Parameter Synthesis for Bounded Cost Reachability in Time Petri Nets

Didier Lime, Olivier Henri Roux, Charlotte Seidner

Parameter Synthesis for Bounded Cost Reachability in Time Petri Nets*

Didier Lime[1], Olivier H. Roux[1], and Charlotte Seidner[2]

[1] École Centrale de Nantes, LS2N UMR CNRS 6004, France
[2] Université de Nantes, LS2N UMR CNRS 6004, France

Abstract. We investigate the problem of parameter synthesis for time Petri nets with a cost variable that evolves both continuously with time, and discretely when firing transitions. More precisely, parameters are rational symbolic constants used for time constraints on the firing of transitions and we want to synthesise all their values such that the cost variable stays within a given budget.
We first prove that the mere existence of such values for the parameters is undecidable. We nonetheless provide a symbolic semi-algorithm that is proved both sound and complete when it terminates. We also show how to modify it for the case when parameters values are integers. Finally, we prove that this modified version terminates if parameters are bounded. While this is to be expected since there are now only a finite number of possible parameter values, this is interesting because the computation is symbolic and thus avoids an explicit enumeration of all those values. Furthermore, the result is a symbolic constraint representing a finite union of convex polyhedra that is easily amenable to further analysis through linear programming.
We finally report on the implementation of the approach in Romeo, a software tool for the analysis of hybrid extensions of time Petri nets.

1 Introduction

So-called *priced* or *cost timed* models are suitable for representing real-time systems whose behaviour is constrained by some resource consuming (be it energy or CPU time, for instance) and for which we need to assess the total cost accumulated during their execution. Such models can even describe whether the evolution of the cost during the run is caused by staying in a given state (continuous cost) or by performing a given action (discrete cost). Thus, the task of finding if the model can reach some "good" states while keeping the overall cost under a given bound (or, further, finding the minimum cost) can prove of interest in many real-life applications, such as optimal scheduling or production line planning.

Timed models, however, require a thorough knowledge of the system for their analysis and are thus difficult to build in the early design stages, when the

* This work is partially supported by the ANR national research program PACS (ANR-14-CE28-0002)

system is not fully identified. Even when all timing constraints are known, the whole design process must often be caried out afresh, whenever the environment changes. To obtain such valuable characteristics as flexibility and robustness, the designer may want to relax constraints on some specifications by allowing them a wider range of values. To this end, parametric reasoning is particularly relevant for timed models, since it allows designers to use parameters instead of definite timing values.

We therefore propose to tackle the definition and analysis of models that support both (linear) cost functions and timed parameters.

Related work Parametric timed automata (PTA) [3] extend timed automata [2] to overcome the limits of checking the correctness of the systems with respect to definite timing constraints. The reachability-emptiness problem, which tests whether there exists a parameter valuation such that the automaton has an accepting run, is fundamental to any verification process but is undecidable [3]. L/U automata [13] use each parameter either as a lower bound or as an upper bound on clocks. The reachability-emptiness problem is decidable for this model, but the state-space exploration, which would allow for explicit synthesis of all the suitable parameter valuations, still might not terminate [15]. To obtain decidability results, the approach described in [15] does not rely on syntactical restrictions on guards and invariants, but rather on restricting the parameter values to bounded integers. From a practical point of view, this subclass of PTA is not that restrictive, since the time constraints of timed automata are usually expressed as natural (or perhaps rational) numbers.

In [4], the authors have proved the decidability of the optimal-cost problem for Priced Timed Automata with non-negative costs. In [7,8,16], the computation of the optimal-cost to reach a goal location is based on a forward exploration of zones extended with linear cost functions. In [12], the authors have improved this approach, so as to ensure termination of the forward exploration algorithm, even when clocks are not bounded and costs are negative, provided that the automaton has no negative cost cycles. In [1], the considered model is a timed arc Petri net, under weak firing semantics, extended with rate costs associated with places and firing costs associated with transitions. The computation of the optimal-cost for reaching a goal marking is based on similar techniques to [4]. In [11], the authors have investigated the optimal-cost reachability problem for time Petri nets where each transition has a firing cost and each marking has a rate cost (represented as a linear rate cost function over markings). To compute the optimal-cost to reach a goal marking, the authors have revisited the state class graph method to include costs.

