2011 26th Annual IEEE Symposium on Logic in Computer Science

Computing Optimal Coverability Costs in Priced Timed Petri Nets

Parosh Aziz Abdulla
Uppsala University, Sweden

Richard Mayr
University of Edinburgh, UK

***Abstract*—We consider timed Petri nets, i.e., unbounded Petri nets where each token carries a real-valued clock. Transition arcs are labeled with time intervals, which specify constraints on the ages of tokens. Our cost model assigns token storage costs per time unit to places, and firing costs to transitions. We study the cost to reach a given control-state. In general, a cost-optimal run may not exist. However, we show that the infimum of the costs is computable.**

***Keywords*-Formal verification; Petri nets; Timed Automata**

I. Introduction

Petri nets [1], [2] are a widely used model for the study and analysis of concurrent systems. Many different formalisms have been proposed which extend Petri nets with clocks and real-time constraints, leading to various definitions of *Timed Petri nets (TPNs)*. A complete discussion of all these formalisms is beyond the scope of this paper and the interested reader is referred to the surveys in [3], [4].

An important distinction is whether the time model is discrete or continuous. In discrete-time nets, time is interpreted as being incremented in discrete steps and thus the ages of tokens are in a countable domain, commonly the natural numbers. Such discrete-time nets have been studied in, e.g., [5], [6]. In continuous-time nets, time is interpreted as continuous, and the ages of tokens are real numbers. Some problems for continuous-time nets have been studied in [7], [8], [9], [10].

In parallel, there have been several works on extending the model of timed automata [11] with *prices* (*weights*) (see e.g., [12], [13], [14]). Weighted timed automata are suitable models for embedded systems, where we have to take into consideration the fact that the behavior of the system may be constrained by the consumption of different types of resources. Concretely, weighted timed automata extend classical timed automata with a cost function $Cost$ that maps every location and every transition to a nonnegative integer (or rational) number. For a transition, $Cost$ gives the cost of performing the transition. For a location, $Cost$ gives the cost per time unit for staying in the location. In this manner, we can define, for each computation of the system, the accumulated cost of staying in locations and performing transitions along the computation.

Here we consider a very expressive model that subsumes all models mentioned above. *Priced Timed Petri Nets* (*PTPN*) are a generalization of classic Petri nets [1] with real-valued (i.e., continuous-time) clocks, real-time constraints, and prices for computations.

Each token is equipped with a real-valued clock, representing the age of the token. The firing conditions of a transition include the usual ones for Petri nets. Additionally, each arc between a place and a transition is labeled with a time-interval whose bounds are natural numbers (or possibly ∞ as upper bound). These intervals can be open, closed or half open. When firing a transition, tokens which are removed/added from/to places must have ages lying in the intervals of the corresponding transition arcs. Furthermore, we add special *read-arcs* to our model. These affect the enabledness of transitions, but, unlike normal arcs, they do not remove the token from the input place. Read arcs preserve the exact age of the input token, unlike the scenario where a token is first removed and then replaced. Read arcs are necessary in order to make PTPN subsume the classic priced timed automata of [14]. We assign a cost to computations via a cost function $Cost$ that maps transitions and places of the Petri net to natural numbers. For a transition t, $Cost(t)$ gives the cost of performing the transition, while for a place p, $Cost(p)$ gives the cost per time unit per token in the place.

PTPN are a continuous-time model which subsumes the continuous-time TPN of [7], [8], [9], [10] and the priced timed automata of [12], [13], [14]. It should be noted that PTPN are infinite-state in several different ways. First, the Petri net itself is unbounded. So the number of tokens (and thus the number of clocks) can grow beyond any bound, i.e., the PTPN can create and destroy arbitrarily many clocks. In that PTPN differ from the priced timed automata of [12], [13], [14], which have only a finite number of control-states and only a fixed finite number of clocks. Secondly, every single clock value is a real number of which there are uncountably many.

Our contribution. We study the cost to reach a given control-state in a PTPN. In Petri net terminology, this is called a control-state reachability problem or a coverability problem. The related reachability problem (i.e., reaching a particular configuration) is undecidable for (continuous-time and discrete-time) TPN [5], even without taking costs into account. In general, a cost-optimal computation may not exist (e.g., even in priced timed automata it can happen that there is no computation of cost 0, but there exist computations of cost $\leq \epsilon$ for every $\epsilon > 0$). However, we show that the infimum of the costs is computable.

Supported by Royal Society grant JP080268.

1043-6871/11 $26.00 © 2011 IEEE
DOI 10.1109/LICS.2011.40

This cost problem had been shown to be decidable for the much simpler model of *discrete-time* PTPN in [15]. However, discrete-time PTPN do not subsume the priced timed automata of [14]. Moreover, the techniques from [15] do not carry over to the continuous-time domain (e.g., arbitrarily many delays of length 2^{-n} for $n = 1, 2, \ldots$ can can happen in ≤ 1 time).

Outline of Used Techniques. Since the PTPN model is very expressive, several powerful new techniques are developed to analyze them. These are interesting in their own right and can be instantiated to solve other problems.

In Section II we define PTPN and the priced coverability problem, and describe its relationship with priced timed automata and Petri nets. Then, in Sections III–V, we reduce the priced coverability problem for PTPN to a coverability problem in an abstracted untimed model called AC-PTPN. This abstraction is done by an argument similar to a construction in [14], where parameters indicating a feasible computation are contained in a polyhedron, which is described by a totally unimodular matrix. However, our class of matrices is more general than in [14], because PTPN allow the creation of new clocks with a nonzero value. The resulting AC-PTPN are still much more expressive than Petri nets, because their configurations are arbitrarily long sequences of multisets. Moreover, the transitions of AC-PTPN are not monotone, because larger configurations cost more and might thus exceed the cost limit. In order to solve coverability for AC-PTPN, we develop a very general method to solve reachability/coverability problems in infinite-state transition systems which are more general than the well-quasi-ordered/well-structured transition systems of [16], [17]. We call this method the *abstract phase construction*, and it is described in abstract terms in Section VI. In particular, it includes a generalization of the Valk-Jantzen construction [18] to arbitrary well-quasi-ordered domains. In Section VII, we instantiate this abstract method with AC-PTPN and prove the main result. This instantiation is nontrivial and requires several auxiliary lemmas, which ultimately use the decidability of the reachability problem for Petri nets with one inhibitor arc [19]. There exist close connections between timed Petri nets, Petri nets with one inhibitor arc, and transfer nets.

II. Priced Timed Petri Nets

a) Preliminaries: We use $\mathbb{N}, \mathbb{R}_{\geq 0}, \mathbb{R}_{>0}$ to denote the sets of natural numbers (including 0), nonnegative reals, and strictly positive reals, respectively. For a natural number k, we use $\mathbb{N}^k$ and $\mathbb{N}_\omega^k$ to denote the set of vectors of size k over $\mathbb{N}$ and $\mathbb{N} \cup \{\omega\}$, respectively ($\omega$ represents the first limit ordinal). For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{0, \ldots, n\}$. For $x \in \mathbb{R}_{\geq 0}$, we use $\mathit{frac}(x)$ to denote the fractional part of x. We use a set Intrv of intervals. An open interval is written as $(w : z)$ where $w \in \mathbb{N}$ and $z \in \mathbb{N} \cup \{\infty\}$. Intervals can also be closed in one or both directions, e.g. $[w : z]$ is closed in both directions and $[w : z)$ is closed to the left and open to the right.

For a set A, we use A^* and $A^\odot$ to denote the set of finite words and finite multisets over A, respectively. We view a multiset b over A as a mapping $b : A \mapsto \mathbb{N}$. Sometimes, we write finite multisets as lists (possibly with multiple occurrences), so both $[2.4, 2.4, 2.4, 5.1, 5.1]$ and $[2.4^3\ ,\ 5.1^2]$ represent a multiset b over $\mathbb{R}_{\geq 0}$ where $b(2.4) = 3$, $b(5.1) = 2$ and $b(x) = 0$ for $x \neq 2.4, 5.1$. For multisets b_1 and b_2 over A, we say that $b_1 \leq b_2$ if $b_1(a) \leq b_2(a)$ for each $a \in A$. We define $b_1 + b_2$ to be the multiset b where $b(a) = b_1(a) + b_2(a)$, and (assuming $b_1 \leq b_2$) we define $b_2 - b_1$ to be the multiset b where $b(a) = b_2(a) - b_1(a)$, for each $a \in A$. We use $a \in b$ to denote that $b(a) > 0$. We use $\varnothing$ or $[\,]$ to denote the empty multiset and ε to denote the empty word. Let $(A, \leq)$ be a poset. We define a partial order $\leq^w$ on A^* as follows. Let $a_1 \ldots a_n \leq^w b_1 \ldots b_m$ iff there is a subsequence $b_{j_1} \ldots b_{j_n}$ of $b_1 \ldots b_m$ s.t. $\forall k \in \{1, \ldots, n\}. a_k \leq b_{j_k}$. A subset $B \subseteq A$, is said to be *upward closed* in A if $a_1 \in B, a_2 \in A$ and $a_1 \leq a_2$ implies $a_2 \in B$. If A is known from the context, then we say simply that B is *upward closed*. For $B \subseteq A$ we define the *upward closure* $B\uparrow$ to be the set $\{a \in A \mid \exists a' \in B : a' \leq a\}$. A *downward closed* set B and the *downward closure* $B\downarrow$ are defined in a similar manner. We use $a\uparrow$, $a\downarrow$, a instead of $\{a\}\uparrow$, $\{a\}\downarrow$, $\{a\}$, respectively. Given a transition relation $\longrightarrow$, we denote its transitive closure by $\xrightarrow{+}$ and its reflexive-transitive closure by $\xrightarrow{*}$. Given a set of configurations C, let $\mathit{Pre}_{\longrightarrow}(C) = \{c' \mid \exists c \in C. c' \longrightarrow c\}$ and $\mathit{Pre}^*_{\longrightarrow}(C) = \{c' \mid \exists c \in C. c' \xrightarrow{*} c\}$.

