, F, T)$ be a free-choice workflow net, let $c \in \mathbb{N}_{\geq 1}$ and let $\boldsymbol{m} \in \mathbb{N}^P$ be such that $L(c \cdot \boldsymbol{m}) = F(c \cdot \boldsymbol{m})$. If $c \cdot \boldsymbol{m} \to^* \{\mathsf{f}\colon c\}$ and $(\mathcal{N}, c \cdot \boldsymbol{m})$ is bounded, then $\boldsymbol{m} = \{\mathsf{f}\colon 1\}$.*

Proof. Recall that no transition of a workflow net consumes from f, *i.e.* ${}^\bullet t[\mathsf{f}] = 0$ for all $t \in T$. Thus, we either have $\boldsymbol{m}[\mathsf{f}] = 0$ or $\boldsymbol{m}[\mathsf{f}] = 1$.

If $\boldsymbol{m}[\mathsf{f}] = 0$, then there is some transition $t \in F(c \cdot \boldsymbol{m})$ such that $f \in t^\bullet[\mathsf{f}] > 0$. Since $t \in L(c \cdot \boldsymbol{m})$, it follows that from $c \cdot \boldsymbol{m}$, we can reach $\boldsymbol{m}'$ with $\boldsymbol{m}'[\mathsf{f}]$ abritrarily large, as t puts a token into f and can be fired arbitrarily often from $c \cdot \boldsymbol{m}$. This contradicts the fact that $(\mathcal{N}, c \cdot \boldsymbol{m})$ is bounded. Hence, $\boldsymbol{m}[\mathsf{f}] = 1$. We can write $\boldsymbol{m}$ as $\boldsymbol{m} = \{\mathsf{f}\colon 1\} + \boldsymbol{m}'$ where $\boldsymbol{m}'[\mathsf{f}] = 0$. We have $c \cdot \boldsymbol{m} = \{\mathsf{f}\colon c\} + c \cdot \boldsymbol{m}'$. If $\boldsymbol{m}' = \boldsymbol{0}$, then we are done. Otherwise, we obtain a contradiction. Indeed, it cannot be the case that $\{\mathsf{f}\colon c\} + c \cdot \boldsymbol{m}' \to^* \{\mathsf{f}\colon c\}$, as every transition of a workflow net produces at least one token (and none consumes from f). □

Lemma 3. *Let $\mathcal{N}$ be a workflow net. If $\mathcal{N}$ is continuously sound, then $(\mathcal{N}, \{\mathsf{i}\colon k\})$ is bounded for all $k \in \mathbb{N}_{\geq 1}$.*

Proof. Assume for contradiction that there exists $k \in \mathbb{N}_{\geq 1}$ such that $(\mathcal{N}, \{\mathsf{i}\colon k\})$ is unbounded, but $\mathcal{N}$ is continuously sound. There exist marking $\boldsymbol{m}$ and $\boldsymbol{m}' > \boldsymbol{m}$ such that $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m} \to^* \boldsymbol{m}'$. By Lemma 1, we have $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{n} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{n}'$, where $\boldsymbol{n} \coloneqq (1/k) \cdot \boldsymbol{m}$ and $\boldsymbol{n}' \coloneqq (1/k) \cdot \boldsymbol{m}'$. As $\mathcal{N}$ is continuously sound, it must hold that $\boldsymbol{n} \to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$. It follows that

$$\boldsymbol{n}' = \boldsymbol{n} + (\boldsymbol{n}' - \boldsymbol{n}) \to^*_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\} + (\boldsymbol{n}' - \boldsymbol{n}).$$

This contradicts the assumption that $\mathcal{N}$ is continuously sound, as each transition of a workflow net produces at least one token, and none consumes from f. □

A.5 Missing definition of the arc weight encoding of Section 7

Recall that under our definition, Petri nets do not have arc weights as $F\colon ((P \times T) \cup (T \times P)) \to \{0, 1\}$. Petri nets with arc weights are defined exactly as Petri nets but with $F\colon ((P \times T) \cup (T \times P)) \to \mathbb{N}$. An example of the arc weight encoding described in the main text is shown in Figure 7.

In this section, we will use t^{-1} to denote the reverse transition of transition t, as done in the coNP membership proof of Theorem 2.

Formally, to simulate a transition t, we add places $P_{p,t}$ and transitions $T_{p,t}$ for each place p with $b \coloneqq {}^\bullet t[p] > 1$, and places $P'_{p,t}$ and transitions $T'_{p,t}$ for each place p with $b' \coloneqq t^\bullet[p] > 1$.

From now on, when we define a transition t, we assume that ${}^\bullet t[p'] = 0$ and $t^\bullet[p'] = 0$ for each place p' except those given explicitly. We define $P_{p,t}$ as follows. We denote by $b_1, b_2, \ldots, b_n$ the binary representation of b, that is, $b = \sum_{i=1}^{n} b_i \cdot 2^{i-1}$, and similarly $b'_1, b'_2, \ldots, b'_{n'}$ for b'. The set $P_{p,t}$ consists of $2n - 1$ places. For every $i \in [1..n-1]$, we add two places l_i and r_i; and an additional place r_n. The set $T_{p,t}$ contains the following transitions:

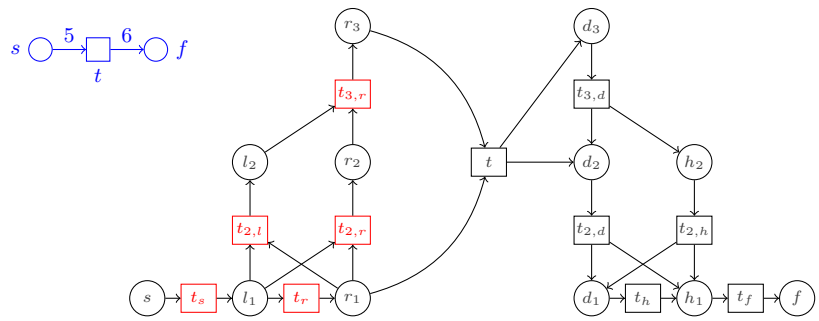


Fig. 7. *Top left (in blue):* A Petri net $\mathcal{N}$ with arc weights. *Center:* A Petri net $\mathcal{N}_{\text{enc}}$ without arc weights that simulates behaviour of $\mathcal{N}$. For each transition colored in red, the reverse transition is also part of $\mathcal{N}_{\text{enc}}$, and is merely not drawn to avoid overcrowding the figure. For ease of presentation, places and transitions of $\mathcal{N}_{\text{enc}}$ contain their names (not values).

- t_p, where ${}^\bullet t_p[p] = t_p^\bullet[l_1] \coloneqq 1$;
- t_r, as well as its reverse t_r^{-1}, where ${}^\bullet t_r[l_1] = t_r^\bullet[r_1] \coloneqq 1$;
- for each $i \in [2..n-1]$ the transitions $t_{i,l}$, $t_{i,r}$ and their reverses $t_{i,l}^{-1}$, $t_{i,r}^{-1}$, where ${}^\bullet t_{i,l}[r_{i-1}] = {}^\bullet t_{i,l}[l_{i-1}] \coloneqq 1$, ${}^\bullet t_{i,r} = {}^\bullet t_{i,l}$, and $t_{i,l}^\bullet[l_i] = t_{i,r}^\bullet[r_i] \coloneqq 1$,
- the transition $t_{n,r}$ and its reverse $t_{n,r}^{-1}$, where ${}^\bullet t_{n,r}[l_{n-1}] = {}^\bullet t_{n,r}[r_{n-1}] \coloneqq 1$ and $t_{n,r}^\bullet[r_n] \coloneqq 1$.

We further redefine t to have ${}^\bullet t[p] \coloneqq 0$ and ${}^\bullet t[r_i] \coloneqq 1$ for all i such that $b_i = 1$.

The set $P'_{p,t}$ consists of $2n'-1$ places. We have d_i and h_i for each $i \in [1..n'-1]$, and an additional place d'_n. The set $T'_{p,t}$ contains the following transitions:

- t_p, where ${}^\bullet t_p[h_1] = t_p^\bullet[p] \coloneqq 1$,
- $t_{1,h}$, where ${}^\bullet t_{1,h}[d_1] = t_{1,h}^\bullet[h_1] \coloneqq 1$,
- for each $i \in [2..n-1]$, the transitions $t_{i,d}$ and $t_{i,h}$, where ${}^\bullet t_{i,d}[d_i] = {}^\bullet t_{i,h}[h_i] \coloneqq 1$, $t_{i,d}^\bullet[d_{i-1}] = t_{i,d}^\bullet[h_{i-1}] \coloneqq 1$, and $t_{i,h}^\bullet \coloneqq t_{i,d}^\bullet$.

We further redefine t to have $t^\bullet[p] \coloneqq 0$ and $t^\bullet[d_i] \coloneqq 1$ for each i such that $b'_i = 1$.

Given a Petri net $\mathcal{N} = (P, T, F)$, let us denote by $\mathcal{N}_{\text{enc}} = (P', T', F')$ the transformed $\mathcal{N}$ where all transitions with arc weights are modified by the gadget defined above. To avoid any confusion, we denote markings in $\mathcal{N}$ as $\boldsymbol{m}$ and $\boldsymbol{m}'$, and markings in $\mathcal{N}_{\text{enc}}$ as $\boldsymbol{n}$ and $\boldsymbol{n}'$. As $\mathcal{N}_{\text{enc}}$ does not remove (but only adds) places, we may treat markings on $\mathcal{N}$ as markings on $\mathcal{N}_{\text{enc}}$, where all places in $P' \setminus P$ are marked with zero token.

Recall that $\boldsymbol{\sigma}$ is a vector mapping each transition t to the number of times t is used in run σ. In the following, let $p \in P$ and $t \in T$ be such that ${}^\bullet t[p] = b \geq 1$. Let $b_1, \ldots, b_n$ be the binary representation of b. Furthermore, let $P_{p,t}$ and $T_{p,t}$ be defined as above.

We are ready to state some helpful lemmas.

Lemma 7. *Let $i \in [1..n]$. We have $\{p\colon 2^{i-1}\} \to^\sigma \{r_i\colon 1\}$ in $\mathcal{N}_{\text{enc}}$ with $\operatorname{supp}(\boldsymbol{\sigma}) \subseteq T_{p,t}$. Further, if $i < n$, then $\{p\colon 2^{i-1}\} \to^{\sigma'} \{l_i\colon 1\}$ in $\mathcal{N}_{\text{enc}}$ with $\operatorname{supp}(\boldsymbol{\sigma'}) \subseteq T_{p,t}$.*

Proof. We proceed by induction. For $i = 1$, we have

$$\{p\colon 2^{1-1}\} = \{p\colon 1\} \to^{t_p} \{l_1\colon 1\} \to^{t_r} \{r_1\colon 1\}.$$

For $i > 1$, we have $\{p\colon 2^{i-1}\} = \{p\colon 2^{i-2} + 2^{i-2}\} \to^* \{r_{i-1}\colon 1, l_{i-1}\colon 1\}$ by the induction hypothesis. Thus, we have $\{r_{i-1}\colon 1, l_{i-1}\colon 1\} \to^{t_{i,r}} \{r_i\colon 1\}$. If $i < n$, then we additionally have $\{r_{i-1}\colon 1, l_{i-1}\colon 1\} \to^{t_{i,l}} \{l_i\colon 1\}$. We conclude the proof by pointing out that for all i, $t_p, t_r, t_{i,r}, t_{i,l} \in T_{p,t}$. □

The proof of the lemma below follows by the fact that all transitions of $T_{p,t}$ are reversible.

Lemma 8. *Let $i \in [1..n]$. We have $\{r_i\colon 1\} \to^{\sigma} \{p\colon 2^{i-1}\}$ in $\mathcal{N}_{\text{enc}}$ with $\text{supp}(\boldsymbol{\sigma}) \subseteq T_{p,t}$. Further, if $i < n$, then $\{l_i\colon 1\} \to^{\sigma'} \{p\colon 2^{i-1}\}$ in $\mathcal{N}_{\text{enc}}$ with $\text{supp}(\boldsymbol{\sigma'}) \subseteq T_{p,t}$.*

For the next lemma, let $p \in P$ and $t \in T$ be such that $t^{\bullet}[p] = b \geq 1$. Let $b_1, \ldots, b_n$ be the binary representation of b. Let $P'_{p,t}$ and $T'_{p,t}$ be as defined above.

Lemma 9. *Let $i \in [1..m]$. We have $\{d_i\colon 1\} \to^{\sigma} \{p\colon 2^{i-1}\}$ in $\mathcal{N}_{\text{enc}}$ with $\text{supp}(\boldsymbol{\sigma}) \subseteq T'_{p,t}$. Further, if $i < n$, then $\{h_i\colon 1\} \to^{\sigma'} \{p\colon 2^{i-1}\}$ in $\mathcal{N}_{\text{enc}}$ with $\text{supp}(\boldsymbol{\sigma'}) \subseteq T'_{p,t}$.*

Proof. We proceed by induction on i. If $i = 1$, then we have $2^{i-1} = 1$ and hence $\{d_1\colon 1\} \to^{t_h} \{h_1\colon 1\} \to^{t_p} \{p\colon 1\}$.

For $i > 1$, we have $\{d_i\colon 1\} \to^{t_{i,d}} \{d_{i-1}\colon 1, h_{i-1}\colon 1\}$. If $i < n$, then we additionally have $\{h_i\colon 1\} \to^{t_{i,h}} \{d_{i-1}\colon 1, h_{i-1}\colon 1\}$. It follows from the induction hypothesis that $\{d_{i-1}\colon 1, h_{i-1}\colon 1\} \to^{\sigma\sigma'} \{p\colon 2^{i-2} + 2^{i-2}\} = \{p\colon 2^{i-1}\}$. We conclude by pointing out that, for all i, we have $t_p, t_h, t_{i,d}, t_{i,h} \in T'_{p,t}$. □

Definition 1. *Let $U \subseteq T$. A vector $\boldsymbol{x}\colon P \to \mathbb{Q}$ is a* place invariant *over U if the following holds for all $t \in U$:*

$$\sum_{p \in P} {}^{\bullet}t[p] \cdot \boldsymbol{x}[p] = \sum_{p \in P} t^{\bullet}[p] \cdot \boldsymbol{x}[p]. \tag{6}$$

Proposition 11 (adapted from [16, Prop. 2.27]). *Let $U \subseteq T$ and let $\boldsymbol{x}$ be a place invariant over U. If $\boldsymbol{m} \to^{\sigma} \boldsymbol{m'}$ with $\text{supp}(\boldsymbol{\sigma}) \subseteq U$, then $\boldsymbol{x} \cdot \boldsymbol{m} = \boldsymbol{x} \cdot \boldsymbol{m'}$.*

Let us define the vector $I_{p,t}$ with $I_{p,t}[p] \coloneqq 1$, $I_{p,t}[r_i] \coloneqq 2^{i-1}$ and $I_{p,t}[l_i] \coloneqq 2^{i-1}$, where r_i and l_i are the places previously defined in $P_{p,t}$. It is easy to see that $I_{p,t}$ is a place invariant of $T_{p,t}$.

Let $R \coloneqq \{t \in T \mid {}^{\bullet}t[p] \geq 2\}$ and $S \coloneqq \{t \in T \mid t^{\bullet}[p] \geq 2\}$. We further define the vector $I_p\colon \{p\} \cup \bigcup_{t \in R} P_{p,t} \cup \bigcup_{t \in S} P'_{p,t} \to \mathbb{Q}$, where $P_{p,t} = \emptyset$ if ${}^{\bullet}t[p] \leq 1$ and $P'_{p,t} = \emptyset$ if $t^{\bullet}[p] \leq 1$. We define $I_p[p] \coloneqq 1$ and $I_p[p'] \coloneqq 2^{i-1}$ if $p' \in \{r_i, l_i, d_i, h_i\}$ for some i. Note that this is well-defined by our choice of domain of I_p. It is easy to convince oneself that I_p is a place invariant of $T' \setminus T$.

We introduce some notation. For a transition $t \in T$, let $G \coloneqq \{p \in P \mid {}^{\bullet}t[p] \geq 2\}$ and $H \coloneqq \{p \in P \mid t^{\bullet}[p] \geq 2\}$. For a place $p \in G$, we write $b(p) \coloneqq {}^{\bullet}t[p]$. For $i \in \mathbb{N}$, we write $n(i)$ to denote the number of bits in the binary representation of

i. Let $b_1(p), \ldots, b_{n(b(p))}(p)$ denote the bits of the binary representation of $b(p)$. Let $r_i(p)$ denote the place r_i in $P_{p,t}$. Similarly, given $p \in H$, we write $c(p) \coloneqq t^\bullet[p]$, we let $c_1(p), \ldots, c_{n(c(p))}(p)$ be the bits of the binary representation of $c(p)$, and we further write $d_i(p)$ to denote the place d_i of $P'_{p,t}$. In the following, we denote by t the transition in $\mathcal{N}$, and by t' the corresponding transition in $\mathcal{N}_{\text{enc}}$.

Lemma 10. *Let $t \in T$ and let $\boldsymbol{m}, \boldsymbol{m}'$ be markings of $\mathcal{N}$ with $\boldsymbol{m}' = \boldsymbol{m} + \Delta(t)$. It holds that $\boldsymbol{m} \to^t \boldsymbol{m}'$ in $\mathcal{N}$ iff $\boldsymbol{m} \to^{\pi t' \pi'} \boldsymbol{m}'$ in $\mathcal{N}_{\text{enc}}$, where* $\text{supp}(\boldsymbol{\pi}) \subseteq \bigcup_{p \in G} T_{p,t}$ *and* $\text{supp}(\boldsymbol{\pi'}) \subseteq \bigcup_{p \in H} T'_{p,t}$.

Proof. $\Rightarrow$) By definition of $\mathcal{N}_{\text{enc}}$, $\boldsymbol{m}[p] \geq {}^\bullet t'[p]$ for all $p \in P \setminus G$. By definition of $\mathcal{N}_{\text{enc}}$, it holds that ${}^\bullet t'[r_i(p)] = b_i(p)$ for all $i \in [1..n(b(p))]$. Note that $\boldsymbol{m}[p] \geq b(p) = \sum_{i=1}^{n(b(p))} 2^{i-1} \cdot b_i(p)$. Thus, it follows from Lemma 7 that $\{p\colon b(p)\} \to^\sigma \sum_{i=1}^{n(b(p))} \{r_i(p)\colon b_i(p)\}$. So, in particular,

$$\boldsymbol{m}[p] \to^\sigma \boldsymbol{m} - \{p\colon b(p)\} + \sum_{i=1}^{n(b(p))} \{r_i(p)\colon b_i(p)\}.$$

Since the transitions in σ do not have an effect on places other than $P_{p,t} \cup \{p\}$, we can invoke Lemma 7 individually for each $p \in G$, and thus obtain

$$\boldsymbol{m} \to^{\sigma_1 \cdots \sigma_{|G|}} \boldsymbol{m} + \sum_{p \in G} \sum_{i=1}^{n(b(p))} \{r_i(p)\colon b_i(p)\} - \{p\colon b(p)\},$$

where $\text{supp}(\boldsymbol{\sigma_1}), \ldots, \text{supp}(\boldsymbol{\sigma_{|G|}}) \subseteq \bigcup_{p \in G} T_{p,t}$. By definition, t' is enabled in this marking and its firing leads to

$$\begin{aligned}\boldsymbol{m} - \sum_{p \in G} \{p\colon b(p)\} - \sum_{p \in P \setminus G} \{p\colon {}^\bullet t[p]\} + \sum_{p \in P \setminus H} \{p\colon t^\bullet[p]\} + \sum_{p \in H} \sum_{i=1}^{n(c(p))} \{d_i(p)\colon c_i(p)\} \\ = \boldsymbol{m} - {}^\bullet t + \sum_{p \in P \setminus H} \{p\colon t^\bullet[p]\} + \sum_{p \in H} \sum_{i=1}^{n(c(p))} \{d_i(p)\colon c_i(p)\}.\end{aligned}$$

Let us denote the latter marking as $\boldsymbol{m}'$. By invoking Lemma 9 individually on each $d_i(p)$, it follows that for each $p \in H$:

$$\begin{aligned}\boldsymbol{m}' \to^{\sigma'_1 \cdots \sigma'_{|H|}} \boldsymbol{m} - {}^\bullet t + \sum_{p \in P \setminus H} \{p\colon t^\bullet[p]\} + \sum_{p \in H} \sum_{i=1}^{n(c(p))} \{p\colon 2^{i-1} c_i(p)\} = \\ \boldsymbol{m} - {}^\bullet t + \sum_{p \in P \setminus H} \{p\colon t^\bullet[p]\} + \sum_{p \in H} \{p\colon t^\bullet[p]\} = \\ \boldsymbol{m} - {}^\bullet t + t^\bullet = \boldsymbol{m} + \Delta(t).\end{aligned}$$

We conclude this direction by noting that $\text{supp}(\boldsymbol{\sigma_1}), \ldots, \text{supp}(\boldsymbol{\sigma_{|H|}}) \subseteq \bigcup_{p \in H} T'_{p,t}$ by Lemma 9.

$\Leftarrow$) We have $\boldsymbol{m} \to^{\sigma t' \sigma'} \boldsymbol{m}'$. Let us denote by $\boldsymbol{m}_1$ the marking such that $\boldsymbol{m} \to^{\sigma} \boldsymbol{m}_1$. It must be the case that

$$\boldsymbol{m}_1 \geq {}^{\bullet}t' = \sum_{p \in P \setminus G} {}^{\bullet}t[m] + \sum_{p \in G} \sum_{i \in n(b(p))} \{r_i(p) \colon b_i(p)\}.$$

Recall that for each $p \in G$, $I_{p,t}$ is a place invariant of $T_{p,t}$. In particular, among transitions from $T' \setminus T$, places in $\{p\} \cup P_{p,t}$ are only affected by transitions in $T_{p,t}$. So, $I_{p,t} \cdot \boldsymbol{m} = I_{p,t} \cdot \boldsymbol{m}_1$ by Proposition 11. Since $\boldsymbol{m}_1 \geq \sum_{i \in n(b(p))} \{r_i(p) \colon b_i(p)\}$, we have $I_{p,t} \cdot \boldsymbol{m}_1 \geq \sum_{i \in n(b(p))} 2^{i-1} b_i(p)$. Thus, the same must hold for $I_{p,t} \cdot \boldsymbol{m}$. But among places in $\{p\} \cup P_{p,t}$, $\boldsymbol{m}$ marks only p, as it is (by projection) a marking of $\mathcal{N}$. Since $I_{p,t}[p] = 1$, it must hold that $\boldsymbol{m}[p] \geq \sum_{i \in n(b(p))} 2^{i-1} b_i(p) = b(p)$, where the last equality follows from the fact that $b_1(p), \ldots, b_{n(b(p))}(p)$ is the binary representation of $b(p)$. So, $\boldsymbol{m}[p] \geq {}^{\bullet}t[p]$ holds by definition of $b(p)$. Therefore, $\boldsymbol{m}$ enables t, and consequently $\boldsymbol{m} \to^{t} \boldsymbol{m} + \Delta(t) = \boldsymbol{m}'$, and we are done. □

Lemma 11. *Let $\boldsymbol{m}, \boldsymbol{m}'$ be markings of $\mathcal{N}$. If $\boldsymbol{m} \to^{\sigma} \boldsymbol{m}'$ in $\mathcal{N}_{\text{enc}}$ and $\text{supp}(\boldsymbol{\sigma}) \subseteq T' \setminus T$, then $\boldsymbol{m} = \boldsymbol{m}'$.*

Proof. We argue for each place $p \in P$ individually that $\boldsymbol{m}[p] = \boldsymbol{m}'[p]$.