Our contribution We propose in Section 2 an extension of time Petri nets with costs (both discrete and continuous with time) and timing parameters, i.e., rational symbolic constants used in the constraints on the firing times of transitions.

Within this formalism, we define two problems dealing with parametric reachability within a bounded cost. We prove in Section 3 that the existence of a

parameter valuation to reach a given marking under a given bounded cost is undecidable. This proof adapts a 2-counter machine encoding first proposed in [14] for PTA. To our knowledge it is the first time a direct Petri net encoding is provided and the adaptation is not trivial. We give in Section 4 a symbolic semi-algorithm that computes all such parameter valuations when it terminates, and we prove its correctness. We propose in Section 5 a variant of this semi-algorithm that computes integer parameter valuations and prove in Section 6 its termination provided those parameter valuations are bounded and the cost of each run is uniformly lower-bounded for integer parameter valuations. This technique is symbolic and avoids the explicit enumeration of all possible parameter valuations. The basic underlying idea of using the integer hull operator was first investigated in [15] for PTA, but this is the first time that it is adapted and proved to work with state classes for time Petri nets, and the fact that it naturally also preserves costs for integer parameter valuations is new and very interesting. We finally describe in Section 7 the implementation of the approach in the tool Romeo by analysing a small scheduling case-study.

2 Parametric Cost Time Petri Nets

2.1 Preliminaries

We denote the set of natural numbers (including 0) by $\mathbb{N}$, the set of integers by $\mathbb{Z}$, the set of rational numbers by $\mathbb{Q}$ and the set of real numbers by $\mathbb{R}$. We note $\mathbb{Q}_{\geq 0}$ (resp. $\mathbb{R}_{\geq 0}$) the set of non-negative rational (resp. real) numbers. For $n \in \mathbb{N}$, we let $[\![0, n]\!]$ denote the set $\{i \in \mathbb{N} \mid i \leq n\}$. For a finite set X, we denote its size by $|X|$.

Given a set X, we denote by $\mathcal{I}(X)$, the set of non empty real intervals that have their finite end-points in X. For $I \in \mathcal{I}(X)$, $\underline{I}$ denotes its left end-point if I is left-bounded and $-\infty$ otherwise. Similarly, $\overline{I}$ denotes the right end-point if I is right-bounded and ∞ otherwise. We say that an interval I is non-negative if $I \subseteq \mathbb{R}_{\geq 0}$. Moreover, for any $d \in \mathbb{R}_{\geq 0}$ and any non-negative interval I, we let $I \ominus d$ be the interval defined by $\{\theta - d \mid \theta \in I \wedge \theta - d \geq 0\}$. Note that this is again a non-negative interval.

Given sets V and X, a V-valuation (or simply valuation when V is clear from the context) of X is a mapping from X to V. We denote by V^X the set of V-valuations of X. When X is finite, given an arbitrary fixed order on X, we often equivalently consider V-valuations as vectors of $V^{|X|}$. Given a V-valuation v of X and $Y \subseteq X$, we denote by $v_{|Y}$ the projection of v on Y, i.e., the valuation on Y such that $\forall x \in Y, v_{|Y}(x) = v(x)$.