b) Priced Timed Petri Nets: A *Priced Timed Petri Net (PTPN)* is a tuple $\mathcal{N} = (Q, P, T, \mathit{Cost})$ where Q is a finite set of control-states and P is a finite set of places. T is a finite set of transitions, where each transition $t \in T$ is of the form $t = (q_1, q_2, \mathit{In}, \mathit{Read}, \mathit{Out})$. We have that $q_1, q_2 \in Q$ are the source and target control-state, respectively, and $\mathit{In}, \mathit{Read}, \mathit{Out} \in (P \times \mathit{Intrv})^\odot$ are finite multisets over $P \times \mathit{Intrv}$ which define the input-arcs, read-arcs and output-arcs of t, respectively. $\mathit{Cost} : P \cup T \to \mathbb{N}$ is the cost function assigning firing costs to transitions and storage costs to places. Note that it is not a restriction to use integers for time bounds and costs in PTPN. By the same standard technique as in timed automata, the problem for rational numbers can be reduced to the integer case (by multiplying all numbers with the lcm of the divisors). To simplify the presentation we use a one-dimensional cost. This can be generalized to multidimensional costs; see Section X. We let cmax denote the maximum integer appearing on the arcs of a given PTPN. A *configuration* of $\mathcal{N}$ is a tuple (q, M) where $q \in Q$ is a control-state and M is a *marking* of $\mathcal{N}$. A marking is a multiset over $P \times \mathbb{R}_{\geq 0}$, i.e., $M \in (P \times \mathbb{R}_{\geq 0})^\odot$. The marking M defines the numbers and ages of tokens in each place in the net. We identify a token in a marking M by the pair (p, x) representing its place and age in M. Then, $M(p, x)$ defines the number of tokens with age x in place p. Abusing notation, we define, for each place p, a multiset $M(p)$ over $\mathbb{R}_{\geq 0}$, where $M(p)(x) = M(p, x)$.

For a marking M of the form $[(p_1, x_1)\ ,\ \ldots\ ,\ (p_n, x_n)]$ and $x \in \mathbb{R}_{>0}$, we use M^{+x} to denote the marking $[(p_1, x_1 + x)\ ,\ \ldots\ ,\ (p_n, x_n + x)]$.

c) Computations: We define two transition relations on the set of configurations: timed transition and discrete transition. A *timed transition* increases the age of each token

by the same real number. Formally, for $x \in \mathbb{R}_{>0}, q \in Q$, we have $(q, M_1) \xrightarrow{x}_{Time} (q, M_2)$ if $M_2 = M_1^{+x}$. We use $(q, M_1) \longrightarrow_{Time} (q, M_2)$ to denote that $(q, M_1) \xrightarrow{x}_{Time} (q, M_2)$ for some $x \in \mathbb{R}_{>0}$.

We define the set of *discrete transitions* $\longrightarrow_{Disc}$ as $\bigcup_{t \in T} \longrightarrow_t$, where $\longrightarrow_t$ represents the effect of *firing* the discrete transition t. To define $\longrightarrow_t$ formally, we need the auxiliary predicate $match$ that relates markings with the inputs/reads/outputs of transitions. Let $M \in (P \times \mathbb{R}_{\geq 0})^{\odot}$ and $\alpha \in (P \times Intrv)^{\odot}$. Then $match(M, \alpha)$ holds iff there exists a bijection $f : M \mapsto \alpha$ s.t. for every $(p, x) \in M$ we have $f((p, x)) = (p', \mathcal{I})$ with $p' = p$ and $x \in \mathcal{I}$. Let $t = (q_1, q_2, In, Read, Out) \in T$. Then we have a discrete transition $(q_1, M_1) \longrightarrow_t (q_2, M_2)$ iff there exist $I, O, R, M_1^{rest} \in (P \times \mathbb{R}_{\geq 0})^{\odot}$ s.t. the following conditions are satisfied:

- $M_1 = I + R + M_1^{rest}$
- $match(I, In)$, $match(R, Read)$ and $match(O, Out)$.
- $M_2 = O + R + M_1^{rest}$

We say that t is *enabled* in (q_1, M_1) if the first two conditions are satisfied. A transition t may be fired iff for each input-arc and each read-arc, there is a token with the right age in the corresponding input place. These tokens in I matched to the input arcs will be removed when the transition is fired, while the tokens in R matched to the read-arcs are kept. The newly produced tokens in O have ages which are chosen nondeterministically from the relevant intervals on the output arcs of the transitions. This semantics is lazy, i.e., enabled transitions do not need to fire and can be disabled again.

We write $\longrightarrow = \longrightarrow_{Time} \cup \longrightarrow_{Disc}$ to denote all transitions. For sets C, C' of configurations, we write $C \xrightarrow{*} C'$ to denote that $c \xrightarrow{*} c'$ for some $c \in C$ and $c' \in C'$. A *computation* π (from c to c') is a sequence of transitions $c_0 \longrightarrow c_1 \longrightarrow \ldots \longrightarrow c_n$ such that $c_0 = c$ and $c_n = c'$. We write $c \xrightarrow{\pi} c'$ to denote that π is a computation from c to c'. Similarly, we write $C \xrightarrow{\pi} C'$ to denote that $\exists c_1 \in C, c_n \in C'. c_1 \xrightarrow{\pi} c_n$.

d) Costs: The cost of a computation consisting of one discrete transition $t \in T$ is defined as $Cost((q_1, M_1) \longrightarrow_t (q_2, M_2)) := Cost(t)$. The cost of a computation consisting of one timed transition is defined by $Cost\left((q, M) \xrightarrow{x} (q, M^{+x})\right) := x * \sum_{p \in P} |M(p)| * Cost(p)$. The cost of a computation is the sum of all transition costs in it, i.e., $Cost((q_1, M_1) \longrightarrow (q_2, M_2) \longrightarrow \ldots \longrightarrow (q_n, M_n)) := \sum_{1 \leq i < n} Cost((q_i, M_i) \longrightarrow (q_{i+1}, M_{i+1}))$. We write $C \xrightarrow{v} C'$ to denote that there is a computation π such that $C \xrightarrow{\pi} C'$ and $Cost(\pi) \leq v$. We define $OptCost(C, C')$ to be the infimum of the set $\left\{v \mid C \xrightarrow{v} C'\right\}$, i.e., the infimum of the costs of all computations leading from C to C'. We use the infimum, because the minimum does not exist in general. We partition the set of places $P = P_c \cup P_f$ where $Cost(p) > 0$ for $p \in P_c$ and $Cost(p) = 0$ for $p \in P_f$. The places in P_c are called cost-places and the places in P_f are called free-places.

e) Relation of PTPN to Other Models: PTPN subsume the priced timed automata of [12], [13], [14] via the following simple encoding. For every one of the finitely many clocks of the automaton we have one place in the PTPN with exactly one token on it whose age encodes the clock value. We assign cost zero to these places. For every control-state s of the automaton we have one place p_s in the PTPN. Place p_s contains exactly one token iff the automaton is in state s, and it is empty otherwise. An automaton transition from state s to state s' is encoded by a PTPN transition consuming the token from p_s and creating a token on $p_{s'}$. The transition guards referring to clocks are encoded as read-arcs to the places which encode clocks, labeled with the required time intervals. Note that open and half-open time intervals are needed to encode the strict inequalities used in timed automata. Clock resets are encoded by consuming the timed token (by an input-arc) and replacing it (by an output-arc) with a new token on the same place with age 0. The cost of staying in state s is encoded by assigning a cost to place p_s, and the cost of performing a transition is encoded as the cost of the corresponding PTPN transition. Also PTPN subsume fully general unbounded (i.e., infinite-state) Petri nets (by setting all time intervals to $[0 : \infty)$ and thus ignoring the clock values).

Note that (just like for timed automata) the problems for continuous-time PTPN cannot be reduced to (or approximated by) the discrete-time case. Replacing strict inequalities with non-strict ones might make the final control-state reachable, when it originally was unreachable.

f) The Priced Coverability Problem: We will consider two variants of the cost problem, the *Cost-Threshold* problem and the *Cost-Optimality* problem. They are both characterized by an *initial* control state q_{init} and a *final* control state q_{fin}.

Let $C_{init} = (q_{init}, [])$ be the initial configuration and $\mathcal{C}_{fin} = \{(q_{fin}, M) \mid M \in (P \times \mathbb{R}_{\geq 0})^{\odot}\}$ the set of final configurations defined by the control-state q_{fin}. I.e., we start from a configuration where the control state is q_{init} and where all the places are empty, and then consider the cost of computations that takes us to q_{fin}. (If C_{init} contained tokens with a non-integer age then the optimal cost might not be an integer.)

In the *Cost-Threshold* problem we ask the question whether $OptCost(C_{init}, \mathcal{C}_{fin}) \leq v$ for a given threshold $v \in \mathbb{N}$.

In the *Cost-Optimality* problem, we want to compute $OptCost(C_{init}, \mathcal{C}_{fin})$. (Example in Appendix A.)