Recall that I_p is a place invariant over $T' \setminus T$. Therefore, $I_p \cdot \boldsymbol{m} = I_p \cdot \boldsymbol{m}'$ by Proposition 11. Note also that in the domain of I_p, the only place in P is p. Since $\boldsymbol{m}$ and $\boldsymbol{m}'$ are markings of $\mathcal{N}$, and consequently all places in the domain of I_p other than p must be unmarked, it follows that $I_p[p] \cdot \boldsymbol{m}[p] = I_p[p] \cdot \boldsymbol{m}'[p]$. Thus, $\boldsymbol{m}[p] = \boldsymbol{m}'[p]$. □

Lemma 12. *Let $\boldsymbol{m}, \boldsymbol{m}'$ be markings of $\mathcal{N}$. If $\boldsymbol{m} \to^{*} \boldsymbol{m}'$ in $\mathcal{N}_{\text{enc}}$, then $\boldsymbol{m} \to^{*} \boldsymbol{m}'$ in $\mathcal{N}$.*

Proof. Let $\boldsymbol{m} \to^{\pi} \boldsymbol{m}'$. If $\text{supp}(\boldsymbol{\pi}) \subseteq T' \setminus T$, then $\boldsymbol{m} = \boldsymbol{m}'$ by Lemma 11, and we are done. So, assume that $t \in T$ for some $t \in \pi$. We factor run π so that $\pi = \sigma_1 t_1 \sigma'_1 \cdots \sigma_n t_n \sigma'_n$ with $t_1, \ldots, t_n \in T$ and

$$\text{supp}(\boldsymbol{\sigma_1}), \text{supp}(\boldsymbol{\sigma'_1}), \ldots, \text{supp}(\boldsymbol{\sigma_n}), \text{supp}(\boldsymbol{\sigma'_n}) \subseteq T \setminus T'.$$

It follows from Lemma 10 that $\{\mathsf{i} \colon 1\} \to^{t_1 \cdots t_n} \{\mathsf{f} \colon k\}$. □

Proposition 12. *For any workflow net $\mathcal{N}$ and any $k \in \mathbb{N}_{\geq 1}$, $\mathcal{N}$ is k-quasi-sound iff $\mathcal{N}_{\text{enc}}$ is k-quasi-sound.*

Proof. This follows immediately from Lemmas 10 and 12. □

Proposition 13. *For any workflow net $\mathcal{N}$ and any $k \in \mathbb{N}_{\geq 1}$, $\mathcal{N}$ is k-sound iff $\mathcal{N}_{\text{enc}}$ is k-sound.*

Proof. $\Rightarrow$) Assume $\mathcal{N}$ is k-sound. Let $\boldsymbol{m}$ be a marking of $\mathcal{N}_{\text{enc}}$ such that $\{\mathsf{i} \colon k\} \to^{*} \boldsymbol{m}$ in $\mathcal{N}_{\text{enc}}$. If $\boldsymbol{m}$ is also a marking of $\mathcal{N}$, then $\{\mathsf{i} \colon k\} \to^{*} \boldsymbol{m}$ in $\mathcal{N}$ by Lemma 12. Thus, $\boldsymbol{m} \to^{*} \{\mathsf{f} \colon k\}$ in $\mathcal{N}$ by k-soundness, and $\boldsymbol{m} \to^{*} \{\mathsf{f} \colon k\}$ in $\mathcal{N}_{\text{enc}}$ by Lemma 10. If $\boldsymbol{m}$ is not a marking on $\mathcal{N}$, then, for each place $p \in P' \setminus P$,

we can invoke Lemmas 8 and 9 in order to obtain a marking $\boldsymbol{m}'$ which marks only places in P. So, we have $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m} \to^* \boldsymbol{m}'$ in $\mathcal{N}_{\text{enc}}$, and it follows by Lemma 12 that $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m}'$ in $\mathcal{N}$. Thus, $\boldsymbol{m}' \to^* \{\mathsf{f}\colon k\}$ in $\mathcal{N}$ by k-soundness, and $\boldsymbol{m}' \to^* \{\mathsf{f}\colon k\}$ in $\mathcal{N}_{\text{enc}}$ by Lemma 10, which shows that $\mathcal{N}_{\text{enc}}$ is k-sound.

$\Leftarrow$) Assume $\mathcal{N}_{\text{enc}}$ is k-sound. Let $\boldsymbol{m}$ be a marking of $\mathcal{N}$ such that $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m}$ in $\mathcal{N}$. If follows from Lemma 10 that $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m}$ in $\mathcal{N}_{\text{enc}}$. By k-soundness of $\mathcal{N}_{\text{enc}}$, we have $\boldsymbol{m} \to^* \{\mathsf{f}\colon k\}$ in $\mathcal{N}_{\text{enc}}$. Thus, $\boldsymbol{m} \to^* \{\mathsf{f}\colon k\}$ in $\mathcal{N}$ by Lemma 12. □

A.6 Missing proofs of Section 7

Let us prove the properties claimed about the instances of Figure 4.

Proposition 14. *It is the case that*

1. *$\mathcal{N}_c$ is c-unsound and k-sound for all $k \in [1..c-1]$.*
2. *$\mathcal{N}_{sound\text{-}c}$ is kc-sound for all $k \in \mathbb{N}_{\geq 1}$,*
3. *$\mathcal{N}_{\neg quasi\text{-}c}$ is not structurally quasi-sound, and*
4. *$\mathcal{N}_{\neg sound\text{-}c}$ is (mc)-quasi-sound for all $m \in \mathbb{N}_{\geq 1}$, not k-quasi-sound for any other number $k \in \mathbb{N}_{\geq 1}$, and not structurally sound.*

Proof. Items 2 and 3. They follow from the definitions of the unique transition.

Item 1. We first focus on k-soundness. Let $k \in [1..c-1]$ and let $\boldsymbol{m}$ be a marking such that $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m}$. We must show that $\boldsymbol{m} \to^* \{\mathsf{f}\colon k\}$.

Recall the definition of a place invariant from Definition 1.

Let $\boldsymbol{x}[\mathsf{i}] \coloneqq c+1$, $\boldsymbol{x}[p] \coloneqq 1$, $\boldsymbol{x}[r] \coloneqq c$ and $\boldsymbol{x}[\mathsf{f}] \coloneqq c+1$. It is readily seen that $\boldsymbol{x}$ is a place invariant. Recall Proposition 11: for any two markings $\boldsymbol{n}$ and $\boldsymbol{n}'$, if $\boldsymbol{n} \to^* \boldsymbol{n}'$, then $\boldsymbol{x} \cdot \boldsymbol{n} = \boldsymbol{x} \cdot \boldsymbol{n}'$. Since $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m}$, we have $\boldsymbol{x} \cdot \{\mathsf{i}\colon k\} = (c+1) \cdot k = \boldsymbol{x} \cdot \boldsymbol{m}$.

From marking $\boldsymbol{m}$, transition t_{i} can be fired $\boldsymbol{m}[\mathsf{i}]$ times, which leads to marking

$$\boldsymbol{m}_1 \coloneqq \{p\colon \boldsymbol{m}[p] + (c+1) \cdot \boldsymbol{m}[\mathsf{i}], r\colon \boldsymbol{m}[r], \mathsf{f}\colon \boldsymbol{m}[\mathsf{f}]\}.$$

From $\boldsymbol{m}_1$, transition t_r can be fired $\boldsymbol{m}_1[\mathsf{i}] \div c$ times, which leads to marking

$$\boldsymbol{m}_2 \coloneqq \{p\colon \boldsymbol{m}_1[p] \bmod c, r\colon \boldsymbol{m}_1[r] + \boldsymbol{m}_1[p] \div c, \mathsf{f}\colon \boldsymbol{m}_1[\mathsf{f}]\}.$$

Recall that from place invariant $\boldsymbol{x}$, we have

$$(c+1) \cdot k = (c+1) \cdot \boldsymbol{m}[\mathsf{i}] + \boldsymbol{m}[p] + c \cdot \boldsymbol{m}[r] + (c+1) \cdot \boldsymbol{m}[\mathsf{f}].$$

By reorganizing this equation, we obtain

$$\boldsymbol{m}[p] + \boldsymbol{m}[i] = (c+1)(k - \boldsymbol{m}[\mathsf{f}]) - c \cdot (\boldsymbol{m}[\mathsf{i}] + \boldsymbol{m}[r]). \tag{7}$$

This means that

$$\begin{aligned}
\boldsymbol{m}_2[p] &= \boldsymbol{m}_1[p] \bmod c && \text{(by def. of } \boldsymbol{m}_2\text{)} \\
&= (\boldsymbol{m}[p] + (c+1) \cdot \boldsymbol{m}[i]) \bmod c && \text{(by def. of } \boldsymbol{m}_1\text{)} \\
&= (\boldsymbol{m}[p] + \boldsymbol{m}[i]) \bmod c \\
&= k - \boldsymbol{m}[\mathsf{f}] && \text{(by (7))}.
\end{aligned} \tag{8}$$

Since $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m}_2$, from place invariant $\boldsymbol{x}$, we obtain

$$(c+1)\cdot k = (c+1)\cdot \boldsymbol{m}_2[\mathsf{i}] + \boldsymbol{m}_2[p] + c\cdot \boldsymbol{m}_2[r] + (c+1)\cdot \boldsymbol{m}_2[\mathsf{f}].$$

By reorganizing this equation, we obtain

$$c\cdot \boldsymbol{m}_2[r] = (c+1)\cdot(k - \boldsymbol{m}_2[\mathsf{i}] - \boldsymbol{m}_2[\mathsf{f}]) - \boldsymbol{m}_2[p]. \tag{9}$$

This means that

$$\begin{aligned} c\cdot \boldsymbol{m}_2[r] &= (c+1)\cdot(k - \boldsymbol{m}_2[\mathsf{i}] - \boldsymbol{m}_2[\mathsf{f}]) - \boldsymbol{m}_2[p] && \text{(by (9))}\\ &= (c+1)\cdot(k - \boldsymbol{m}_1[\mathsf{f}]) - (k - \boldsymbol{m}[\mathsf{f}]) && \text{(by def. of } \boldsymbol{m}_2 \text{ and (8))}\\ &= (c+1)\cdot(k - \boldsymbol{m}[\mathsf{f}]) - (k - \boldsymbol{m}[\mathsf{f}]) && \text{(by def. of } \boldsymbol{m}_1\text{)}\\ &= c\cdot(k - \boldsymbol{m}[\mathsf{f}]). \end{aligned}$$

Altogether, we have $\boldsymbol{m}_2[r] = (k - \boldsymbol{m}[\mathsf{f}]) = \boldsymbol{m}_2[p]$. Thus, from $\boldsymbol{m}_2$, transition t_f can be fired $k - \boldsymbol{m}[\mathsf{f}]$ times, which leads to marking

$$\{\mathsf{f}\colon \boldsymbol{m}_1[\mathsf{f}] + (k - \boldsymbol{m}[\mathsf{f}])\} = \{\mathsf{f}\colon \boldsymbol{m}[\mathsf{f}] + k - \boldsymbol{m}[\mathsf{f}]\} = \{\mathsf{f}\colon k\}.$$

This concludes the proof of k-soundness as $\boldsymbol{m} \to^* \boldsymbol{m}_1 \to^* \boldsymbol{m}_2 \to^* \{\mathsf{f}\colon k\}$.

It remains to consider the case where $k = c$. We have

$$\{\mathsf{i}\colon c\} \to^{t_\mathsf{i}^c} \{p\colon (c+1)\cdot c\} \to^{t_r^{c+1}} \{r\colon (c+1)\}.$$

No transition is enabled in the latter marking. So, we have $\{r\colon (c+1)\} \not\to^* \{\mathsf{f}\colon c\}$ and hence c-unsoundness follows. We are done proving this item.

Item 4. Let $k \in \mathbb{N}_{\geq 1}$ be a number that is not a multiple of c. Let us first show that $\{\mathsf{i}\colon k\} \not\to^* \{\mathsf{f}\colon k\}$. For the sake of contradiction, assume there exists a run ρ such that $\{\mathsf{i}\colon k\} \to^\rho \{\mathsf{f}\colon k\}$. Note that ρ needs to fire t_i exactly k times, since no other transition consumes from i. Without loss of generality, let us reorder ρ into a run ρ' such that any firing of t_i happens at the beginning. Let us write $\rho' = t_\mathsf{i}^k\sigma$, where σ does not contain t_i. We have that $\{\mathsf{i}\colon 1\} \to^{t_\mathsf{i}^k} \{u\colon k, d\colon k\}$. The only transition consuming from u is t_u. Since k is not a multiple of c, and since t_u consumes c tokens from u, place u can never be emptied. Thus $\{u\colon k, d\colon k\} \not\to^* \{\mathsf{f}\colon 1\}$.

Next, let us show that $\{\mathsf{i}\colon mc\} \to^* \{\mathsf{f}\colon mc\}$ for any $m \in \mathbb{N}_{\geq 1}$. It follows from

$$\{\mathsf{i}\colon mc\} \to^{t_\mathsf{i}^{mc}} \{u\colon mc, d\colon mc\} \to^{t_d^{m(c-1)}} \{u\colon mc, d\colon m, \mathsf{f}\colon m(c-1)\} \to^{t_u^m} \{\mathsf{f}\colon mc\}.$$

Finally, we show that $N_{\neg\text{sound}}$ is not structurally sound. It suffices to show that it is mc-unsound for all $m \in \mathbb{N}_{\geq 1}$. Note that

$$\{\mathsf{i}\colon mc\} \to^{t_\mathsf{i}^{mc}} \{u\colon mc, d\colon mc\} \to^{t_u^m} \{d\colon (m-1)c, \mathsf{f}\colon m\} \to^{t_d^{((m-1)c)-1}} \{d\colon 1, \mathsf{f}\colon mc-1\}.$$

No transition is enabled in the latter marking, so mc-unsoundness follows. □

Appendix D

Continuous One-Counter Automata

This paper was published as a peer-reviewed journal article. It is in turn based on a conference paper [24].

Cite as: M. Blondin, T. Leys, F. Mazowiecki, P. Offtermatt, and G. A. Pérez. Continuous one-counter automata. *ACM Trans. Comput. Logic (TOCL)*, 24(1), jan 2023.

Summary We propose the novel model of continuous one-counter automata (COCA). These are automata with one counter where the effects of a transition that is fired can be non-deterministically scaled down by a nonzero factor between zero and one. Further, states are guarded by upper and lower bounds on the counter value. We additionally propose two variants of the model. Globally-guarded COCA (GG-COCA) have the same upper and lower bound on each state, while parametric COCA are allowed to have parameters on updates and tests. We show that the reachability problem 1) is in NC^2in GG-COCA (even when in addition to the global bound, states can additionally place equality constraints on the counter), 2) is in polynomial time in COCA, and 3) is NP-complete in parametric COCA.

Contribution of this author This author was chiefly responsible for obtaining the results from Section 3 and for obtaining the NP-hardness result in Section 5, and

contributed to the development of the results in Section 4 by discussing and improving upon early sketches of the polynomial-time algorithm for reachability in COCA. The author also contributed the improved upper bound on the complexity of reachability in parametric COCA from Σ_2 to NP (Theorem 43), which is the main addition of the journal article [23] to the conference article [24]. Additionally, the author participated in the development of the manuscript, particularly taking a leading role on Section 3 and Section 5, and contributing to the presentation of the results in Section 4.

Continuous One-counter Automata

MICHAEL BLONDIN, Université de Sherbrooke, Canada
TIM LEYS, University of Antwerp, Belgium
FILIP MAZOWIECKI, Max Planck Institute for Software Systems, Germany
PHILIP OFFTERMATT, Université de Sherbrooke, Canada, and Max Planck Institute for Software Systems, Germany
GUILLERMO PÉREZ, University of Antwerp – Flanders Make, Belgium

3

We study the reachability problem for continuous one-counter automata, COCA for short. In such automata, transitions are guarded by upper- and lower-bound tests against the counter value. Additionally, the counter updates associated with taking transitions can be (non-deterministically) scaled down by a nonzero factor between zero and one. Our three main results are as follows: we prove (1) that the reachability problem for COCA with global upper- and lower-bound tests is in NC^2; (2) that, in general, the problem is decidable in polynomial time; and (3) that it is NP-complete for COCA with parametric counter updates and bound tests.

CCS Concepts: • **Theory of computation → Logic and verification**; **Automata over infinite objects**; **Problems, reductions and completeness;**

Additional Key Words and Phrases: Counter automata, reachability problem, parametric automata, continuous relaxation

ACM Reference format:
Michael Blondin, Tim Leys, Filip Mazowiecki, Philip Offtermatt, and Guillermo Pérez. 2023. Continuous One-counter Automata. *ACM Trans. Comput. Logic* 24, 1, Article 3 (January 2023), 31 pages.
https://doi.org/10.1145/3558549

1 INTRODUCTION

Counter machines form a fundamental computational model that captures the behavior of infinite-state systems. Unfortunately, their central decision problems, like the *(configuration) reachability problem*, are undecidable as the model is Turing-complete [20, 21]. To circumvent this issue, numerous restrictions of counter machines have been studied in the literature. For instance, **vector addition systems with states (VASS)** arise from restricting the types of tests that can be used to

M. Blondin was supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC); G. Pérez, by the "SAILor" Junior Research Project (G030020N) from the Research Foundation – Flanders (FWO).
Authors' addresses: M. Blondin, Université de Sherbrooke, 2500 boulevard de l'Université, Sherbrooke, Québec, Canada, J1K 2R1; email: michael.blondin@usherbrooke.ca; T. Leys, University of Antwerp, Middelheimlaan 1, Antwerpen, 2020, Belgium; email: tim.leys@uantwerp.be; F. Mazowiecki, Max Planck Institute for Software Systems, Campus E1 5, Saarbrücken, Saarland, Germany; email: filipm@mpi-sws.org; P. Offtermatt, Université de Sherbrooke, 2500 boulevard de l'Université, Sherbrooke, Québec, Canada, J1K 2R1, and Max Planck Institute for Software Systems, Campus E1 5, Saarbrücken, Saarland, Germany; email: philip.offtermatt@usherbrooke.ca; G. Pérez, University of Antwerp - Flanders Make, Middelheimlaan 1, Antwerpen, 2020, Belgium; email: guillermoalberto.perez@uantwerpen.be.

1529-3785/2023/01-ART3
https://doi.org/10.1145/3558549

guard transitions [11, 17–19]. One-counter automata are yet another well-studied model [6, 12, 13], in this case arising from the restriction to a single counter, hence the name.

We consider one-counter automata that can use tests of the form "$\leq c$" and "$\geq d$"—where c and d are constants—to guard their transitions. As a natural extension of finite-state automata, one-counter automata allow for better conservative approximations of classical static-analysis problems like instruction reachability (see, e.g., *program graphs* as defined in [3]). They also enable the verification of programs with lists [7] and XML-stream validation [10]. Furthermore, their reachability problem seems intrinsically connected to that of **timed automata (TA)**. The reachability problem for two-clock TA is known to be logspace-equivalent to the same problem for **succinct one-counter automata (SOCA)**, that is, where constants used in counter updates and tests are encoded in binary [14]. An analog of this connection holds when SOCA are enriched with parameters that can be used on updates: reachability for two-parametric-clock TA reduces to the (existential) reachability problem for parametric one-counter automata [9]. Interestingly, Alur et al. [1] observe that the former subsumes a long-standing open problem of Ibarra et al. [15] concerning "simple programs."

All of the above connections from interesting problems to reachability for SOCA and parametric SOCA indicate that efficient algorithms for the problem are very much desirable. Unfortunately, it is known that reachability for SOCA (with upper- and lower-bound tests) is PSPACE-complete [12]. For parametric SOCA, the situation is even worse as the general problem is not even known to be decidable. In this work, we study *continuous relaxations* of these problems and show that their complexities belong in tractable complexity classes. We thus give the first efficient conservative approximation for the reachability problem for SOCA and parametric SOCA.

The continuous relaxations. We observe that the model considered by Fearnley and Jurdziński [12] is not precisely our SOCA; rather, they consider *bounded* 1*-dimensional VASS*. In such VASS, the counter is not allowed to take negative values. Additionally, it is not allowed to take values greater than some global upper bound. Note that inequality tests against constants can be added to such VASS as "syntactic sugar" since these can be implemented making use of the upper and lower bounds. These observations allow us to adapt Blondin and Haase's definition of continuous VASS [4] to introduce ***globally guarded continuous one-counter automata*** **(GG-COCA)** that have *global* upper- and lower-bound tests: Transitions are allowed to be "partially taken" in the sense that the respective counter updates can be scaled by some factor $\alpha \in (0, 1]$.

In contrast to the situation in the discrete world, because of the continuous semantics, adding arbitrary upper- and lower-bound tests to COCA does result in the more expressive model of *locally guarded COCA*, or simply COCA for short. Importantly, COCA are a "tighter" relaxation of SOCA than GG-COCA (via the translation to bounded 1-VASS). Finally, we also study the reachability problem for *parametric COCA*. These are COCA where counter updates can be variables $x \in X$ whose values range over the rationals; bound tests can also be against variables from X. The resulting model can be seen as a continuous relaxation of Ibarra's simple programs [9, 15].

Contributions. Our main contributions are threefold (see Theorem 1). First, we show that the reachability problem for GG-COCA is in NC^2. This closes a complexity gap for continuous VASS, where reachability is known to be NP-hard when at least two counters are available [4, Lemma 4.13]. Thus, our result shows that an improvement of this lower bound to the case of one counter is unlikely. Second, we give a polynomial-time algorithm for the same problem for COCA. Finally, we show that the reachability problem for parametric COCA is NP-complete. The last upper bound improves on the conference version of this work [5], where we showed that the reachability problem for parametric COCA lies in the polynomial hierarchy.

On the way, we prove that the reachability problem for GG-COCA enriched with equality tests is in NC^2, and that the reachability problem for parametric COCA where only counter updates are allowed to be parametric is equivalent to the integer-valuation restriction of the problem.

Other related work. To complete a full circle of connections between timed and counter automata, we note that the closest model to ours is that of one-clock TA. The value of the clock in such automata evolves (continuously) at a fixed positive rate and can be reset by some transitions. COCA can simulate clock delays using +1 self-loops and resets using −1 self-loops and bound tests "≤0" and "≥0." Our model thus generalizes one-clock TA.

The reachability problem for (non-parametric) one-clock TA is NL-complete [16]. The NL membership proof from [16] relies on the fact that clock delays can always occur and do so without changing the state. This allows partitioning the counter values into *regions*. Only the current region is important, not the precise clock value, as the next region can always be reached by letting time pass. This does not hold in the more general framework of COCA. Here, we need to know how far away the next region is, since the counter value in COCA cannot necessarily be incremented at will in every state. Hence, the proof does not directly extend to COCA.

The reachability problem for parametric one-clock TA with integer-valued parameters is known to lie in NEXP [9]. Since non-parametric clocks can be removed at the cost of an exponential blow-up [1], it is also argued in [9] that the problem belongs to N2EXP if an arbitrary number of non-parametric clocks is allowed [9]. For the latter problem, the authors also prove that it is NEXP-hard. Our NP upper bound for update-parametric COCA with integer-valued parameters improves the latter two bounds.

Organization. In Section 2 we introduce the basic notation and models. Then, in Sections 3, 4, and 5 we prove membership in NC^2, membership in polynomial time, and NP-completeness for reachability of GG-COCA, COCA, and parametric COCA, respectively. Finally, we conclude in Section 6.

2 PRELIMINARIES

We write $\mathbb{Q}_{\geq 0}$ for the set of non-negative rationals, and $\mathbb{Q}_{>0}$ for the set of positive rationals. We use symbols "[" and "]" for closed intervals, and "(" and ")" for open intervals of rational numbers. For example, $[a, b)$ denotes $\{q \in \mathbb{Q} \mid a \leq q < b\}$. Intervals do not have to be bounded; e.g., we allow $[3, +\infty)$. We denote the set of all intervals over $\mathbb{Q}$ by Γ. We write $\overline{X} \subseteq \mathbb{Q}$ to denote the *closure* of a set $X \subseteq \mathbb{Q}$, i.e., X enlarged with its limit points. For example, $\overline{(3, 5)} = [3, 5]$, $\overline{[1, 4) \cup (4, 5]} = [1, 5]$, and $\overline{[2, 3) \cup (4, 5)} = [2, 3] \cup [4, 5]$. Note that $\overline{(-\infty, +\infty)}$ remains $(-\infty, +\infty)$. Throughout the article, numbers are encoded in *binary* and we assume intervals to be encoded as pairs of endpoints, together with binary flags indicating whether the endpoints are contained or not. We present rational numbers as quotients of integers. We assume the sizes of formulas to be the number of symbols needed to write them when numbers are encoded in binary.

