Approaching the Coverability Problem Continuously

Michael Blondin[1,2⋆], Alain Finkel[2], Christoph Haase[2⋆⋆], and Serge Haddad[2,3⋆⋆⋆]

[1] DIRO, Université de Montréal, Canada
[2] LSV, CNRS & ENS Cachan, Université Paris-Saclay, France
[3] Inria, France

Abstract. The coverability problem for Petri nets plays a central role in the verification of concurrent shared-memory programs. However, its high EXPSPACE-complete complexity poses a challenge when encountered in real-world instances. In this paper, we develop a new approach to this problem which is primarily based on applying forward coverability in continuous Petri nets as a pruning criterion inside a backward-coverability framework. A cornerstone of our approach is the efficient encoding of a recently developed polynomial-time algorithm for reachability in continuous Petri nets into SMT. We demonstrate the effectiveness of our approach on standard benchmarks from the literature, which shows that our approach decides significantly more instances than any existing tool and is in addition often much faster, in particular on large instances.

1 Introduction

Counter machines and Petri nets are popular mathematical models for modeling and reasoning about distributed and concurrent systems. They provide a high level of abstraction that allows for employing them in a great variety of application domains, ranging, for instance, from modeling of biological, chemical and business processes to the formal verification of concurrent programs.

Many safety properties of real-world systems reduce to the *coverability problem* in Petri nets: Given an initial and a target configuration, does there exist a sequence of transitions leading from the initial configuration to a configuration larger than the target configuration? For instance, in an approach pioneered by German and Sistla [18] multi-threaded non-recursive finite-state programs with shared variables, which naturally occur in predicate-abstraction-based verification frameworks, are modeled as Petri nets such that every program location corresponds to a place in a Petri net, and the number of tokens of a place

⋆ Supported by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT), by the French Centre national de la recherche scientifique (CNRS), and by the "Chaire Digiteo, ENS Cachan — École Polytechnique".
⋆⋆ Supported by Labex Digicosme, Univ. Paris-Saclay, project VERICONISS.
⋆⋆⋆ Supported by ERC project EQualIS (FP7-308087).

indicates how many threads are currently at the corresponding program location. Coverability can then, for instance, be used in order to detect whether a mutual exclusion property could be violated when a potentially unbounded number of threads is executed in parallel. The coverability problem was one of the first decision problems for Petri nets that was shown decidable and EXPSPACE-complete [20,3,23]. Despite this huge worst-case complexity, over the course of the last twenty years, a plethora of tools has emerged that have shown to be able to cope with a large number of real-world instances of coverability problems in a satisfactory manner.

Our contribution. We present a new approach to the coverability problem and its implementation. When run on standard benchmarks that we obtained from the literature, our approach proves more than 91% of safe instances to be safe, most of the time much faster when compared to existing tools, and none of those tools can individually prove more than 84% of safe instances to be safe. We additionally demonstrate that our approach is also competitive when run on unsafe instances. In particular, it decides 142 out of 176 (80%) instances of our benchmark suite, while the best competitor only decides 122 (69%) instances.

Our approach is conceptually extremely simple and exploits recent advances in the theory of Petri nets as well as the power of modern SMT-solvers inside a backward-coverability framework. In [13], Fraca and Haddad solved long-standing open problems about the complexity of decision problems for so-called continuous Petri nets. This class was introduced by David and Alla [4] and allows for transitions to be fired a non-negative real number of times—hence places may contain a non-negative real number of tokens. The contribution of [13] was to present polynomial-time algorithms that decide all of coverability, reachability and boundedness in this class. A further benefit of [13] is to show that continuous Petri nets over the reals are equivalent to continuous Petri nets over the rationals, and, moreover, to establish a set of simple sufficient and necessary conditions in order to decide reachability in continuous Petri nets. The first contribution of our paper is to show that these conditions can efficiently be encoded into a sentence of *linear size* in the existential theory of the non-negative rational numbers with addition and order ($\mathrm{FO}(\mathbb{Q}_+, +, >)$). This encoding paves the way for deciding coverability in continuous Petri nets inside SMT-solvers and is particularly useful in order to efficiently answer *multiple coverability queries* on the same continuous Petri net due to caching strategies present in modern SMT-solvers. Moreover, we show that our encoding in effect *strictly subsumes* a recently introduced CEGAR-based approach to coverability described by Esparza *et al.* in [9]; in particular we can completely avoid the potentially exponentially long CEGAR-loop, cf. the related work section below. The benefit of coverability in continuous Petri nets is that it provides a way to over-approximate coverability under the standard semantics: any configuration that is not coverable in a continuous Petri net is also not coverable under the standard semantics. This observation can be exploited inside a backward-coverability framework as follows. Starting at the target configuration to be covered, the classical backward-coverability algorithm [1] repeatedly computes the set of all minimal predecessor configurations that by application of one transition cover the target or some earlier computed configuration until a fixed point is reached, which is guaranteed to happen due to Petri nets being well-structured transition systems [12]. The crux to the performance of the algorithm lies in the size of the set of minimal elements that is computed during each iteration, which may grow exponentially.[4] This is where continuous coverability becomes beneficial. In our approach, if a minimal element is not continuously coverable, it can safely be discarded since none of its predecessors is going to be coverable either, which substantially shrinks the predecessor set. In effect, this heuristic yields a powerful pruning technique, enabling us to achieve the aforementioned advantages when compared to other approaches on standard benchmarks.

Some proof details are only sketched in the main part of this paper; full details can be found in the appendix.

Related Work. Our approach is primarily related to the work by Esparza *et al.* [9], by Kaiser, Kroening and Wahl [19], and by Delzanno, Raskin and van Begin [6]. In [9], Esparza *et al.* presented an implementation of a semi-decision procedure for disproving coverability which was originally proposed by Esparza and Melzer [10]. It is based on the Petri-net state equation and traps as sufficient criteria in order to witness non-coverability. As shown in [10], those conditions can be encoded into an equi-satisfiable system of linear inequalities called the *trap inequation* in [10]. This approach is, however, prone to numerical imprecision that become problematic even for instances of small size [10, Sec. 5.3]. For that reason, the authors of [9] resort to a CEGAR-based variant of the approach described in [10] which has the drawback that in the worst case, the CEGAR loop has to be executed an exponential number of times leading to an exponential number of queries to the underlying SMT-solver. We will show in Section 4.3 that the conditions used in [9] are strictly subsumed by a subset of the conditions required to witness coverability in continuous Petri nets: whenever the procedure described in [9] returns uncoverable then coverability does not hold in the continuous setting either, but not *vice versa*. Thus, a single satisfiability check to our formula in existential $\mathrm{FO}(\mathbb{Q}_+, +, >)$ encoding continuous coverability that we develop in this paper completely subsumes the CEGAR-approach presented in [9]. Another difference to [9] is that here we present a sound *and* complete decision procedure.

Regarding the relationship of our work to [19], Kaiser *et al.* develop in their paper an approach to coverability in richer classes of well-structured transition systems that is also based on the classical backward-analysis algorithm. They also employ forward analysis in order to prune the set of minimal elements during the backward iteration, and in addition a widening heuristic in order to over-approximate the minimal basis. Our approach differs in that our minimal basis is always precise yet as small as possible modulo continuous coverability. Thus no backtracking as in [19] is needed, which is required when the widened basis turns

[4] This problem is commonly referred to as the *symbolic state explosion problem*, cf. [7].

out to be too inaccurate. Another difference is that for the forward analysis, a Karp-Miller tree is incrementally built in the approach described in [19], whereas we use the continuous coverability over-approximation of coverability.

The idea of using an over-approximation of the reachability set of a Petri net in order to prune minimal basis elements inside a backward coverability framework was first described by Delzanno *et al.* [6], who use place invariants as a pruning criterion. However, computing such invariants and checking if a minimal basis element can be pruned potentially requires exponential time.

Finally, a number of further techniques and tools for deciding Petri net coverability or more general well-structured transition systems have been described in the literature. They are, for instance, based on efficient data structures [14,11,7,15] and generic algorithmic frameworks such as EEC [16] and IC3 [21].