III. Computations in δ-Form

We show that, in order to solve the cost problems it is sufficient to consider computations of a certain form where the ages of all the tokens are arbitrarily close to an integer.

The decomposition of a PTPN marking M into its fractional parts $M_{-m}, \ldots, M_{-1}, M_0, M_1, \ldots, M_n$, is uniquely defined by the following properties:

- $M = M_{-m} + \cdots + M_{-1} + M_0 + M_1 + \cdots + M_n$.
- If $(p, x) \in M_i$ and $i < 0$ then $frac(x) \geq 1/2$. If $(p, x) \in M_0$ then $frac(x) = 0$. If $(p, x) \in M_i$ and $i > 0$ then $frac(x) < 1/2$.
- Let $(p_i, x_i) \in M_i$ and $(p_j, x_j) \in M_j$. Then $frac(x_i) = frac(x_j)$ iff $i = j$, and if $-m \leq i < j < 0$ or $0 \leq i < j \leq n$ then $frac(x_i) < frac(x_j)$.
- $M_i \neq \varnothing$ if $i \neq 0$ (M_0 can be empty, but the other M_i must be non-empty in order to get a unique representation.)

We say that a timed transition $(q, M) \xrightarrow{x} (q, M')$ is *detailed* iff at most one fractional part of any token in M changes its status about reaching or exceeding the next higher integer value. Formally, let ϵ be the fractional part of the token ages in M_{-1}, or $\epsilon = 1/2$ if M_{-1} does not exist. Then $(q, M) \xrightarrow{x} (q, M')$ is *detailed* iff either $0 < x < 1 - \epsilon$ (i.e., no tokens reach the next integer), or $M_0 = \varnothing$ and $x = \epsilon$ (no tokens had integer age, but those in M_{-1} reach integer age). Every computation of a PTPN can be transformed into an equivalent one (w.r.t. reachability and cost) where all timed transitions are detailed. Thus we may assume w.l.o.g. that timed transitions are detailed. This property is needed to obtain a one-to-one correspondence between PTPN steps and the steps of A-PTPN, defined in the next section.

For $\delta \in (0 : 1/5]$ the marking $[(p_1, x_1), \ldots, (p_n, x_n)]$ is in *δ-form* if, for all $i : 1 \le i \le n$, it is the case that either (i) $frac(x_i) < \delta$ (low fractional part), or (ii) $frac(x_i) > 1-\delta$ (high fractional part). I.e., the age of each token is close to (within $< \delta$) an integer. We choose $\delta \le 1/5$ to ensure that the cases (i) and (ii) do not overlap, and that they still do not overlap for a new $\delta' \le 2/5$ after a delay of $\le 1/5$ time units.

The occurrence of a discrete transition t is said to be in δ-form if its output O is in δ-form, i.e., the ages of the newly generated tokens are close to an integer. This is not a property of the transition t as such, but a property of its occurrence, because it depends on the particular choice of O.

Let $\mathcal{N} = (Q, P, T, Cost)$ be a PTPN and $C_{init} = (q_{init}, [])$ and $\mathcal{C}_{fin} = \{(q_{fin}, M) \mid M \in (P \times \mathbb{R}_{\ge 0})^{\oplus}\}$ as in the last section.

For $0 < \delta \le 1/5$, the computation π is in δ-form iff (1) every occurrence of a discrete transition $c_i \longrightarrow_t c_{i+1}$ is in δ-form, and (2) for every timed transition $c_i \xrightarrow{x} c_{i+1}$ we have either $x \in (0 : \delta)$ or $x \in (1 - \delta : 1)$. We show that, in order to find the infimum of the possible costs, it suffices to consider computations in δ-form, for arbitrarily small values of $\delta > 0$.

Lemma 1. *Let $C_{init} \xrightarrow{\pi} \mathcal{C}_{fin}$, where π is $C_{init} = c_0 \longrightarrow \ldots \longrightarrow c_{length} \in \mathcal{C}_{fin}$. Then for every $\delta > 0$ there exists a computation π' in δ-form where $C_{init} \xrightarrow{\pi'} \mathcal{C}_{fin}$, where π' is $C_{init} = c'_0 \longrightarrow \ldots \longrightarrow c'_{length} \in \mathcal{C}_{fin}$ s.t. $Cost(\pi') \le Cost(\pi)$, π and π' have the same length and $\forall i : 0 \le i \le length. |c_i| = |c'_i|$. Furthermore, if π is detailed then π' is detailed.*

Corollary 2. *For every $\delta > 0$ we have $OptCost(C_{init}, \mathcal{C}_{fin}) = \inf\{Cost(\pi) \mid C_{init} \xrightarrow{\pi} \mathcal{C}_{fin}, \pi$ in δ-form$\}$.*

IV. Abstract PTPN

We now reduce the Cost-Optimality problem to a simpler case without explicit clocks by defining a new class of systems called *abstract PTPN* (for short A-PTPN), whose computations represent PTPN computations in δ-form, for infinitesimally small values of $\delta > 0$. For each PTPN $\mathcal{N} = (Q, P, T, Cost)$, we define a corresponding A-PTPN $\mathcal{N}'$ (sometimes denoted by $aptpn(\mathcal{N})$). The A-PTPN $\mathcal{N}'$ is syntactically of the same form $(Q, P, T, Cost)$ as $\mathcal{N}$. However, $\mathcal{N}'$ induces a different transition system (its configurations and operational semantics are different). Below we define the set of markings of the A-PTPN, and then describe the transition relation. We will also explain the relation to the markings and the transition relation induced by the original PTPN.

g) Markings and Configurations: Fix a $\delta : 0 < \delta \le 2/5$. A marking M of $\mathcal{N}$ in δ-form is encoded by a marking $aptpn(M)$ of $\mathcal{N}'$ which is described by a triple $\left(w^{high}, b_0, w^{low}\right)$ where $w^{high}, w^{low} \in \left((P \times [cmax + 1])^{\oplus}\right)^*$ and $b_0 \in (P \times [cmax + 1])^{\oplus}$. The ages of the tokens in $aptpn(M)$ are integers and therefore only carry the integral parts of the tokens in the original PTPN. However, the marking $aptpn(M)$ carries additional information about the fractional parts of the tokens as follows. The tokens in w^{high} represent tokens in M that have *high* fractional parts (their values are at most δ below the next integer); the tokens in w^{low} represent tokens in M that have *low* fractional parts (their values at most δ above the previous integer); while tokens in b_0 represent tokens in M that have *zero* fractional parts (their values are equal to an integer). Furthermore, the ordering among the fractional parts of tokens in w^{high} (resp. w^{low}) is represented by the positions of the multisets to which they belong in w^{high} (resp. w^{low}). Let $M = M_{-m}, \ldots, M_{-1}, M_0, M_1, \ldots, M_n$ be the decomposition of M into fractional parts. Then we define $aptpn(M) := \left(w^{high}, b_0, w^{low}\right)$ with $w^{high} = b_{-m} \ldots b_{-1}$, and $w^{low} = b_1 \ldots b_n$, where $b_i((p, \lfloor x \rfloor)) = M_i((p, x))$ if $x \le cmax$. (This is well defined, because M_i contains only tokens with one particular fractional part.) Furthermore, $b_i((p, cmax+1)) = \sum_{y > cmax} M((p, y))$, i.e., all tokens whose age is $> cmax$ are abstracted as tokens of age $cmax + 1$, because the PTPN cannot distinguish between token ages $> cmax$. Note that w^{high} and w^{low} represent tokens with fractional parts in increasing order. An A-PTPN configuration is a control-state plus a marking. If we apply $aptpn$ to a set of configurations (i.e., $aptpn(\mathcal{C}_{fin})$), we implicitly restrict this set to the subset of configurations in $2/5$-form.

h) Transition Relation: The transitions on the A-PTPN are defined as follows. For every discrete transition $t = (q_1, q_2, In, Read, Out) \in T$ we have $(q_1, b_{-m} \ldots b_{-1}, b_0, b_1 \ldots b_n) \longrightarrow_t (q_2, c_{-m'} \ldots c_{-1}, c_0, c_1 \ldots c_{n'})$ if the following conditions are satisfied: For every $i : -m \le i \le n$ there exist $b_i^I, b_i^R, b_i^{rest}, \hat{O}, b_0^O \in (P \times [cmax + 1])^{\oplus}$ s.t. for every $0 < \epsilon < 1$ we have

- $b_i = b_i^I + b_i^R + b_i^{rest}$ for $-m \le i \le n$
- $match((\sum_{i \ne 0} b_i^I)^{+\epsilon} + b_0^I, In)$
- $match((\sum_{i \ne 0} b_i^R)^{+\epsilon} + b_0^R, Read)$
- $match(\hat{O}^{+\epsilon} + b_0^O, Out)$
- There is a strictly monotone injection $f : \{-m, \ldots, n\} \mapsto \{-m', \ldots, n'\}$ where $f(0) = 0$ s.t. $c_{f(i)} \ge b_i - b_i^I$ and $c_0 = b_0 - b_0^I + b_0^O$ and $\sum_{i \ne 0} c_i = (\sum_{i \ne 0} b_i - b_i^I) + \hat{O}$.

The intuition is that the A-PTPN tokens in b_i for $i \ne 0$ represent PTPN tokens with a little larger, and strictly positive, fractional part. Thus their age is incremented by $\epsilon > 0$ before it is matched to the input, read and output arcs. The fractional parts of the tokens that are not involved in the transition stay the same. However, since all the time intervals in the PTPN

have integer bounds, the fractional parts of newly created tokens are totally arbitrary. Thus they can be inserted at any position in the sequence, between any positions in the sequence, or before/after the sequence of existing fractional parts. This is specified by the last condition on the sequence $c_{-m'} \dots c_{-1}, c_0, c_1 \dots c_{n'}$.