2.1 One-counter Automata

A GG-COCA is a triple $\mathcal{V} = (Q, T, \tau)$, where Q and $T \subseteq Q \times \mathbb{Q} \times Q$ are finite sets of *states* and *transitions*, and $\tau \in \Gamma$. A *configuration* of $\mathcal{V}$ is a pair $(q, a) \in Q \times \mathbb{Q}$, denoted $q(a)$. A *run* from $p(a)$ to $q(b)$ in $\mathcal{V}$ is a sequence $\alpha_1 t_1 \cdots \alpha_n t_n$, where $\alpha_i \in (0, 1]$ and $t_i = (q_{i-1}, z_i, q_i) \in T$, for which there exist configurations $q_0(a_0), \ldots, q_n(a_n)$ such that $q_0(a_0) = p(a)$, $q_n(a_n) = q(b)$ and $a_i = a_{i-1} + \alpha_i \cdot z_i$ for all $i \in \{1, \ldots, n\}$. We say that such a run is *admissible* if $a_0, \ldots, a_n \in \tau$. For readers familiar with one-counter automata, note that the model of one-counter nets is obtained by setting $\tau = [0, +\infty)$ and $\alpha_i = 1$ for all i.

A *(locally guarded)* COCA is a triple $\mathcal{W} = (Q, T, \tau)$, where (Q, T) is as for a GG-COCA and $\tau : Q \to \Gamma$ assigns intervals to states. *Configurations* and runs of $\mathcal{W}$ are defined as for a GG-COCA. A run is *admissible* if each of its configurations $q_i(a_i)$ satisfies $a_i \in \tau(q_i)$. Hence, a GG-COCA can be seen as a COCA where $\tau(q)$ is the same for all $q \in Q$.

The set Γ_X of *parameterized intervals* over a set X is the set of intervals whose endpoints belong either to $\mathbb{Q} \cup \{-\infty, +\infty\}$ or X. A *parametric COCA* is a tuple $\mathcal{P} = (Q, T, \tau, X)$, where Q, X, and $T \subseteq Q \times (\mathbb{Q} \cup X) \times Q$ are finite sets of *states*, *parameters*, and *transitions*, and where $\tau : Q \to \Gamma_X$. A *valuation* of X is a function $\mu : X \to \mathbb{Q}$. We write $\mathcal{P}^\mu = (Q, T^\mu, \tau^\mu)$ to denote the COCA obtained from $\mathcal{P}$ by replacing each parameter $x \in X$, occurring in T or τ, with $\mu(x)$. We say that there is a run from $p(a)$ to $q(b)$ in $\mathcal{P}$ if there exists a valuation μ such that $\mathcal{P}^\mu$ has a run from $p(a)$ to $q(b)$. In particular, $\mathcal{P}$ is a COCA if $X = \emptyset$. Otherwise, the notion of a run only makes sense w.r.t. a valuation μ, i.e., in the COCA $\mathcal{P}^\mu$.

In summary, we deal with three increasingly richer models: GG-COCA $\subseteq$ COCA $\subseteq$ parametric COCA. In all variants, the *size* of the automaton is $(|Q| + |T|) \cdot s$, where s is the maximal number of bits required to encode a number in T or τ.

2.2 Runs, Paths, Cycles, and Linear Path Schemes

Let $\mathcal{W} = (Q, T, \tau)$ be a COCA. We write $\text{Paths}_{p,q} \subseteq T^*$ to denote the set of paths from state $p \in Q$ to state $q \in Q$ in the graph induced by T. Let $\pi = t_1 \cdots t_n \in T^*$ be a path. We write $\pi_i = t_i$ to denote the ith element of π. Let $\rho = \alpha_1 t_1 \cdots \alpha_n t_n$ be a run where each $t_i = (q_{i-1}, z_i, q_i)$. The *underlying path* of ρ is $\text{path}(\rho) \coloneqq t_1 \cdots t_n \in \text{Paths}_{q_0,q_n}$. We further define $\rho[i..j] \coloneqq \alpha_i t_i \cdots \alpha_j t_j$, $\rho_i \coloneqq \rho[i..i]$, $\text{in}(\rho) \coloneqq q_0$, $\text{out}(\rho) \coloneqq q_n$, and $\Delta(\rho) \coloneqq \sum_{i=1}^n \alpha_i z_i$. By convention, $\rho[i..j] \coloneqq \varepsilon$ if $j < i$, and $\Delta(\varepsilon) \coloneqq 0$. We write $p(a) \to_\rho q(b)$ to denote the fact that ρ is admissible from $p(a)$ to $q(b)$. Since states p and q are determined by ρ, we may omit them and simply write $a \to_\rho b$. For every $\beta \in (0, 1]$, we define $\beta\rho \coloneqq (\beta\alpha_1)t_1 \cdots (\beta\alpha_n)t_n$. Note that $\beta\rho$ is a run, but it is not necessarily admissible, even if ρ is. For a path $\pi = t_1 \cdots t_n$, we denote as $|\pi| \coloneqq n$ the *length* of π.

Let $\pi = t_1 \cdots t_n \in \text{Paths}_{p,q}$ be such that each $t_i = (q_{i-1}, z_i, q_i)$. We say that π is a *cycle* if $p = q$, and *simple* if π does not repeat any state. Let $\Delta(\pi) \coloneqq z_1 + \ldots + z_n$, $\Delta^+(\pi) \coloneqq \sum_{i=1}^n \max(0, z_i)$ and $\Delta^-(\pi) \coloneqq \sum_{i=1}^n \min(0, z_i)$, with $\Delta(\varepsilon) = \Delta^+(\varepsilon) = \Delta^-(\varepsilon) \coloneqq 0$. In particular, $\Delta^-(\pi) \leq 0 \leq \Delta^+(\pi)$. Moreover, scaling the positive or negative transitions of a path π arbitrarily close to zero yields a run of effect arbitrarily close to $\Delta^-(\pi)$ or $\Delta^+(\pi)$.

We say that a *linear path scheme* is a regular expression of the form

$$L = \sigma_0 \theta_0^* \sigma_1 \theta_1^* \cdots \sigma_{k-1} \theta_{k-1}^* \sigma_k,$$

where each σ_i is a path, each θ_j is a cycle, and $\sigma_0\theta_0\sigma_1\theta_1 \cdots \sigma_{k-1}\theta_{k-1}\sigma_k$ is a path. For ease of notation, we denote by a regular expression L also the language described by L. The *size* of L is defined as $|L| \coloneqq |\sigma_0\theta_0\sigma_1\theta_1 \cdots \sigma_{k-1}\theta_{k-1}\sigma_k|$, that is, the length of the underlying path of L.

We write $p(a) \to_\pi q(b)$ to denote the existence of a run ρ such that $p(a) \to_\rho q(b)$ and $\text{path}(\rho) = \pi$. As for runs, we may omit states and simply write $a \to_\pi b$. The *reachability function* given by π is defined as $\text{Post}_\pi(a) \coloneqq \{b \in \mathbb{Q} \mid p(a) \to_\pi q(b)\}$. We generalize this notion to sets of paths and numbers:

$$\text{Post}_S(A) \coloneqq \bigcup_{\pi \in S} \bigcup_{a \in A} \text{Post}_\pi(a).$$

Note that linear path schemes express sets of paths, so this notion also naturally extends to them. If $S = \text{Paths}_{p,q}$, then we write $\text{Post}_{p,q}(a)$ and $\text{Post}_{p,q}(A)$. For example, for the COCA of Figure 1,

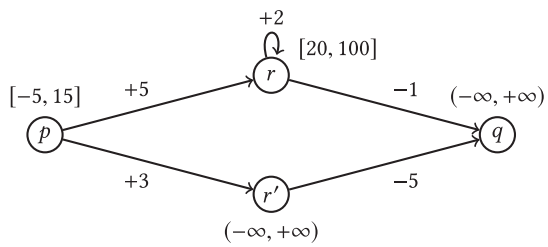


Fig. 1. A COCA; each state s is labeled with the interval $\tau(s)$.

the following holds:

$$\mathrm{Post}_{p,q}(a) = \begin{cases} (10, 18) \cup [19, 100) & \text{if } a = 15, \\ (a-5, a+3) & \text{if } a \in [-5, 15), \\ \emptyset & \text{otherwise.} \end{cases}$$

Finally, we define the set of starting points as $\mathrm{enab}(\pi) := \{a \in \mathbb{Q} \mid \mathrm{Post}_\pi(a) \neq \emptyset\}$ and $\mathrm{enab}(S) := \bigcup_{\pi \in S} \mathrm{enab}(\pi)$.

2.3 Our Contribution

In this work, we study the *reachability problem* that asks the following question: given a GG-COCA or a COCA $\mathcal{W}$ with configurations $p(a)$ and $q(b)$, is there an admissible run from $p(a)$ to $q(b)$? In other words, by abbreviating "$\mathrm{Paths}_{p,q}$" with "$*$," the problem asks whether $p(a) \to_* q(b)$ holds. For parametric COCAs, the *(existential) reachability problem* asks whether $p(a) \to_* q(b)$ for some parameter valuation.

We will establish the following complexity results:

Theorem 1. *The reachability problem is*

(1) in NC^2 for GG-COCAs;
(2) in P for COCAs; and
(3) NP-complete for parametric COCAs.

Recall that NC is the class of problems solvable in polylogarithmic parallel time, i.e., $\mathrm{NC} = \bigcup_{i \geq 0} \mathrm{NC}^i$, where NC^i is the class of problems decidable by logspace-uniform families of circuits of polynomial size, depth $\mathcal{O}(\log^i n)$, and bounded fan-in (e.g., see [2, 22] for a thorough definition). It is well known that $\mathrm{NL} \subseteq \mathrm{NC}^2 \subseteq \mathrm{P}$. We also refer to the functional variant of NC^i as NC^i.

The two first results of Theorem 1 are obtained by characterizing reachability functions and by showing how to efficiently compute a representation in terms of a small number of intervals. More precisely, we show that for every $a \in \mathbb{Q}$:

(1) if $\mathcal{W}$ is a GG-COCA, then $\mathrm{Post}_{p,q}(a)$ consists of at most two intervals whose representations are computable in NC^2 (Corollary 7);
(2) if $\mathcal{W}$ is a COCA, then $\mathrm{Post}_{p,q}(a)$ is made of $|\mathcal{W}|^{\mathcal{O}(1)}$ intervals, and a representation of $\mathrm{Post}_{p,q}(a)$ is computable in polynomial time (Lemma 21).

To derive the third result, i.e., NP-completeness, we need to look more carefully at the second result. It turns out that the reachability question for COCA can be reduced to *linear path schemes* (Lemma 32). This allows us to reduce reachability of parametric COCA to determining the satisfiability of an existential linear arithmetic formula.

3 GLOBALLY GUARDED COCA REACHABILITY

In this section, we prove that the reachability problem for GG-COCAs belongs in NC^2 by showing how to compute a representation of $\mathrm{Post}_{p,q}(a)$ from some a. As a first step, we describe how to utilize graph reachability to check whether $\mathrm{Post}_{p,q}(a)$ is nonempty in Section 3.1. In Section 3.2, we observe that due to the fact that COCAs allow continuous scaling factors when firing transitions, the closure $\overline{\mathrm{Post}_{p,q}(a)}$ is an interval $[b, c]$ that differs from $\mathrm{Post}_{p,q}(a)$ in at most the three points a, b, and c. We utilize results on computing shortest and longest paths in graphs to identify the endpoints b and c in Section 3.3. In Section 3.4, we reduce checks for membership of a, b, and c in $\mathrm{Post}_{p,q}(a)$ to graph reachability queries, using similar techniques as in Section 3.1, and thus obtain an NC^2 algorithm for deciding reachabilitxy in GG-COCAs. Finally, we generalize the achieved results to GG-COCAs with equality tests in Section 3.5 by using the fact that equality tests are passed by a single configuration to construct a graph where nodes represent equality tests and edges correspond to reachability in the GG-COCA.

In the remainder, we fix a GG-COCA $\mathcal{V} = (Q, T, \tau)$ where T and τ use numbers from $\mathbb{Z}$ rather than $\mathbb{Q}$. This assumption merely simplifies the presentation. Indeed, for any $\lambda \in \mathbb{Q}_{\geq 0}$, $p(a) \to_* q(b)$ holds in $\mathcal{V}$ iff $p(\lambda a) \to_* q(\lambda b)$ holds after all numbers are multiplied by λ. Hence, λ could be precomputed in NC^2 as the product of all denominators of fractions occurring within T and τ.

3.1 Testing Emptiness

We first aim to show that deciding whether $\mathrm{Post}_{p,q}(a) \neq \emptyset$ can be checked in NC^2. To this end, we first state some simple graph properties checkable in NC^2. For a path π, let us write $\mathrm{first}(\pi)$ (resp. $\mathrm{last}(\pi)$) to denote the first (resp. last) index such that $\Delta(\pi_i) \neq 0$ if there are any, and $\mathrm{first}(\pi) = \mathrm{last}(\pi) := \infty$ if none exist.

This lemma follows from standard results on NC^2:

Lemma 2. *Let $\mathcal{V} = (Q, E, \tau)$ be a COCA whose weights are encoded in binary, and let $p, q \in Q$ be nodes. Deciding whether $S = \emptyset$ is in NC^2, where S is the set of paths $\pi \in \mathrm{Paths}_{p,q}$ that satisfy a fixed subset of these conditions*[1]*:*

(a) $\Delta^+(\pi) \neq 0$ *(resp.* $\Delta^-(\pi) \neq 0$*);*
(b) $\Delta^+(\pi) = 0$ *(resp.* $\Delta^-(\pi) = 0$*);*
(c) $\Delta(\mathrm{first}(\pi)) < 0$ *(resp.* $\Delta(\mathrm{first}(\pi) > 0)$*);*
(d) $\Delta(\mathrm{last}(\pi)) < 0$ *(resp.* $\Delta(\mathrm{last}(\pi) > 0)$*).*

Furthermore, for any such set S, the following value can be computed in NC^2: $\mathrm{opt}\{w(\pi) \mid \pi \in S \text{ and } |\pi| \leq |Q|)\}$*, where* $\mathrm{opt} \in \{\min, \max\}$ *and* $w \in \{\Delta^+, \Delta^-\}$.

Proof. For each condition, we focus only on one of the two stated cases, while the other case will follow similarly. Each time, we will give a graph H such that reachability from p to q in the multi-graph $G = (Q, E)$ via a path that satisfies the condition is equivalent to graph reachability between two fixed nodes of H, which can be decided in $\mathrm{NL} \subseteq \mathrm{NC}^2$.

(a) Let us define H as joining G with a modified copy $\overline{G}$ that remains in $\overline{G}$ with nonpositive edges and moves into G with positive edges. More formally, let $H := (Q', E')$, where $Q' := Q \cup \{\overline{q} \mid q \in Q\}$. Further, $E' := E \cup \{(\overline{p}, z, \overline{q}) \mid (p, z, q) \in E, z \leq 0\} \cup \{(\overline{p}, z, q) \mid (p, z, q) \in E, z > 0\}$. It is easy to see that any path $\overline{\pi}$ from $\overline{p}$ to q in H corresponds to a path π from p to q in G such that $\Delta^+(\pi) \neq 0$.

(b) We set $H := (Q, E')$, where $E' := \{(p', z, q') \in E \mid z \leq 0\}$. Clearly, reachability from p to q in H is equivalent to reachability from p to q in G via a path π with $\Delta^+(\pi) = 0$.

[1]The set of conditions may be empty, in which case $S = \mathrm{Paths}_{p,q}$.

(c) We again define H as joining G with a modified copy $\overline{G}$. In $\overline{G}$, we omit all positive edges, while edges with weight zero remain in $\overline{G}$ and negative edges lead to G. Formally, let $H := (Q', E')$, where $Q' := Q \cup \{\overline{q} \mid q \in Q\}$ and $E' := E \cup \{(\overline{p}, 0, \overline{q}) \mid (p, 0, q) \in E\} \cup \{(\overline{p}, z, q) \mid (p, z, q) \in E \wedge z < 0\}$. A path $\overline{\pi}$ from $\overline{p}$ to q in H corresponds to a path π from p to q in G such that $\Delta(\text{first}(\pi)) < 0$.

(d) We again join G with a modified copy $\overline{G}$. Now, $\overline{G}$ omits all positive edges, and for each negative edge in G, we add a copy that leads from G to $\overline{G}$. We define $H := (Q', E')$ with $Q' := Q \cup \{\overline{q} \mid q \in Q\}$ and $E' := E \cup \{(p, z, \overline{q}) \mid (p, z, q) \in E \wedge z < 0\} \cup \{(\overline{p}, z, \overline{q}) \mid (p, z, q) \in E \wedge z \leq 0\}$. A path $\overline{\pi}$ from p to $\overline{q}$ in H corresponds to a path π from p to q in G such that $\Delta(\text{last}(\pi)) < 0$.

Note that for each condition, we construct a graph H from a given input graph G. To require many conditions at once, we simply apply the transformations for each condition sequentially and obtain a graph H' such that paths of H' satisfy all imposed conditions and correspond to paths in the original graph G. Observe that H' is of polynomial size.

Finally, let us argue that the following value can be computed in NC^2: $\text{opt}\{w(\pi) \mid \pi \in S \text{ and } |\pi| \leq |Q|)\}$, where $\text{opt} \in \{\min, \max\}$ and $w \in \{\Delta^+, \Delta^-\}$. Let us first deal with $w = \Delta^+$, for which it suffices to treat edges with negative weight as having zero weight.

The problem of finding a shortest weighted path in a graph with edges of nonnegative weights is in NC^2 (e.g., see [8, Example 12.4]). The procedure relies on the fact that there must be an acyclic shortest path, and hence that it suffices to consider paths of length at most $|Q|$. We can easily adapt the standard procedure for maximization: instead of successively minimizing among weighted paths of length $1, 2, 4, 8, \ldots, |Q|$, one maximizes among those paths.[2] It follows that the claim holds for $\text{opt} \in \{\min, \max\}$. The case of Δ^- can be handled similarly by flipping the signs of weights and the optimization type (max/min). □

Now, let us give necessary and sufficient conditions for enabledness of a path from a value. Intuitively, these conditions stem from the observation that effects may be scaled arbitrarily small, yet it is still impossible to apply a transition with negative effect when starting at $\inf \tau$, and similar for transitions with positive effect when starting at $\sup \tau$.

Lemma 3. *Let $a \in \mathbb{Z}$, $p, q \in Q$, and $\pi \in \text{Paths}_{p,q}$. We have $a \in \text{enab}(\pi)$ iff $a \in \tau$ and any of these conditions hold:*

(a) $a \notin \{\inf \tau, \sup \tau\}$;
(b) $a = \inf \tau = \sup \tau$, $\text{first}(\pi) = \infty$;
(c) $a = \inf \tau < \sup \tau$, $\text{first}(\pi) \neq \infty \implies \Delta(\pi_{\text{first}(\pi)}) > 0$;
(d) $a = \sup \tau > \inf \tau$, $\text{first}(\pi) \neq \infty \implies \Delta(\pi_{\text{first}(\pi)}) < 0$.

Proof. Having $a \in \tau$ is obviously necessary, so we assume it holds throughout the proof.

$\Leftarrow$) We proceed by induction on $|\pi|$. If $|\pi| = 0$, then the claim is trivial as the empty path is admissible from a. Assume $|\pi| = n > 0$ and π satisfies a condition. Let $t := \pi_1$ and $\sigma := \pi[2..n]$. If (a) holds, then $a \to_{\beta t} a'$ for some $a' \in \tau \setminus \{\inf \tau, \sup \tau\}$ and sufficiently small $\beta \in (0, 1]$. If (b) holds, then $a \to_t a' = a$ as $\Delta(t) = 0$. If (c) or (d) holds, then either $a \to_t a' = a$ if $\Delta(t) = 0$, or $a \to_{\beta t} a'$ for some $a' \in \tau \setminus \{\inf \tau, \sup \tau\}$ and sufficiently small $\beta \in (0, 1]$ otherwise. In all cases, σ satisfies one of the conditions w.r.t. value a'. Thus, were are done by the induction hypothesis.

$\Rightarrow$) Toward a contradiction, let us assume that $a \in \text{enab}(\pi)$ and that no condition is satisfied. If $a = \inf \tau = \sup \tau$ and $\text{first}(\pi) \neq \infty$, then there is obviously a contradiction. Otherwise, either (i) $a = \inf \tau$ and $\Delta(\pi_{\text{first}(\pi)}) < 0$ or (ii) $a = \sup \tau$ and $\Delta(\pi_{\text{first}(\pi)}) > 0$. We only consider (ii) as (i) is symmetric. Let $a \to_{\pi[1..\text{first}(\pi)-1]} a'$. We have $a' = a$ by definition of $\text{first}(\cdot)$. Moreover,

[2]Note that finding a maximal simple path is NP-complete, but this is not a problem since we allow non-simple paths.

$a' + \beta \cdot \Delta(\pi_{\text{first}(\pi)}) > a' = a = \sup \tau$ for any $\beta \in (0, 1]$. Since exceeding $\sup \tau$ is forbidden, we obtain the contradiction $a \notin \text{enab}(\pi[1..\text{first}(\pi)]) \supseteq \text{enab}(\pi)$. □

Corollary 4. *Given $a \in \mathbb{Z}$ and $p, q \in Q$, deciding whether $a \in \text{enab}(\text{Paths}_{p,q})$, or equivalently $\text{Post}_{p,q}(a) \neq \emptyset$, is in NC^2.*

Proof. We report "empty" if $a \notin \tau$. Otherwise, let

$$S_0 := \{\pi \in \text{Paths}_{p,q} \mid \Delta^+(\pi) = \Delta^-(\pi) = 0\},$$
$$S_+ := \{\pi \in \text{Paths}_{p,q} \mid \text{first}(\pi) \neq \infty \implies \Delta(\pi_{\text{first}(\pi)}) > 0\},$$
$$S_- := \{\pi \in \text{Paths}_{p,q} \mid \text{first}(\pi) \neq \infty \implies \Delta(\pi_{\text{first}(\pi)}) < 0\}.$$

By Lemma 3, it suffices if one of the following holds:

(a) $a \notin \{\inf \tau, \sup \tau\}$ and $\text{Paths}_{p,q} \neq \emptyset$;
(b) $a = \inf \tau = \sup \tau$ and $S_0 \neq \emptyset$;
(c) $a = \inf \tau < \sup \tau$ and $S_+ \neq \emptyset$;
(d) $a = \sup \tau > \sup \tau$ and $S_- \neq \emptyset$.

All of the above can be checked in NC^2 by Lemma 2. □

3.2 Characterization of Reachability Sets

As a step toward computing a representation of $\text{Post}_{p,q}(a)$, we characterize $\text{Post}_{p,q}(a)$ in terms of its closure. To this end, we note that admissible runs remain admissible whenever they are scaled down. Consequently, $\overline{\text{Post}_{p,q}(a)}$ is a closed interval that differs from $\text{Post}_{p,q}(a)$ in at most three points.

Proposition 5 (Adapted from [4, Lemma 4.2(c)]). *Let $\beta \in (0, 1]$ and let ρ be an admissible run from configuration $p(a)$. It is the case that run $\beta\rho$ is also admissible from $p(a)$.*

Proof. We will show that for all $i \in \{0, \ldots, |\rho|\}$, either $a \leq a + \Delta(\beta\rho[1..i]) \leq a + \Delta(\rho[1..i])$ or $a \geq a + \Delta(\beta\rho[1..i]) \geq a + \Delta(\rho[1..i])$. Since $a + \Delta(\rho[1..i]) \in \tau$ holds by the admissibility of ρ from a, it follows that $a + \Delta(\beta\rho[1..i]) \in \tau$, and so that $\beta\rho$ is admissible.