2 Preliminaries

We denote by $\mathbb{Q}$, $\mathbb{Z}$ and $\mathbb{N}$ the set of rationals, integers, and natural numbers, respectively, and by $\mathbb{Q}_+$ the set of non-negative rationals. Throughout the whole paper, numbers are encoded in binary, and rational numbers as pairs of integers encoded in binary. Let $\mathbb{D} \subseteq \mathbb{Q}$, $\mathbb{D}^E$ denotes the set of vectors indexed by a finite set E. A vector $\boldsymbol{u}$ is denoted by $\boldsymbol{u} = (u_i)_{i\in E}$. Given vectors $\boldsymbol{u} = (u_i)_{i\in E}, \boldsymbol{v} = (v_i)_{i\in E} \in \mathbb{D}^E$, addition $\boldsymbol{u} + \boldsymbol{v}$ is defined component-wise, and $\boldsymbol{u} \leq \boldsymbol{v}$ whenever $u_i \leq v_i$ for all $i \in E$. Moreover, $\boldsymbol{u} < \boldsymbol{v}$ whenever $\boldsymbol{u} \leq \boldsymbol{v}$ and $\boldsymbol{u} \neq \boldsymbol{v}$. Let $E' \subseteq E$ and $\boldsymbol{v} \in \mathbb{D}^E$, we sometimes write $\boldsymbol{v}[E']$ as an abbreviation for $(v_i)_{i\in E'}$. The *support of* $\boldsymbol{v}$ is the set $[\![\boldsymbol{v}]\!] \stackrel{\text{def}}{=} \{i \in E : \boldsymbol{v}_i \neq 0\}$.

Given finite sets of indices E and F, and $\mathbb{D} \subseteq \mathbb{Q}$, $\mathbb{D}^{E\times F}$ denotes the set of matrices over $\mathbb{D}$ with rows and columns indexed by elements from E and F, respectively. Let $\mathbf{M} \in \mathbb{D}^{E\times F}$, $E' \subseteq E$ and $F' \subseteq F$, we denote by $\mathbf{M}_{E'\times F'}$ the $\mathbb{D}^{E'\times F'}$ sub-matrix obtained from $\mathbf{M}$ whose row and columns indices are restricted respectively to E' and F'.

Petri Nets. In what follows, we introduce the syntax and semantics of Petri nets. While we provide a single syntax for nets, we introduce a discrete (i.e. in $\mathbb{N}$) and a continuous (i.e. in $\mathbb{Q}_+$) semantics.

Definition 1. *A Petri net is a tuple* $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$*, where* P *is a finite set of* places*;* T *is a finite set of* transitions *with* $P \cap T = \emptyset$*; and* $\mathbf{Pre}, \mathbf{Post} \in \mathbb{N}^{P\times T}$ *are the* backward *and* forward incidence matrices*, respectively.*

A (discrete) *marking* of $\mathcal{N}$ is a vector of $\mathbb{N}^P$. A *Petri net system (PNS)* is a pair $\mathcal{S} = (\mathcal{N}, \boldsymbol{m}_0)$, where $\mathcal{N}$ is a Petri net and $\boldsymbol{m}_0 \in \mathbb{N}^P$ is the *initial marking.* The *incidence matrix* $\mathbf{C}$ of $\mathcal{N}$ is the $P \times T$ integer matrix defined by $\mathbf{C} \stackrel{\text{def}}{=} \mathbf{Post} - \mathbf{Pre}$. The *reverse net* of $\mathcal{N}$ is $\mathcal{N}^{-1} \stackrel{\text{def}}{=} (P, T, \mathbf{Post}, \mathbf{Pre})$. Let $p \in P$ and $t \in T$, the *pre-sets* of p and t are the sets ${}^\bullet p \stackrel{\text{def}}{=} \{t' \in T : \mathbf{Post}(p, t') > 0\}$

and ${}^\bullet t \stackrel{\text{def}}{=} \{p' \in P : \mathbf{Pre}(p',t) > 0\}$, respectively. Likewise, the *post-sets* of p and t are $p^\bullet \stackrel{\text{def}}{=} \{t' \in T : \mathbf{Pre}(p,t') > 0\}$ and $t^\bullet = \{p' \in P : \mathbf{Post}(p',t) > 0\}$, respectively. Those definitions can canonically be lifted to subsets of places and of transitions, e.g., for $Q \subseteq P$ we have ${}^\bullet Q = \bigcup_{p \in Q} {}^\bullet p$. We also introduce the *neighbors* of a subset of places/transitions by: ${}^\bullet Q^\bullet = {}^\bullet Q \cup Q^\bullet$. Let $S \subseteq T$, then $\mathcal{N}_S$ is the sub-net defined by $\mathcal{N}_S \stackrel{\text{def}}{=} ({}^\bullet S^\bullet, S, \mathbf{Pre}_{{}^\bullet S^\bullet \times S}, \mathbf{Post}_{{}^\bullet S^\bullet \times S})$.

We say that a transition $t \in T$ is *enabled* at a marking $\boldsymbol{m}$ whenever $\boldsymbol{m}(p) \geq \mathbf{Pre}(p,t)$ for every $p \in {}^\bullet t$. A transition t that is enabled can be *fired*, leading to a new marking $\boldsymbol{m}'$ such that for all places $p \in P$, $\boldsymbol{m}'(p) = \boldsymbol{m}(p) + \mathbf{C}(p,t)$. We write $\boldsymbol{m} \xrightarrow{t} \boldsymbol{m}'$ whenever t is enabled at $\boldsymbol{m}$ leading to $\boldsymbol{m}'$, and write $\boldsymbol{m} \to \boldsymbol{m}'$ if $\boldsymbol{m} \xrightarrow{t} \boldsymbol{m}'$ for some $t \in T$. By $\to^*$ we denote the reflexive transitive closure of $\to$. A word $\sigma = t_1 t_2 \cdots t_k \in T^*$ is a *firing sequence of* $(\mathcal{N}, \boldsymbol{m}_0)$ whenever there exist markings $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k$ such that

$$\boldsymbol{m}_0 \xrightarrow{t_1} \boldsymbol{m}_1 \xrightarrow{t_2} \cdots \xrightarrow{t_{k-1}} \boldsymbol{m}_{k-1} \xrightarrow{t_k} \boldsymbol{m}_k.$$

Given a marking $\boldsymbol{m}$, the *reachability* problem asks whether $\boldsymbol{m}_0 \to^* \boldsymbol{m}$. The reachability problem is decidable, EXPSPACE-hard [3] and in $\mathbf{F}_{\omega^3}$ [22], a non-primitive-recursive complexity class. In this paper, however, we are interested in deciding coverability, an EXPSPACE-complete problem [3,23].

Definition 2. *Given a Petri net system* $\mathcal{S} = (P, T, \mathbf{Pre}, \mathbf{Post}, \boldsymbol{m}_0)$ *and a marking* $\boldsymbol{m} \in \mathbb{N}^P$*, the* coverability *problem asks whether* $\boldsymbol{m}_0 \to^* \boldsymbol{m}'$ *for some* $\boldsymbol{m}' \geq \boldsymbol{m}$.