Lemma 3. *Let (q, M) be a PTPN configuration in δ-form for some $\delta \leq 1/5$. There is an occurrence of a discrete transition in δ-form $(q, M) \longrightarrow_t (q', M')$ if and only if $aptpn((q, M)) \longrightarrow_t aptpn((q', M'))$.*

Additionally there are A-PTPN transitions that encode the effect of PTPN detailed timed transitions $\xrightarrow{x}$ for $x \in (0 : \delta)$ or $x \in (1-\delta : 1)$ for sufficiently small $\delta > 0$. We call these *abstract timed transitions*. For any multiset $b \in (P \times [cmax + 1])^{\odot}$ let $b^+ \in (P \times [cmax + 1])^{\odot}$ be defined by $b^+((p, x + 1)) = b((p, x))$ for $x \leq cmax$ and $b^+((p, cmax + 1)) = b((p, cmax + 1)) + b((p, cmax))$, i.e., the age $cmax + 1$ represents all ages $> cmax$. There are 4 different types of abstract timed transitions. (In the following all b_i are nonempty.)

- Type 1 $(q_1, b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n) \longrightarrow (q_1, b_{-m} \dots b_{-1}, \varnothing, b_0 b_1 \dots b_n)$. This simulates a very small delay $\delta > 0$ where the tokens of integer age in b_0 now have a positive fractional part, but no tokens reach an integer age.
- Type 2 $(q_1, b_{-m} \dots b_{-1}, \varnothing, b_1 \dots b_n) \longrightarrow (q_1, b_{-m} \dots b_{-2}, b^+_{-1}, b_1 \dots b_n)$. This simulates a very small delay $\delta > 0$ in the case where there were no tokens of integer age and the tokens in b_{-1} just reach the next higher integer age.
- Type 3 $(q_1, b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n) \longrightarrow (q_1, b^+_{-m} \dots b^+_{-2} b^+_{-1} b_0 \dots b_k, \varnothing, b^+_{k+1} \dots b^+_n)$ for some $k \in \{0, \dots, n\}$. This simulates a delay in $(1 - \delta : 1)$ where the tokens in $b_0 \dots b_k$ do not quite reach the next higher integer and no token gets an integer age.
- Type 4 $(q_1, b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n) \longrightarrow (q_1, b^+_{-m} \dots b^+_{-2} b^+_{-1} b_0 \dots b_k, b^+_{k+1}, b^+_{k+2} \dots b^+_n)$ for some $k \in \{0, \dots, n-1\}$. This simulates a delay in $(1 - \delta : 1)$ where the tokens in $b_0, \dots b_k$ do not quite reach the next higher integer and the tokens on b_{k+1} just reach the next higher integer age.

Lemma 4. *Let (q, M) be a PTPN configuration in δ-form for some $\delta \leq 1/5$ and $x \in (0 : \delta)$. There is a PTPN detailed timed transition $(q, M) \xrightarrow{x} (q, M^{+x})$ if and only if there is a A-PTPN abstract timed transition of type 1 or 2 s.t. $aptpn((q, M)) \longrightarrow aptpn((q, M^{+x}))$.*

Lemma 5. *Let (q, M) be a PTPN configuration in δ-form for some $\delta \leq 1/5$ and $x \in (1 - \delta : 1)$. There is a PTPN timed transition $(q, M) \xrightarrow{x} (q, M^{+x})$ if and only if there is a A-PTPN transition of either type 3 or 4 s.t. $aptpn((q, M)) \longrightarrow aptpn((q, M^{+x}))$.*

The cost model for A-PTPN is defined as follows. For every transition $t \in T$ we have $Cost((q_1, M_1) \longrightarrow_t (q_2, M_2)) := Cost(t)$, just like in PTPN. For abstract timed transitions of types 1 and 2 we define the cost as zero. For abstract timed transitions $(q, M_1) \longrightarrow (q, M_2)$ of types 3 and 4, we define $Cost((q, M_1) \longrightarrow (q, M_2)) := \sum_{p \in P} |M_1(p)| * Cost(p)$ (i.e., as if the elapsed time had length 1). The intuition is that, as δ converges to zero, the cost of the PTPN timed transitions of length in $(0 : \delta)$ (types 1 and 2) or in $(1 - \delta : 1)$ (types 3 and 4) converges to the cost of the corresponding abstract timed transitions in the A-PTPN. The following Lemma 6, which follows from Lemmas 3,4,5, shows this formally.

Lemma 6.

1) *Let c_0 be a PTPN configuration where all tokens have integer ages. For every PTPN computation $\pi = c_0 \longrightarrow \dots \longrightarrow c_n$ in detailed form and δ-form s.t. $n * \delta \leq 1/5$ there exists a corresponding A-PTPN computation $\pi' = aptpn(c_0) \longrightarrow \dots \longrightarrow aptpn(c_n)$ s.t.*

$$|Cost(\pi) - Cost(\pi')| \leq n * \delta * (\max_{0 \leq i \leq n} |c_i|) * (\max_{p \in P} Cost(p))$$

2) *Let c'_0 be a A-PTPN configuration $(\epsilon, b_0, \epsilon)$. For every A-PTPN computation $\pi' = c'_0 \longrightarrow \dots \longrightarrow c'_n$ and every $0 < \delta \leq 1/5$ there exists a PTPN computation $\pi = c_0 \longrightarrow \dots \longrightarrow c_n$ in detailed form and δ-form s.t. $c'_i = aptpn(c_i)$ for $0 \leq i \leq n$ and*

$$|Cost(\pi) - Cost(\pi')| \leq n * \delta * (\max_{0 \leq i \leq n} |c'_i|) * (\max_{p \in P} Cost(p))$$

Theorem 7. *The infimum of the costs in a PTPN coincide with the infimum of the costs in the corresponding A-PTPN.*

$$\inf\{Cost(\pi) \mid C_{init} \xrightarrow{\pi} \mathcal{C}_{fin}\} = \inf\{Cost(\pi') \mid aptpn(C_{init}) \xrightarrow{\pi'} aptpn(\mathcal{C}_{fin})\}$$

V. Abstracting Costs in A-PTPN

Given an A-PTPN, the cost-threshold problem is whether there exists a computation $aptpn(C_{init}) \xrightarrow{\pi} aptpn(\mathcal{C}_{fin})$ s.t. $Cost(\pi) \leq v$ for a given threshold v.

We now reduce this question to a question about simple coverability in a new model called AC-PTPN. The idea is to encode the cost of the computation into a part of the control-state. For every A-PTPN and cost threshold $v \in \mathbb{N}$ there is a corresponding AC-PTPN that is defined as follows.

For every A-PTPN configuration $(q, b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n)$ there are AC-PTPN configurations $((q, y), b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n)$ for all integers $0 \leq y \leq v$, where y represents the remaining allowed cost of the computation. We define a finite set of functions ac_y for $0 \leq y \leq v$ that map A-PTPN configurations to AC-PTPN configurations s.t. $ac_y((q, b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n)) = ((q, y), b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n)$.

For every discrete transition $t = (q_1, q_2, In, Read, Out) \in T$ with $(q_1, b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n) \longrightarrow_t (q_2, c_{-m'} \dots c_{-1}, c_0, c_1 \dots c_{n'})$ in the A-PTPN, we have instead $((q_1, y), b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n) \longrightarrow_t ((q_2, y - Cost(t), c_{-m'} \dots c_{-1}, c_0, c_1 \dots c_{n'})$ in the AC-PTPN for $v \geq y \geq Cost(t)$. I.e., we deduct the cost of the transition from the remaining allowed cost of the computation.

For every A-PTPN abstract timed transition of the types 1 and 2 $(q_1, \dots) \longrightarrow (q_1, \dots)$ we have corresponding AC-PTPN abstract timed transitions of types 1 and 2 where $((q_1, y), \dots) \longrightarrow ((q_1, y), \dots)$ for all $0 \leq y \leq v$. I.e., infinitesimally small delays do not cost anything.

For every A-PTPN abstract timed transition of type 3 $(q_1, b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n) \longrightarrow (q_1, b^+_{-m} \dots b^+_{-2} b^+_{-1} b_0 \dots b_k, \varnothing, b^+_{k+1} \dots b^+_n)$ we have corresponding AC-PTPN abstract timed transitions of type 3 where $((q_1, y), b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n) \longrightarrow ((q_1, y - z), b^+_{-m} \dots b^+_{-2} b^+_{-1} b_0 \dots b_k, \varnothing, b^+_{k+1} \dots b^+_n)$ where $z = \sum_{i=-m}^{n} \sum_{p \in P} |b_i(p)| * Cost(p)$ and $v \geq y \geq z$.

Transitions of type 4 are handled analogously.

Lemma 8. *There is an A-PTPN computation $aptpn(C_{init}) \xrightarrow{\pi} aptpn(\mathcal{C}_{fin})$ with $Cost(\pi) \leq v$ iff there is a corresponding AC-PTPN computation $ac_v(aptpn(C_{init})) \xrightarrow{\pi'} \bigcup_{0 \leq y \leq v} ac_y(aptpn(\mathcal{C}_{fin}))$*

Proof: Directly from the definition of AC-PTPN. ∎

Note that, unlike A-PTPN, AC-PTPN are not monotone. This is because steps of type 3/4 with more tokens on cost-places cost more, and thus cost-constraints might block transitions from larger configurations.