By definition, we have $a + \Delta(\beta\rho[1..i]) = a + \beta\Delta(\rho[1..i])$. Additionally, $\beta \in (0, 1]$. Hence, if $\Delta(\rho[1..i]) \geq 0$, then we have $a + \Delta(\rho[1..i]) \geq a + \Delta(\beta\rho[1..i]) \geq a$. If $\Delta(\rho[1..i]) < 0$, then $a + \Delta(\rho[1..i]) \leq a + \Delta(\beta\rho[1..i]) < a$, so we are done. □

Lemma 6. *For every $b \in \overline{\text{Post}_{p,q}(a)}$, it is the case that $(a, b) \subseteq \text{Post}_{p,q}(a)$ and $(b, a) \subseteq \text{Post}_{p,q}(a)$.*

Proof. We only prove $(a, b) \subseteq \text{Post}_{p,q}(a)$ as the other inclusion is symmetric. We assume that $a < b$ as we are otherwise done. Let $c \in (a, b)$. Since $b \in \overline{\text{Post}_{p,q}(a)}$, there exists $b' \in \text{Post}_{p,q}(a)$ such that $b' \in [c, b]$. Let ρ be an admissible run from $p(a)$ to $q(b')$. By definition, $\Delta(\rho) = b' - a$. Let $\beta := (c - a)/(b' - a) \in (0, 1]$. By Proposition 5, $\beta\rho$ is admissible from $p(a)$. Since $\Delta(\beta\rho) = c - a$, this concludes the proof of the main claim. □

Corollary 7. *Set $\overline{\text{Post}_{p,q}(a)}$ is a closed interval. Moreover,*

$$\overline{\text{Post}_{p,q}(a)} \setminus \text{Post}_{p,q}(a) \subseteq \{\inf \overline{\text{Post}_{p,q}(a)}, a, \sup \overline{\text{Post}_{p,q}(a)}\}.$$

Proof. Let $b := \inf \overline{\text{Post}_{p,q}(a)}$ and $c := \sup \overline{\text{Post}_{p,q}(a)}$. For the sake of contradiction, suppose there is some $v \in \overline{\text{Post}_{p,q}(a)} \setminus \text{Post}_{p,q}(a)$ such that $v \notin \{b, a, c\}$. By Lemma 6, we have $(a, b) \cup (a, c) \cup (b, a) \cup (c, a) \subseteq \text{Post}_{p,q}(a) \subseteq \overline{\text{Post}_{p,q}(a)}$. Since $v \in (b, c) \setminus \{a\}$, we obtain $v \in \text{Post}_{p,q}(a)$, which is a contradiction. □

3.3 Identifying the Endpoints

We now show that a representation of the interval $\overline{\mathrm{Post}_{p,q}(a)}$ can be obtained by identifying its endpoints in NC^2. Some simple observations follow from Proposition 5 and Lemma 6:

Proposition 8. *The following statements hold:*

(a) If $\inf \tau \notin \{-\infty, \infty\}$ *and* $\mathrm{Post}_{p,q}(\inf \tau) \neq \emptyset$*, then* $\inf \overline{\mathrm{Post}_{p,q}(\inf \tau)} = \inf \tau$*.*
(b) If $\sup \tau \notin \{-\infty, \infty\}$ *and* $\mathrm{Post}_{p,q}(\sup \tau) \neq \emptyset$*, then* $\sup \overline{\mathrm{Post}_{p,q}(\sup \tau)} = \sup \tau$*.*
(c) Let $v \in \tau \setminus \{\inf \tau, \sup \tau\}$ *and let* ρ *be a run. There exists* $\varepsilon \in (0, 1]$ *such that for all* $\beta \in (0, \varepsilon]$ *there exists* $v_\beta > 0$ *such that* $v \to_{\beta\rho} v_\beta$*. Moreover,* $\lim_{\beta \to 0} v_\beta = v$*.*

Proof.

(a) Let $a \coloneqq \inf \tau$ and $c \coloneqq \sup \overline{\mathrm{Post}_{p,q}(a)}$. By Lemma 6, we have $(a, c) \subseteq \overline{\mathrm{Post}_{p,q}(a)}$. Since the latter is closed by definition, we have $\inf \overline{\mathrm{Post}_{p,q}(a)} = \inf \tau$.
(b) The proof is symmetric to (a).
(c) Since $v \notin \{\inf \tau, \sup \tau\}$, there is a small enough $\varepsilon \in (0, 1]$ such that $v + |\varepsilon \cdot \Delta(\rho[1..i])| \in \tau$ for all $i \in \{1, \ldots, |\rho|\}$. By definition, $\varepsilon\rho$ is admissible from v. Let $v_\beta \coloneqq v + \beta \cdot \Delta(\rho)$. By Proposition 5, $v \to_{\beta\rho} v_\beta$ is admissible for all $\beta \in (0, \varepsilon]$. Moreover, $\lim_{\beta \to 0} v_\beta = \lim_{\beta \to 0} v + \beta \cdot \Delta(\rho) = v$. □

The two forthcoming lemmas characterize the endpoints of $\overline{\mathrm{Post}_{p,q}(a)}$ through so-called admissible cycles. We say that a cycle θ is (a, p, q)-*admissible* if its first transition t satisfies $\Delta(t) \neq 0$, $a \in \mathrm{enab}(\mathrm{Paths}_{p,\mathrm{in}(t)})$, and $\mathrm{Paths}_{\mathrm{in}(t),q} \neq \emptyset$. We say that such an admissible cycle is *positive* if $\Delta(t) > 0$, and *negative* if $\Delta(t) < 0$. Such cycles can be iterated to approach the endpoints of τ, by scaling all transitions but t arbitrarily close to zero.

Lemma 9. *If* $\mathrm{Post}_{p,q}(a) \neq \emptyset$ *and* $\mathcal{V}$ *has an* (a, p, q)*-admissible cycle* θ*, then the following holds:*

(a) $\inf \overline{\mathrm{Post}_{p,q}(a)} = \inf \tau$*, if* θ *is negative;*
(b) $\sup \overline{\mathrm{Post}_{p,q}(a)} = \sup \tau$*, if* θ *is positive.*

Proof. Let $\theta = t\pi$, where t is the first transition of θ and π is the remaining path. Let $r \coloneqq \mathrm{in}(t)$. Since $a \in \mathrm{enab}(\mathrm{Paths}_{p,r})$, there is an admissible run ρ_1 from $p(a)$ that ends in state r. Similarly, since $\mathrm{Paths}_{r,q} \neq \emptyset$, there is a run ρ_3 from r to q.

We only show (b) as (a) is symmetric. We assume that $a < \sup \tau$, as otherwise we are done by Proposition 8(b). We make a case distinction on whether $\sup \tau = \infty$.

Case $\sup \tau \neq \infty$. We must show that we can reach values arbitrarily close to $\sup \tau$, i.e., that for every $\varepsilon \in (0, 1]$, there exists a value $b \in [\sup \tau - \varepsilon, \sup \tau)$ and a run $p(a) \to_\rho q(b)$.

By Proposition 5 and $a < \sup \tau$, we have $p(a) \to_{(1/2)\rho_1} r(a')$ for some $a' < \sup \tau$. Let

$$m \coloneqq \sum_{i=1}^{|\pi|} |\Delta(\pi_i)|, \alpha_t \coloneqq \frac{\varepsilon}{4|\Delta(t)|} \text{ and } \alpha_\pi \coloneqq \frac{\varepsilon}{4m + 1}.$$

Let $\rho_2 \coloneqq \alpha_t t\, \alpha_\pi \pi$. By definition, we have $\Delta(\alpha_t t) = \varepsilon/4$ and $|\Delta(\alpha_\pi \pi[1..i])| < \varepsilon/4$ for all $i \in \{1, \ldots, |\pi|\}$. Consequently, it is the case that $\Delta(\rho_2[1..i]) \in (0, \varepsilon/2)$ for all $i \in \{2, \ldots, |\rho_2|\}$.

Hence, there exists $k \geq 0$ such that ρ_2^k is admissible from a' and $\sup \tau - \varepsilon/2 \leq a' + \Delta(\rho_2^k) < \sup \tau$. Therefore, we have

$$r(a') \to_{\rho_2^k} r(b') \text{ where } b' \in [\sup \tau - \varepsilon/2, \sup \tau).$$

By Proposition 8(c), we can scale the run ρ_3 so that it is admissible from $r(b')$ and reaches a value arbitrarily close to b' in state q. More formally, there exists $\beta \in (0, 1]$ such that

$$r(b') \to_{\beta\rho_3} q(b) \text{ where } b \in [b' - \varepsilon/2, \sup \tau).$$

We are done since $p(a) \to_{(1/2)\rho_1} r(a') \to_{\rho_2^k} r(b') \to_{\beta\rho_3} q(b)$ and $b \in [\sup \tau - \varepsilon, \sup \tau)$.

Case $\sup \tau = \infty$. We must show that we can reach arbitrarily large values. Let $b \geq a$. For all $\ell \geq 0$, the run $\rho'_\ell := (1/2)\rho_1 \rho_2^\ell$ is admissible from a, and such that $\Delta(\rho'_\ell) > 0$. Thus, there exists $\ell \geq 0$ such that $\Delta(\rho'_\ell) \geq (b - a) + \Delta^-(\rho_3)$. We are done since

$$a \to_{\rho'_\ell} b' \to_{\rho_3} b'', \text{ where } b' \geq b + \Delta^-(\rho_3) \text{ and } b'' \geq b. \qquad \square$$

Lemma 10. *Let* $\mathrm{Post}_{p,q}(a) \neq \emptyset$, $b := \inf \overline{\mathrm{Post}_{p,q}(a)}$ *and* $c := \sup \overline{\mathrm{Post}_{p,q}(a)}$. *If* $\mathcal{V}$ *has no* (a, p, q)-*admissible cycle that is*

(a) negative, then $b \neq -\infty$ *and* $b = \max(\inf \tau, a + \min\{\Delta^-(\pi) \mid \pi \in \mathrm{Paths}_{p,q}, a \in \mathrm{enab}(\pi)\})$*;*
(b) positive, then $c \neq +\infty$ *and* $c = \min(\sup \tau, a + \max\{\Delta^+(\pi) \mid \pi \in \mathrm{Paths}_{p,q}, a \in \mathrm{enab}(\pi)\})$.

Proof. We only prove (b) as (a) is symmetric. Assume $\mathcal{V}$ has no positive (a, p, q)-admissible cycle. Let $D^+ := \{\Delta^+(\pi) \mid \pi \in \mathrm{Paths}_{p,q}, a \in \mathrm{enab}(\pi)\}$. We show that $\max D^+$ is well defined. For the sake of contradiction, suppose that D^+ is infinite. By a pigeonhole argument, we obtain a run ρ admissible from a and such that ρ contains at least two occurrences of a transition t with $\Delta(t) > 0$. Let $\mathrm{path}(\rho) = \pi t \pi' t \pi''$, where π, π', π'' are paths. The cycle $\theta := t\pi'$ is a positive admissible cycle, which yields a contradiction.

Note that $c \leq \min(\sup \tau, a + \max D^+)$, so $c \neq +\infty$. It remains to show that $c = \min(\sup \tau, a + \max D^+)$. Let $\pi \in \mathrm{Paths}_{p,q}$ be such that $a \in \mathrm{enab}(\pi)$ and $\Delta^+(\pi) = \max D^+$. By definition, there exists a run $\rho = \alpha_1 t_1 \cdots \alpha_n t_n$ admissible from $p(a)$ and such that $\mathrm{path}(\rho) = \pi$. Since $a \in \tau$, there exists $\lambda \in (0, 1]$ such that $a + \lambda \cdot \max D^+ = \min(\sup \tau, a + \max D^+)$. For all $\varepsilon \in (0, 1)$, let $\rho_\varepsilon := \alpha'_1 t_1 \cdots \alpha'_n t_n$ be the run such that

$$\alpha'_i := \begin{cases} (1 - \varepsilon) \cdot \lambda & \text{if } \Delta(t_i) \geq 0, \\ \varepsilon \cdot (1/|\Delta(t_i)|) \cdot (1/n) & \text{otherwise.} \end{cases}$$

Informally, if we were allowed to scale transitions by 0, then we would be done by using ρ_0 from a, as it would never decrease and reach exactly $a + \lambda \cdot \max D^+ = \min(\sup \tau, a + \max D^+)$.

Formally, we choose a small $\varepsilon \in (0, 1]$ as follows. If $a > \inf \tau$, then we pick ε so that $a - \varepsilon \geq \inf \tau$. Otherwise, we pick ε so that $(1 - \varepsilon) \cdot \lambda \geq \varepsilon$. We claim that the run ρ_δ is admissible from a for every $\delta \in (0, \varepsilon]$. First note that the top guard is never exceeded since $a + \Delta^+(\rho_\delta) = a + (1 - \delta) \cdot \lambda \cdot \max D^+ \leq a + \lambda \cdot \max D^+ \leq \sup \tau$. Let us now consider the bottom guard.

If $a > \inf \tau$, then $a + \Delta^-(\rho_\delta) \geq a - \delta \geq a - \varepsilon \geq \inf \tau$. Otherwise, if $a = \inf \tau$, then either $\Delta^-(\rho_\delta) = \Delta^+(\rho_\delta) = 0$, in which case admissibility is trivial, or the first transition t_i such that $\Delta(t_i) \neq 0$ is such that $\Delta(t_i) \geq 1$. In that case, the following holds for every $j \geq i$:

$$\begin{aligned} a + \Delta(\rho_\delta[1..j]) &\geq a + (1 - \delta) \cdot \lambda \cdot \Delta(t_i) + \Delta^-(\rho_\delta[i + 1..j]) \\ &\geq a + (1 - \varepsilon) \cdot \lambda + \Delta^-(\rho_\delta[i + 1..j]) \\ &\geq a + (1 - \varepsilon) \cdot \lambda - \varepsilon \geq a = \inf \tau. \end{aligned}$$

This shows the admissibility of ρ_δ. Thus, for all $\delta \in (0, \varepsilon]$, we have $a \to_{\rho_\delta} a_\delta$, where $a_\delta \geq \min(\sup \tau, a + \max D^+) - \delta \cdot \lambda \cdot \max D^+$. We are done since $\lim_{\delta \to 0} a_\delta = \min(\sup \tau, a + \max D^+)$. $\square$

In the forthcoming propositions, we show how the previous characterizations can be turned into NC^2 procedures.

Lemma 11. *On input* $a \in \mathbb{Z}$ *and* $p, q \in Q$, *deciding if* $\mathcal{V}$ *has a positive or negative* (a, p, q)-*admissible cycle is in* NC^2.

Proof. We consider the positive case; the negative one is symmetric. Testing whether there is a positive (a, p, q)-admissible cycle beginning with a transition t with $\Delta(t) > 0$ amounts to testing whether (i) $a \in \mathrm{enab}(\mathrm{Paths}_{p,\mathrm{in}(t)})$, (ii) $\mathrm{Paths}_{\mathrm{in}(t),q} \neq \emptyset$, and (iii) $\mathrm{Paths}_{\mathrm{out}(t),\mathrm{in}(t)} \neq \emptyset$. Condition (i) can be checked in NC^2 by Corollary 4. Conditions (ii) and (iii) are graph reachability queries, which can be tested in $\mathrm{NL} \subseteq \mathrm{NC}^2$. There are at most $|T|$ transitions with positive effect, so the conditions can be tested in parallel for each $t \in T$. □

Proposition 12. *On input $a \in \mathbb{Z}$ and states p, q, the values $\inf \overline{\mathrm{Post}_{p,q}(a)}$ and $\sup \overline{\mathrm{Post}_{p,q}(a)}$ can be computed in NC^2.*

Proof. We explain how to compute $c := \sup \overline{\mathrm{Post}_{p,q}(a)}$ in NC^2. The procedure for $\inf \overline{\mathrm{Post}_{p,q}(a)}$ is symmetric. By Corollary 4, testing whether $\mathrm{Post}_{p,q}(a) = \emptyset$ is in NC^2. If it holds, then trivially $c = -\infty$. Otherwise, assume that $\mathrm{Post}_{p,q}(a) \neq \emptyset$, and hence $a \in \tau$. Additionally, if $a = \sup \tau$, then $c = \sup \tau$ by Proposition 8(b). So, we assume $a < \sup \tau$.

By Lemma 11, it can be decided in NC^2 whether there exists a positive (a, p, q)-admissible cycle. If such a cycle exists, then $c = \sup \tau$ by Lemma 9. Otherwise, by Lemma 10, we have $c = \min(\sup \tau, a + \max D^+)$, where $D^+ := \{\Delta^+(\pi) \mid \pi \in \mathrm{Paths}_{p,q}, a \in \mathrm{enab}(\pi)\}$.

Consider a path $\pi \in \mathrm{Paths}_{p,q}$ that satisfies $a \in \mathrm{enab}(\pi)$ and which can be decomposed as $\pi = \sigma\theta\sigma'$, where θ is a cycle. As θ is (a, p, q)-admissible by definition, it cannot contain a positive transition. Otherwise, $\mathcal{V}$ would admit a positive (a, p, q)-admissible cycle, which is a contradiction. Hence, $\Delta^+(\pi) = \Delta^+(\sigma\sigma')$. Recall that $a \neq \sup \tau$. Then $a \in \mathrm{enab}(\sigma\sigma')$ follows from Lemma 3(a) or (c).

The above shows that there is a simple path π_{max} such that $\max D^+ = \Delta^+(\pi_{max})$ and $a \in \mathrm{enab}(\pi_{max})$. Therefore, to obtain $\max D^+$, it is sufficient to compute $\max E^+$, where $E^+ := \{\Delta^+(\pi) \mid \pi \in \mathrm{Paths}_{p,q}, a \in \mathrm{enab}(\pi), |\pi| \leq |Q|\}$.

Now, let us make a case distinction based on whether $a = \inf \tau$. Assume this is true. By Lemma 3,

$$\max E^+ = \max(\{\Delta^+(\pi) \mid \pi \in \mathrm{Paths}_{p,q}, \mathrm{first}(\pi) = \infty, |\pi| \leq |Q|\} \cup \{\Delta^+(\pi) \mid \pi \in \mathrm{Paths}_{p,q}, \Delta(\pi_{\mathrm{first}(\pi)}) > 0, |\pi| \leq |Q|\}).$$

By Lemma 2, the two sets that are joined in the expression can be computed in NC^2. If $a \neq \inf \tau$, then we have $a \in \tau \setminus \{\inf \tau, \sup \tau\}$, so by Lemma 3(a) all paths are admissible from a, and hence $\max E^+ = \max\{\Delta^+(\pi) \mid \pi \in \mathrm{Paths}_{p,q} \wedge |\pi| \leq |Q|\}$. This value can be computed in NC^2 by Lemma 2. □

3.4 Computing the Representation

To obtain a representation of $\mathrm{Post}_{p,q}(a)$, it remains to explain how to check in NC^2 which of the three limit elements $\inf \overline{\mathrm{Post}_{p,q}(a)}$, $\sup \overline{\mathrm{Post}_{p,q}(a)}$, and a belong to $\mathrm{Post}_{p,q}(a)$. Intuitively, for each of these three elements, membership is equivalent to the existence of paths satisfying some conditions.

Proposition 13. *Let $\mathrm{Post}_{p,q}(a) \neq \emptyset$. It is the case that $a \in \mathrm{Post}_{p,q}(a)$ iff at least one of these conditions holds:*

(a) there exists a path $\pi \in \mathrm{Paths}_{p,q}$ whose transitions are all zero, i.e., $\Delta(\pi) = \Delta^+(\pi) = \Delta^-(\pi) = 0$;
(b) there exist $\pi \in \mathrm{Paths}_{p,q}$ and indices i, j such that $\Delta(\pi_i) > 0$ and $\Delta(\pi_j) < 0$. If $a = \inf \tau$, then we also require that $\Delta(\pi_k) = 0$ for all $k < i$ and $k > j$. Similarly, if $a = \sup \tau$, then we also require $\Delta(\pi_k) = 0$ for all $k < j$ and $k > i$.

Proof. $\Leftarrow$) If (a) holds, then trivially $a \in \mathrm{Post}_{p,q}(a)$. Assume (b) holds. Let $\rho := 1t_1 \ldots 1t_n$, where $\pi = t_1 \ldots t_n$. Suppose $a \notin \{\inf \tau, \sup \tau\}$. By Proposition 8(c), for all β small enough, it is the case

that $a \to_{\beta\rho} a_\beta$, where $|a - a_\beta| < 1/2$. Let $p(a) = q_0(a_0), \ldots, q_n(a_n) = q(a_\beta)$ be the sequence of configurations witnessing $a \to_{\beta\rho} a_\beta$. Since n is fixed, we can choose $\beta < 1/2$ small enough so that $|a_i - a| < 1/2$ for all i. If $a_\beta > a$, then we enlarge the coefficient of t_j to $\alpha_j > \beta$ so that $(\alpha_j - \beta) \cdot \Delta(t_j) = a - a_\beta$. By the choice of β, we get an admissible run $\rho' := \beta t_1 \ldots \beta t_{j-1} \alpha_j t_j \beta t_{j+1} \ldots \beta t_n$ that satisfies $a \to_{\rho'} a$. If $a_\beta < a$, then we proceed analogously with index i.

It remains to prove the case where $a = \inf \tau$; the case where $a = \sup \tau$ is symmetric. By assumption, we have $\Delta(t_k) = 0$ for all $k < i$ and $k > j$. For the sake of simplicity, assume $\Delta(t_1) > 0$ and $\Delta(t_n) < 0$. Let $\alpha_1 \in (0, 1)$ be such that $\alpha_1 \cdot \Delta(t_1) < 1/2$. Let $\rho_1 := 1t_2 \ldots 1t_{n-1}$. By Proposition 8(c), there exists $\beta \in (0, 1]$ such that $\inf \tau \to_{\alpha_1 t_1 \beta \rho_1} \delta$, where $\delta < 1$. Since $\Delta(t_n) < 0$, there exists $\alpha_n \in (0, 1)$ such that $\alpha_n \cdot \Delta(t_n) = -\delta$. Thus, we have $p(\inf \tau) \to_{\alpha_1 t_1 \beta \rho_1 \alpha_n t_n} q(\inf \tau)$.

$\Rightarrow$) Let $p(a) \to_\rho q(a)$ and $\pi := \mathrm{path}(\rho)$. Suppose (a) does not hold. If all transitions of π were positive, then we would obtain the contradiction $p(a) \to_\rho q(a')$ with $a' > a$. Similarly, all transitions cannot be negative. For the specific case where $a = \inf \tau$, observe that if the first nonzero transition is negative, then ρ cannot be admissible. Similarly, if the last nonzero transition is positive, then $p(\inf \tau) \to_\rho q(\delta)$ for some $\delta > \inf \tau$. The reasoning for the case $a = \sup \tau$ is symmetric. □

It is easy to see that both conditions of the prior proposition can be checked in NC^2 by Lemma 2.

We introduce and prove a similar characterization of "$\sup \overline{\mathrm{Post}_{p,q}(a)} \in \mathrm{Post}_{p,q}(a)$":

Proposition 14. *Let* $\mathrm{Post}_{p,q}(a) \neq \emptyset$, $b := \inf \overline{\mathrm{Post}_{p,q}(a)}$ *and* $c := \sup \overline{\mathrm{Post}_{p,q}(a)}$. *If* $b < a < c$ *and* $c \in \tau$, *then* $c \in \mathrm{Post}_{p,q}(a)$ *iff there is a state* r *and a path* $\sigma \in \mathrm{Paths}_{r,q}$ *that satisfy* $\Delta^+(\sigma) > 0$, $\Delta^-(\sigma) = 0$, *and either of the following:*

(i) there exists a path $\sigma' \in \mathrm{Paths}_{p,r}$ *such that* $|\sigma|, |\sigma'| \leq |Q|$, $\Delta^-(\sigma') = 0$ *and* $\Delta^+(\pi) \geq c - a$, *where* $\pi := \sigma'\sigma$;
(ii) there exists a path $\sigma' \in \mathrm{Paths}_{p,r}$ *such that* $|\sigma|, |\sigma'| \leq |Q|$ *and* $\Delta^+(\pi) > c - a$, *where* $\pi := \sigma'\sigma$;
(iii) there is a positive (a, p, r)*-admissible cycle* θ.