Continuous Petri nets are Petri nets in which markings may consist of rational numbers[5], and in which transitions may be fired a fractional number of times. Formally, a marking of a continuous Petri net is a vector $\boldsymbol{m} \in \mathbb{Q}_+^P$. Let $t \in T$, the *enabling degree* of t with respect to $\boldsymbol{m}$ is a function $enab(t, \boldsymbol{m}) \in \mathbb{Q}_+ \cup \{\infty\}$ defined by:

$$enab(t,\boldsymbol{m}) \stackrel{\text{def}}{=} \begin{cases} \min\{\boldsymbol{m}(p)/\mathbf{Pre}(p,t) : p \in {}^\bullet t\} & \text{if } {}^\bullet t \neq \emptyset, \\ \infty & \text{otherwise.} \end{cases}$$

We say that t is $\mathbb{Q}$-*enabled* at $\boldsymbol{m}$ if $enab(t,\boldsymbol{m}) > 0$. If t is $\mathbb{Q}$-enabled it may be *fired* by any amount $q \in \mathbb{Q}_+$ such that $0 \leq q \leq enab(t,\boldsymbol{m})$, leading to a new marking $\boldsymbol{m}'$ such that for all places $p \in P$, $\boldsymbol{m}(p)' \stackrel{\text{def}}{=} \boldsymbol{m}(p) + q \cdot \mathbf{C}(p,t)$. In this case, we write $\boldsymbol{m} \xrightarrow{q \cdot t} \boldsymbol{m}'$. The definition of a $\mathbb{Q}$-*firing sequence* $\sigma = q_1 t_1 \cdots q_k t_k \in (\mathbb{Q}_+ \times T)^*$ is analogous to the standard definition of firing sequence, and so are $\to_{\mathbb{Q}}$, $\to_{\mathbb{Q}}^*$ and $\mathbb{Q}$-reachability. The $\mathbb{Q}$-*Parikh image* of the firing sequence σ is the vector $\pi(\sigma) \in \mathbb{Q}_+^T$ such that $\pi(\sigma)(t) \stackrel{\text{def}}{=} \sum_{t_i = t} q_i$. We also adapt the decision problems for Petri nets.

[5] In fact, the original definition allows for real numbers, however for studying decidability and complexity issues, rational numbers are more convenient.

Definition 3. *Given a Petri net system $\mathcal{S} = (P, T, \mathbf{Pre}, \mathbf{Post}, \boldsymbol{m}_0)$ and a marking $\boldsymbol{m} \in \mathbb{Q}_+^P$, the $\mathbb{Q}$-reachability (respectively $\mathbb{Q}$-coverability) problem asks whether $\boldsymbol{m}_0 \to_{\mathbb{Q}}^* \boldsymbol{m}$ (respectively $\boldsymbol{m}_0 \to_{\mathbb{Q}}^* \boldsymbol{m}'$ for some $\boldsymbol{m}' \geq \boldsymbol{m}$).*

Recently $\mathbb{Q}$-reachability and $\mathbb{Q}$-coverability were shown to be decidable in polynomial time [13]. In Section 3.2, we will discuss in detail the approach from [13]. For now, observe that $\boldsymbol{m} \to \boldsymbol{m}'$ implies $\boldsymbol{m} \to_{\mathbb{Q}} \boldsymbol{m}'$, and hence $\boldsymbol{m} \to^* \boldsymbol{m}'$ implies $\boldsymbol{m} \to_{\mathbb{Q}}^* \boldsymbol{m}'$. Consequently, $\mathbb{Q}$-coverability provides an over-approximation of coverability: this fact is the cornerstone of this paper.

Upward Closed Sets. A set $V \subseteq \mathbb{N}^P$ is *upward closed* if for every $\boldsymbol{v} \in V$ and $\boldsymbol{w} \in \mathbb{N}^P$, $\boldsymbol{v} \leq \boldsymbol{w}$ implies $\boldsymbol{w} \in V$. The *upward closure* of a vector $\boldsymbol{v} \in \mathbb{N}^P$ is the set $\uparrow\boldsymbol{v} \stackrel{\text{def}}{=} \{\boldsymbol{w} \in \mathbb{N}^P : \boldsymbol{v} \leq \boldsymbol{w}\}$. This definition can be lifted to sets $V \subseteq \mathbb{N}^P$ in the obvious way, i.e., $\uparrow V \stackrel{\text{def}}{=} \bigcup_{\boldsymbol{v} \in V} \uparrow\boldsymbol{v}$. Due to $\mathbb{N}^P$ being well-quasi-ordered by $\leq$, any upward-closed set V contains a finite set $F \subseteq V$ such that $V = \uparrow F$. Such an F is called a *basis* of V and allows for a finite representation of an upward-closed set. In particular, it can be shown that V contains a unique *minimal basis* $B \subseteq V$ that is minimal with respect to inclusion for all bases $F \subseteq V$. We denote $\mathit{minbase}(F)$ this minimal basis obtained by deleting vectors $\boldsymbol{v} \in F$ such that there exists $\boldsymbol{w} \in F$ with $\boldsymbol{w} < \boldsymbol{v}$ (when F is finite).

3 Deciding Coverability and $\mathbb{Q}$-Reachability

We now introduce and discuss existing algorithms for solving coverability and $\mathbb{Q}$-reachability which form the basis of our approach. The main reason for doing so is that it allows us to smoothly introduce some additional notations and concepts that we require in the next section. For the remainder of this section, we fix some Petri net system $\mathcal{S} = (\mathcal{N}, \boldsymbol{m}_0)$ with $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$, and some marking $\boldsymbol{m}$ to be covered or $\mathbb{Q}$-reached.

3.1 The Backward Coverability Algorithm

The standard backward coverability algorithm, Algorithm 1, is a simple to state algorithm.

- It iteratively constructs minimal bases M, where in the k-th iteration, M is the minimal basis of the (upward closed) set of markings that can cover $\boldsymbol{m}$ after a firing sequence of length at most k. If $\boldsymbol{m}_0 \in \uparrow M$, the algorithm returns true, i.e., that $\boldsymbol{m}$ is coverable. Otherwise, in order to update M, for all $\boldsymbol{m}' \in M$ and $t \in T$ it computes $\boldsymbol{m}'_t(p) \stackrel{\text{def}}{=} \max\{\mathbf{Pre}(p,t),\ \boldsymbol{m}'(p) - \mathbf{C}(p,t)\}$. The singleton $\{\boldsymbol{m}'_t\}$ is the minimal basis of the set of vectors that can cover $\boldsymbol{m}'$ after firing t.
- Thus defining $pb(M)$ as $pb(M) \stackrel{\text{def}}{=} \bigcup_{\boldsymbol{m}' \in M, t \in T} \{\boldsymbol{m}'_t\}$, $M \cup pb(M)$ is a (not necessarily minimal) basis of the upward closed set of markings that can cover $\boldsymbol{m}$ after a firing sequence of length at most $k+1$. This basis can be then minimized in every iteration.

Algorithm 1 Backward Coverability

Require: PNS $\mathcal{S} = (\mathcal{N}, \boldsymbol{m}_0)$ and a marking $\boldsymbol{m} \in \mathbb{N}^P$

```
M := {m};
while m_0 ∉ ↑M do
    B := pb(M) \ ↑M;
    if B = ∅ then
        return false;
    else
        M := minbase(M ∪ B);
return true;
```

The termination of the algorithm is guaranteed due to $\mathbb{N}^P$ being well-quasi-ordered, which entails that M must stabilize and return false in this case. It can be shown that Algorithm 1 runs in 2-EXP [2]. The key point to the (empirical) performance of the algorithm is the size of the set M during its computation: the smaller, the better. Even though one can establish a doubly-exponential lower bound on the cardinality of M during the execution of the algorithm, in general not every element in M is coverable, even when $\boldsymbol{m}$ is coverable.