VI. The Abstract Coverability Problem

We describe a general construction for solving reachability/coverability problems under some abstract conditions. Later we will show how this construction can be applied to AC-PTPN (and thus the A-PTPN and PTPN cost problems).

A. The Generalized Valk-Jantzen Construction

Theorem 9. *(Valk & Jantzen [18]) Given an upward-closed set $V \subseteq \mathbb{N}^k$, the finite set V_{min} of minimal elements of V is effectively computable iff for any vector $\vec{u} \in \mathbb{N}^k_\omega$ the predicate $\vec{u}{\downarrow} \cap V \neq \varnothing$ is decidable.*

We now show a generalization of this result.

Theorem 10. *Let $(\Omega, \leq)$ be a set with a decidable well-quasi-order (wqo) $\leq$, and let $V \subseteq \Omega$ be upward-closed and recursively enumerable. Then the finite set V_{min} of minimal elements of V is effectively constructible if and only if for every finite subset $X \subseteq \Omega$ it is decidable if $V \cap \overline{X{\uparrow}} \neq \varnothing$ (i.e., if $\exists v \in V. v \notin X{\uparrow}$).*

Proof: V_{min} is finite, since $\leq$ is a wqo. For the only-if part, since $X{\uparrow}$ is upward-closed, it suffices to check for each of the finitely many elements of V_{min} if it is not in $X{\uparrow}$. This is possible, because X is finite and $\leq$ is decidable.

For the if-part, we start with $X = \varnothing$ and keep adding elements to X until $X{\uparrow} = V$. In every step we do the check if $\exists v \in V. v \notin X{\uparrow}$. If no, we stop. If yes, we enumerate V and check for every element v if $v \notin X{\uparrow}$ (this is possible since X is finite and $\leq$ decidable). Eventually, we will find such a v, add it to the set X, and do the next step. Consider the sequence of elements $v_1, v_2, \dots$ which are added to X in this way. By our construction $v_j \not\geq v_i$ for $j > i$. Thus the sequence is finite, because $\leq$ is a wqo. Therefore the algorithm terminates and the final set X satisfies $\nexists v \in V. v \notin X{\uparrow}$, i.e., $V \subseteq X{\uparrow}$. Furthermore, by our construction $X \subseteq V$ and thus $X{\uparrow} \subseteq V{\uparrow} = V$. Thus $X{\uparrow} = V$. Finally, we remove all non-minimal elements from X (this is possible since X is finite and $\leq$ decidable) and obtain V_{min}. ∎

Corollary 11. *Let Σ be a finite alphabet and $V \subseteq \Sigma^*$ a recursively enumerable set that is upward-closed w.r.t. the substring ordering $\leq$. The following three properties are equivalent.*

1) *The finite set V_{min} of minimal elements of V is effectively constructible.*
2) *For every finite subset $X \subseteq \Sigma^*$ it is decidable if $\exists v \in V. v \notin X{\uparrow}$.*
3) *For every regular language $R \subseteq \Sigma^*$ it is decidable if $R \cap V = \varnothing$.*

Proof: By Higman's Lemma [20], the substring order $\leq$ is a wqo on Σ^* and thus V_{min} is finite. Therefore the equivalence of (1) and (2) follows from Theorem 10. Property (1) implies that V is an effectively constructible regular language, which implies property (3). Property (2) is equivalent to checking whether $V \cap \overline{X{\uparrow}} \neq \varnothing$ and $\overline{X{\uparrow}}$ is effectively regular because X is finite. Therefore, (3) implies (2) and thus (1). ∎

Note that Theorem 10 (and even Corollary 11, via an encoding of vectors into strings) imply Theorem 9.

B. The Abstract Phase Construction

We define some sufficient abstract conditions on infinite-state transition systems under which a general reachability/coverability problem is decidable. Intuitively, we have two different types of transition relations. The first relation is monotone (w.r.t. a given quasi-order) on the whole state space, while the second relation is only defined/enabled on an upward-closed subspace. The quasi-order is not a well quasi-order on the entire space, but only on the subspace. In particular, this is not a well-quasi-ordered transition system in the sense of [16], [17], but more general.

We call the following algorithm the *abstract phase construction*, because we divide sequences of transitions into phases, separated by occurrences of transitions of the second kind.

Definition 1. *We say that a structure $(S, C, \leq, \rightarrow, \rightarrow_A, \rightarrow_B, init, F)$ satisfies the* abstract phase construction requirements *iff the following conditions hold.*

1. *S is a (possibly infinite) set of states, $C \subseteq S$ is a finite subset, $init \in S$ is the initial state and $F \subseteq S$ is a (possibly infinite) set of final states.*
2. *$\leq$ is a decidable quasi-order on S. Moreover, $\leq$ is a well-quasi-order on the subset $C{\uparrow}$ (where $C{\uparrow} = \{s \in S \mid \exists c \in C. s \geq c\}$).*
3. *$\rightarrow = \rightarrow_A \cup \rightarrow_B$*
4. *$\rightarrow_A \subseteq S \times S$ is a monotone (w.r.t. $\leq$) transition relation on S.*

5.a. *$\rightarrow_B \subseteq C{\uparrow} \times C{\uparrow}$ is a monotone (w.r.t. $\leq$) transition relation on $C{\uparrow}$.*

5.b *For every finite set $X \subseteq C{\uparrow}$ we have that the finitely many minimal elements of the upward-closed set $Pre_{\rightarrow_B}(X{\uparrow})$ are effectively constructible.*

6.a $Pre^*_{\to_A}(F)$ *is upward-closed and decidable.*

6.b *The finitely many minimal elements of* $Pre^*_{\to_A}(F) \cap C\uparrow$ *are effectively constructible.*

7.a *For any finite set* $U \subseteq C\uparrow$*, the set* $Pre^*_{\to_A}(U\uparrow)$ *is decidable.*

7.b *For any finite sets* $U, X \subseteq C\uparrow$*, it is decidable if* $\overline{X\uparrow} \cap Pre^*_{\to_A}(U\uparrow) \cap C\uparrow \neq \emptyset$*. (In other words, it is decidable if* $\exists z \in (\overline{X\uparrow} \cap C\uparrow).\, z \to_A^* U\uparrow$*.)*

(Note that $Pre^*_{\to_A}(U\uparrow)$ is not necessarily constructible, because $\leq$ is not a well-quasi-order on S. Note also that F is not necessarily upward-closed.)

Theorem 12. *If* $(S, C, \leq, \to, \to_A, \to_B, init, F)$ *satisfies the abstract phase construction requirements of Def. 1, then the problem* $init \to^* F$ *is decidable.*

Proof: By Def. 1 (cond. 3), we have $init \to^* F$ iff (1) $init \to_A^* F$, or (2) $init \to_A^* (\to_B \to_A^*)^+ F$.

Condition (1) can be checked directly, by Def. 1 (cond. 6.a).

In order to check condition (2), we first construct a sequence of minimal finite sets $U_k \subseteq C\uparrow$ for $k = 1, 2, \ldots$ such that $U_k\uparrow = \{s \in S \mid \exists j : 1 \leq j \leq k.\ s(\to_B \to_A^*)^j F\}$ and show that this sequence converges.

First we construct the minimal finite set $U_1' \subseteq C\uparrow$ s.t. $U_1'\uparrow = Pre^*_{\to_A}(F) \cap C\uparrow$. This is possible by conditions 6.a and 6.b of Def. 1. Then we construct the minimal finite set $U_1 \subseteq C\uparrow$ s.t. $U_1\uparrow = Pre_{\to_B}(U_1'\uparrow)$. This is possible by conditions 5.a and 5.b of Def. 1. For $k = 1, 2, \ldots$ we repeat the following steps.

- Given the finite set $U_k \subseteq C\uparrow$, we construct the minimal finite set $U_{k+1}' \subseteq C\uparrow$ s.t. $U_{k+1}'\uparrow = Pre^*_{\to_A}(U_k\uparrow) \cap C\uparrow$. This is possible because of Theorem 10, which we instantiate as follows. Let $\Omega = C\uparrow$ and $V = Pre^*_{\to_A}(U_k\uparrow) \cap C\uparrow$. Using the conditions from Def. 1 we have the following: By condition 2, $\leq$ is a decidable well-quasi-order on $C\uparrow$. By condition 4, $V = Pre^*_{\to_A}(U_k\uparrow) \cap C\uparrow$ is upward-closed, since $\to_A$ is monotone. By conditions 7.a and 2, V is decidable, and by condition 7.b the question $\overline{X\uparrow} \cap V \neq \emptyset$ is decidable. Thus, by Theorem 10, the finitely many minimal elements of V, i.e., the set U_{k+1}', are effectively constructible.
- Given U_{k+1}', we construct the minimal finite set $U_{k+1}'' \subseteq C\uparrow$ s.t. $U_{k+1}''\uparrow = Pre_{\to_B}(U_{k+1}'\uparrow)$. This is possible by conditions 5.a and 5.b of Def. 1.
  Then let U_{k+1} be the finite set of minimal elements of $U_{k+1}'' \cup U_k$.

The sequence $U_1\uparrow, U_2\uparrow, \ldots$ is a monotone-increasing sequence of upward-closed subsets of $C\uparrow$, where U_k is the finite set of minimal elements of $U_k\uparrow$. This sequence converges, because $\leq$ is a well-quasi-order on $C\uparrow$ by condition 2 of Def. 1. Therefore, we get $U_n = U_{n+1}$ for some finite index n and $U_n\uparrow = \{s \in S \mid s(\to_B \to_A^*)^* F\}$, because transition $\to_B$ is only enabled in $C\uparrow$ by Def. 1 (cond. 5.a).