Proof. $\Rightarrow$) Assume $c \in \mathrm{Post}_{p,q}(a)$. There is a run ρ such that $p(a) \to_\rho q(c)$. Let ρ' be the run obtained from ρ by repeatedly removing a cycle θ with $\Delta^+(\theta) = 0$, until no further possible. Let $\pi := \mathrm{path}(\rho')$. We have $\Delta^+(\pi) \geq \Delta(\rho') \geq \Delta(\rho) = c - a$. Since $c > a$, there is a maximal index i such that $\Delta(\pi_i) > 0$. Let $r := \mathrm{in}(\pi_i)$, $\sigma' := \pi[1..i-1]$ and $\sigma := \pi[i..|\pi|]$. Note that $\sigma' \in \mathrm{Paths}_{p,r}$ and $\sigma \in \mathrm{Paths}_{r,q}$. Moreover, $\Delta^+(\sigma) > 0$ holds by maximality of i. It must also be the case that $\Delta^-(\sigma) = 0$. Indeed, otherwise the last nonzero transition t of σ, and consequently of ρ, would be negative. Hence, this would contradict $c = \sup \overline{\mathrm{Post}_{p,r}(a)}$ as we could reach values arbitrarily close to $c + \varepsilon$ for some $\varepsilon > 0$ by scaling t arbitrarily close to zero. Observe that if $\Delta^+(\pi) = c - a$, then $\Delta(\rho) = \Delta^+(\pi) = c - a$, which implies $\Delta^-(\rho) = \Delta^-(\pi) = 0$. Therefore, if $|\sigma|, |\sigma'| \leq |Q|$, we have shown (i) or (ii).

Otherwise, π is a nonsimple path. So, by our past cycle elimination, π contains a cycle θ with $\Delta^+(\theta) > 0$. Let us reorder θ into θ' so that the first transition t of θ' satisfies $\Delta(t) > 0$. We have $a \in \mathrm{enab}(\mathrm{Paths}_{p,\mathrm{in}(t)})$ as state $\mathrm{in}(t)$ occurs on the original run ρ that leads to state q. Moreover, $\mathrm{Paths}_{\mathrm{in}(t),r} \neq \emptyset$ holds by maximality of i. Thus, θ' is a positive (a, p, r)-admissible cycle. Hence, we have shown that (iii) holds.

$\Leftarrow$) If (i) holds, then $\Delta^+(\pi) = c - a$ or $\Delta^+(\pi) > c - a$. The latter case is subsumed by (ii), and in the former case we are done as $a \to_\pi c$ due to $\Delta^-(\pi) = 0$. If (iii) holds, then since θ is a positive (a, p, r)-admissible cycle—and hence (a, p, q)-admissible—Lemma 9(b) yields $\sup \overline{\mathrm{Post}_{p,r}(a)} = c = \sup \tau$. Thus, there exists $\varepsilon \in [0, 1]$ such that $c - \varepsilon \in \mathrm{Post}_{p,r}(a)$. Note that $\Delta^+(\sigma) > 0$ implies $\Delta^+(\sigma) \geq 1$, since we assume transition effects to be integral. By $\Delta^+(\sigma) \geq 1$ and $\Delta^-(\sigma) = 0$, we have

$$p(a) \to_* r(c - \varepsilon) \to_{\beta\sigma} q(c) \text{ where } \beta := \varepsilon / \Delta^+(\sigma).$$

If (ii) holds, then we proceed as follows. Recall that $b < a < c$. Therefore, $a \notin \{\inf \tau, \sup \tau\}$, since $\inf \tau \leq b$ and $c \leq \sup \tau$ by definition of b and c. Due to $a \notin \{\inf \tau, \sup \tau\}$, we can scale the negative transitions of σ' arbitrarily close to zero and scale its positive transitions so that either $a + \Delta^+(\sigma') - \varepsilon \in \mathrm{Post}_{p,r}(a)$ or $\sup \tau - \varepsilon \in \mathrm{Post}_{p,r}(a)$ for some $\varepsilon \in (0, 1]$. Since $\Delta^+(\sigma) \geq c - a - \Delta^+(\sigma') + 1$, $\Delta^+(\sigma) \geq 1$, and $\Delta^-(\sigma) = 0$, we can derive either $c \in \mathrm{Post}_{r,q}(a)$ or $\sup \tau \in \mathrm{Post}_{r,q}(a)$. As the latter implies $c = \sup \tau$, we are done proving the claim. □

Finally, these characterizations allow us to conclude our first major result.

Proposition 15. *On input $a \in \mathbb{Z}$ and $p, q \in Q$, computing $\mathrm{Post}_{p,q}(a) \cap \{\inf \overline{\mathrm{Post}_{p,q}(a)}, \sup \overline{\mathrm{Post}_{p,q}(a)}, a\}$ is in NC^2.*

Proof. By Corollary 4, $\mathrm{Post}_{p,q}(a) \neq \emptyset$ can be tested in NC^2. Thus, we assume that it is nonempty. By Proposition 13, it holds that $a \in \mathrm{Post}_{p,q}(a)$ iff at least one of these conditions holds:

(a) there exists a path $\pi \in \mathrm{Paths}_{p,q}$ whose transitions are all zero, i.e., $\Delta(\pi) = \Delta^+(\pi) = \Delta^-(\pi) = 0$;
(b) there exist $\pi \in \mathrm{Paths}_{p,q}$ and i, j such that $\Delta(\pi_i) > 0$ and $\Delta(\pi_j) < 0$. If $a = \inf \tau$, then we also require $\Delta(\pi_k) = 0$ for all $k < i$ and $k > j$. Similarly, if $a = \sup \tau$, then we also require $\Delta(\pi_k) = 0$ for all $k < j$ and $k > i$.

Note that (b) is trivially unsatisfiable if $a = \inf \tau = \sup \tau$.

It remains to argue that both conditions can be checked in NC^2. We can check condition (a) in NC^2 via Lemma 2(b). If $a \notin \{\inf \tau, \sup \tau\}$, then condition (b) can be checked in NC^2 via Lemma 2(a). If $a \in \{\inf \tau, \sup \tau\}$, then condition (b) can be checked in NC^2 via Lemma 2(d) and Lemma 2(c).

Let $b \coloneqq \inf \overline{\mathrm{Post}_{p,q}(a)}$ and $c \coloneqq \sup \overline{\mathrm{Post}_{p,q}(a)}$, which can be computed in NC^2 by Proposition 12. We check whether $b = c$. If it is, we return $\{b, c\}$ since $\mathrm{Post}_{p,q}(a) \neq \emptyset$. Otherwise, we explain how to check whether $c \in \mathrm{Post}_{p,q}(a)$; the case of b can be handled symmetrically. We assume that $b < a < c$, as the first half of the proof handles the case $a \in \{b, c\}$ when checking membership of a.

If $c \notin \tau$, then $c \notin \mathrm{Post}_{p,q}(a)$. Otherwise, by Proposition 14, it holds that $c \in \mathrm{Post}_{p,q}(a)$ iff there is a state $r \in Q$ and a path $\sigma \in \mathrm{Paths}_{r,q}$ that satisfy $\Delta^+(\sigma) > 0$, $\Delta^-(\sigma) = 0$ and either of the following holds:

(i) there exists a path $\sigma' \in \mathrm{Paths}_{p,r}$ such that $|\sigma|, |\sigma'| \leq |Q|$, $\Delta^-(\sigma') = 0$ and $\Delta^+(\pi) \geq c - a$, where $\pi \coloneqq \sigma'\sigma$;
(ii) there exists a path $\sigma' \in \mathrm{Paths}_{p,r}$ such that $|\sigma|, |\sigma'| \leq |Q|$ and $\Delta^+(\pi) > c - a$, where $\pi \coloneqq \sigma'\sigma$;
(iii) there is a positive (a, p, r)-admissible cycle θ.

It remains to show that the conditions can be tested in NC^2. There are $|Q|$ choices for state r, so we can test the conditions for all choices in parallel. Let $S \coloneqq \{\sigma \in \mathrm{Paths}_{r,q} \mid \Delta^+(\sigma) > 0 \text{ and } \Delta^-(\sigma) = 0\}$. We first check whether $S \neq \emptyset$, which can be done in NC^2 by Lemma 2(a)–(b). Moreover, condition (iii) can be checked in NC^2 by Lemma 11. We proceed as follows to check (i). Let

$$W \coloneqq \{\Delta^+(\sigma) \mid \sigma \in S, |\sigma| \leq |Q|\},$$
$$W' \coloneqq \{\Delta^+(\sigma') \mid \sigma' \in \mathrm{Paths}_{p,r}, |\sigma'| \leq |Q|, \Delta^-(\sigma') = 0\}.$$

By Lemma 2, we can compute $m \coloneqq \max W + \max W'$ in NC^2 and check that $m \geq c - a$. Lastly, we define $W'' \coloneqq \{\Delta^+(\sigma') \mid \sigma' \in \mathrm{Paths}_{p,r}, |\sigma'| \leq |Q|\}$ and test whether $\max W + \max W'' > c - a$ to verify condition (ii). □

Theorem 16. *Given $a, a' \in \mathbb{Z}$ and $p, q \in Q$, the following can be done in NC^2: obtaining a representation of* $\mathrm{Post}_{p,q}(a)$ *and testing whether* $a' \in \mathrm{Post}_{p,q}(a)$.

Proof. By Proposition 12, we can compute $b \coloneqq \inf \overline{\mathrm{Post}_{p,q}(a)}$ and $c \coloneqq \sup \overline{\mathrm{Post}_{p,q}(a)}$ in NC^2. By Corollary 7 and Proposition 15, the set $S \coloneqq \overline{\mathrm{Post}_{p,q}(a)} \setminus \mathrm{Post}_{p,q}(a)$, of size at most three, can be computed in NC^2. By Corollary 7, this yields the representation $\mathrm{Post}_{p,q}(a) = [b, c] \setminus S$. Thus, $a' \in \mathrm{Post}_{p,q}(a)$ iff $b \leq a' \leq c$ and $a' \notin S$. □

3.5 Equality Tests

A *GG-COCA with equality tests* is a tuple $\mathcal{V} = (Q, T, \tau, \phi)$, where (Q, T, τ) is a GG-COCA and $\phi\colon Q \to \{[z, z] \mid z \in \mathbb{Q}\} \cup \{(-\infty, \infty)\}$. We say that a run of $\mathcal{V}$ is *admissible* if each of its configurations $q(a)$ satisfies $a \in \tau \cap \phi(q)$.

Using the previous results, we can extend the NC^2 membership of the reachability problem $p(a) \to_* q(b)$ to GG-COCA with equality tests. The proof relies on the fact that each equality test is passed by exactly one configuration. For this reason, we can construct a reachability graph between equality tests using a quadratic number of GG-COCA reachability queries.

Let us assume that p has no incoming edges; if it does, we can simply add a new initial state p' and a single transition $(p', 0, p)$. Similarly, we can assume q has no outgoing edges.

We will reason about reachability in $\mathcal{V}$, where we avoid all equality tests. For every pair of states $p', q' \in Q$, let us define the GG-COCA $\mathcal{V}_{p',q'} \coloneqq (Q_{p',q'}, T_{p',q'})$, where $Q_{p',q'} \coloneqq \{s \in Q \mid \phi(s) = \mathbb{Q}\} \cup \{p', q'\}$. We treat p' as a dedicated input state, and q' as a dedicated output state. That is,

$$T_{p',q'} \coloneqq \{t \in T \mid \mathrm{in}(t) \in Q_{p',q'} \setminus \{q'\}, \mathrm{out}(t) \in Q_{p',q'} \setminus \{p'\}\}.$$

Let us define a graph $\mathcal{G} \coloneqq (V, E)$, where $V \coloneqq \{p(a), q(b)\} \cup \{r(z) \mid r \in Q, \phi(r) = [z, z], z \in \tau\}$. If $a \notin \tau \cap \phi(p)$ or $b \notin \tau \cap \phi(q)$, then we trivially conclude that $p(a)$ cannot reach $q(b)$. Hence, $|V| \leq |Q|$ holds. Intuitively, the nodes of $\mathcal{G}$ correspond to the initial and final configurations plus, for each equality test, the configuration that passes this test. Let us define $E \coloneqq \{(p'(x), q'(y)) \mid p'(x) \to_* q'(y) \text{ in } \mathcal{V}_{p',q'}\}$.

Lemma 17. *It is the case that $p(a) \to_* q(b)$ in $\mathcal{V}$ if and only if there is a path from $p(a)$ to $q(b)$ in $\mathcal{G}$.*

Proof. We show the "if" direction first. Assume that $p(a) \to_\rho q(b)$ for some run ρ. Without loss of generality, we assume that no configuration repeats when starting at $p(a)$ with ρ; otherwise, we can simply shorten ρ. Let $\sigma_1\sigma_2 \cdots \sigma_n$ be the unique maximal decomposition of ρ into runs such that $\phi(\mathrm{out}(\sigma_i)) \neq \mathbb{Q}$ for all $1 \leq i < n$. For ease of notation, let $q_i \coloneqq \mathrm{out}(\sigma_i)$. It holds that $p(a) \to_{\sigma_1} q_1(a_1) \to_{\sigma_2} q_2(a_2) \cdots \to_{\sigma_n} q(b)$, where $\phi(q_i) = [a_i, a_i]$ for all i. Since $\sigma_1\sigma_2 \cdots \sigma_n$ is the maximal decomposition, $\phi(\mathrm{out}((\sigma_i)_j)) = \mathbb{Q}$ holds for all $j < |\sigma_i|$.

Additionally, recall that p has no incoming edges and q has no outgoing edges. Hence, the following holds:

$$\begin{aligned} p(a) &\to_{\sigma_1} q_1(a_1) \text{ in } \mathcal{V}_{p,q_1}, \\ q_{i-1}(a_{i-1}) &\to_{\sigma_i} q_i(a_i) \text{ in } \mathcal{V}_{q_{i-1},q_i} \text{ for all } 1 < i < n, \\ q_{n-1}(a_{n-1}) &\to_{\sigma_n} q(b) \quad \text{ in } \mathcal{V}_{q_{n-1},q}. \end{aligned}$$

We are done as $p(a)q_1(a_1) \cdots q_{n-1}(a_{n-1})q(b)$ is a path of $\mathcal{G}$.

It remains to show the "only if" direction. Suppose there is a path $p(a)q_1(a_1) \cdots q_{n-1}(a_{n-1})q(b)$ in $\mathcal{G}$. Note that if $p'(a') \to_* q'(b')$ in $\mathcal{V}_{p',q'}$, then by definition we also have $p'(a') \to_* q'(b')$ in $\mathcal{V}$.

So we have

$$p(a) \to_* q_1(a_1) \to_* \cdots \to_* q_{n-1}(a_{n-1}) \to_* q(b) \text{ in } \mathcal{V}. \qquad \square$$

THEOREM 18. *The reachability problem for GG-COCA with equality tests is in NC^2.*

PROOF. Let us first argue that the graph $\mathcal{G}$ can be constructed in NC^2. There are at most $|Q|$ nodes in $\mathcal{G}$, and hence at most $|Q|^2$ edges. Note that an edge $(p'(x), q'(y))$ is present iff $p'(x) \to_* q'(y)$ in $\mathcal{V}_{p',q'}$, which can be decided in NC^2 by Theorem 16, as $\mathcal{V}_{p',q'}$ is a GG-COCA. By running these $|Q|^2$ queries in parallel, it follows that $\mathcal{G}$ can be obtained in NC^2.

Once the graph $\mathcal{G}$ has been constructed, by Lemma 17, it suffices to test reachability from $p(a)$ to $q(b)$ in $\mathcal{G}$. Since graph reachability is in NL $\subseteq$ NC^2, we are done. $\square$

4 COCA REACHABILITY

We now turn to the reachability problem $p(a) \to_* q(b)$ for a COCA $\mathcal{W} = (Q, T, \tau)$. In contrast to GG-COCA, the set $\mathrm{Post}_{p,q}(a)$ does not necessarily admit a decomposition into a constant number of intervals. Nevertheless, we show that it can always be decomposed into a polynomial number of intervals with respect to the number of states (see Section 4.1). Then, we present a formalization of the natural forward computation one would employ to obtain under-approximations of the reachability function (see Section 4.2), which can be efficiently stored due to the aforementioned fact. Finally, in an approach reminiscent of the Bellman-Ford algorithm, we introduce a way of "accelerating" our forward computation of under-approximations so as to reach a fixed point in finite time. Concretely, in Section 4.3, we give sufficient (and efficient-to-check) conditions for the existence of certain cycles. We then propose an acceleration scheme based on those cycles. Our polynomial-time algorithm is summarized in Section 4.4.

Throughout this section, we write $\mathcal{I}(R)$ to denote the unique decomposition of a set $R \subseteq \mathbb{Q}$ into maximal disjoint nonempty intervals. For example, $\mathcal{I}([3,4] \cup (4,5) \cup (5,+\infty)) = \{[3,5), (5,+\infty)\}$ and $\mathcal{I}(\emptyset) = \emptyset$.

4.1 Controlling the Number of Intervals

We will prove that for every k the set $\{b \in \mathbb{Q} \mid p(a) \to_\rho q(b), |\rho| = k\}$ decomposes into at most $|Q|^{O(1)}$ intervals. (Note that the bound is independent of k.) To do so, we will bound the size of the decomposition of sets obtained by updating $A = [a, a]$ with operations that suffice to implement continuous counter updates and guard tests. More precisely, these are Minkowski sums, intersections (with elements of $\mathcal{L} = \{\tau(q) \mid q \in Q\}$), and unions (with sets constructed similarly). For technical reasons, we also consider a fourth operation.

Let us fix a bounded interval $A \in \Gamma$ and $\mathcal{L} \subseteq \Gamma$. We write $P_{\mathcal{L}} := \{\inf I, \sup I \mid I \in \mathcal{L}\}$ to denote the set of endpoints within $\mathcal{L}$. Further, for two sets $B, B' \subseteq \mathbb{Q}$, we write $B + B' = \{b + b' \mid b \in B, b' \in B'\}$ to mean the *Minkowski sum* of B and B'. We define the *MIUN-closure* (short for Minkowski sum, Intersection, Union, and New), of interval A w.r.t. $\mathcal{L}$, as the smallest collection $C \subseteq 2^{\mathbb{Q}}$ such that $A \in C$ and

- M: if $B \in C$ and $z \in \mathbb{Q}_{>0}$, then $B + (0, z], B + [-z, 0) \in C$;
- I: if $B \in C$ and $L \in \mathcal{L}$, then $B \cap L \in C$;
- U: if $B, B' \in C$, then $B \cup B' \in C$;
- N: if $B \in C$ and $I \in \Gamma$ s.t. $\overline{I} \cap P_{\mathcal{L}} \neq \emptyset$, then $B \cup I \in C$.

The forthcoming lemma forms the basis of our bound. It is based on so-called *indicator functions* that give us, for every interval I, the set of endpoints of $\mathcal{L}$ and A that belong to the closure of I. As we will see later, the set of endpoints needed to analyze COCA is small. Furthermore, all

MIUN-operations are such that sets of $\mathcal{C}$ decompose into intervals whose closure contains at least one such endpoint.

More formally, for all $B \in \mathcal{C}$, let $\phi_B \colon \mathcal{I}(B) \to 2^{P_{\mathcal{L}} \cup P_A}$ be the function defined as $\phi_B(I) \coloneqq \overline{I} \cap (P_{\mathcal{L}} \cup P_A)$, where $P_A = \{\inf A, \sup A\}$.

Lemma 19. *We have $\phi_B(I) \neq \emptyset$ for all $B \in \mathcal{C}$ and $I \in \mathcal{I}(B)$.*

Proof. We proceed by induction on the definition of MIUN-closures. We define $\mathcal{C}_0 \coloneqq \{A\}$ and $\mathcal{C}_{i+1}$ as $\mathcal{C}_i$ extended with all sets obtained by applying the MIUN-operations applied to any $B, B' \in \mathcal{C}_i$. We will show that the lemma holds for all $\mathcal{C}_i$, which will conclude the proof since $\mathcal{C} = \bigcup_{i \in \mathbb{N}} \mathcal{C}_i$.

We have $\mathcal{I}(A) = \{A\}$ and the claim holds since $P_A \subseteq \phi_A(A)$. For the induction step, we suppose the claim holds for $\mathcal{C}_i$. We have to prove that for all $C \in \mathcal{C}_{i+1}$ and all $I \in \mathcal{I}(C)$ it holds that $\phi_C(I) \neq \emptyset$. Notice that this is trivial if C is obtained from $B \in \mathcal{C}_i$ by application of the New operation.

First, we consider the Minkowski sum. Consider some $B \in \mathcal{C}_i$ with the function ϕ_B and let $I \in \{(0, z], [-z, 0)\}$ for some $z \in \mathbb{Q}_{>0}$. Let $C \coloneqq B + I$. For all $J \in \mathcal{I}(C)$ there exists $K_B \in \mathcal{I}(B)$ such that $K_B \subseteq J$. Thus, $\phi_B(K_B) \subseteq \phi_C(J)$ and the claim holds by the inductive hypothesis for ϕ_B.

Second, we consider intersections. We only deal with intervals of the form $[\ell, +\infty)$, $(\ell, +\infty)$, $(-\infty, \ell)$, or $(-\infty, \ell]$, since intersection with any interval can be expressed by at most two consecutive intersections with intervals of this form. Let $B \in \mathcal{C}_i$ and $L \in \mathcal{L}$. Suppose that $\overline{L} = [\ell, +\infty)$ and let $C \coloneqq B \cap L$. Recall that $\ell \in P_{\mathcal{L}}$. Observe that $\mathcal{I}(B)$ contains at most two intervals I such that $\ell \in \overline{I}$. If such an I exists, then $\ell \in \phi_C(\overline{I \cap L})$. For all other intervals $J \in \mathcal{I}(B)$, we have that $J \cap L$ is either J or $\emptyset$. If the intersection is nonempty, then $\phi_C(J) = \phi_B(J)$ and the claim holds by inductive hypothesis. If $\overline{L}$ is instead of the form $(-\infty, \ell]$, then we proceed similarly.

Finally, we consider unions. Let $B, B' \in \mathcal{C}_i$ and $I \in \mathcal{I}(B \cup B')$. By definition, there exists $J \in \mathcal{I}(B) \cup \mathcal{I}(B')$ with $J \subseteq I$. Therefore, either $\phi_B(J)$ or $\phi_{B'}(J)$ is nonempty and contained in $\phi_{B \cup B'}(I)$. □

Lemma 20. *For every set $B \subseteq \mathbb{Q}$ and every pairwise distinct interval $I_1, I_2, I_3 \in \mathcal{I}(B)$, it is the case that $\overline{I_1} \cap \overline{I_2} \cap \overline{I_3} = \emptyset$.*

A point can belong to at most one interval among disjoint intervals. Moreover, a point can belong to at most two closures; e.g., consider $[0, 1)$ and $(1, 2]$. This is no longer possible for three intervals due to maximality of intervals in $\mathcal{I}(B)$. Thus, the proof of Lemma 20 follows from a simple case analysis.

Now, we show that if $\mathcal{L}$ is finite, and then there is a polynomial bound on the number of intervals within the decomposition of any set from the MIUN-closure $\mathcal{C}$. More formally:

Lemma 21. *If $\mathcal{L}$ is finite, then $\mathcal{I}(B)$ consists of at most $4(|\mathcal{L}| + 1)$ intervals, for every $B \in \mathcal{C}$.*

Proof. By Lemma 20, there are at most two pairwise disjoint intervals that share a point in their closure. By Lemma 19, the indicator function guarantees that $\overline{J} \cap (P_{\mathcal{L}} \cup P_A) \neq \emptyset$ for all $J \in \mathcal{I}(B)$. Thus, $\mathcal{I}(B)$ has at most $2(2|\mathcal{L}| + 2)$ intervals. Otherwise, by the pigeonhole principle, a point of $P_{\mathcal{L}} \cup P_A$ would belong to at least three closures of intervals from $\mathcal{I}(B)$. □

4.2 Approximations of the Reachability Function

It will be convenient to manipulate mappings from states to (under-approximations of) their reachability functions. We consider the mappings $\mathcal{R}_Q \coloneqq \{S \colon Q \to 2^{\mathbb{Q}}\}$. An example of such a mapping is $\mathrm{Reach}_{p(a)}$, defined as $\mathrm{Reach}_{p(a)}(q) \coloneqq \mathrm{Post}_{p,q}(a)$. Given $S, S' \in \mathcal{R}_Q$, we write $S \preceq S'$ iff $S(q) \subseteq S'(q)$ for all $q \in Q$. We seek to define a sequence of mappings $S_0 \preceq S_1 \preceq \cdots$ such that $S_n = \mathrm{Reach}_{p(a)}$ for some $n \in \mathbb{N}$.