2.2 Time Petri Nets with Costs and Parameters

Definition 1 (Parametric Cost Time Petri Net (pcTPN)). *A* Parametric Cost Time Petri Net *(pcTPN) is a tuple* $\mathcal{N} = (P, T, \mathbb{P}, {}^\bullet., .^\bullet, m_0, I_s, \mathsf{cost}_t, \mathsf{cost}_m)$ *where*

– P *is a finite non-empty set of* places*,*
– T *is a finite set of* transitions *such that* $T \cap P = \emptyset$*,*
– $\mathbb{P}$ *is a finite set of* parameters*,*
– ${}^\bullet. : T \to \mathbb{N}^P$ *is the backward incidence mapping,*
– $.^\bullet : T \to \mathbb{N}^P$ *is the forward incidence mapping,*
– $m_0 \in \mathbb{N}^P$ *is the initial marking,*
– $I_s : T \to \mathcal{I}(\mathbb{N} \cup \mathbb{P})$ *is the (parametric)* static firing interval *function,*
– $\mathit{cost}_t : T \to \mathbb{Z}$ *is the* discrete cost *function, and*
– $\mathit{cost}_m : \mathbb{N}^P \to \mathbb{Z}$ *is the* cost rate *function.*

Given a parameterized object x (be it a pcTPN, a function, an expression, etc.), and a $\mathbb{Q}$-valuation v of parameters, we denote by $v(x)$ the corresponding non-parameterized object, in which each parameter a has been replaced by the value $v(a)$.

A *marking* is an $\mathbb{N}$-valuation of P. For a marking $m \in \mathbb{N}^P$, $m(p)$ represents a number of *tokens* in place p. A transition $t \in T$ is said to be *enabled* by a given marking $m \in \mathbb{N}^P$ if for all places p, $m(p) \geq {}^\bullet t(p)$. We also write $m \geq {}^\bullet t$. We denote by $\mathsf{en}(m)$ the set of transitions that are enabled by the marking m: $\mathsf{en}(m) = \{t \in T \mid m \geq {}^\bullet t\}$.

Firing an enabled transition t from marking m leads to a new marking $m' = m - {}^\bullet t + t^\bullet$. A transition $t' \in T$ is said to be *newly enabled* by the firing of a transition t from a given marking $m \in \mathbb{N}^P$ if it is enabled by the new marking but not by $m - {}^\bullet t$ (or it is itself fired). We denote by $\mathsf{newen}(m, t)$ the set of transitions that are newly enabled by the firing of t from the marking m: $\mathsf{newen}(m, t) = \left\{t' \in \mathsf{en}(m - {}^\bullet t + t^\bullet) \mid t' \notin \mathsf{en}(m - {}^\bullet t) \text{ or } t = t'\right\}$

A *state* of the net $\mathcal{N}$ is a tuple (m, I, c, v) in $\mathbb{N}^P \times \mathcal{I}(\mathbb{R}_{\geq 0})^T \times \mathbb{R} \times \mathbb{Q}_{\geq 0}^{\mathbb{P}}$, where: m is a marking of $\mathcal{N}$, I is called the interval function and associates a *temporal interval* to each transition enabled by m. Value c is the cost associated with that state and valuation v assigns a rational value to each parameter for the state.

Definition 2 (Semantics of a pcTPN). *The semantics of a pcTPN is a timed transition system* $(Q, Q_0, \to)$ *where:*

– $Q \subseteq \mathbb{N}^P \times \mathcal{I}(\mathbb{R}_{\geq 0})^T \times \mathbb{R} \times \mathbb{Q}_{\geq 0}^{\mathbb{P}}$
– $Q_0 = \{(m_0, I_0, 0, v) | v \in \mathbb{Q}_{\geq 0}^{\mathbb{P}}, \forall t \in T, v(I_s(t)) \neq \emptyset\}$ *where* $\forall t \in \mathsf{en}(m_0), I_0(t) = I_s(t)$
– $\to$ *consists of two types of transitions:*
  - *discrete transitions:* $(m, I, c, v) \xrightarrow{t \in T} (m', I', c', v)$ *iff*
    * $m \geq {}^\bullet t$, $m' = m - {}^\bullet t + t^\bullet$ *and* $v(\underline{I(t)}) = 0$,
    * $\forall t' \in \mathsf{en}(m')$
      · $I'(t') = I_s(t')$ *if* $t' \in \mathsf{newen}(m, t)$,
      · $I'(t') = I(t')$ *otherwise*
    * $c' = c + \mathit{cost}_t(t)$
  - *time transitions:* $(m, I, c, v) \xrightarrow{d \in \mathbb{R}_{\geq 0}} (m, I \ominus d, c', v)$, *iff* $\forall t \in \mathsf{en}(m)$, $\overline{(I \ominus d)(t)} \geq 0$ *and* $c' = c + \mathit{cost}_m(m) * d$.