Finally, by Def. 1 (cond. 7.a) we can do the final check whether $init \in Pre^*_{\to_A}(U_n\uparrow)$ and thus decide condition (2). ■

In the following section we use Theorem 12 to solve the optimal cost problem for PTPN. However, it also has many other applications, when used with different instantiations.

Remark 1. *Theorem 12 can be used to obtain a simple proof of decidability of the coverability problem for Petri nets with one inhibitor arc. Normal Petri net transitions are described by* $\longrightarrow_A$*, while the inhibited transition is described by* $\longrightarrow_B$*. (This uses the decidability of the normal Petri net reachability problem [21] to prove conditions 7.a and 7.b).*

A different instantiation could be used to show the decidability of the reachability problem for generalized classes of lossy FIFO-channel systems, where, e.g., an extra type of transition $\longrightarrow_B$ *is only enabled when some particular channel is empty.*

VII. The Main Result

Here we state the main computability result of the paper. Its proof refers to several auxiliary lemmas that will be shown in the following sections.

Theorem 13. *Consider a PTPN* $\mathcal{N} = (Q, P, T, Cost)$ *with initial configuration* $C_{init} = (q_{init}, [])$ *and set of final configurations* $\mathcal{C}_{fin} = \{(q_{fin}, M) \mid M \in (P \times \mathbb{R}_{\geq 0})^{\odot}\}$*. Then* $OptCost(C_{init}, \mathcal{C}_{fin})$ *is computable.*

Proof: $OptCost(C_{init}, \mathcal{C}_{fin}) = \inf\{Cost(\pi) \mid C_{init} \xrightarrow{\pi} \mathcal{C}_{fin}\} = \inf\{Cost(\pi') \mid aptpn(C_{init}) \xrightarrow{\pi'} aptpn(\mathcal{C}_{fin})\}$, by Theorem 7. Thus it suffices to consider the computations $aptpn(C_{init}) \xrightarrow{\pi'} aptpn(\mathcal{C}_{fin})$ of the corresponding A-PTPN. In particular, $OptCost(C_{init}, \mathcal{C}_{fin}) \in \mathbb{N}$.

To compute this value, it suffices to solve the cost-threshold problem for any given threshold $v \in \mathbb{N}$, i.e., to decide if $aptpn(C_{init}) \xrightarrow{\pi} aptpn(\mathcal{C}_{fin})$ for some π with $Cost(\pi) \leq v$.

To show this, we first decide if $aptpn(C_{init}) \xrightarrow{\pi} aptpn(\mathcal{C}_{fin})$ for any π (i.e., reachability). This can be reduced to the cost-threshold problem by setting all place and transition costs to zero and solving the cost-threshold problem for $v = 0$. If no, then no final state is reachable and we represent this by $\inf\{Cost(\pi) \mid C_{init} \xrightarrow{\pi} \mathcal{C}_{fin}\} = \infty$. If yes, then we can find the optimal cost v by solving the cost-threshold problem for threshold $v = 0, 1, 2, 3, \ldots$ until the answer is yes.

Now we show how to solve the cost-threshold problem. By Lemma 8, this question is equivalent to a reachability problem $ac_v(aptpn(C_{init})) \xrightarrow{*} \bigcup_{0 \leq y \leq v} ac_y(aptpn(\mathcal{C}_{fin}))$ in the corresponding AC-PTPN. This reachability problem is decidable by Lemma 16. ■

Before showing the auxiliary lemmas, we give a lower bound on the cost-threshold problem.

Theorem 14. *Consider a PTPN* $\mathcal{N} = (Q, P, T, Cost)$ *with initial configuration* $C_{init} = (q_{init}, [])$ *and set of final states* $\mathcal{C}_{fin} = \{(q_{fin}, M) \mid M \in (P \times \mathbb{R}_{\geq 0})^{\odot}\}$*. Then the question if* $OptCost(C_{init}, \mathcal{C}_{fin}) = 0$ *is at least as hard as the reachability problem for Petri nets with one inhibitor arc.*

Theorem 14 implies that $OptCost(C_{init}, \mathcal{C}_{fin}) = 0$ is at least as hard as the reachability problem for standard Petri nets and thus EXPSPACE-hard [22].

To prove Lemma 16, we need some auxiliary definitions.

Definition 2. *We define the partial order $\leq^f$ on AC-PTPN configurations. Given two AC-PTPN configurations $\beta = (q_\beta, (b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n))$ and $\gamma = (q_\gamma, (c_{-m'} \dots c_{-1}, c_0, c_1 \dots c_{n'}))$ we have $\beta \leq^f \gamma$ iff $q_\beta = q_\gamma$ and there exists a strictly monotone function $f : \{-m, \dots, n\} \mapsto \{-m', \dots, n'\}$ where $f(0) = 0$ s.t.*

1) $c_{f(i)} - b_i \in (P_f \times [cmax + 1])^\odot$, for $-m \leq i \leq n$.

2) $c_j \in (P_f \times [cmax + 1])^\odot$, if $\nexists\, i \in \{-m, \dots, n\}. f(i) = j$.

(Intuitively, γ is obtained from β by adding tokens on free-places, while the tokens on cost-places are unchanged.) In this case, if $\alpha = (q_\beta, (c_{-m'} - b_{f^{-1}(-m')}, \dots, c_{-1} - b_{f^{-1}(-1)}, c_0 - b_0, c_1 - b_{f^{-1}(1)}, \dots, c_{n'} - b_{f^{-1}(n')}))$ then we write $\alpha \oplus \beta = \gamma$. (Note that α is not uniquely defined, because it depends on the choice of the function f. However one such α always exists and only contains tokens on P_f.)

The partial order $\leq^c$ on configurations of AC-PTPN is defined analogously with P_c instead of P_f, i.e., γ is obtained from β by adding tokens on cost-places.

The partial order $\leq^{fc}$ on configurations of AC-PTPN is defined analogously with P instead of P_f, i.e., γ is obtained from β by adding tokens on any places, and $\leq^{fc} = \leq^c \cup \leq^f$.

Lemma 15. *$\leq^f$, $\leq^c$ and $\leq^{fc}$ are decidable quasi-orders on the set of all AC-PTPN configurations.*

For every AC-PTPN configuration c, $\leq^f$, is a well-quasi-order on the set $\{c\}\uparrow = \{s \mid c \leq^f s\}$ (i.e., here $\uparrow$ denotes the upward-closure w.r.t. $\leq^f$).

$\leq^{fc}$ is a well-quasi-order on the set of all AC-PTPN configurations.

Lemma 16. *Given an instance of the PTPN cost problem and a given threshold $v \in \mathbb{N}$, the reachability question $ac_v(aptpn(C_{init})) \xrightarrow{*} \bigcup_{0 \leq y \leq v} ac_y(aptpn(\mathcal{C}_{fin}))$ in the corresponding AC-PTPN is decidable.*

Proof: We instantiate a structure $(S, C, \leq, \rightarrow, \rightarrow_A, \rightarrow_B, init, F)$, show that it satisfies the requirements of Def. 1, and then apply Theorem 12.

Let S be the set of all AC-PTPN configurations of the form $((q, y), b_{-m} \dots b_{-1}, b_0, b_1 \dots b_n)$ where $y \leq v$.

Let C be the set of all AC-PTPN configurations of the form $((q, y), b_{-m'} \dots b_{-1}, b_0, b_1 \dots b_{n'})$ where $y \leq v$, and $b_i \in (P_c \times [cmax + 1])^\odot$ and $\sum_{j=-m'}^{n'} |b_j| \leq v$. In other words, the configurations in C only contain tokens on cost-places and the size of these configurations is limited by v. C is finite, because P_c, $cmax$ and v are finite.

Let $\leq := \leq^f$ of Def. 2, i.e., in this proof $\uparrow$ denotes the upward-closure w.r.t. $\leq^f$. By Lemma 15, $\leq$ is decidable, $\leq$ is a quasi-order on S, and $\leq$ is a well-quasi-order on $\{c\}\uparrow$ for every AC-PTPN configuration c. Therefore $\leq^f$ is a well-quasi-order on $C\uparrow$, because C is finite.

Let $init := ac_v(aptpn(C_{init}))$ and $F := \bigcup_{0 \leq y \leq v} ac_y(aptpn(\mathcal{C}_{fin}))$. In particular, F is upward-closed w.r.t. $\leq^f$ and w.r.t. $\leq^{fc}$. Thus conditions 1 and 2 of Def. 1 are satisfied.

Let $\rightarrow_A$ be the transition relation induced by the discrete AC-PTPN transitions and the abstract timed AC-PTPN transitions of types 1 and 2. These are monotone w.r.t. $\leq^f$. Thus condition 4 of Def. 1 is satisfied.

Let $\rightarrow_B$ be the transition relation induced by abstract timed AC-PTPN transitions of types 3 and 4. These are monotone w.r.t. $\leq^f$, but only enabled in $C\uparrow$, because otherwise the cost would be too high. (Remember that every AC-PTPN configuration stores the remaining allowed cost, which must be non-negative.) Moreover, timed AC-PTPN transitions of types 3 and 4 do not change the number or type of the tokens in a configuration, and thus $\rightarrow_B \subseteq C\uparrow \times C\uparrow$. So we have condition 5.a of Def. 1. Condition 5.b is satisfied, because there are only finitely many token ages $\leq cmax$ and the number and type of tokens is unchanged.