For all states $q \in Q$, we define the *successor mapping-update function* $\mathrm{Succ}_q \colon \mathcal{R}_Q \to 2^{\mathbb{Q}}$ as follows:

$$\begin{aligned}\mathrm{Succ}_q(S) \coloneqq\ & S(q)\\ & \cup \bigcup \{(S(r) + (0, z]) \cap \tau(q) \mid (r, z, q) \in T, z > 0\}\\ & \cup \bigcup \{(S(r) + [z, 0)) \cap \tau(q) \mid (r, z, q) \in T, z < 0\}\\ & \cup \bigcup \{S(r) \cap \tau(q) \mid (r, 0, q) \in T\}.\end{aligned}$$

Let $\mathrm{Succ} \colon \mathcal{R}_Q \to \mathcal{R}_Q$ be defined as $\mathrm{Succ}(S)(q) \coloneqq \mathrm{Succ}_q(S)$. Below, we state the key property enjoyed by Succ. In words, its i-fold composition coincides with the set of configurations reachable via runs of length at most i. It can easily be proven by induction on the definition of Succ.

Lemma 22. *Let $S_0 \in \mathcal{R}_Q$ and $S_i \coloneqq \mathrm{Succ}(S_{i-1})$ for all $i \geq 1$. The following holds:*

$$S_i(q) = \bigcup_{p \in Q} \{b \in \mathbb{Q} \mid a \in S_0(p), p(a) \to_\rho q(b) \text{ and } |\rho| \leq i\}.$$

Now we can state a proposition that shows how the previous section relates to these definitions. Let us fix a configuration $p(a)$. We will focus on the MIUN-closure C of $A \coloneqq [a, a]$ with respect to $\mathcal{L} \coloneqq \{\tau(q) \mid q \in Q\}$. We say that a mapping $S \in \mathcal{R}_Q$ is *C-valid* if $S(q) \in C$ for all $q \in Q$.

Proposition 23. *Let $S \in \mathcal{R}_Q$ be a C-valid mapping. We have $S \preceq \mathrm{Succ}(S)$, and $\mathrm{Succ}(S)$ is a C-valid mapping. Moreover, for every $q \in Q$, if $b \in \mathrm{Succ}(S)(q) \setminus S(q)$, then there exists $r(c)$ such that $c \in S(r)$ and $r(c) \to_t q(b)$ for some transition t.*

Proof. From the definition of Succ, we have that the following holds for all $q \in Q$. If $b \in \mathrm{Succ}(S)(q) \setminus S(q)$, then there exists $r(c)$ such that $c \in S(r)$ and $r(c) \to_t q(b)$ for some transition t. Hence, $S \preceq \mathrm{Succ}(S)$ follows directly from the definition of Succ. To prove that $\mathrm{Succ}(S)(q) \in C$, it suffices to observe that $\mathrm{Succ}_q(S)$ is defined using Minkowski sums, intersections, and unions, which are building blocks of MIUN-closures. □

4.3 Accelerations

Unfortunately, applying Succ might not give us $\mathrm{Reach}_{p(a)}$ in a small or even finite number of steps, e.g., if $\mathrm{Reach}_{p(a)}(q)$ is unbounded for some $q \in Q$. We introduce another operation on mappings to resolve this. We start by defining some special form of cycles.

Let us fix a mapping $S_0 \in \mathcal{R}_Q$ and let $S_{i+1} \coloneqq \mathrm{Succ}(S_i)$ for every $i \geq 0$. We say that a run $\rho = \alpha_1 t_1 \cdots \alpha_n t_n$ is a *positively expanding cycle* from S_0 if it is admissible and there exist configurations $p_0(a_0), p_1(a_1), \ldots, p_n(a_n)$ such that

(1) $p_0 = p_n$ and $\Delta(\rho) > 0$;
(2) $a_0 \in S_0(p_0)$ and $p_0(a_0) \to_{\rho[..i]} p_i(a_i)$ for all $i \geq 1$; and
(3) $a_i \in S_i(p_i) \setminus S_{i-1}(p_i)$ for all $i \geq 1$.

Moreover, letting $I_0, \ldots, I_n$ be the sequence of intervals such that $a_i \in I_i \in \mathcal{I}(S_i(p_i))$ for all $i \in \{0, \ldots, n\}$, we require

(4) $I_0 \subseteq I_n$;
(5) for all $i \geq 1$, there is a unique interval $I'_i \in \mathcal{I}(S_{i-1}(p_i))$ such that $I'_i \subseteq I_i$; and
(6) $a_i \geq \sup(I'_i)$ for every $i \geq 1$.

Intuitively, the third condition states that each a_i is a "new" value, and the fifth and sixth conditions state that a_i expands some interval toward the top.

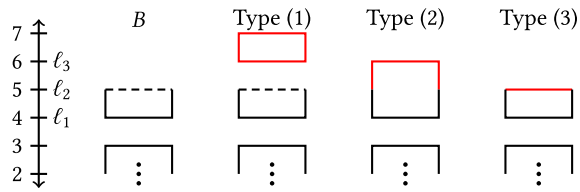


Fig. 2. Left: A set B such that $\mathcal{I}(B) = \{(-\infty, 3], [4, 5)\}$. Right: Example of the three possible types of progressing extensions of B. Dashed lines denote open interval borders; $\ell_1 = 4, \ell_2 = 5$, and $\ell_3 = 6$ denote values in $P_{\mathcal{L}} \cup P_A$.

For example, consider a state $q \in Q$ with guard $\tau(q) = [0, \infty)$, a self-loop $\rho = (q, 1, q)$, and a mapping S_0 s.t. $S_0(q) = \{0\}$. The sequence of configurations we get from following ρ is $q(0)q(1)$. By applying Succ, we get that $S_1(q) = [0, 1]$. It is easy to see that the first two conditions are met. For the third condition, we have that $1 \in [0, 1] \setminus \{0\}$. According to our definition, we let $I_0 = [0, 0]$ and $I_1 = [0, 1]$. Since $I_0 \subseteq I_1$, conditions four and five are met, where $I'_1 = I_0$. Finally, we have that $1 > \sup(I_0)$. Thus, ρ is a positively expanding cycle.

Similarly, we say that ρ is a *negatively expanding cycle* from S_0 if in the first item we replace $\Delta(\rho) > 0$ with $\Delta(\rho) < 0$, and in the last item we replace $a_i \geq \sup(I'_i)$ with $a_i \leq \inf(I'_i)$.

The following property follows from the definitions:

Lemma 24. *It holds that* $a_0, a_n \in I_n \subseteq \mathrm{Reach}_{p_0(a_0)}(p_0)$.

It transpires that the Succ function always yields expanding cycles after a polynomial number of applications. The proof of this claim relies on our bounds for interval decompositions of sets from the MIUN-closure. We will also need to define a measure on the mappings from states to interval decompositions, which progresses with an increasing number applications of Succ, and finally leads us to find an expanding cycle.

Let C be the MIUN-closure of A w.r.t. $\mathcal{L}$, and let $B, B' \in C$ be such that $B \subseteq B'$. We say that B' is a *progressing extension* of B if

(1) there is $I' \in \mathcal{I}(B')$ such that $B \cap I' = \emptyset$;

or if there are $I \in \mathcal{I}(B)$ and $I' \in \mathcal{I}(B')$ such that $I \subseteq I'$ and at least one of the following holds:

(2) either $\phi_{B'}(I') \setminus \phi_B(I) \neq \emptyset$, or
(3) there exists $\ell \in \phi_B(I)$ such that $\ell \notin I$ and $\ell \in I'$.

See Figure 2 for a pictorial description of progressing extensions. Observe that in case (3) we necessarily have that $\ell \in \overline{I}$.

Let us prove a bound on the number of progressing extensions in any $\subseteq$-increasing sequence.

Lemma 25. *Let* $B_0, B_1, B_2, \ldots \in C$ *be a sequence such that* $B_i \subseteq B_{i+1}$ *for all* $i \in \mathbb{N}$. *The set of* $i \in \mathbb{N}$ *such that* B_{i+1} *is a progressing extension of* B_i *has cardinality at most* $|\mathcal{L}|^{O(1)}$.

Proof. Let $P \coloneqq P_{\mathcal{L}} \cup P_A$. First, observe that (3) can happen only if there exists some $\ell \in P$ such that $\ell \in B_{i+1} \setminus B_i$, and thus at most $|P|$ times.

Let $\phi_{B_i}(\mathcal{I}(B_i)) \subseteq P$ be the image of all intervals of $\mathcal{I}(B_i)$. Note that $\phi_{B_i}(\mathcal{I}(B_i)) \subseteq \phi_{B_{i+1}}(\mathcal{I}(B_{i+1}))$. A strict inclusion can happen at most $|P|$ times. Thus, we can assume that $\phi_{B_i}(\mathcal{I}(B_i)) = \phi_{B_{i+1}}(\mathcal{I}(B_{i+1}))$. Note that (1) can happen at most $|P|$ times due to Lemma 20 and because $\phi_{B_{i+1}}(I') \neq \emptyset$ for all $I' \in \mathcal{I}(B_{i+1})$. Indeed, for all $\ell \in P$, Lemma 20 tells us there are no pairwise distinct intervals $I_1, I_2, I_3 \in \mathcal{I}(B_{i+1})$ such that $\ell \in \phi_{B_{i+1}}(I_1)$, $\ell \in \phi_{B_{i+1}}(I_2)$, and $\ell \in \phi_{B_{i+1}}(I_3)$.

Now, assume that (1) and (3) are not the case and that $\phi_{B_i}(\mathcal{I}(B_i)) = \phi_{B_{i+1}}(\mathcal{I}(B_{i+1}))$. Since (1) does not hold, we have $|\mathcal{I}(B_{i+1})| \leq |\mathcal{I}(B_i)|$. Note that a strict inequality can happen at most $|P|$ times, so we can assume that $|\mathcal{I}(B_{i+1})| = |\mathcal{I}(B_i)|$. We define a function $f\colon \mathcal{I}(B_i) \to \mathcal{I}(B_{i+1})$. Recall that $B_i \subseteq B_{i+1}$. So, for every $I \in \mathcal{I}(B_i)$ there exists a unique $f(I) \in \mathcal{I}(B_{i+1})$ such that $I \subseteq f(I)$. Uniqueness follows from maximality of intervals within $\mathcal{I}(B_{i+1})$. Then, we have that $\overline{I} \subseteq \overline{f(I)}$ and therefore $\overline{I} \cap P \subseteq \overline{f(I)} \cap P$. It follows that $\phi_{B_i}(I) \subseteq \phi_{B_{i+1}}(f(I))$. Since (1) does not hold, the function f is a surjection. Moreover, $\phi_{B_i}(I) = \phi_{B_{i+1}}(f(I))$. Thus, (2) can happen at most $|P|$ times by Lemma 20. □

Now, let us show an interesting property of extensions that are *not* progressing:

Lemma 26. *Let $B, B' \in C$ be such that $B \subseteq B'$, where B' is not a progressing extension of B. There is a bijection $f\colon \mathcal{I}(B) \to \mathcal{I}(B')$ s.t. $\phi_B(I) = \phi_{B'}(f(I))$ for all $I \in \mathcal{I}(B)$.*

Proof. Since $B \subseteq B'$, for all $I \in \mathcal{I}(B)$ there is a unique $f(I) \in \mathcal{I}(B')$ such that $I \subseteq f(I)$. We show that if f is not a bijection, then it will contradict that $B \subseteq B'$ is not a progressing extension.

First, we prove that f is an injection. Suppose this is not the case and that $f(I) = f(J)$ for some $I, J \in \mathcal{I}(B)$. Then $\phi_B(I) \cup \phi_B(J) \subseteq \phi_{B'}(f(I))$. If there exists $\ell \in \phi_B(I) \cap \phi_B(J)$, then it must be the case that $\ell \notin I, \ell \notin J$, and $\ell \in f(I)$. This is a contradiction because the extension is progressing due to (3). Otherwise, there is $\ell \in \phi_B(I) \setminus \phi_B(J)$. Since $\phi_B(I) \subseteq \phi_B(I) \cup \phi_B(J) \subseteq \phi_{B'}(f(I))$, we get a contradiction because the extension is progressing due to (2).

Now, we prove that f is a surjection. If we suppose this does not hold, then there is an interval $I' \in \mathcal{I}(B')$ such that $f^{-1}(I') = \emptyset$. Thus, $I' \cap B = \emptyset$, which is a contradiction because this would mean the extension is progressing due to (1).

To conclude, we note that for all $I \in \mathcal{I}(B)$, since $I \subseteq f(I)$, we have that $\overline{I} \subseteq \overline{f(I)}$. It follows that $\overline{I} \cap (P_{\mathcal{L}} \cup P_A) \subseteq \overline{f(I)} \cap (P_{\mathcal{L}} \cup P_A)$. Hence, by definition of the indicator functions, we have that $\phi_B(I) \subseteq \phi_{B'}(f(I))$. If the inclusion is strict for some I, then we get a contradiction because the extension is progressing due to (2). □

Now, we are able to prove the following proposition. The computational part is a simple backward construction of a run containing a cycle.

Proposition 27. *Let $S_0 \in \mathcal{R}_Q$ let $S_{i+1} \coloneqq \mathrm{Succ}(S_i)$ for all $i \in \mathbb{N}$. For some n polynomially bounded in $|Q|$, at least one of the following holds:*

- *$S_n = S_{n+1}$,*
- *there is a positively expanding cycle ρ from S_n, or*
- *there is a negatively expanding cycle ρ from S_n.*

Moreover, it can be determined in time $|Q|^{O(1)}$ whether the second or third case hold, and then ρ and its witnessing configurations can be computed in time $|Q|^{O(1)}$.

Proof. The value of the polynomially bounded number n will be determined by the proof. By definition, we have $S_i \preceq S_{i+1}$ for all $i \in \mathbb{N}$. By Lemma 25, there is a polynomial number of indices i (w.r.t. $|Q|$) such that $S_i(q) \subseteq S_{i+1}(q)$ is a progressing extension for some $q \in Q$. Thus, there exists an index j polynomial in $|Q|$ such that the extensions $S_i(q) \subseteq S_{i+1}(q)$ are not progressing for all $j \leq i \leq j + k$, where k is sufficiently large (but polynomially bounded in $|Q|$). To simplify the notation we will assume that $j = 0$ and consider $S_0 \preceq S_1 \preceq \cdots \preceq S_k$.

By Lemma 26, there is a bijection $f_i\colon \mathcal{I}(S_i(q)) \to \mathcal{I}(S_{i+1}(q))$ for every $q \in Q$ and $0 \leq i < k$. Thus, for every $q \in Q$, the sets $\mathcal{I}(S_1(q)), \ldots, \mathcal{I}(S_k(q))$ have the same number m_q of intervals. By Lemma 21, m_q is polynomially bounded in $|Q|$. Let us denote the intervals $I_1^{q,i}, \ldots, I_{m_q}^{q,i}$, where

$f_i(I_j^{q,i}) = I_j^{q,i+1}$ for all $0 \leq i < k$, $q \in Q$ and $1 \leq j \leq m_q$. By Lemma 26, $\phi_{S_i(q)}(I_j^{q,i}) = \phi_{S_{i+1}(q)}(I_j^{q,i+1})$. Since $S_i \preceq S_{i+1}$, we conclude that $I_j^{q,1} \subseteq I_j^{q,2} \subseteq \cdots \subseteq I_j^{q,k}$ for all $q \in Q$ and $1 \leq j \leq m_q$.

Recall that T is the set of transitions of the guarded COCA. Let $\text{active}(q,i,j) \coloneqq \{t \in T \mid I_j^{q,i} \cap \text{enab}(t) \neq \emptyset\}$. Since $I_j^{q,i} \subseteq I_j^{q,i+1}$, we have $\text{active}(q,i,j) \subseteq \text{active}(q,i+1,j)$, and a strict inclusion can occur at most $|T|$ times. Since it is polynomially bounded in $|Q|$ and $|T|$, we can assume that $\text{active}(q,i,j) = \text{active}(q,i+1,j)$ for all $q \in Q$, $0 \leq i < k$, and $1 \leq j \leq m_q$. Suppose $S_i \neq S_{i+1}$ for every $0 \leq i < k$. Then, we prove that there is a positively or negatively expanding cycle from S_0. Since $S_i \neq S_{i+1}$, there exists $v_k \in S_k(q_k)$ for some $q_k \in Q$ such that $v_k \notin S_{k-1}(q_k)$. Moreover, because $S_k = \text{Succ}(S_{k-1})$, we have that there exist $q_{k-1} \in Q$ and $v_{k-1} \in S_{k-1}(q_{k-1})$ such that $q_{k-1}(v_{k-1}) \rightarrow_t q_k(v_k)$ holds for some $t \in T$. Now, since $v_k \notin S_{k-1}(q_k)$, we necessarily have that $v_{k-1} \notin S_{k-2}(q_{k-1})$. Tracing back in this fashion, we can find a sequence of transitions $t_1, \ldots, t_k$, scalars $\alpha_1, \ldots, \alpha_k \in (0,1]$, and configurations $q_0(v_0), \ldots, q_k(v_k)$ such that $v_i \in S_i(q_i)$, $v_i \notin S_{i-1}(q_i)$, and $q_{i-1}(v_{i-1}) \rightarrow_{\alpha_i t_i} q_i(v_i)$ for all $0 < i \leq k$. We show that an infix of this sequence defines a positively or negatively expanding cycle.

Let j_k be the unique index with $v_k \in I_{j_k}^{q_k,k}$. By definition, we have $v_k \notin I_{j_k}^{q_k,k-1}$. If $v_k \geq \sup(I_{j_k}^{q_k,k-1})$, then we construct a positively expanding cycle, and otherwise we construct a negatively expanding one. We will prove only the former case; the latter case follows the same steps. We claim the following holds:

$$v_i \geq \sup\left(I_{j_i}^{q_i,i-1}\right) \text{ for all } 0 < i \leq k. \tag{1}$$

Let us argue that Equation (1) allows us to conclude. For a large enough k, we can find an infix $\rho \coloneqq q_a(v_a), \ldots, q_b(v_b)$ such that $a < b$, $q_a = q_b$, and $j_a = j_b$. The following inequalities thus hold:

$$v_b \geq \sup\left(I_{j_b}^{q_b,b-1}\right) \geq \sup\left(I_{j_a}^{q_a,a}\right) \geq v_a.$$

Since $v_a \in I_{j_a}^{q_a,a}$ and $v_b \notin I_{j_a}^{q_a,a}$, we obtain $\Delta(\rho) > 0$. The remaining conditions of the positively expanding cycle follow directly from the definition.

It remains to prove Equation (1). We proceed by induction, going from $i = k$ down to $i = 1$. The base case follows by assumption. For the inductive step, toward a contradiction, suppose that $v_i < \sup(I_{j_i}^{q_i,i-1})$. Observe that, by construction, $v_i \notin I_{j_i}^{q_i,i-1}$. Hence, it must also be the case that $v_i \leq \inf(I_{j_i}^{q_i,i-1})$. Recall that $\text{active}(q_i, i-1, j_i) = \text{active}(q_i, i, j_i)$. Since $v_i \rightarrow_{\alpha_{i+1}t_{i+1}} v_{i+1}$, there exists $\overline{v} \in I_{j_i}^{q_i,i-1}$ and $\beta \in (0,1]$ such that $\overline{v} \rightarrow_{\beta t_{i+1}} \overline{w}$ for some $\overline{w} \in S_i(q_{i+1})$.

We show that $v_{i+1} < \overline{w}$ holds for all choices of β. Note that if there exists some β such that $v_{i+1} = \overline{w}$, then $v_{i+1} \in S_i(q_{i+1})$ and we get a contradiction with the definition of v_{i+1}. It follows that either $v_{i+1} > \overline{w}$ for all choices of β or $v_{i+1} < \overline{w}$ for all choices of β. Let $t_{i+1} = (q_i, z_{i+1}, q_{i+1})$. Since $v_i \leq \inf(I_{j_i}^{q_i,i-1})$ and $\overline{v} \in I_{j_i}^{q_i,i-1}$, we have that

$$v_{i+1} = v_i + \alpha_{i+1}z_{i+1} \leq \overline{v} + \alpha_{i+1}z_{i+1}.$$

Thus, it must be the case that $v_{i+1} < \overline{w}$ for all choices of β.

We now prove that $\overline{w} \in I_{j_{i+1}}^{q_{i+1},i+1}$. Recall that $v_i, \overline{v} \in I_{j_i}^{q_i,i}$ and thus $[v_i, \overline{v}] \subseteq I_{j_i}^{q_i,i}$. Also, $[v_i + \alpha_{i+1}z_{i+1}, \overline{v} + \beta z_{i+1}] = [v_{i+1}, \overline{w}]$. Hence, for every $v_{i+1} \leq w' \leq \overline{w}$ there exists $v_i \leq v' \leq \overline{v}$ and $\gamma \in (0,1]$ such that $v' \rightarrow_{\gamma t_{i+1}} w'$. Thus, $\overline{w}$ and v_{i+1} belong to the same interval in $S_{i+1}(q_{i+1})$ as required.

Since $\overline{w} \in I_{j_{i+1}}^{q_{i+1},i+1}$ and $\overline{w} \in S_i(q_{i+1})$, we have $\overline{w} \in I_{j_{i+1}}^{q_{i+1},i}$. We have reached a contradiction, since by the inductive hypothesis we have

$$v_{i+1} \geq \sup\left(I_{j_{i+1}}^{q_{i+1},i}\right).$$

Observe that since Succ is easily computable in polynomial time, one can also find $q_0(v_0), \ldots, q_k(v_k)$ and their corresponding intervals in polynomial time. □

We are ready to define the *acceleration* operation. Let ρ be a positively or negatively expanding cycle from $S \in \mathcal{R}_Q$ and let $p_0(a_0), p_1(a_1), \ldots, p_n(a_n)$ be the configurations witnessing the run. Let $I_0, \ldots, I_n$ be the intervals given by the definition of expanding cycles. If ρ is positively expanding, then we define $\delta_i^+ := \sup \tau(p_i) - a_i$ for all $i \in \{1, \ldots, n\}$. If ρ is a negatively expanding cycle, then we define $\delta_i^- := a_i - \inf \tau(p_i)$. Let $j \in \{1, \ldots, n\}$ be such that

$$\delta_j^+ = \min\{\delta_i^+ \mid 1 \leq i \leq n\} \text{ or } \delta_j^- = \min\{\delta_i^- \mid 1 \leq i \leq n\}.$$

We define Acc so that, given ρ and the mapping S, it outputs a new mapping $\mathrm{Acc}(S, \rho) = S'$. If ρ is positively expanding from S, then $S'(q) := S(q)$ for all $q \neq p_j$ and

$$S'(p_j) := S(p_j) \cup \underbrace{I_j \cup \Big(\tau(p_j) \cap [a_j, +\infty)\Big)}_{=K \subseteq \mathbb{Q}}.$$

Recall that $a_j \in I_j$, and $a_j \in \tau(p_j)$ since $a_0 \to_{\rho[1..j]} a_j$, so K is an interval. Also, since ρ is positively expanding and $j \geq 1$, we have $a_j \notin S(p_j)$ and $S'(p_j) \setminus S(p_j) \neq \emptyset$. Similarly, if ρ is negatively expanding, then $S'(q) := S(q)$ for all $q \neq p_j$ and

$$S'(p_j) := S(p_j) \cup I_j \cup \Big((-\infty, a_j] \cap \tau(p_j)\Big).$$

Lemma 28. *Let $S \in \mathcal{R}_Q$ be a $\mathcal{C}$-valid mapping such that $S \preceq \mathrm{Reach}_{p(a)}$, and let ρ be a positively or negatively expanding cycle from S. If $S' = \mathrm{Acc}(S, \rho)$, then $S \preceq S'$, S' is a $\mathcal{C}$-valid mapping, and $S' \preceq \mathrm{Reach}_{p(a)}$. Moreover, for every $q \in Q$, if $b \in \mathrm{Acc}(S, \rho)(q) \setminus S(q)$, then there exists $r(c)$ such that $c \in S(r)$ and $r(c) \to_\pi q(b)$, where $\pi \in \rho^*$.*

Proof. We have $S \preceq S'$ directly from the definition of Acc. Similarly, S' is $\mathcal{C}$-valid because the operation to define $S'(p_j)$ is the "New" operation since the closure of the added interval always contains one of the endpoints from $\tau(p_j)$. It remains to prove that $S' \preceq \mathrm{Reach}_{p(a)}$.