3.2 The $\mathbb{Q}$-Reachability Algorithm

We now present the fundamental concepts of the polynomial-time $\mathbb{Q}$-reachability algorithm of Fraca and Haddad [13]. The key insight underlying their algorithm is that $\mathbb{Q}$-reachability can be characterized in terms of three simple criteria. The algorithm relies on the notions of *firing set* and *maximal firing set*, denoted $\mathit{fs}(\mathcal{N}, \boldsymbol{m})$ and $\mathit{maxfs}(\mathcal{N}, \boldsymbol{m})$, and defined as follows:

$$\mathit{fs}(\mathcal{N}, \boldsymbol{m}) \stackrel{\text{def}}{=} \{[\![\pi(\sigma)]\!] : \sigma \in (\mathbb{Q}_+ \times T)^*, \text{ there is } \boldsymbol{m}' \in \mathbb{Q}_+^P \text{ s.t. } \boldsymbol{m} \xrightarrow{\sigma}_{\mathbb{Q}} \boldsymbol{m}'\}$$

$$\mathit{maxfs}(\mathcal{N}, \boldsymbol{m}) \stackrel{\text{def}}{=} \bigcup_{T' \in \mathit{fs}(\mathcal{N}, \boldsymbol{m})} T'.$$

Thus, $\mathit{fs}(\mathcal{N}, \boldsymbol{m})$ is the set of supports of firing sequences starting in $\boldsymbol{m}$. Even though $\mathit{fs}(\mathcal{N}, \boldsymbol{m})$ can be of size exponential with respect to $|T|$, deciding $T' \in \mathit{fs}(\mathcal{N}, \boldsymbol{m})$ for some $T' \subseteq T$ can be done in polynomial time, and $\mathit{maxfs}(\mathcal{N}, \boldsymbol{m})$ is also computable in polynomial time [13]. The following proposition characterizes the set of $\mathbb{Q}$-reachable markings.

Proposition 4 ([13, Thm. 20]). *A marking $\boldsymbol{m}$ is $\mathbb{Q}$-reachable in $\mathcal{S} = (\mathcal{N}, \boldsymbol{m}_0)$ if and only if there exists $\boldsymbol{x} \in \mathbb{Q}_+^T$ such that*

(i) $\boldsymbol{m} = \boldsymbol{m}_0 + \mathbf{C} \cdot \boldsymbol{x}$
(ii) $[\![\boldsymbol{x}]\!] \in \mathit{fs}(\mathcal{N}, \boldsymbol{m}_0)$
(iii) $[\![\boldsymbol{x}]\!] \in \mathit{fs}(\mathcal{N}^{-1}, \boldsymbol{m})$

In this characterization, $\boldsymbol{x}$ is supposed to be the Parikh image of a firing sequence. The first item expresses the state equation of $\mathcal{S}$ with respect to $\boldsymbol{m}_0$,

Algorithm 2 $\mathbb{Q}$-reachability [13]

Require: PNS $\mathcal{S} = (\mathcal{N}, \boldsymbol{m}_0)$ with $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ and a marking $\boldsymbol{m}$

if $\boldsymbol{m} = \boldsymbol{m}_0$ **then**
  return *true*;
$T' := T$;
while $T' \neq \emptyset$ **do**
  $S := \emptyset$;
  for all $t \in T'$ **do**
    $\boldsymbol{x} := \mathit{solve}(\mathbf{C}_{P \times T'} \cdot \boldsymbol{x} = \boldsymbol{m} - \boldsymbol{m}_0 \wedge \boldsymbol{x}(t) > 0 \wedge \boldsymbol{x} \in \mathbb{Q}_+^{T'})$;
    if $\boldsymbol{x} \neq \mathit{undef}$ **then**
      $S := S \cup [\![\boldsymbol{x}]\!]$;
  if $S = \emptyset$ **then**
    return *false*;
  $T' := \mathit{maxfs}(\mathcal{N}_S, \boldsymbol{m}_0[{}^\bullet S^\bullet])) \cap \mathit{maxfs}(\mathcal{N}_S^{-1}, \boldsymbol{m}[{}^\bullet S^\bullet]))$
  if $T' = S$ **then**
    return *true*;
return *false*;

$\boldsymbol{m}$ and $\boldsymbol{x}$. The two subsequent items express that the support of the solution of the state equation has to lie in the firing sets of $\mathcal{S}$ and its reverse. As such, the characterization in Proposition 4 yields an NP algorithm. By employing a greatest fixed point computation, Algorithm 2, which is a decision variant of the algorithm presented in [13], turns those criteria into a polynomial-time algorithm (see [13] for a proof of its correctness). In order to use Algorithm 2 for deciding coverability, it is sufficient, for each place p, to add a transition to $\mathcal{N}$ that can at any time non-deterministically decrease p by one token. Denote the resulting Petri net system by $\mathcal{S}'$, it can easily checked that $\boldsymbol{m}$ is $\mathbb{Q}$-coverable in $\mathcal{S}$ if and only if $\boldsymbol{m}$ is $\mathbb{Q}$-reachable in $\mathcal{S}'$.

4 Backward Coverability Modulo $\mathbb{Q}$-Reachability

We now present our decision algorithm for the Petri net coverability problem.

4.1 Encoding $\mathbb{Q}$-Reachability into Existential FO($\mathbb{Q}_+, +, >$)

Throughout this section, when used in formulas, $\boldsymbol{w}$ and $\boldsymbol{x}$ are vectors of first-order variables indexed by P representing markings, and $\boldsymbol{y}$ is a vector of first-order variables indexed by T representing the $\mathbb{Q}$-Parikh image of a transition sequence.

Condition (i) of Proposition 4, which expresses the state equation, is readily expressed as a system of linear equations and thus directly corresponds to a formula $\Phi(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})$ which holds whenever a marking $\boldsymbol{x}$ is reached starting at marking $\boldsymbol{w}$ by firing every transition $\boldsymbol{y}(t)$ times (without any consideration whether such a firing sequence would actually be admissible):

$$\Phi_{eqn}^{\mathcal{N}}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \boldsymbol{x} = \mathbf{C} \cdot \boldsymbol{y} + \boldsymbol{w}.$$

Next, we show how to encode Conditions (ii) and (iii) into suitable formulas. To this end, we require an effective characterization of membership in the firing set $fs(\mathcal{N}, \boldsymbol{w})$ defined in Section 3.2. The following characterization can be derived from [13, Cor. 19]. First, we define a monotonic increasing function $incfs_{\mathcal{N},\boldsymbol{w}} : 2^T \to 2^T$ as follows:

$$incfs_{\mathcal{N},\boldsymbol{w}}(S) \stackrel{\text{def}}{=} S \cup \{t \in T(\mathcal{N}) : {}^\bullet t \subseteq [\![\boldsymbol{w}]\!] \cup \{s^\bullet : s \in S\}\} .$$

From [13, Cor. 19], it follows that $T' \in fs(\mathcal{N}, \boldsymbol{w})$ if and only if $T' = \text{lfp}(incfs_{\mathcal{N}_{T'},\boldsymbol{w}})$, where lfp is the least fixed point operator[6], i.e.,

$$T' = incfs_{\mathcal{N}_{T'},\boldsymbol{w}}(\cdots(incfs_{\mathcal{N}_{T'},\boldsymbol{w}}(\emptyset))\cdots).$$

Clearly, the least fixed point is reached after at most $|T'|$ iterations.

In order to decide whether $[\![\boldsymbol{y}]\!] \in fs(\mathcal{N}, \boldsymbol{w})$, we simulate this fixed-point computation in an existential $\text{FO}(\mathbb{Q}_+, +, >)$-formula $\Phi^{\mathcal{N}}_{fs}(\boldsymbol{w}, \boldsymbol{y})$. Our approach is inspired by a technique of Verma, Seidl and Schwentick that was used to show that the reachability relation for communication-free Petri nets is definable by an existential Presburger arithmetic formula of linear size [27]. The basic idea is to introduce additional first-order variables $\boldsymbol{z}$ indexed by $P \cup T$ that, given a firing set, capture the relative order in which transitions of this set are fired and the order in which their input places are marked. This order corresponds to the computation of $\text{lfp}(incfs_{\mathcal{N}_{[\![\boldsymbol{y}]\!]},\boldsymbol{w}})$ and is encoded via a numerical value $\boldsymbol{z}(t)$ (respectively $\boldsymbol{z}(p)$), representing an index that must be strictly greater than zero for a transition (respectively an input place of a transition) of this set. In addition, input places have to be marked before the firing of a transition:

$$\Phi^{\mathcal{N}}_{dt}(\boldsymbol{y}, \boldsymbol{z}) \stackrel{\text{def}}{=} \bigwedge_{t \in T} \left(\boldsymbol{y}(t) > 0 \to \bigwedge_{p \in {}^\bullet t} 0 < \boldsymbol{z}(p) \leq \boldsymbol{z}(t) \right) .$$