Condition 3 is satisfied, because $\rightarrow = \rightarrow_A \cup \rightarrow_B$ by the definition of AC-PTPN.

Now we show the conditions 6.a and 6.b. F is upward-closed w.r.t. $\leq^{fc}$ and $\rightarrow_A$ is monotone w.r.t. $\leq^{fc}$ (not only w.r.t $\leq^f$). By Lemma 15, $\leq^{fc}$ is a decidable wqo on the set of AC-PTPN configurations. Therefore, $Pre^*_{\rightarrow_A}(F)$ is upward-closed w.r.t. $\leq^{fc}$ and effectively constructible (i.e., its finitely many minimal elements w.r.t. $\leq^{fc}$), because the sequence $Pre^{\leq i}_{\rightarrow_A}(F)$ for $i = 1, 2, \dots$ converges. Let K be this finite set of minimal (w.r.t. $\leq^{fc}$) elements of $Pre^*_{\rightarrow_A}(F)$. We obtain condition 6.a., because K is finite and $\leq^{fc}$ is decidable. Moreover, $Pre^*_{\rightarrow_A}(F)$ is also upward-closed w.r.t. $\leq^f$. The set C is a finite set of AC-PTPN configurations and $C\uparrow$ is the upward-closure of C w.r.t. $\leq^f$. Therefore $Pre^*_{\rightarrow_A}(F) \cap C\uparrow$ is upward closed w.r.t. $\leq^f$. Now we show how to construct the finitely many minimal (w.r.t. $\leq^f$) elements of $Pre^*_{\rightarrow_A}(F) \cap C\uparrow$. For every $k \in K$ let $\alpha(k) := \{k' \mid k' \in C\uparrow, k \leq^c k'\}$, i.e., those configurations which have the right control-state for $C\uparrow$, but whose number of tokens on cost-places is bounded by v, and who are larger (w.r.t. $\leq^c$) than some base element in K. In particular, $\alpha(k)$ is finite and constructible, because v is finite, and $\leq^c$ and $\leq^f$ are decidable. Note that $\alpha(k)$ can be empty (if k has the wrong control-state or too many tokens on cost-places). Let $K' := \bigcup_{k \in K} \alpha(k)$, which is finite and constructible. We show that $Pre^*_{\rightarrow_A}(F) \cap C\uparrow = K'\uparrow$. Consider the first inclusion. If $x \in K'\uparrow$ then $\exists k' \in K', k \in K. k \leq^c k' \leq^f x, k' \in C\uparrow$. Therefore $k \leq^{fc} x$ and $x \in Pre^*_{\rightarrow_A}(F)$. Also $k' \in C\uparrow$ and $k' \leq^f x$ and thus $x \in C\uparrow$. Now we consider the other inclusion. If $x \in Pre^*_{\rightarrow_A}(F) \cap C\uparrow$ then there is a $k \in K$ s.t. $k \leq^{fc} x$. Moreover, the number of tokens on cost-places in x is bounded by v and the control-state is of the form required by $C\uparrow$, because $x \in C\uparrow$. Since, $k \leq^{fc} x$, the same holds for k and thus there is some $k' \in \alpha(k)$ s.t. $k' \leq^f x$. Therefore $x \in K'\uparrow$. To summarize, K' is the finite set of minimal (w.r.t. $\leq^f$) elements of $Pre^*_{\rightarrow_A}(F) \cap C\uparrow$ and thus condition 6.b holds.

Conditions 7.a and 7.b are satisfied by Lemma 20.

Therefore, Theorem 12 yields the decidability of the reachability problem $init \rightarrow^* F$, i.e., $ac_v(aptpn(C_{init})) \xrightarrow{*} \bigcup_{0 \leq y \leq v} ac_y(aptpn(\mathcal{C}_{fin}))$. ∎

Lemma 20 will be shown in Section IX. Its proof uses the simultaneous-disjoint transfer nets of Section VIII.

VIII. Simultaneous-Disjoint-Transfer Nets

Simultaneous-disjoint-transfer nets (SD-TN) [10] are a subclass of transfer nets [23]. SD-TN subsume ordinary Petri nets. A SD-TN N is described by a tuple $(Q, P, T, Trans)$.

- Q is a finite set of control-states
- P is a finite set of places
- T is a finite set of ordinary transitions. Every transition $t \in T$ has the form $t = (q_1, q_2, I, O)$ where $q_1, q_2 \in Q$ and $I, O \in P^{\odot}$.
- $Trans$ describes the set of simultaneous-disjoint transfer transitions. Although these transitions can have different control-states and input/output places, they all share the *same transfer* (thus the 'simultaneous'). The transfer is described by the relation $ST \subseteq P \times P$, which is global for the SD-TN N. Intuitively, for $(p, p') \in ST$, in a transfer every token in p is moved to p'. The transfer transitions in $Trans$ have the form (q_1, q_2, I, O, ST) where $q_1, q_2 \in Q$ are the source and target control-state, $I, O \in P^{\odot}$ are like in a normal Petri net transition, and $ST \subseteq P \times P$ is the same global transfer relation for all these transitions. For every transfer transition (q_1, q_2, I, O, ST) the following 'disjointness' restrictions must be satisfied:
  - Let $(sr, tg), (sr', tg') \in ST$. Then either $(sr, tg) = (sr', tg')$ or $|\{sr, sr', tg, tg'\}| = 4$. Furthermore, $\{sr, tg\} \cap (I \cup O) = \emptyset$.

Let $(q, M) \in Q \times P^{\odot}$ be a configuration of N. The firing of normal transitions $t \in T$ is defined just as for ordinary Petri nets. A transition $t = (q_1, q_2, I, O) \in T$ is enabled at configuration (q, M) iff $q = q_1$ and $M \geq I$. Firing t yields the new configuration (q_2, M') where $M' = M - I + O$.

A transfer transition $(q_1, q_2, I, O, ST) \in Trans$ is enabled at (q, M) iff $q = q_1$ and $M \geq I$. Firing it yields the new configuration (q_2, M') where

$$
\begin{array}{ll}
M'(p) = M(p) - I(p) + O(p) & \text{if } p \in I \cup O \\
M'(p) = 0 & \text{if } \exists p'.\, (p, p') \in ST \\
M'(p) = M(p) + M(p') & \text{if } (p', p) \in ST \\
M'(p) = M(p) & \text{otherwise}
\end{array}
$$

The restrictions above ensure that these cases are disjoint. Note that after firing a transfer transition all source places of transfers are empty, since, by the restrictions defined above, a place that is a source of a transfer can neither be the target of another transfer, nor receive any tokens from the output of this transfer transition.

Theorem 17. *The reachability problem for SD-TN is decidable, and has the same complexity as the reachability problem for Petri nets with one inhibitor arc.*

IX. Encoding AC-PTPN Computations by SD-TN

In this section, we fix an AC-PTPN $\mathcal{N}$, described by the tuple $(Q, P, T, Cost)$ and the cost-threshold v. We use the partial order $\leq := \leq^f$ on AC-PTPN configurations; see Def. 2. We describe an encoding of the configurations of $\mathcal{N}$ as words over some alphabet Σ. We define $\Sigma := (P \times [cmax + 1]) \cup (Q \times \{y \mid 0 \leq y \leq v\}) \cup \{\#, \$\}$, i.e., the members of Σ are elements of $P \times [cmax + 1]$, the control-states of $\mathcal{N}$, and the two "separator" symbols $\#$ and \$. For a multiset $b = [a_1, \ldots, a_n] \in (P \times [cmax + 1])^{\odot}$, we define the encoding $enc(b)$ to be the word $a_1 \cdots a_n \in (P \times [cmax + 1])^*$. For a word $w = b_1 \cdots b_n \in \left((P \times [cmax + 1])^{\odot}\right)^*$, we define $enc(w) := enc(b_n) \# \cdots \# enc(b_1)$, i.e., it consists of the reverse concatenation of the encodings of the individual multisets, separated by $\#$. For a marking $M = (w_1, b, w_2)$, we define $enc(M) := enc(w_2) \$ enc(b) \$ enc(w_1)$. In other words, we concatenate the encoding of the components in reverse order: first w_2 then b and finally w_1, separated by \$. Finally for a configuration $c = ((q, y), M)$, we define $enc(c) := (q, y)\, enc(M)$, i.e., we append the pair (q, y) in front of the encoding of M. We call a finite automaton $\mathcal{A}$ over Σ a *configuration-automaton* if whenever $w \in L(\mathcal{A})$ then $w = enc(c)$ for some AC-PTPN configuration c.

Lemma 18. *Given a finite set C of AC-PTPN configurations, we can construct a configuration-automaton $\mathcal{A}$ s.t. $L(\mathcal{A}) = enc(C\uparrow)$.*

Lemma 19. *We can construct a configuration-automaton $\mathcal{A}$ s.t. $L(\mathcal{A}) = enc(S)$, where S is the set of all configurations of a given AC-PTPN.*

Lemma 20. *Consider an instance of the PTPN cost problem, a given threshold $v \in \mathbb{N}$, and a structure $(S, C, \leq, \rightarrow, \rightarrow_A, \rightarrow_B, init, F)$, instantiated as in Lemma 16.*

Then conditions 7.a and 7.b. of Def. 1 are decidable.