We assume that $\Delta(\rho) > 0$; the proof is similar for the other case. Let $\rho = \alpha_1 t_1 \cdots \alpha_n t_n$. Let $p_0(a_0), p_1(a_1), \ldots, p_n(a_n)$ and $I_0, \ldots, I_n$ be the configurations and intervals given by the definition of positively expanding cycles. Let j be an index minimizing δ_j^+. We must prove that $S'(p_j) \preceq \mathrm{Reach}_{p(a)}$. Since $S \preceq \mathrm{Reach}_{p(a)}$ and $a_0 \in S(p_0)$ by definition of positively expanding cycles, it suffices to show that for every $b \in S'(p_j) \setminus S(p_j)$ there is an admissible run from $p_0(a_0)$ to $p_j(b)$.

If $\delta_j^+ = +\infty$, then $\sup \tau(p_i) = +\infty$ for all $i \in \{1, \ldots, n\}$. Since $\Delta(\rho) > 0$, for all $\alpha, \beta \in (0, 1]$ and $m \in \mathbb{N}$ the run $\rho' := (\beta\rho)^m \alpha \rho[1..j]$ is admissible from any $p_n(a')$ with $a' \geq a_n$ to state p_j. Note that $\Delta(\rho') = m\beta\Delta(\rho) + \alpha\Delta(\rho[1..j])$, which can be any positive rational number by properly choosing α, β, and m. Thus, $b \in \mathrm{Reach}_{p(a)}(p_j)$ for every $b > a_n$.

It remains to consider the case $a_j \leq a_n$ to prove the claim for every $b \in [a_j, a_n]$. Let $\varepsilon \in (0, a_n - a_0]$. Since $a_0 < a_0 + \varepsilon \leq a_n$, we have $a_0 + \varepsilon \in I_n \subseteq \mathrm{Reach}_{p(a)}(p_0)$, where the latter follows from Lemma 24. Note that ρ is admissible from all $p_n(a')$ with $a' \geq a_n$, and hence from $a_n + \varepsilon$. Thus, $p(a) \to_* p_0(a_0 + \varepsilon) \to_\rho p_n(a_n + \varepsilon)$, and $p_0(a_0 + \varepsilon) \to_{\rho[1..j]} p_j(a_j + \varepsilon)$. This shows that $b \in \mathrm{Reach}_{p(a)}(p_j)$ for all $b \in [a_j, a_n]$.

Now, suppose $\delta_j^+ < +\infty$. If $a_j = \sup \tau(p_j)$, then we are done because $a_j \in I_j \subseteq \mathrm{Reach}_{p(a)}(p_j)$. Otherwise, let $b \in [a_j, +\infty) \cap \tau(p_j)$. We need to prove that $b \in \mathrm{Reach}_{p(a)}(p_j)$. Note that, by definition, we have $0 \leq b - a_j \leq \delta_j^+$.

Let $m \in \mathbb{N}$ and $c \in \mathbb{Q}_{\geq 0}$ be the unique numbers that satisfy $b - a_j = m\Delta(\rho) + c$ and $c < \Delta(\rho)$. Since $a_0 \leq a_0 + c \leq a_0 + \Delta(\rho) = a_n$, then by Lemma 24 we conclude that $a_0 + c \in \mathrm{Reach}_{p(a)}(p_0)$. Notice that $a_j + c + m\Delta(\rho) = b$. It thus remains to prove that $p_0(a_0 + c) \to_{\rho^m \rho[1..j]} p_j(b)$. We prove something stronger, namely that $p_0(a_0 + c) \to_{\rho^{m+1}} p_n(a_n + b - a_j)$. Since $\Delta(\rho) > 0$, for the bottom

guards it suffices to check whether the configurations are large enough when ρ is applied the first time. Indeed, since ρ is admissible from $p_0(a_0)$, we get $a_i + c + \Delta(\rho_i) \geq a_i + \Delta(\rho_i) \geq \inf \tau(q_i)$. Similarly, for the top guards, it suffices to check whether the configurations are small enough when ρ is applied last.

Indeed, since $b - a_j \leq \delta_j^+$, we have $a_i + c + m\Delta(\rho) = a_i + b - a_j \leq a_i + \delta_j^+ \leq a_i + \delta_i^+ = \sup \tau(p_i)$. If $\sup \tau(p_j) \notin \tau(p_j)$, then $b - a_j < \delta_j^+$ and the previous inequalities are strict. □

4.4 Polynomial Time Algorithm

We summarize how to obtain the polynomial time algorithm for deciding $p(a) \rightarrow_* q(b)$. We begin with the mapping $R_0 \in \mathcal{R}_Q$ defined as $R_0(p) := [a, a]$ and $R_0(r) := \emptyset$ for every $r \neq p$. Clearly, $R_0 \leq \mathrm{Reach}_{p(a)}$. The next mappings $R_1, R_2, \ldots$ are defined as follows. Suppose we have defined $R_0, \ldots, R_i$. Let $S_0^i := R_i$ and $S_{j+1}^i := \mathrm{Succ}(S_j^i)$ for all $j \geq 0$.

By Proposition 27, we will either find an expanding cycle ρ from some S_n^i, where n is bounded polynomially, or we will find some $S_n^i = S_{n+1}^i$, again for n bounded polynomially. If there is an expanding cycle—a fact that, by Proposition 27, we can check in polynomial time—then we define $R_{i+j} := S_j^i$ for $1 \leq j < n$ and $R_{i+n} := \mathrm{Acc}(R_{i+n-1}, \rho)$. Otherwise, we define $R_{i+j} := S_j^i$ for $j \in \{1, \ldots, n\}$ and the algorithm returns R_{i+n}. By Lemmas 22 and 28, we have $R_i \leq \mathrm{Reach}_{p(a)}$ for all defined R_i. Hence, if the algorithm terminates, then by Lemma 22 it returns $\mathrm{Reach}_{p(a)}$.

The rest of this section is devoted to proving that the above-described algorithm has a polynomial worst-case running time. It suffices to argue that expanding cycles can only be found some polynomial number of times.

Proposition 29. *The algorithm computes a representation of* $\mathrm{Reach}_{p(a)}$ *in time* $|Q|^{O(1)}$.

By Proposition 27, it suffices to show that Acc can be applied at most polynomially many times. We show that accelerating leads to a progressing extension.

Lemma 30. *Let ρ be an expanding cycle from R and let $R' := \mathrm{Acc}(R, \rho)$. There is some state p_j such that $R(p_j) \subseteq R'(p_j)$ is a progressing extension.*

Proof. Let $p_j \in Q$ be such that $R'(p_j) = R(p_j) \cup I_j \cup J$ for some interval J. In the proof, we write $\alpha_i t_i$, $p_i(a_i)$, and I_i, as in the definition of expanding cycles. We will assume that ρ is positively expanding; the other case is similar. Thus, $J \in \{[a_j, g], [a_j, g), [a_j, +\infty)\}$, where $g := \sup \tau(p_j)$. Recall that by definition, $a_j \notin R(p_j)$ and $a_j \in I_j \in \mathcal{I}(\mathrm{Succ}^k(R)(p_j))$ for some k. Thus, $I_j \cup J$ is an interval.

If $(I_j \cup J) \cap R(p_j) = \emptyset$, then $R'(p_j)$ is a progressing extension due to (1). For the remaining case, let $b \in (I_j \cup J) \cap R(p_j)$ and $K \in \mathcal{I}(R(p_j))$ be such that $b \in K$. Note that $K \cup I_j \cup J$ is an interval. If $b < a_j$, then, because $a_j \notin R(p_j)$, either $g \notin K$ or K has an upper bound if $J = [a_j, +\infty)$. Thus, $R'(p_j)$ is a progressing extension due to (2) or (3). Finally, suppose that $b > a_j$. By definition of ρ, there is a unique interval $I_j' \in \mathcal{I}(\mathrm{Succ}^{k-1}(R)(p_j))$ such that $I_j' \subseteq I_j$. Moreover, $a_j \notin I_j'$ and $a_j \geq \sup I_j'$. Thus, $K \cap I_j'$ is empty and $g \notin I_j'$ or I_j' has an upper bound if $J = [a_j, +\infty)$. Since $K \cup I_j \cup J \in \Gamma$, $R'(p_j)$ is a progressing extension due to (2) or (3). □

5 PARAMETRIC COCA REACHABILITY

This section is dedicated to characterizing the complexity of existential reachability in parametric COCAs. We will show the following result:

Theorem 31. *The existential reachability problem for parametric COCAs is NP-complete.*

NP-membership will be based on results from Section 4 and heavily rely on the fact that reachability in COCAs can be expressed as reachability in one of many linear path schemes (see

Lemma 32), and that the reachability relation of such path schemes can be expressed as a small existential linear formula. In Section 5.1, we describe a straightforward formula for describing the reachability relation in simple paths. In Section 5.2, we similarly give a formula for the reachability relation of simple cycles, which requires a more careful analysis of the behavior of cycles in COCAs (see particularly Lemma 38). We conclude the proof of NP-membership in Section 5.3, where the full formula is assembled. We briefly switch to the setting where only updates can be parameterized in Section 5.4. We show that in this setting, any rational valuation of the parameters can be turned into an integral valuation of the parameters while maintaining reachability. Finally, we conclude the proof of NP-completeness in Section 5.5 by showing that existential reachability in parametric COCA is NP-hard, even when they are acyclic and parameters occur only on updates or only on guards. To do so, we give a reduction from 3-SAT.

For the rest of this section, let $\mathcal{V} = (Q, T, \tau, X)$ be a parametric COCA, and let $a, b \in \mathbb{Z}$. We will show that existential reachability from $p(a)$ to $q(b)$ can be witnessed by an existential linear formula φ of polynomial size. Membership in NP will follow from the fact that we can both guess φ and check satisfiability of φ in NP. This formula will be obtained based on the following corollary, which can be derived from the last section:

Lemma 32. *If $b \in \mathrm{Post}_{p,q}(a)$, then there exists a path π from some linear path scheme $\sigma_0\theta_0^*\sigma_1\theta_1^* \cdots \sigma_k$ such that $p(a) \to_\pi q(b)$, where*

- $k \leq |Q|^{O(1)}$,
- $|\sigma_i| \leq |Q|^{O(1)}$ *for each* $0 \leq i \leq k$, *and*
- $|\theta_j| \leq |Q|^{O(1)}$ *for each* $0 \leq j < k$.

Proof. The proof is by induction on the definition of the sequence of R_is. We argue that, for all $q \in Q$, if $b \in R_i(q)$, then there is a path π of the claimed form such that $p(a) \to_\pi q(b)$. Note that this is sufficient in view of Proposition 29.

For the induction, let us first focus on the last two items from the claim. The base case is trivial and the inductive step is a straightforward application of Proposition 23 and Lemma 28 (with Proposition 27 giving us the required polynomial bounds). Finally, we still need to argue that k is also polynomial. However, this follows from our induction on the sequence of R_is together with the polynomial bound on the length of the sequence established in Lemmas 30 and 25. □

In order to exploit Lemma 32, we will further need the forthcoming technical lemmas.

Lemma 33. *Let $t \in T$, $\alpha, \beta \in (0, 1]$ and $a, a', b, b' \in \mathbb{Q}$ be such that $a' = a + \alpha\Delta(t)$ and $b' = b + \beta\Delta(t)$. If $a \leq b$ and $a' \geq b'$, then $b' = a + \alpha'\Delta(t)$ and $a' = b + \beta'\Delta(t)$ for some $\alpha', \beta' \in (0, 1]$.*

Proof. The claim is trivial whenever $a = b$ or $a' = b'$; hence assume $a < b$ and $a' > b'$. Note that $\Delta(t) \neq 0$, as we would otherwise derive the contradiction $a < b = b' < a' = a$. Let

$$\alpha' := \frac{b' - a}{\Delta(t)} \text{ and } \beta' := \frac{a' - b}{\Delta(t)}.$$

We have $a + \alpha'\Delta(t) = b'$ and $b + \beta'\Delta(t) = a'$ as desired. It remains to show that $\alpha', \beta' \in (0, 1]$. From $a' > b'$, we have

$$\alpha' = (b' - a)/\Delta(t) < (a' - a)/\Delta(t) = \alpha \leq 1,$$
$$\beta' = (a' - b)/\Delta(t) > (b' - b)/\Delta(t) = \beta > 0.$$

From $a < b$, we symmetrically derive $\alpha' > 0$ and $\beta' \leq 1$. □

Corollary 34. *Let $a, a', b, b' \in \mathbb{Q}$, and let $\pi = t_1 \ldots t_n$ be a path such that $a \to_\pi a'$ and $b \to_\pi b'$. If $a \leq b$ and $a' \geq b'$, then $a \to_\pi b'$ and $b \to_\pi a'$.*

Proof. Let σ_1, σ_2 be runs such that $a \to_{\sigma_1} a'$ and $b \to_{\sigma_2} b'$. Let us define $a_i := a + \Delta(\sigma_1[1..i])$ and $b_i := b + \Delta(\sigma_2[1..i])$. In particular, $a_0 = a$, $b_0 = b$, $a_n = a'$, and $b_n = b'$. By assumption, it is the case that $a_0 \leq b_0$ and $a_n \geq b_n$. Clearly, if $a_0 = b_0$, then $a_0 \to_{\sigma_2} b'$ and we are done. So we can make the stronger assumption that $a_0 < b_0$.

Let i be the first index such that $a_i \geq b_i$. We have $i > 0$, as $a_0 < b_0$. By minimality of i, it holds that $a_{i-1} < b_{i-1}$. Thus, by Lemma 33, it holds that $a_{i-1} \to_{t_i} b_i$ and $b_{i-1} \to_{t_i} a_i$. Hence, we are done since $a \to_{\sigma_1[1..i-1]} a_{i-1} \to_{t_i} b_i \to_{\sigma_2[i+1..n]} b'$ and $b \to_{\sigma_2[1..i-1]} b_{i-1} \to_{t_i} a_i \to_{\sigma_1[i+1..n]} a'$. □

Lemma 35. *Let $\pi = t_1 \cdots t_n$ be a path, and let $a, b, a', b' \in \mathbb{Q}$ be such that $b \leq a < b' \leq a'$. If $a \to_\pi a'$ and $b \to_\pi b'$, then $a \to_\pi b'$.*

Proof. If $b' = a'$, we are done. So assume $b' < a'$. Let σ_1, σ_2 be runs with $\text{path}(\sigma_1) = \text{path}(\sigma_2) = \pi$ such that $a \to_{\sigma_1} a'$ and $b \to_{\sigma_2} b'$. Let us define $\beta := (b' - a)/(a' - a)$. As $b' > a'$, it holds that $\beta \in (0, 1]$. So we can define a new run $\sigma_1' := \beta\sigma_1$. Clearly, σ_1' has the right effect to go from a to b':

$$a + \Delta(\sigma_1') = a + \frac{b' - a}{a' - a}\Delta(\sigma_1) = a + \frac{b' - a}{a' - a}(a' - a) = b'.$$

It remains to show that σ_1' is an admissible run from a. Let $a_i := a + \Delta(\sigma_1[1..i])$, $a_i' := a + \Delta(\sigma_1'[1..i])$ and $b_i := b + \Delta(\sigma_2[1..i])$. If there exists i such that $a_i' \leq b_i$, then by Lemma 33, it follows that $a \to_{\pi[1..i]} b_i \to_{\sigma_2[i+1..n]} b'$, and so we are done. Thus, let us assume that $a_i' > b_i$ for all i. By the definition of σ_1', it is the case that $a_i \geq a_i'$. So, for all i, we have $a_i \geq a_i' \geq b_i$. Since σ_1 and σ_2 are respectively admissible runs from a and b, it follows that σ_1' is an admissible run from a. Thus, $a \to_{\sigma_1'} b'$, and hence we are done. □

For completeness, we note that we can also state and prove an equivalent formulation of the lemma when $b \geq a > b' \geq a'$.

5.1 Formula for σ

Let $I \in \Gamma_X$, let a be a number, and let μ be a valuation of X. Recall that Γ_X is the set of all intervals with endpoints from $\mathbb{Q} \cup \{-\infty, +\infty\} \cup X$. We write I^μ to mean I where each parameter $x \in X$ is replaced with $\mu(x)$. We define a formula $\phi_{\in I}(a, \mu)$ that is satisfied if and only if $a \in I$ under μ:

$$\phi_{\in I}(a, \mu) := (\inf I^\mu \prec_{\inf} a \prec_{\sup} \sup I^\mu),$$

where $\prec_{\inf} := <$ if the lower end of I is open, and $\prec_{\inf} := \leq$ otherwise. Similarly, we define $\prec_{\sup} := <$ if the upper end of I is open, and $\prec_{\sup} := \leq$ otherwise. For example, we have $\phi_{\in[4,9)}(a, \mu) = 4 \leq a < 9$. Note that this also generalizes to the presence of parameters in the endpoints of I. For example, if $x, y \in X$ are parameters, then $\phi_{\in(x,y]}(a, \mu) = \mu(x) < a \leq \mu(y)$.

Let $t = (q_0, z, q_1) \in T$ be a transition. We give a formula $\phi_t(a, b, \mu)$ that is satisfied iff $b \in \text{Post}_{t^\mu}(a)$:

$$\begin{aligned}\phi_t(a, b, \mu) := {} & [\phi_{\in\tau(q_0)}(a, \mu)] \wedge [z^\mu = 0 \to a = b] \wedge \\ & [z^\mu < 0 \to (a + z^\mu \leq b < a)] \wedge [z^\mu > 0 \to (a + z^\mu \geq b > a)] \wedge \phi_{\in\tau(q_1)}(b, \mu).\end{aligned}$$

Clearly, we can generalize the formula $\phi_t(a, b, \mu)$ to work for paths instead of transitions. Let $\pi = t_1 t_2 \cdots t_n$. We give a formula $\phi_\pi(a, b, \mu)$ as follows:

$$\phi_\pi(a, b, \mu) := \exists a_0, a_1, \ldots, a_n : (a_0 = a) \wedge (a_n = b) \wedge \bigwedge_{1 \leq i \leq n} \phi_{t_i}(a_{i-1}, a_i, \mu).$$

The following is straightforward:

Lemma 36. *It is the case that $\phi_\pi(a, b, \mu)$ holds iff $b \in \text{Post}_{\pi^\mu}(a)$. Furthermore, the size of ϕ_π is linear in $|\pi|$ and the sum of non-parametric endpoints and updates along π.*

5.2 Formula for θ^*

We define a formula $\phi_{\theta^*}(a, b, \mu)$ as follows:

$$\phi_{\theta^*}(a, b, \mu) \coloneqq (a = b) \vee \phi_{\theta^+}(a, b, \mu).$$

Intuitively, we differentiate two cases: To reach b from a, we iterate θ

- zero times: $a = b$,
- one or more times: $\phi_{\theta^+}(a, b, \mu)$.

It remains to give a formula for the case where we take θ one or more times. Formula $\phi_{\theta^+}(a, b, \mu)$ is split into two cases, based on whether $a < b$ or $b < a$. Note that we do not need to handle the case $a = b$, as this is trivially included in the case where we iterate zero times. So, we define

$$\phi_{\theta^+}(a, b, \mu) \coloneqq [a \neq b] \wedge [a < b \rightarrow \phi^+_{\theta^+}(a, b, \mu)] \wedge [a > b \rightarrow \phi^-_{\theta^+}(a, b, \mu)].$$

Formally, we require that if $a < b$, then $\phi^+_{\theta^+}(a, b, \mu)$ is satisfied iff $b \in \mathrm{Post}_{\theta\theta^+}(a) \cup \mathrm{Post}_{\theta}(a)$. The requirement for $\phi^-_{\theta\theta^+}(a, b, \mu)$, assuming $a > b$, is symmetric.

Let us give a necessary and sufficient condition for the membership of a value b' in $\mathrm{Post}_{\theta^\mu(\theta^\mu)^+}(a) \setminus \mathrm{Post}_{\theta^\mu}(a)$. We first proceed to give a series of technical lemmas.

Lemma 37. *Let $a < b$. If $a \rightarrow_{(\theta^\mu)^+} b$, then there exists $n \geq 1$ and values $a_0, \ldots, a_n$ such that $a_0 \rightarrow_{\theta^\mu} a_1 \rightarrow_{\theta^\mu} \cdots \rightarrow_{\theta^\mu} a_n$ and $a_i < b$ for all i.*

Proof. Let $\pi \in \theta^+$ be a path such that $a \rightarrow_\pi b$. Let n be the number of times π iterates θ, that is, $n = |\pi|/|\theta|$. By definition of π, n is a natural number. Let $a_i = a + \Delta(\pi[1..(i \cdot |\theta|)])$. We can assume that $a_i \neq b$ for all i; otherwise we can easily shorten π.

For the sake of contradiction, assume there exists some i such that $a_i > b$. Clearly, if such an i does not exist, we are done. So let i be the smallest such index. Note that $0 < i < n$, since $a < b$. By definition, $a_{i-1} \rightarrow_\theta a_i$. Further, by minimality of i, it holds that $a_{i-1} < b$. Recall that we have $a_{n-1} \rightarrow_\theta b$. In the following, we show that $a_{i-1} \rightarrow_{\theta^\mu} b$, which finishes the proof by minimality of i.

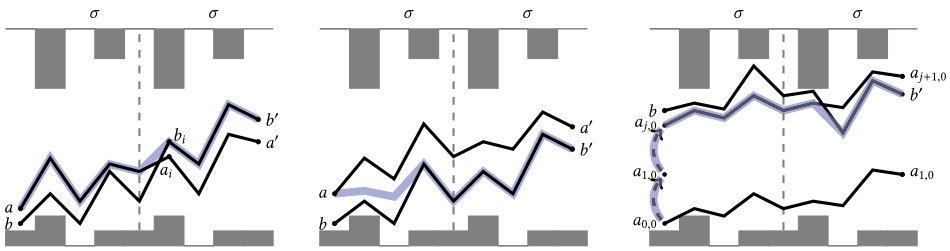
We distinguish three cases based on the order of a_{n-1} and a_{i-1}. First, if $a_{n-1} = a_{i-1}$, then we are clearly done. If $a_{n-1} < a_{i-1}$, then together with the fact that $b < a$, we invoke Lemma 35 to derive that $a_{i-1} \rightarrow_{\theta^\mu} b$. Lastly, if $a_{n-1} > a_{i-1}$, then together with the fact that $b < a_i$, it follows by Corollary 34 that $a_{i-1} \rightarrow_{\theta^\mu} b$. □

Now, we can prove a useful lemma that allows us to assume that iterations of the cycle behave in a monotonic manner:

Lemma 38. *Let $a < b$. If $a \rightarrow_{(\theta^\mu)^+} b$, then there exists $n \geq 1$ and values $a_0, \ldots, a_n$ such that $a_0 \rightarrow_{\theta^\mu} a_1 \rightarrow_{\theta^\mu} \cdots \rightarrow_{\theta^\mu} a_n$ and $a = a_0 < a_1 < \cdots < a_n = b$.*

Proof. Let $\pi \in (\theta^\mu)^+$ such that $a \rightarrow_\pi b$. Let n be the number of times π iterates θ. Note that $n \geq 1$, since π iterates θ at least once by membership in $(\theta^\mu)^+$. Let $a_0 \coloneqq a$, $a_n \coloneqq b$ and let a_i be the value reached after the ith iteration of θ when following π. We have $a_i \rightarrow_{\pi[1..|\theta|]} a_{i+1}$ for all $0 \leq i < n$. By Lemma 37, we can assume that $a_i \leq b$ for all i. Note that we can assume that $a_i \neq a_j$ for all i, j; otherwise we can trivially shorten π.