Moreover, a place is either initially marked or after the firing of a transition of the firing set. So:

$$\Phi^{\mathcal{N}}_{mk}(\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{z}) \stackrel{\text{def}}{=} \bigwedge_{p \in P} \left(\boldsymbol{z}(p) > 0 \to \left(\boldsymbol{w}(p) > 0 \vee \bigvee_{t \in {}^\bullet p} \boldsymbol{y}(t) > 0 \wedge \boldsymbol{z}(t) < \boldsymbol{z}(p) \right) \right) .$$

We can now take the conjunction of the formulas above in order to obtain a logical characterization of $fs(\mathcal{N}, \boldsymbol{w})$:

$$\Phi^{\mathcal{N}}_{fs}(\boldsymbol{w}, \boldsymbol{y}) \stackrel{\text{def}}{=} \exists \boldsymbol{z} : \Phi^{\mathcal{N}}_{dt}(\boldsymbol{y}, \boldsymbol{z}) \wedge \Phi^{\mathcal{N}}_{mk}(\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{z}).$$

Having logically characterized all conditions of Proposition 4, we can define the global $\mathbb{Q}$-reachability relation for a Petri net system $\mathcal{S} = (\mathcal{N}, \boldsymbol{w})$ as follows:

$$\Phi_{\mathcal{S}}(\boldsymbol{w}, \boldsymbol{x}) \stackrel{\text{def}}{=} \exists \boldsymbol{y} : \Phi^{\mathcal{N}}_{eqn}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) \wedge \Phi^{\mathcal{N}}_{fs}(\boldsymbol{w}, \boldsymbol{y}) \wedge \Phi^{\mathcal{N}^{-1}}_{fs}(\boldsymbol{x}, \boldsymbol{y}).$$

In summary, we have thus proved the following result in this section.

[6] In [13, Cor. 19], an algorithm is presented that basically computes $\text{lfp}(incfs_{\mathcal{N}_{T'},\boldsymbol{w}})$.

Algorithm 3 Backward Coverability Modulo $\mathbb{Q}$-Reachability

Require: PNS $\mathcal{S} = (\mathcal{N}, \boldsymbol{m}_0)$ and a marking $\boldsymbol{m} \in \mathbb{N}^P$
1: $M := \{\boldsymbol{m}\}$; $\Phi(\boldsymbol{x}) := \exists \boldsymbol{y} : \Phi_{\mathcal{S}}(\boldsymbol{m}_0, \boldsymbol{y}) \wedge \boldsymbol{y} \geq \boldsymbol{x}$;
2: **if not** $\mathbb{Q}$-*coverable*$(\mathcal{S}, \boldsymbol{m})$ **then**
3:    **return** *false*
4: **while** $\boldsymbol{m}_0 \notin \uparrow M$ **do**
5:    $B := pb(M) \setminus \uparrow M$;
6:    $D := \{\boldsymbol{v} \in B : unsat(\Phi(\boldsymbol{v}))\}$;
7:    $B := B \setminus D$;
8:    **if** $B = \emptyset$ **then**
9:       **return** *false*;
10:    **else**
11:       $M := minbase(M \cup B)$;
12:       $\Phi(\boldsymbol{x}) := \Phi(\boldsymbol{x}) \wedge \bigwedge_{\boldsymbol{v} \in D} \boldsymbol{x} \ngeq \boldsymbol{v}$;
13: **return** *true*;

Proposition 5. *Let $\mathcal{S} = (\mathcal{N}, \boldsymbol{m}_0)$ be a Petri net system and $\boldsymbol{m}$ be a marking. There exists an existential FO$(\mathbb{Q}_+, +, >)$-formula $\Phi_{\mathcal{S}}(\boldsymbol{w}, \boldsymbol{x})$ computable in linear time such that $\boldsymbol{m}$ is $\mathbb{Q}$-reachable in $\mathcal{S}$ if and only if $\Phi_{\mathcal{S}}(\boldsymbol{m}_0, \boldsymbol{m})$ is valid.*

Checking satisfiability of $\Phi_{\mathcal{S}}$ is in NP, see e.g. [25]. It is a valid question to ask why one would prefer an NP-algorithm over a polynomial-time one. We address this question in the next section. For now, note that in order to obtain an even more accurate over-approximation, we can additionally restrict $\boldsymbol{y}$ to be interpreted in the natural numbers while retaining membership of satisfiability in NP, due to the following variant of Proposition 4: If a marking is reachable in $\mathcal{S}$ then there exists some $\boldsymbol{y} \in \mathbb{N}^T$ such that Conditions (i), (ii) and (iii) of Proposition 4 hold.

Remark 6. Proposition 5 additionally allows us to improve the best known upper bound for the *inclusion problem* of continuous Petri nets, which is EXP [13]. Given two Petri net systems $\mathcal{S} = (\mathcal{N}, \boldsymbol{m}_0)$ and $\mathcal{S}' = (\mathcal{N}', \boldsymbol{m}_0')$ over the same set of places, this problem asks whether the set of reachable markings of $\mathcal{S}$ is included in $\mathcal{S}'$, i.e., whether $\forall \boldsymbol{m}. \Phi_{\mathcal{S}}(\boldsymbol{m}_0, \boldsymbol{m}) \to \Phi_{\mathcal{S}'}(\boldsymbol{m}_0', \boldsymbol{m})$ is valid. The latter is a Π_2-sentence of FO$(\mathbb{Q}_+, +, >)$ and decidable in Π_2^{P} [25]. Hence, inclusion between continuous Petri nets is in Π_2^{P}.

4.2 The Coverability Decision Procedure

We now present Algorithm 3 for deciding coverability. This algorithm is an extension of the classical backward reachability algorithm that incorporates $\mathbb{Q}$-reachability checks during its execution in order to keep the set of minimal basis elements small.

First, on Line 1 we derive an open formula $\Phi(\boldsymbol{x})$ from $\Phi_{\mathcal{S}}$ such that $\Phi(\boldsymbol{x})$ holds if and only if $\boldsymbol{x}$ is $\mathbb{Q}$-coverable in $\mathcal{S}$. Then, on Line 2, the algorithm checks whether the marking $\boldsymbol{m}$ is $\mathbb{Q}$-coverable using the polynomial-time algorithm from [13] and returns that $\boldsymbol{m}$ is not coverable if this is not the case. Otherwise, the algorithm enters a loop which iteratively computes a basis M of the backward coverability set starting at $\boldsymbol{m}$ whose elements are in addition $\mathbb{Q}$-coverable in $\mathcal{S}$. To this end, on Line 5 the algorithm computes a set B of new basis elements obtained from one application of pb, and on Line 7 it removes from B the set D which contains all elements of B which are not $\mathbb{Q}$-coverable. If as a result B is empty the algorithm concludes that $\boldsymbol{m}$ is not coverable in $\mathcal{S}$. Otherwise, on Line 11 it adds the elements of B to M. Finally, Line 12 makes sure that in future iterations of the loop the underlying SMT solver can immediately discard elements that lie in $\uparrow D$. The latter is technically not necessary, but it provides some guidance to the SMT solver. The proof of the following proposition is deferred to Appendix A.

Proposition 7. *Let $\mathcal{S} = (\mathcal{N}, \boldsymbol{m}_0)$ be a PNS and $\boldsymbol{m}$ be a marking. Then $\boldsymbol{m}$ is coverable in $\mathcal{S}$ if and only if Algorithm 3 returns true.*

Remark 8. In our actual implementation, we use a slight variation of Algorithm 3 in which the instruction $M := \mathit{minbase}(M \cup B)$ in Line 11 is replaced by $M := \mathit{minbase}(M \cup \mathit{min}_{c,k} B)$. Here, $c, k \in \mathbb{N}$ are parameters to the algorithm, and $\mathit{min}_{c,k} B$ is the set of the $c + |B|/k$ elements of B with the smallest sum-norm. In this way, the empirically chosen parameters c and k create a bottleneck that gives priority to elements with small sum-norms, as they are more likely to allow for discarding elements with larger sum-norms in future iterations.