A run of a pcTPN $\mathcal{N}$ is a (finite or infinite) sequence $q_0 a_0 q_1 a_1 q_2 a_2 \cdots$ such that $q_0 \in Q_0$, for all $i > 0$, $q_i \in Q$, $a_i \in T \cup \mathbb{R}_{\geq 0}$ and $q_i \xrightarrow{a_i} q_{i+1}$. The set of runs of $\mathcal{N}$ is denoted by $\mathsf{Runs}(\mathcal{N})$. We note $(m, I, c, v) \xrightarrow{t@d} (m', I', c', v)$ for the sequence of elapsing $d \geq 0$ followed by the firing of the transition t. We denote by $\mathsf{sequence}(\rho)$ the projection of the run ρ over T: for a run $\rho = q_0 \xrightarrow{t_0@d_0} q_1 \xrightarrow{t_1@d_1} q_2 \xrightarrow{t_2@d_2} q_3 \xrightarrow{t_3@d_3} \cdots$, we have $\mathsf{sequence}(\rho) = t_0 t_1 t_2 t_3 \cdots$. We write $q \overset{t}{\hookrightarrow} q'$ if there exists $d \geq 0$ such that $q \xrightarrow{t@d} q'$.

For a finite run ρ we denote by $\mathsf{last}(\rho)$ the last state of ρ and by $\mathsf{lastm}(\rho)$ its marking. A state (m, I, c, v) is said to be *reachable* if there exists a finite run ρ of the net, with $\mathsf{last}(\rho) = (m, I, c, v)$. A marking m is reachable for parameter valuation v, if there exists some I and c such that (m, I, c, v) is reachable.

For $k \in \mathbb{N}$ and parameter valuation v, the (Cost) Time Petri net $v(\mathcal{N})$ is said to be k-bounded if for all reachable markings m, and all places p, $m(p) \leq k$. We say that $v(\mathcal{N})$ is bounded if there exists k such that it is k-bounded.

The *cost* $\mathsf{cost}(\rho)$ of a finite run ρ, with last state (m, I, c, v) is c. Since we are interested in minimising the cost, the *cost* of a sequence of transitions σ is defined as $\mathsf{cost}(\sigma) = \inf_{\rho \in \mathsf{Runs}(\mathcal{N}), \mathsf{sequence}(\rho) = \sigma} \mathsf{cost}(\rho)$. For the sake of the clarity of the presentation, we consider only closed intervals (or right-open to infinity) so this infimum is actually a minimum.

2.3 Parametric Cost Problems

Given a set of target markings Goal, the problems we are interested in are:

1. the existential problem: Given a finite maximum cost value $c_{\max}$, is there a parameter valuation v such that some marking in Goal is reachable with a cost less than $c_{\max}$ in $v(\mathcal{N})$?
2. the synthesis problem: Given a finite maximum cost value $c_{\max}$, compute all the parameter valuations v such that some marking in Goal is reachable with a cost less than $c_{\max}$ in $v(\mathcal{N})$.

We prove in Section 3 that the existential problem is undecidable.

3 Undecidability Results

The existential parametric time bounded reachability problem for bounded parametric time Petri nets asks whether a given target marking is reachable for some valuation of the parameter(s) within $c_{\max}$ time units. This is a special case of the existential cost bounded reachability problem defined in Section 2, with no discrete cost and a uniform cost rate of 1. Proposition 1 therefore implies the undecidability of that more general problem.