If $a_0 < a_1 < \cdots < a_n$, then we are done. So assume there exists i such that $a_i > a_{i+1}$. Let i be the smallest such index. Note that $i < n-1$, as we assume $a_i < b = a_n$ for all i. Let $j > i$ be the smallest index such that $a_j > a_i$. Note that such an index must exist, since $a_n > a_i$. Recall that $a_i \rightarrow_{\theta^\mu} a_{i+1}$ and $a_{j-1} \rightarrow_{\theta^\mu} a_j$. Further, by minimality of our choice of j, $a_{j-1} < a_i$ and $a_j > a_i > a_{i+1}$. Then, we can invoke Corollary 34, from which it follows that $a_i \rightarrow_{\theta^\mu} a_j$. Thus, we can shorten π by going



(a) Case where $b < a$, but there exists i such that $b_i \geq a_i$.

(b) Case where $b_i < a_i$ for all i.

(c) Case where $b_i > a_i$ for all i.

Fig. 3. Illustration depicting how a reaches b' in the different cases.

directly from a_i to a_j. Note that $a_i < a_j$. It is easy to see that we can iteratively repeat this process to remove all occurrences where $a_i > a_{i+1}$. Thus, the statement follows. □

Now, we are ready to state the necessary and sufficient condition:

Lemma 39. *Let $b' > a$. It is the case that $b' \in \mathrm{Post}_{\theta^\mu(\theta^\mu)^+}(a) \cup \mathrm{Post}_{\theta^\mu}(a)$ if and only if $a \to_{\theta^\mu} b'$ or*

(1) there exists $a' > a$ such that $a \to_{\theta^\mu\theta^\mu} a'$, and
(2) there exists $b < b'$ such that $b \to_{\theta^\mu\theta^\mu} b'$.

Proof. $\Leftarrow$) If $a \to_{\theta^\mu} b'$, then we are done. So assume $a \not\to \theta^\mu b$ and that (1) and (2) both hold. That is, $a \to_{\sigma_1\sigma_2} a'$ and $b \to_{\sigma_1'\sigma_2'} b'$ with $a < a', b < b'$, and $\mathrm{path}(\sigma_1) = \mathrm{path}(\sigma_2) = \mathrm{path}(\sigma_1') = \mathrm{path}(\sigma_2') = \theta$. Let $\sigma \coloneqq \sigma_1\sigma_2$ and $\sigma' \coloneqq \sigma_1'\sigma_2'$. We define $a_i \coloneqq a + \Delta(\sigma[1..i])$ and $b_i \coloneqq b + \Delta(\sigma'[1..i])$ as the counter values along runs $a \to_\sigma a'$ and $b \to_{\sigma'} b'$, respectively. In particular, $a_0 = a$ and $b_0 = b$. Further, $a_{2n} = a'$ and $b_{2n} = b'$. We differentiate cases based on the order of a and b.

Case $b < a$: Assume there exists i such that $b_i \geq a_i$. By Corollary 34, it follows that $a \to_{\mathrm{path}(\sigma[1..i])} b_i$. By definition of σ', we have $b_i \to_{\sigma'[i+1..2n]} b'$. Thus, $a \to_{\theta^\mu\theta^\mu} b'$, and we are done. So it remains to handle the case where $b_i < a_i$ for all i. Recall that we assume $b' > a$. Thus, we have $b' > a$, $b \leq a$, and $b' \leq a'$. So, we can invoke Lemma 35, from which it follows that $a \to_{\theta^\mu\theta^\mu} b'$ as desired. See Figures 3(a) and 3(b) for illustrations of both subcases.

Case $b > a$: This is similar to the previous case. If there exists i such that $b_i \leq a_i$, then we derive $a \to_{\theta^\mu\theta^\mu} b'$ from Corollary 34. Thus, assume that $b_i > a_i$ for all i. Intuitively, we now iterate σ to obtain larger and larger configurations, until we find a configuration that exceeds the respective configuration on the run from b. See Figure 3(c) for a sketch of this case.

Formally, let $a_{j,0} \coloneqq a_0 + j \cdot \Delta(\sigma)$, and $a_{j,i} \coloneqq a_{j,0} + \Delta(\sigma[1..i])$. Recall that $b_i = b + \Delta(\sigma'[1..i])$. Let j be the first index such that there exists i such that $a_{j,i} \geq b_i$, and let i be the smallest value for which this holds for that choice of j. Note that such a j must exist, since $\Delta(\sigma) > 0$, as $a < a_n$. Furthermore, note that by minimality of j, it follows that $i > 0$, as $a_{j,0} = a_{j-1,n}$.

We argue that $\sigma^j\sigma[1..i-1]$ is an admissible run from a to $a_{j,i-1}$. We first show that for all j', i' that $a_{0,i'} \leq a_{j',i'}$. It is clear that $a_{j',i'} = j' \cdot \Delta(\sigma) + \Delta(\sigma[1..i']) = j' \cdot \Delta(\sigma) + a_{0,i'} > a_{0,i'}$ because $\Delta(\sigma) > 0$. Further, for $j' < j$ and for all i', it holds that $a_{j',i'} < b_{i'}$ by minimality of j. In a similar manner, $a_{j,i'} < b_{i'}$ holds for all $i' < i$ by minimality of j and i. Therefore, we have

$$a_{0,i'} < a_{j',i'} < b_{i'} \text{ for all } j' < j \text{ and } i',$$
$$a_{0,i'} < a_{j,i'} < b_{i'} \text{ for all } i' < i.$$

Note the subtle difference between the statements, as the latter statement is concerned with the part of the run from $a_{j,0}$ to $a_{j,i}$, and the former statement is concerned with the parts of the run from $a_{0,0}$ to $a_{0,n}$, from $a_{1,0}$ to $a_{1,n}$, and so on.

Recall that run σ from a gives rise to values $a_{0,i}$, and that run σ' from b gives rise to values b_i. Thus, admissibility of $\sigma^j\sigma[1..i-1]$ from a follows from admissibility of σ from a and of σ' from b. So we have $a \to_{\sigma^j\sigma[1..i-1]} a_{j,i-1}$. We further have $a_{j,i} \geq b_i$ and $a_{j,i-1} < b_{i-1}$ by choice of j, i. By definition of $a_{j,i}$ and b_i, there exist α, β such that $a_{j,i} = a_{j,i-1} + \alpha\Delta(t_i)$ and $b_i = b_{i-1} + \beta\Delta(t_i)$. By Lemma 33, there thus exists α' such that $a_{j,i-1} + \alpha'\Delta(t_i) = b_i$. So we derive that $a_{j,i-1} \to_{t_i} b_i$.

Thus, we finally derive that

$$a \to_{\sigma^j\sigma[1..i-1]} a_{j,i-1} \to_{t_i} b_i \to_{\sigma[i+1..2n]} b'.$$

Hence, $a \to_{\mathrm{path}(\sigma)^{j+1}} b'$, and by definition of σ, $a \to_{\theta^{2(j+1)}} b$. So $b \in \mathrm{Post}_{\theta^\mu(\theta^\mu)^+}(a)$. By assumption, $a \not\to \theta^\mu b$, so $b \in \mathrm{Post}_{\theta^\mu(\theta^\mu)^+}(a) \setminus \mathrm{Post}_{\theta^\mu}(a)$.

$\Rightarrow$) It holds that $b' \in \mathrm{Post}_{\theta^\mu(\theta^\mu)^+}(a) \cup \mathrm{Post}_{\theta^\mu}(a)$. Thus, $a \to_{\theta^\mu(\theta^\mu)^+} b'$ or $a \to_{\theta^\mu} b'$. If $a \to_{\theta^\mu} b'$, then clearly we are done. So assume $a \not\to \theta^\mu b'$. By Lemma 38, there exist values $a_0, \ldots, a_n$ such that $a_0 \to_{\theta^\mu} a_1 \to_{\theta^\mu} \cdots \to_{\theta^\mu} a_n$ and $a = a_0 < a_1 < \cdots < a_n = b'$. Note that because $a \not\to \theta^\mu b'$, the sequence must contain at least three elements. So take the first three elements a_0, a_1, a_2. Since $a_0 < a_1 < a_2$ and $a = a_0 \to_{\theta^\mu} a_1 \to_{\theta^\mu} a_2$, we have $a \to_{\theta^\mu\theta^\mu} a_2$, and so (1) is satisfied. By a similar argument, $a_{n-2} < a_{n-1} < a_n$ and $a_{n-2} \to_{\theta^\mu} a_{n-1} \to_{\theta^\mu} a_n = b$, and hence (2) is satisfied. □

Now we are ready to define and prove the correctness of $\phi^+_{\theta^+}(a, b, \mu)$. Let

$$\phi^+_{\theta^+}(a, b, \mu) \coloneqq (a > b) \wedge [\phi_\theta(a, b, \mu) \vee \exists a' > a, \exists b' < b : \phi_{\theta\theta}(a, a', \mu) \wedge \phi_{\theta\theta}(b', b, \mu)].$$

Lemma 40. *It is the case that $\phi^+_{\theta^+}(a, b, \mu)$ is satisfied iff $a < b$ and $b \in \mathrm{Post}_{\theta^\mu(\theta^\mu)^+}(a) \cup \mathrm{Post}_{\theta^\mu}(a)$.*

Proof. It follows immediately from the sufficient and necessary condition of Lemma 39. □

Corollary 41. *It is the case that $\phi_{\theta^*}(a, b, \mu)$ is satisfied iff $b \in \mathrm{Post}_{(\theta^*)^\mu}(a)$. Furthermore, the size of ϕ_{θ^*} is linear in the length of θ and nonparametric updates and endpoints on θ.*

5.3 Formula for a Linear Path Scheme

Lemma 42. *For every linear path scheme $L = \sigma_0\theta_0^*\sigma_1\theta_1^*\ldots\sigma_k$, there exists a formula of existential linear arithmetic ϕ_L such that $\phi_L(a, b, \mu)$ holds iff $b \in \mathrm{Post}_{L^\mu}(a)$, and ϕ_L has size polynomial in $|L|$.*

Proof. We define the following formula whose correctness follows Lemma 36 and Corollary 41:

$$\phi_L(a, b, \mu) \coloneqq \exists a_0, a'_0, \ldots, a_k, a'_k : (a_0 = a) \wedge (a'_k = b) \wedge \bigwedge_{0 \leq i \leq k} \phi_{\sigma_i}(a_i, a'_i, \mu) \wedge \bigwedge_{0 \leq i < k} \phi_{\theta_i^*}(a'_i, a_{i+1}, \mu).$$

□

Theorem 43. *The existential reachability problem for parametric COCAs is in NP.*

Proof. By Lemma 32, $b \in \mathrm{Post}_{p,q}(a)$ iff there exists a linear path scheme $L = \sigma_0\theta_0^*\sigma_1\theta_1^* \cdots \sigma_k$ of polynomial size such that $b \in \mathrm{Post}_L(a)$. By Lemma 42, from any linear path scheme L, we can construct, in polynomial time, an existential linear formula ϕ_L such that $\phi_L(a, b, \mu)$ holds iff $b \in \mathrm{Post}_{L^\mu}(a)$. By quantifying existentially over μ, we obtain a formula $\phi^\exists_L(a, b) \coloneqq \exists\mu : \phi_L(a, b, \mu)$, which is satisfied iff $b \in \mathrm{Post}_L(a)$.

Thus, to determine whether $p(a) \to_* q(b)$, we (1) guess a linear path scheme L, (2) construct $\phi^\exists_L$, and (3) check whether $\phi^\exists_L(a, b)$ holds. Since (3) can be achieved in NP [23], we are done. □

5.4 Integer Valuations

We briefly consider parametric COCAs where only updates can be parameterized. In this setting, a rational valuation that witnesses reachability can be turned into an *integer* valuation witnessing reachability. This follows by rescaling the factors of the witnessing run so that it remains admissible.

Lemma 44. *Let μ be a valuation under which $p(a) \to_* q(b)$. For any valuation μ' such that $\mu'(x) = \lambda\mu(x)$ with $\lambda \in \mathbb{N}_{\geq 1}$, it is the case that $p(a) \to_* q(b)$ under μ'.*

Proof. Let $\lambda \in \mathbb{N}_{\geq 1}$ and let μ' be defined w.r.t. μ and λ. Let $s(v) \to_{\alpha t} s'(v')$ be consecutive configurations from the run ρ witnessing $p(a) \to_\rho q(b)$ under valuation μ. If $\Delta(t) \in \mathbb{Q}$, then no rescaling is needed as the update of t is nonparametric. Otherwise, we have $v' - v = \alpha \cdot \mu(\Delta(t))$. Let $\beta \coloneqq \alpha/\lambda$. Since $\alpha \in (0, 1]$ and $\lambda \geq 1$, we have $\beta \in (0, 1]$. Therefore, we have $s(v) \to_{\beta t} s'(v')$. Hence, by rescaling each transition, we obtain a run ρ' such that $p(a) \to_{\rho'} q(b)$ under μ'. □

Now, consider a rational valuation μ witnessing $p(a) \to_* q(b)$. Since μ is rational, each parameter value $\mu(x)$ can be represented as a fraction a_x/b_x. By Lemma 44, we know that valuation $\mu'(x) = \lambda\mu(x)$, where $\lambda \coloneqq \prod_{x \in X} b_x$, also witnesses reachability. Moreover, it is integral, hence:

Corollary 45. *The (existential) reachability problem for parametric COCAs, where valuations must be integral, is equivalent to the rational variant if guards are nonparametric.*

5.5 Hardness

To conclude our treatment of parametric COCAs, we establish NP-hardness of the reachability problem, even for the special case of acyclic COCAs.

Theorem 46. *The reachability problem for acyclic parametric COCAs is NP-hard, even when parameters occur only on updates or only on guards.*

Proof. The reductions for the case where parameters are only on updates and only on guards are very similar, differing only in the construction of certain gadgets that otherwise have the same purpose, so we present these two reductions in parallel.

We give a reduction from 3-SAT. Let $\varphi = \bigwedge_{1 \leq j \leq m} C_j$ be a 3-CNF formula over variables $X = \{x_1, \ldots, x_n\}$.

Let us give two acyclic parametric COCAs $\mathcal{P}$ and $\mathcal{P}'$, both with parameters X. Each one will guess an assignment to X and check whether it satisfies φ. Additionally, $\mathcal{P}$ uses parameters only on guards; $\mathcal{P}'$, only on updates. We sketch the constructions in the following.

The first part is done by sequentially composing n copies of the gadget depicted at the top of Figure 4, for $\mathcal{P}$ on the left-hand side, and for $\mathcal{P}'$ on the right-hand side. The gadgets function as follows: (1) state p_i is entered with counter value 0, (2) the counter is set to x_i, (3) membership of the counter value in $\{0, 1\}$ is checked, and (4) the counter is reset to zero upon leaving to q_i. The only way to traverse the chain of n such gadgets from p_1 to q_n is to have $x_i \in \{0, 1\}$ for each $x_i \in X$.

The second part is achieved by chaining a gadget for each clause similar to the one depicted on the bottom of Figure 4 for $C_j = (x_1 \vee x_2 \vee \neg x_4)$. The left-hand side depicts the gadget for $\mathcal{P}$, and the right-hand side depicts it for $\mathcal{P}'$. In words, it (1) enters state r_j with the counter value set to 0, (2) nondeterministically picks a variable x_i of some literal of C_j and increments the counter by x_i, (3) checks whether the counter holds the right value w.r.t. the literal polarity, and (4) resets the counter to zero upon leaving to state s_j. Thus, the chain of gadgets can be traversed from r_1 to s_m iff φ is satisfied by the assignment.

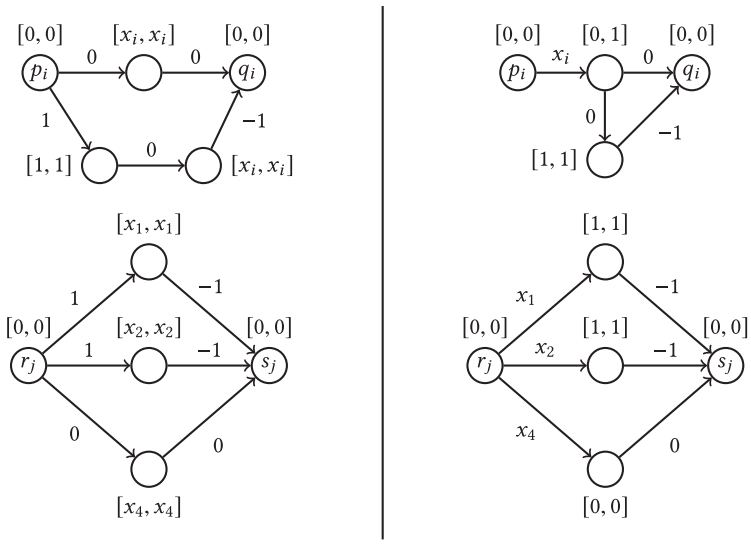


Fig. 4. Gadgets of the reduction from 3-SAT, where parameters occur only on guards (*left*) or updates (*right*), for variable x_i (*top*) and clause $C_j = (x_1 \vee x_2 \vee \neg x_4)$ (*bottom*).

Altogether, these statements are equivalent: (1) formula φ is satisfiable, (2) there exists a valuation $\mu\colon X \to \mathbb{Q}$ such that $p_1(0) \to_* s_m(0)$ holds in $\mathcal{P}^\mu$, and (3) there exists a valuation $\mu'\colon X \to \mathbb{Q}$ such that $p_1(0) \to_* s_m(0)$ holds in $\mathcal{P}'^{\mu'}$. □

Thus, together with Theorem 43, NP-completeness of the existential reachability problem in parametric COCAs follows.

6 CONCLUSION

In this work, we have introduced globally guarded COCA and COCA as over-approximations of SOCA, and we have given efficient algorithms for their reachability problems. For both models, the only hardness result we are aware of is the NL-hardness that follows trivially from the directed-graph reachability problem. Giving tighter hardness results for the complexity of reachability in GG-COCA and COCA seems desirable yet challenging. In particular, one goal could be to show an equivalence with other problems known to be in NC^2 or P-hard. However, it seems necessary to encode information in the counter that, intuitively, encodes more precise information than that the counter value is in a certain range. If only information of this sort is available, then the problem is equivalent to reachability in one-clock timed automata, which is in NL [16]. However, the continuous semantics seem to make encoding more precise information nontrivial. Consequently, it remains open whether our algorithms are computationally optimal.

For parametric COCA, we have shown that the reachability problem is NP-complete when numbers are encoded in binary. Our construction for establishing NP-hardness works regardless of whether numbers are encoded in unary or in binary, but it works only when there is an arbitrary number of parameters. For a fixed number of parameters, including no parameters at all, the only hardness result is the trivial NL-hardness bound derived from graph reachability. The complexity results are summarized in Table 1.

Table 1. Overview of the Complexity for COCA Reachability, Depending on Whether There Are Zero, a Fixed Number, or an Arbitrary Number of Parameters, and Whether Numbers Are Encoded in Unary or in Binary

	Unary	**Binary**
None	NL-complete	$\in$ P
Fixed	NL-complete	$\in$ NP
Arbitrary	NP-complete	NP-complete

REFERENCES

[1] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. 1993. Parametric real-time reasoning. In *Proc. 25th Annual ACM Symposium on Theory of Computing (STOC'93)*. ACM, 592–601. https://doi.org/10.1145/167088.167242

[2] Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity - A Modern Approach*. Cambridge University Press.

[3] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. MIT Press.

[4] Michael Blondin and Christoph Haase. 2017. Logics for continuous reachability in Petri nets and vector addition systems with states. In *Proc. 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'17)*. IEEE Computer Society, 1–12. https://doi.org/10.1109/LICS.2017.8005068

[5] Michael Blondin, Tim Leys, Filip Mazowiecki, Philip Offtermatt, and Guillermo A. Pérez. 2021. Continuous one-counter automata. In *Proc. 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'21)*. 1–13. https://doi.org/10.1109/LICS52264.2021.9470525

[6] Benedikt Bollig, Karin Quaas, and Arnaud Sangnier. 2019. The complexity of flat freeze LTL. *Logical Methods in Computer Science* 15, 3 (2019). https://doi.org/10.23638/LMCS-15(3:33)2019

[7] Ahmed Bouajjani, Marius Bozga, Peter Habermehl, Radu Iosif, Pierre Moro, and Tomás Vojnar. 2011. Programs with lists are counter automata. *Formal Methods in System Design* 38, 2 (2011), 158–192. https://doi.org/10.1007/s10703-011-0111-7

[8] Daniel Pierre Bovet and Pierluigi Crescenzi. 1994. *Introduction to the Theory of Complexity*. Prentice Hall London.

[9] Daniel Bundala and Joël Ouaknine. 2017. On parametric timed automata and one-counter machines. *Information and Computation* 253 (2017), 272–303. https://doi.org/10.1016/j.ic.2016.07.011

[10] Cristiana Chitic and Daniela Rosu. 2004. On validation of XML streams using finite state machines. In *Proc. 7th International Workshop on the Web and Databases (WebDB'04)*. ACM, 85–90. https://doi.org/10.1145/1017074.1017096

[11] Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. 2019. The reachability problem for Petri nets is not elementary. In *Proc. 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC'19)*. ACM, 24–33. https://doi.org/10.1145/3313276.3316369

[12] John Fearnley and Marcin Jurdziński. 2015. Reachability in two-clock timed automata is PSPACE-complete. *Information and Computation* 243 (2015), 26–36. https://doi.org/10.1016/j.ic.2014.12.004

[13] Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. 2009. Reachability in succinct and parametric one-counter automata. In *Proc. 20th International Conference on Concurrency Theory (CONCUR'09)*. Springer, 369–383. https://doi.org/10.1007/978-3-642-04081-8_25

[14] Christoph Haase, Joël Ouaknine, and James Worrell. 2012. On the relationship between reachability problems in timed and counter automata. In *Proc. 6th International Workshop Reachability Problems (RP'12)*. Springer, 54–65. https://doi.org/10.1007/978-3-642-33512-9_6

[15] Oscar H. Ibarra, Tao Jiang, Nicholas Q. Trân, and Hui Wang. 1993. New decidability results concerning two-way counter machines and applications. In *Proc. 20th International Colloquium on Automata, Languages and Programming (ICALP'93)*. Springer, 313–324. https://doi.org/10.1007/3-540-56939-1_82

[16] François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. 2004. Model checking timed automata with one or two clocks. In *Proc. 15th International Conference on Concurrency Theory (CONCUR'04)*. Springer, 387–401. https://doi.org/10.1007/978-3-540-28644-8_25

[17] Jérôme Leroux and Sylvain Schmitz. 2019. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *Proc. 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'19)*. 1–13. https://doi.org/10.1109/LICS.2019.8785796

[18] Richard J. Lipton. 1976. *The Reachability Problem Requires Exponential Space*. Technical Report 62. Yale University. http://cpsc.yale.edu/sites/default/files/files/tr63.pdf.

[19] Ernst W. Mayr. 1981. An algorithm for the general Petri net reachability problem. In *Proc. 13th Annual ACM Symposium on Theory of Computing (STOC'81)*. ACM, 238–246. https://doi.org/10.1145/800076.802477
[20] Marvin L. Minsky. 1961. Recursive unsolvability of Post's problem of "Tag" and other topics in theory of Turing machines. *Annals of Mathematics* 74, 3 (1961), 437–455.
[21] Marvin L. Minsky. 1967. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc.
[22] Christos H. Papadimitriou. 1994. *Computational Complexity*. Addison-Wesley.
[23] Eduardo D. Sontag. 1985. Real addition and the polynomial hierarchy. *Inform. Process. Lett.* 20, 3 (1985), 115–120.

Received 1 November 2021; revised 25 March 2022; accepted 7 June 2022