This variation of Algorithm 3 has the same correctness properties as the original one: It can be shown that using $\mathit{min}_{c,k} B$ instead of B in Line 11 computes the same set $\uparrow M$ at the expense of delaying its stabilization.

Before we conclude this section, let us come back to the question why in our approach we choose using $\Phi_{\mathcal{S}}$ (whose satisfiability is in NP) over Algorithm 2 which runs in polynomial time. In Algorithm 3, we invoke Algorithm 2 only once in Line 2 in order to check if $\mathcal{S}$ is not $\mathbb{Q}$-coverable, and thereafter only employ $\Phi_{\mathcal{S}}$ which gets incrementally updated during each iteration of the loop. The reason is that in practice as observed in our experimental evaluation below, Algorithm 2 turns out to be often faster for a *single* $\mathbb{Q}$-coverability query. Otherwise, as soon $\Phi_{\mathcal{S}}$ has been checked for satisfiability once, future satisfiability queries are significantly faster than Algorithm 2, which is a desirable behavior inside a backward coverability framework. Moreover we can constraint solutions to be in $\mathbb{N}$ instead of $\mathbb{Q}$, leading to a more precise over approximation.

4.3 Relationship to the CEGAR-approach of Esparza et al.

In [9], Esparza *et al.* presented a semi-decision procedure for coverability that is based on [10] and employs the Petri net state equation and traps inside a CEGAR-framework. A *trap in $\mathcal{N}$* is a non-empty subset of places $Q \subseteq P$ such that $Q^\bullet \subseteq {}^\bullet Q$, and $Q \subseteq P$ is a *siphon in $\mathcal{N}$* whenever ${}^\bullet Q \subseteq Q^\bullet$. Given a marking $\boldsymbol{m}$, a trap (respectively siphon) is *marked in $\boldsymbol{m}$* if $\sum_{p \in Q} \boldsymbol{m}(p) > 0$. An important property of traps is that if a trap is marked in $\boldsymbol{m}$, it will remain marked after

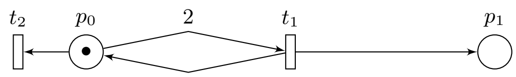

Fig. 1. A Petri net that cannot mark p_1.

any firing sequence starting in $\boldsymbol{m}$. Conversely, when a siphon is unmarked in $\boldsymbol{m}$ it remains so after any firing sequence starting in $\boldsymbol{m}$. By definition, Q is a trap in $\mathcal{N}$ if and only if Q is a siphon in $\mathcal{N}^{-1}$. The coverability criteria that [9] builds upon are derived from [10] and can be summarized as follows.

Proposition 9 ([9]). *If $\boldsymbol{m}$ is $\mathbb{Q}$-reachable (respectively reachable) in $(\mathcal{N}, \boldsymbol{m}_0)$ then there exists $\boldsymbol{x} \in \mathbb{Q}_+^T$ (respectively $\boldsymbol{x} \in \mathbb{N}^T$) such that:*

(i) $\boldsymbol{m} = \boldsymbol{m}_0 + \mathbf{C} \cdot \boldsymbol{x}$
(ii) for all traps $Q \subseteq P$, if Q is marked in $\boldsymbol{m}_0$ then Q is marked in $\boldsymbol{m}$

As in our approach, in [9] those criteria are checked using an SMT-solver. The for-all quantifier is replaced in [9] by incrementally enumerating all traps in a CEGAR-style fashion. It is shown in [13, Prop. 18] that Condition (iii) of Proposition 4 is equivalent to requiring that $\mathcal{N}_{[\![\boldsymbol{x}]\!]}^{-1}$ has no unmarked siphon in $\boldsymbol{m}$, which appears to be similar to Condition (ii) of Proposition 9. In fact, we show the following.

Proposition 10. *Conditions (i) and (iii) of Proposition 4 strictly imply Conditions (i) and (ii) of Proposition 9 (when interpreted over $\mathbb{Q}_+$).*

Proof. We only show strictness, the remainder of the proof is deferred to Appendix B. To this end, consider the Petri net $(\mathcal{N}, \boldsymbol{m}_0)$ depicted in Figure 1 with $\boldsymbol{m} = (0, 1)$. Clearly $\boldsymbol{m}$ is not reachable. There is a single solution to the state equation $\boldsymbol{x} = (1, 0)$. There is a single trap $\{p_1\}$ which is unmarked in $\boldsymbol{m}_0$. So the conditions of Proposition 9 hold, and hence the algorithm of [9] does not decide this net safe. On the contrary in $\mathcal{N}_{[\![\boldsymbol{x}]\!]}^{-1}$, the reverse net without t_2, $\{p_0\}$ is a siphon that is unmarked in $\boldsymbol{m}$. So Condition (iii) of Proposition 4 does not hold. □

This proposition shows that the single formula stated in Proposition 5 strictly subsumes the approach from [9]. Moreover, it provides a theoretical justification for why the approach of [9] performs so well in practice: the conditions are a strict subset of the conditions developed for $\mathbb{Q}$-reachability in [13].

5 Experimental Evaluation

We evaluate the backward coverability modulo $\mathbb{Q}$-reachability algorithm on standard benchmarks from the literature with two goals in mind. First, we demonstrate that our approach is competitive with existing approaches. In particular, we prove significantly more safe instances of our benchmarks safe in less time when compared to any other approach. Overall our algorithm decides 142 out of 176 instances, the best competitor decides 122 instances. Second, we demonstrate that $\mathbb{Q}$-coverability is a powerful pruning criterion by analyzing the relative number of minimal bases elements that get discarded during the execution of Algorithm 3.

We implemented Algorithm 3 in a tool called QCover in the programming language Python.[7] The underlying SMT-solver is z3 [5]. For the $min_{c,k}$ heuristic mentioned in Remark 8, we empirically chose $c = 10$ and $k = 5$. We observed that any sane choice of c and k leads to an overall speed-up, though different values lead to different (even increasing) running times on individual instances. QCover takes as input coverability instances in the mist file format.[8] The basis of our evaluation is the benchmark suite that was used in order to evaluate the tool Petrinizer, see [9] and the references therein. This suite consists of five benchmark categories: `mist`, consisting of 27 instances from the mist toolkit; `bfc`, consisting of 46 instances used for evaluating BFC; `medical` and `bug_tracking`, consisting of 12 and 41 instances derived from the provenance analysis of messages of a medical and a bug-tracking system, respectively; and `soter`, consisting of 50 instances of verification conditions derived from Erlang programs [8].

We compare QCover with the following tools: Petrinizer [9], mist [14] and bfc [19] in their latest versions available at the time of writing of this paper. mist implements a number of algorithms, we use the backward algorithm that uses places invariant pruning [15].[9] All benchmarks were performed on a single computer equipped with four Intel® Core™ 2.00 GHz CPUs, 8 GB of memory and Ubuntu Linux 14.04 (64 bits). The execution time of the tools was limited to 2000 seconds (i.e. 33 minutes and 20 seconds) per benchmark instance. The running time of every tool on an instance was determined using the sum of the user and sys time reported by the Linux tool `time`.

Figure 2 contains three tables which display the number of safe instances shown safe, unsafe instances shown unsafe, and the total number of instances of our benchmark suite decided by each individual tool. As expected, our algorithm outperforms all competitors on safe instances, since in this case a proof of safety (i.e. non-coverability) effectively requires the computation of the whole backward coverability set, and this is where pruning via $\mathbb{Q}$-coverability becomes most beneficial. On the other hand, QCover remains competitive on unsafe instances, though a tool such as BFC handles those instances better since its heuristics are more suited for proving unsafety (i.e. coverability). Nevertheless, QCover is the overall winner when comparing the number of safe and unsafe instances decided, being far ahead at the top of the leader-board deciding 142 out of 176 instances.