Proposition 1. *Existential parametric time bounded reachability is undecidable for bounded parametric time Petri nets.*

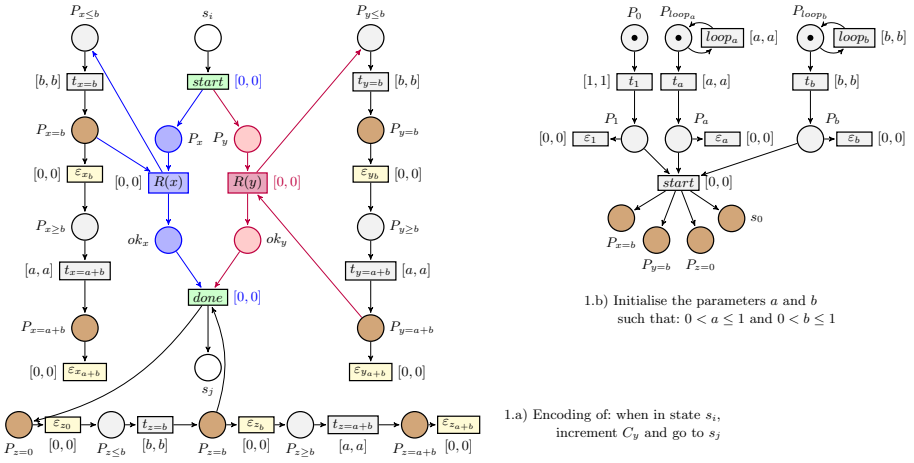

Fig. 1. Increment Gadget (left) and Initial gadget (right)

Proof. Given a bounded parametric time Petri net $\mathcal{N}$, we want to decide whether there exists some parameter valuation v such that some given marking can be reached within $c_{\max}$ time units in $v(\mathcal{N})$. The idea of this proof was first sketched in [14] for parametric timed automata. We encode the halting problem for two-counter machines, which is undecidable [18], into the existential problem for parametric timed Petri nets. Recall that a 2-counter machine $\mathcal{M}$ has two non-negative counters (here C_x and C_y), a finite number of states and a finite number of transitions, which can be of the form: 1) when in state s_i, increment a counter and go to s_j; 2) when in state s_i, decrement a counter and go to s_j; 3) when in state s_i, if a counter is null then go to s_j, otherwise block. The machine starts in state s_0 and halts when it reaches a particular state s_{halt}.

Given such a machine $\mathcal{M}$, we now provide an encoding as a parametric time Petri net $\mathcal{N}_{\mathcal{M}}$: each state s_i of the machine is encoded as place, which we also call s_i. The encoding of the 2-counter machine $\mathcal{M}$ is as follows: it uses two rational-valued parameter a and b, and three gadgets shown in Figure 1.a modelling three clocks x, y, z. Recall that, for a state (m, I, c, v), the enabling time of an enabled transition t is $v(\overline{I_s(t)} - \overline{I(t)})$. For the gadget modelling the clock x, the value of the clock x is equal to: i) the enabling time of the transition $t_{x=b}$ when $P_{x\leq b}$ is marked; ii) b when $P_{x\leq b}$ is marked; iii) the sum of b and the enabling time of the transition $t_{x=a+b}$ when $P_{x\geq b}$ is marked (note that this value is lower than $a+b$); iv) $a+b$ when $P_{x=a+b}$ is marked; v) an unknown (an irrelevant) value in all other cases.

The gadget encoding the increment instruction of C_y is given in Figure 1.a. The clocks x and y store the value of each counter C_x and C_y as follows $x = b - a.C_x$ and $y = b - a.C_y$ when $z = 0$. The zero-test gadget is given in Figure 2. We use the initial gadget in Figure 1.b to initialise a and b such that $0 < a \leq 1$ and $0 < b \leq 1$. The system is studied over 1 time unit.