Proof:

7.a Consider a configuration c. We can trivially construct a configuration-automaton $\mathcal{A}$ s.t. $L(\mathcal{A}) = \{enc(c)\}$. Thus the question $c \in Pre^*_{\rightarrow_A}(U\uparrow)$ can be decided by applying Lemma 21 to $\mathcal{A}$ and U.

7.b Consider finite sets of AC-PTPN configurations $U, X \subseteq C\uparrow$. By Lemma 18, we can construct configuration-automata $\mathcal{A}_1, \mathcal{A}_2$ with $L(\mathcal{A}_1) = enc(X\uparrow)$ and $L(\mathcal{A}_2) = enc(C\uparrow)$. Furthermore, by Lemma 19, we can construct a configuration-automaton $\mathcal{A}_3$ with $L(\mathcal{A}_3) = enc(S)$. Therefore, by elementary operations on finite automata, we can construct a configuration-automaton $\mathcal{A}_4$ with $L(\mathcal{A}_4) = \overline{L(\mathcal{A}_1)} \cap L(\mathcal{A}_3) \cap L(\mathcal{A}_2)$, and we obtain that $L(\mathcal{A}_4) = enc\left(\overline{X\uparrow} \cap C\uparrow\right)$. Note that the complement operation on words is not the same as the complement operation on the set of AC-PTPN configurations. Thus the need for intersection with $\mathcal{A}_3$. The question $\exists z \in (\overline{X\uparrow} \cap C\uparrow).\, z \rightarrow^*_A U\uparrow$ of 7.b can be decided by applying Lemma 21 to $\mathcal{A}_4$ and U. ∎

Lemma 21. *Given a configuration-automaton $\mathcal{A}$, C as in Lemma 16, and a finite set $U \subseteq C\uparrow$, it is decidable if there exists some AC-PTPN configuration $c_{init} \in enc^{-1}(L(\mathcal{A}))$ s.t. $c_{init} \rightarrow^*_A U\uparrow$*

Proof: (Sketch) The idea is to translate the AC-PTPN into an SD-TN which simulates its computation. The automaton $\mathcal{A}$ is also encoded into the SD-TN and runs in parallel. $\mathcal{A}$ outputs an encoding of c_{init}, a nondeterministically chosen initial AC-PTPN configuration from $L(\mathcal{A})$. Since the SD-TN

cannot encode sequences, it cannot store the order information in the sequences which are AC-PTPN configurations. Instead this is encoded into the behavior of $\mathcal{A}$, which outputs parts of the configuration c_{init} 'just-in-time' before they are used in the computation (with exceptions; see below). Several abstractions are used to unify groups of tokens with different fractional parts, whenever the PTPN is unable to distinguish them. AC-PTPN timed transitions of types 1 and 2 are encoded as SD-TN transfer transitions, e.g., all tokens with integer age advance to an age with a small fractional part. Since this operation must affect all tokens, it cannot be done by ordinary Petri net transitions, but requires the simultaneous-disjoint transfer of SD-TN. Another complication is that the computation of the AC-PTPN might use tokens (with high fractional part) from c_{init}, which the automaton $\mathcal{A}$ has not yet produced. This is handled by encoding a 'debt' on future outputs of $\mathcal{A}$ in special SD-TN places. These debts can later be 'paid back' by outputs of $\mathcal{A}$ (but not by tokens created during the computation). At the end, the computation must reach an encoding of a configuration in $U\uparrow$ and all debts must be paid. This yields a reduction to a reachability problem for the constructed SD-TN, which is decidable by Theorem 17. ∎

X. Conclusion and Extensions

We have shown that the infimum of the costs to reach a given control-state is computable in priced timed Petri nets with continuous time. This subsumes the corresponding results for less expressive models such as priced timed automata [14] and priced discrete-timed Petri nets [15].

For simplicity of presentation, we have used a one-dimensional cost model, i.e., with a cost $\in \mathbb{R}_{\geq 0}$, but our result on decidability of the Cost-Threshold problem can trivially be generalized to a multidimensional cost model (provided that the cost is linear in the elapsed time). However, in a multidimensional cost model, the Cost-Optimality problem is not defined, because the infimum of the costs does not exist, due to trade-offs between different components. E.g., one can construct a PTPN (and even a priced timed automaton) with a 2-dimensional cost where the feasible costs are $\{(x, 1-x) \mid x \in \mathbb{R}_{\geq 0}, 0 < x \leq 1\}$, i.e., with uncountably many incomparable values.

Another simple generalization is to make token storage costs on places dependent on the current control-state, e.g., storing one token on place p for one time unit costs 2 if in control-state q_1, but 3 if in control-state q_2. Our constructions can trivially be extended to handle this.

Other extensions are much harder. If the token storage costs are not linear in the elapsed time then the infimum of the costs is not necessarily an integer and our abstraction to A-PTPN would not work. It is an open question how to compute optimal costs in such cases.

Finally, some extensions make the cost-problems undecidable. If one considers the reachability problem (instead of our control-state reachability problem) then the question is undecidable for TPN [5], even without considering costs. If one allows negative costs (i.e., rewards) in the model then all cost-problems (even control-state reachability/coverability) become undecidable, even for discrete-time PTPN [15].

References

[1] C. Petri, "Kommunikation mit Automaten," Ph.D. dissertation, University of Bonn, 1962.

[2] J. Peterson, "Petri nets," *Computing Surveys*, vol. 9, no. 3, pp. 221–252, 1977.

[3] F. D. J. Bowden, "Modelling time in Petri nets," in *Proc. Second Australian-Japan Workshop on Stochastic Models*, 1996.

[4] B. Bérard, F. Cassez, S. Haddad, O. Roux, and D. Lime, "Comparison of different semantics for time Petri nets," in *Proceedings of ATVA 2005*, ser. LNCS, vol. 3707. Springer, 2005, pp. 81–94.

[5] V. V. Ruiz, F. C. Gomez, and D. de Frutos Escrig, "On non-decidability of reachability for timed-arc Petri nets," in *Proc. 8th Int. Workshop on Petri Net and Performance Models (PNPM'99), 8-10 October 1999, Zaragoza, Spain*, 1999, pp. 188–196.

[6] D. de Frutos Escrig, V. V. Ruiz, and O. M. Alonso, "Decidability of properties of timed-arc Petri nets," in *ICATPN 2000*, ser. LNCS, vol. 1825, 2000, pp. 187–206.

[7] P. Abdulla and A. Nylén, "Timed Petri nets and BQOs," in *Proc. ICATPN'2001: 22nd Int. Conf. on application and theory of Petri nets*, ser. LNCS, vol. 2075, 2001, pp. 53 –70.

[8] ——, "Undecidability of LTL for timed Petri nets," in *INFINITY 2002, 4th International Workshop on Verification of Infinite-State Systems*, 2002.

[9] P. Abdulla, J. Deneux, P. Mahata, and A. Nylén, "Forward reachability analysis of timed Petri nets," in *Proc. FORMATS-FTRTFT'04*, ser. LNCS, vol. 3253. Springer, 2004, pp. 343–362.

[10] P. Abdulla, P. Mahata, and R. Mayr, "Dense-timed Petri nets: Checking zenoness, token liveness and boundedness," *Logical Methods in Computer Science*, vol. 3, no. 1, 2007.

[11] R. Alur and D. Dill, "A theory of timed automata," *TCS*, vol. 126, pp. 183–235, 1994.

[12] R. Alur, S. L. Torre, and G. J. Pappas, "Optimal paths in weighted timed automata," in *HSCC*, 2001, pp. 49–62.

[13] K. G. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson, and J. Romijn, "As cheap as possible: Efficient cost-optimal reachability for priced timed automata," in *Proc. 13th Int. Conf. on Computer Aided Verification*, ser. LNCS, vol. 2102, 2001.

[14] P. Bouyer, T. Brihaye, V. Bruyère, and J. Raskin, "On the optimal reachability problem of weighted timed automata," *Formal Methods in System Design*, vol. 31, no. 2, pp. 135–175, 2007.

[15] P. Abdulla and R. Mayr, "Minimal cost reachability/coverability in priced timed Petri nets," in *Proc. of FOSSACS 2009*, ser. LNCS, vol. 5504. Springer, 2009.

[16] P. Abdulla, K. Čerāns, B. Jonsson, and Y. Tsay, "Algorithmic analysis of programs with well quasi-ordered domains," *Information and Computation*, vol. 160, pp. 109–127, 2000.

[17] A. Finkel and P. Schnoebelen, "Well-structured transition systems everywhere!" *TCS*, vol. 256, no. 1-2, pp. 63–92, 2001.

[18] R. Valk and M. Jantzen, "The residue of vector sets with applications to decidability problems in Petri nets," *Acta Inf.*, vol. 21, 1985.

[19] K. Reinhardt, "Reachability in Petri nets with inhibitor arcs," in *Proc. RP08, 2nd Workshop on Reachability Problems*, 2008.

[20] G. Higman, "Ordering by divisibility in abstract algebras," *Proc. London Math. Soc. (3)*, vol. 2, no. 7, pp. 326–336, 1952.

[21] E. Mayr, "An algorithm for the general Petri net reachability problem," *SIAM Journal of Computing*, vol. 13, pp. 441–460, 1984.

[22] R. Lipton, "The reachability problem requires exponential space," Department of Computer Science, Yale University, Tech. Rep. 62, January 1976.

[23] G. Ciardo, "Petri nets with marking-dependent arc cardinality: Properties and analysis," in *Proc. 15th Int. Conf. Applications and Theory of Petri Nets*, ser. LNCS, vol. 815. Springer-Verlag, 1994, pp. 179–198.