QCover not only decides more instances, it often does so faster than its competitors. Figure 3 contains two graphs which show the cumulative number of instances proven safe and the total number of instances decided on all suites

[7] QCover is available at `http://www-etud.iro.umontreal.ca/~blondimi/qcover/`.

[8] `https://github.com/pierreganty/mist/wiki#input-format-of-mist`

[9] `https://github.com/pierreganty/mist/wiki#coverability-checkers-included-in-mist`

Suite	QCover	Petrinizer	mist	bfc	Total
mist	**23**	20	22	20	23
medical	**11**	4	**11**	3	12
bfc	**2**	**2**	**2**	**2**	2
bug_tracking	**32**	**32**	0	19	40
soter	**37**	**37**	0	19	38
Total	**105**	95	35	63	115

Suite	QCover	Petrinizer	mist	bfc	Total
mist	3	—	**4**	**4**	4
medical	—	—	—	—	0
bfc	26	—	29	**42**	44
bug_tracking	0	—	0	**1**	1
soter	8	—	6	**12**	12
Total	37	0	39	**59**	61

Suite	QCover	Petrinizer	mist	bfc	Total
mist	**26**	20	**26**	24	27
medical	**11**	4	**11**	3	12
bfc	28	2	31	**44**	46
bug_tracking	**32**	**32**	0	20	41
soter	**45**	37	6	31	50
Total	**142**	95	74	122	176

Fig. 2. Number of safe instances (top-left), unsafe instances (top-right) and total instances (bottom) decided by every tool. Bold numbers indicate the tool(s) which decide(s) the largest number of instances in the respective category.

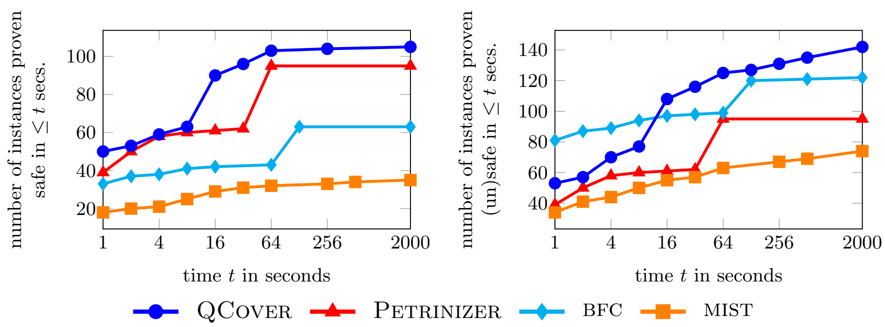

Fig. 3. Cumulative number of instances proven safe (left) and total number of instances decided (right) within a fixed amount of time.

by each tool within a certain amount of time. When it comes to safety, QCover is always ahead of all other tools. However, when looking at all instances decided, BFC first has an advantage. We observed that this advantage occurs on instances of comparably small size. As soon as large instances come into play, QCover wins the race. Besides different heuristics used, one reason for this might be the choice of the implementation language (C for BFC vs. Python for QCover). In particular, BFC can decide a non-negligible number of instances in less than 10ms, which QCover never achieves.

Finally, we consider the effectiveness of using $\mathbb{Q}$-coverability as a pruning criterion. To this end, consider Figure 4 in which we plotted the number of times a certain percentage of basis elements was removed due to not being $\mathbb{Q}$-coverable. Impressively, in some cases more than 95% of the basis elements get discarded. Overall, on average we discard 56% of the basis elements, which substantiates the usefulness of using $\mathbb{Q}$-coverability as a pruning criterion.

Before we conclude, let us mention that already 83 instances are proven safe by only checking the state equation, and that additionally checking for the criteria (ii) and (iii) of Proposition 4 increases this number to 101 instances.

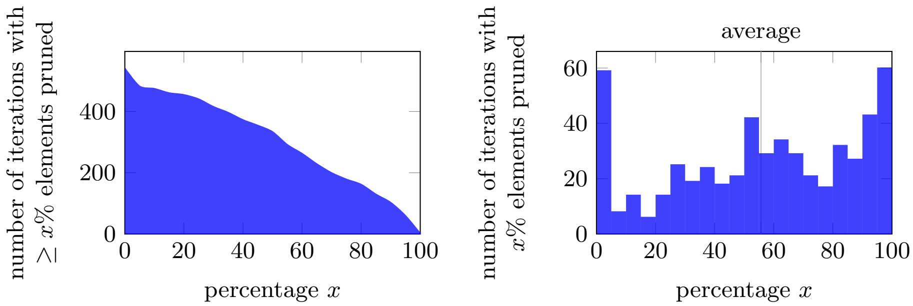

Fig. 4. Number of times a certain percentage of basis elements was removed due to $\mathbb{Q}$-coverability pruning.

If we use Algorithm 2 instead of our $\mathrm{FO}(\mathbb{Q}_+, +, >)$ encoding then we can only decide 132 instances in total. Finally, in our experiments, interpreting variables over $\mathbb{Q}$ instead of $\mathbb{N}$ resulted in no measurable overall performance gain.

In summary, our experimental evaluation shows that the backward coverability modulo $\mathbb{Q}$-reachability approach to the Petri net coverability problem developed in this paper is highly efficient when run on real-world instances, and superior to existing tools and approaches when compared on standard benchmarks from the literature.

6 Conclusion

In this paper, we introduced backward coverability modulo $\mathbb{Q}$-reachability, a novel approach to the Petri net coverability problem that is based on using coverability in continuous Petri nets as a pruning criterion inside a backward coverability framework. A key ingredient for the practicality of this approach is an existential $\mathrm{FO}(\mathbb{Q}_+, +, >)$-characterization of continuous reachability, which we showed to strictly subsume a recently introduced coverability semi-decision procedure [9]. Finally, we demonstrated that our approach significantly outperforms existing ones when compared on standard benchmarks.

There are a number of possible avenues for future work. It seems promising to combine the forward analysis approach based on incrementally constructing a Karp-Miller tree that is used in BFC [19] with the $\mathbb{Q}$-coverability approach introduced in this paper. In particular, recently developed minimization and acceleration techniques for constructing Karp-Miller trees should prove beneficial, see e.g. [17,24,26]. Another way to improve the empirical performance of our algorithm is to internally use more efficient data structures such as sharing trees [7]. It seems within reach that a tool which combines all of the aforementioned techniques and heuristics could decide all of the benchmark instances we used in this paper within reasonable resource restrictions.

Acknowledgments. We would like to thank Vincent Antaki for an early implementation of Algorithm 2. We would also like to thank Gilles Geeraerts for his support with the MIST file format.

References

1. Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Inf. Comput.*, 160(1-2):109–127, 2000.
2. Laura Bozzelli and Pierre Ganty. Complexity analysis of the backward coverability algorithm for VASS. In Giorgio Delzanno and Igor Potapov, editors, *Reachability Problems, RP*, volume 6945 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2011.
3. E. Cardoza, Richard J. Lipton, and Albert R. Meyer. Exponential space complete problems for Petri nets and commutative semigroups: Preliminary report. In *Symposium on Theory of Computing, STOC*, pages 50–54, 1976.
4. René David and Hassane Alla. Continuous Petri nets. In *Proceedings of the 8th European Workshop on Application and Theory of Petri nets*, pages 275–294, 1987.
5. Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.
6. Giorgio Delzanno, Jean-François Raskin, and Laurent Van Begin. Attacking symbolic state explosion. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Computer Aided Verification, CAV*, volume 2102 of *Lecture Notes in Computer Science*, pages 298–310. Springer, 2001.
7. Giorgio Delzanno, Jean-François Raskin, and Laurent Van Begin. Covering sharing trees: a compact data structure for parameterized verification. *STTT*, 5(2-3):268–297, 2004.
8. Emanuele D'Osualdo, Jonathan Kochems, and C.-H. Luke Ong. Automatic verification of Erlang-style concurrency. In Francesco Logozzo and Manuel Fähndrich, editors, *Static Analysis, SAS*, volume 7935 of *Lecture Notes in Computer Science*, pages 454–476. Springer, 2013.
9. Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp Meyer, and Filip Niksic. An SMT-based approach to coverability analysis. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification, CAV*, volume 8559 of *Lecture Notes in Computer Science*, pages 603–619. Springer, 2014.
10. Javier Esparza and Stephan Melzer. Verification of safety properties using integer programming: Beyond the state equation. *Formal Methods in System Design*, 16(2):159–189, 2000.
11. Alain Finkel, Jean-François Raskin, Mathias Samuelides, and Laurent Van Begin. Monotonic extensions of Petri nets: Forward and backward search revisited. *Electr. Notes Theor. Comput. Sci.*, 68(6):85–106, 2002.
12. Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.
13. Estíbaliz Fraca and Serge Haddad. Complexity analysis of continuous Petri nets. *Fundamenta Informaticae*, 137(1):1–28, 2015.
14. Pierre Ganty. Algorithmes et structures de données efficaces pour la manipulation de contraintes sur les intervalles (in French). Master's thesis, Université Libre de Bruxelles, Belgium, 2002.
15. Pierre Ganty, Cédric Meuter, Giorgio Delzanno, Gabriel Kalyon, Jean-François Raskin, and Laurent Van Begin. Symbolic data structure for sets of k-uples. Technical Report 570, Université Libre de Bruxelles, Belgium, 2007.