Increment: We start from some encoding configuration: $x = b - a.C_x$, $y = b - a.C_y$ and $z = 0$ in a marking such that the places $P_{z=0}$ and s_i are marked. After the firing of the transition *start*, there is an interleaving of the transitions $R(x)$ and $R(y)$ that go through the gadget. Finally, we can fire the transition *done* when $z = b$ (i.e. $b - a.C_x$ later) and we have $z = 0$, $x = b - a.C_x$ and $y = b - a(C_y + 1)$ as expected. Moreover, $v(\mathcal{N}_\mathcal{M})$ will block for all the parameter valuations v which not correctly encode the machine.

Decrement: By replacing the arc from $P_{z=b}$ to *done* by an arc from $P_{z=a+b}$ to *done*, the only difference in the previous reasoning is that the elapsing time to fire *done* is increased of a. Then we obtain $z = 0$, $x = b + a - a.C_x = b - a.(C_x - 1)$ and $y = b - a.C_y$ corresponding to the decrement of C_x.

We can obtain symmetrically (by swapping x and y) the increment of C_x and the decrement of C_y

Both the increment gadget and the zero-test gadget require b time units, and the decrement gadget requires $(a + b)$ time units. Since the system executes over 1 time unit, for any value of $a > 0$ and $b > 0$, the number of operations that the machine can perform is finite. We consider two cases:

1. Either the machine halts, both counters C_x and C_y are bounded (let c their maximum value) and the halting and finite execution of the machine is within m steps. If $c = 0$ then the machine is a sequence of m zero-test taking $m.b$ time units and the parametric Petri net $\mathcal{N}_\mathcal{M}$ can go within 1 time unit to a marking m_{halt} if $0 < a \leq 1$ and $0 < b \leq \frac{1}{m}$. If $c > 0$, since an instruction requires at most $a + b$ time units, if $a + b \leq \frac{1}{m}$ and if $0 < a \leq \frac{b}{c}$ then there exists a run that correctly simulates the machine, and eventually reaches m_{halt} within 1 time unit.
   This set of valuations is non-empty: for example if $c = 0$, then we can choose $a = b = \frac{1}{m}$ and if $c > 0$, then, since $m \geq c$, we can choose $a = \frac{b}{m}$ and $b = \frac{1}{1+m}$ hence $a = \frac{1}{m(1+m)}$.
2. Or the machine does not halt. A step requires at least b time units then for any value v of the parameters, after a maximum number of steps (at most $\frac{1}{b}$), one whole time unit will elapse without $v(\mathcal{N}_\mathcal{M})$ reaching m_{halt}. □

4 A Symbolic Semi-algorithm for Parameter Synthesis

4.1 State Classes

We now introduce the notion of state classes for pcTPNs. It was originally introduced for time Petri nets in [10,9], and extended for timing parameters in [21], and for costs in [11]. We show that those two extensions seamlessly blend together.

For an arbitrary sequence of transitions $\sigma = t_1 \dots t_n \in T^*$, let C_σ be the set of all states that can be reached by the sequence σ from any initial state q_0 : $C_\sigma = \{q \in Q | q_0 \overset{t_1}{\hookrightarrow} q_1 \cdots \overset{t_n}{\hookrightarrow} q\}$. All the states of C_σ share the same marking and can therefore be written as a pair (m, D) where m is the common marking