16. Gilles Geeraerts, Jean-François Raskin, and Laurent Van Begin. Expand, enlarge and check: New algorithms for the coverability problem of WSTS. *J. Comput. Syst. Sci.*, 72(1):180–203, 2006.
17. Gilles Geeraerts, Jean-François Raskin, and Laurent Van Begin. On the efficient computation of the minimal coverability set of petri nets. *Int. J. Found. Comput. Sci.*, 21(2):135–165, 2010.
18. Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992.
19. Alexander Kaiser, Daniel Kroening, and Thomas Wahl. A widening approach to multithreaded program verification. *ACM Trans. Program. Lang. Syst.*, 36(4):14:1–14:29, 2014.
20. Richard M. Karp and Raymond E. Miller. Parallel program schemata: A mathematical model for parallel computation. In *Switching and Automata Theory*, pages 55–61. IEEE Computer Society, 1967.
21. Johannes Kloos, Rupak Majumdar, Filip Niksic, and Ruzica Piskac. Incremental, inductive coverability. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification, CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 158–173. Springer, 2013.
22. Jérôme Leroux and Sylvain Schmitz. Demystifying reachability in vector addition systems. In *Logic in Computer Science, LICS*, pages 56–67. IEEE, 2015.
23. Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.*, 6:223–231, 1978.
24. Pierre-Alain Reynier and Frédéric Servais. Minimal coverability set for petri nets: Karp and miller algorithm with pruning. *Fundam. Inform.*, 122(1-2):1–30, 2013.
25. Eduardo D. Sontag. Real addition and the polynomial hierarchy. *Inf. Process. Lett.*, 20(3):115–120, 1985.
26. Antti Valmari and Henri Hansen. Old and new algorithms for minimal coverability sets. *Fundam. Inform.*, 131(1):1–25, 2014.
27. Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. On the complexity of equational horn clauses. In Robert Nieuwenhuis, editor, *Automated Deduction - CADE-20*, volume 3632 of *Lecture Notes in Computer Science*, pages 337–352. Springer, 2005.

A Proof of Propositon 7

Proof. Let B_n and M_n be respectively the value of B and M at Line 7 and Line 11 in the n-th iteration of the while-loop. It is possible to show by induction on n that

$$\uparrow M_n = \bigcup_{\boldsymbol{y} \in \uparrow \boldsymbol{m}} \left\{ \boldsymbol{x} \in \mathbb{N}^d : \mathbb{Q}\text{-}\mathit{coverable}(\mathcal{S}, \boldsymbol{x}) \wedge \exists \sigma \in T^* \text{ s.t. } \boldsymbol{x} \xrightarrow{\sigma} \boldsymbol{y} \wedge |\sigma| \leq n \right\}$$
$$B_n \subseteq \uparrow M_n \setminus \uparrow M_{n-1}.$$

Since $\boldsymbol{m}$ can be covered from $\boldsymbol{m}_0$ if and only if $\mathbb{Q}$-*coverable*$(\mathcal{S}, \boldsymbol{m_0})$ and $\boldsymbol{m}_0 \to^* \boldsymbol{m}$, it follows that Line 9 and 13 are correct and thus that the algorithm is correct. Regarding termination, it is well-known that, for well-quasi-ordered sets, any inclusion chain of upward closed sets stabilizes. Therefore, the chain $\uparrow M_1 \subseteq \uparrow M_2 \subseteq \ldots$ stabilizes to some $\uparrow M_n$. Thus, after a finite number of iterations, either B becomes empty at Line 8 or $\boldsymbol{m}_0 \in \uparrow M$ at Line 4. Hence, the algorithm always terminates. □

B Proof of Proposition 10

Proof. We write *marked*$(Q, \boldsymbol{m})$ if Q is marked in $\boldsymbol{m}$, and *unmarked*$(Q, \boldsymbol{m})$ otherwise. We establish the contrapositive: assuming that for all $\boldsymbol{x} \in \mathbb{Q}_+^T$ some condition of Proposition 9 does not hold, we show that for all $\boldsymbol{x} \in \mathbb{Q}_+^T$ some Condition (i) and (iii) of Proposition 4 does not hold. To this end, let $\boldsymbol{x} \in \mathbb{Q}_+^T$, if Condition (i) of Proposition 9 does not hold, i.e., $\boldsymbol{m} \neq \boldsymbol{m}_0 + \mathbf{C} \cdot \boldsymbol{x}$, then Condition (i) of Proposition 4 does not hold either. Thus, assume that $\boldsymbol{m} = \boldsymbol{m}_0 + \mathbf{C} \cdot \boldsymbol{x}$. We then have the following:

$$\begin{aligned}
&\text{there is a trap } Q \text{ in } \mathcal{N} \text{ s.t. } \mathit{marked}(Q, \boldsymbol{m}_0) \text{ and } \mathit{unmarked}(Q, \boldsymbol{m}) \\
\Longrightarrow\ &\text{there is a siphon } Q \text{ in } \mathcal{N}^{-1} \text{ s.t. } \mathit{marked}(Q, \boldsymbol{m}_0) \text{ and } \mathit{unmarked}(Q, \boldsymbol{m}) \\
\overset{(*)}{\Longrightarrow}\ &\text{there is a siphon } Q' \text{ in } \mathcal{N}_{T'}^{-1} \text{ s.t. } \mathit{unmarked}(Q', \boldsymbol{m}), \text{ where } T' = [\![\boldsymbol{x}]\!] \\
\Longrightarrow\ &[\![\boldsymbol{x}]\!] \notin \mathit{fs}(\mathcal{N}^{-1}, \boldsymbol{m}).
\end{aligned}$$

In order to show the implication $(*)$, we first observe that $\boldsymbol{m}[Q] \neq \boldsymbol{m}_0[Q]$, since Q is marked in $\boldsymbol{m}_0$ and not marked in $\boldsymbol{m}$. Let $T' \overset{\text{def}}{=} [\![\boldsymbol{x}]\!]$ and $P' \overset{\text{def}}{=} {}^\bullet T'^\bullet$, we claim that $Q' \overset{\text{def}}{=} Q \cap P'$ is a siphon in $\mathcal{N}_{T'}$. Let $t \in T'$ and suppose that $t \in {}^\bullet Q'$. Since $Q' \subseteq Q$, we have $t \in {}^\bullet Q$ and hence $t \in Q^\bullet$, which yields $t \in Q'^\bullet$ as all the neighbour places of t belong to P'. It remains to show that $Q' \neq \emptyset$. To the contrary, assume that $Q' = \emptyset$, then $\boldsymbol{x}(t) = 0$ for every $t \in {}^\bullet Q^\bullet$ and hence $\boldsymbol{m}[Q] = \boldsymbol{m}_0[Q]$, a contradiction. □