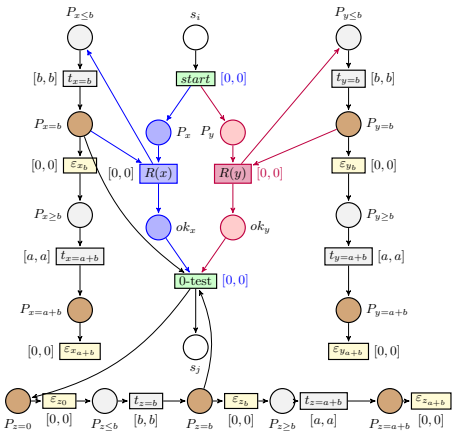

Fig. 2. Encoding 0-test over bounded-time: when in state s_i, if $C_x = 0$ then go to s_j

and, if we note $\mathsf{en}(m) = \{t_1, \ldots, t_n\}$, then D is the set of points $(\theta_1, \ldots, \theta_n, c, v)$ such that $(m, I, c, v) \in C_\sigma$ and for all $t_i \in \mathsf{en}(m)$, $\theta_i \in I(t_i)$. For short, we will often write $(\vec{\theta}, c, v)$ for such a point, with $\vec{\theta} = (\theta_1, \ldots, \theta_n)$ and a small abuse of notation. We denote by Θ the set of θ_i variables, of which we have one per transition of the net: for the sake of simplicity, we will usually use the same index to denote for instance that θ_i corresponds to transition t_i.

C_σ is called a *state class* and D is its *firing domain*.

Lemma 1 equivalently characterises state classes, as a straigthforward reformulation of the definition:

Lemma 1. *For all classes $C_\sigma = (m, D)$, $(\vec{\theta}, c, v) \in D$ if and only if there exists a run ρ in $v(\mathcal{N})$, and $I : \mathsf{en}(m) \to \mathcal{I}(\mathbb{Q}_{\geq 0})$, such that $\mathsf{sequence}(\rho) = \sigma$, $(m, I, c) = \mathsf{last}(\rho)$, and $\vec{\theta} \in I$.*

From Lemma 1, we can then deduce a characterisation of the "next" class, obtained by firing a firable transition from some other class. This is expressed by Lemma 2.

Lemma 2. *Let $C_\sigma = (m, D)$ and $C_{\sigma.t_f} = (m', D')$, we have:*

$$(\vec{\theta'}, c', v) \in D' \text{ iff } \exists (\vec{\theta}, c, v) \in D \text{ s.t. } \begin{cases} \forall t_i \in \mathsf{en}(m), \theta_i - \theta_f \geq 0 \\ \forall t_i \in \mathsf{en}(m - {}^\bullet t_f), \theta'_i = \theta_i - \theta_f \\ \forall t_i \in \mathsf{newen}(m, t_f), \theta'_i \in v(I_s)(t_i) \\ c' = c + \mathsf{cost}_m(m) * \theta_f + \mathsf{cost}_t(t_f) \end{cases}$$

Proof. Consider $(\vec{\theta'}, c', v) \in D'$. Then by Lemma 1, there exists a run ρ' in $v(\mathcal{N})$, and $I' : \mathsf{en}(m) \to \mathcal{I}(\mathbb{Q}_{\geq 0})$, such that $\mathsf{sequence}(\rho') = \sigma.t_f$, $(m', I', c') = \mathsf{last}(\rho')$, and $\vec{\theta'} \in I'$. Consider the prefix ρ of ρ' such that $\mathsf{sequence}(\rho) = \sigma$. The last state of ρ can be written (m, I, c, v) for some I and c. We know that t_f is fired from

(m, I, c, v) so there exists some delay d such that $\underline{I(t_f)} \leq d$ and for all other transitions t_i enabled by m, $\overline{I(t_i)} \geq d$. Furthermore, $c = c' - \mathsf{cost}_m(m) * d - \mathsf{cost}_t(t_f)$. It follows that there exists a point $\vec{\theta} \in I$ with the desired properties.

The other direction is similar. □

Note that according to Lemma 2, D' is not empty if and only if there exists $(\vec{\theta}, c, v)$ in D such that for all $t_i \in \mathsf{en}(m)$, $\theta_i \geq \theta_f$. In that case we say that t_f is *firable* from (m, D) and note $t_f \in \mathsf{firable}((m, D))$.

From Lemma 2, it follows that $C_{\sigma.t_f}$ can be computed from C_σ using Algorithm 1. Note that it is formally the same algorithm as in [11].

Given a class C and a transition t firable from C, we note $\mathsf{Next}(C, t)$ the result of applying Algorithm 1 to C and t.

Algorithm 1 Successor (m', D') of (m, D) by firing t_f

1: $m' \leftarrow m - {}^\bullet t_f + t_f^\bullet$
2: $D' \leftarrow D \wedge \bigwedge_{i \neq f, t_i \in \mathsf{en}(m)} \theta_f \leq \theta_i$
3: for all $t_i \in \mathsf{en}(m - {}^\bullet t_f), i \neq f$, add variable θ'_i to D', constrained by $\theta_i = \theta'_i + \theta_f$
4: add variable c' to D', constrained by $c' = c + \theta_f * \mathsf{cost}_m(m) + \mathsf{cost}_t(t_f)$
5: eliminate (by projection) variables c, θ_i for all i from D'
6: for all $t_j \in \mathsf{newen}(m, t_f)$, add variable θ'_j to D', constrained by $\theta'_j \in I_s(t_j)$

Let $C_0 = (m_0, D_0)$ be the initial class. Domain D_0 is defined by the constraints $\forall t_i \in \mathsf{en}(m_0), \theta_i \in I_s(t_i)$, $\forall t \in T, I_s(t) \neq \emptyset$, and $c = 0$. This gives a convex polyhedron of $\mathbb{R}_{\geq 0}^{|\mathsf{en}(m_0)|+|\mathbb{P}|+1}$; since all the operations on domains in Algorithm 1 are polyhedral, all the domains of state classes are also convex polyhedra. Note that only enabled transitions are constrained in the domain of a state class.

Naturally, we define the *cost* of state class C_σ as $\mathsf{cost}(C_\sigma) = \mathsf{cost}(\sigma)$.

4.2 The Synthesis Semi-algorithm

In Algorithm 2, we explore the symbolic state-space in a classic manner. Whenever a goal marking is encountered we collect the parameter valuations that allowed that marking to be reached with a cost less or equal to $c_{\max}$.

The PASSED list records the visited symbolic states. Instead of checking new symbolic states for membership, we test a weaker relation denoted by $\preccurlyeq$: does there exist a visited state allowing more behaviors with a cheaper cost?

For any state class $C = (m, D)$ and any point $(\vec{\theta}, v) \in D_{|\Theta \cup \mathbb{P}}$, the optimal cost of $(\vec{\theta}, v)$ in D is defined by $\mathsf{cost}_D(\vec{\theta}, v) = \inf_{(\vec{\theta}, c, v) \in D} c$.

Definition 3. *Let $C = (m, D)$ and $C' = (m', D')$ be two parametric cost state classes. We say that C is subsumed by C', which we denote by $C \preccurlyeq C'$ iff $m = m'$, $D_{|\Theta \cup \mathbb{P}} \subseteq D'_{|\Theta \cup \mathbb{P}}$, and for all $(\vec{\theta}, v) \in D_{|\Theta \cup \mathbb{P}}$, $\mathsf{cost}_{D'}(\vec{\theta}, v) \leq \mathsf{cost}_D(\vec{\theta}, v)$.*

Algorithm 2 Symbolic semi-algorithm computing all parameter valuations such that some markings are reachable with a bounded cost.

1: $\mathsf{PolyRes} \leftarrow \emptyset$
2: $\textsc{Passed} \leftarrow \emptyset$
3: $\textsc{Waiting} \leftarrow \{(m_0, D_0)\}$
4: **while** $\textsc{Waiting} \neq \emptyset$ **do**
5: select $C_\sigma = (m, D)$ from $\textsc{Waiting}$
6: **if** $m \in \mathsf{Goal}$ **then**
7: $\mathsf{PolyRes} \leftarrow \mathsf{PolyRes} \cup \big(D \cap (c \leq c_{\max})\big)_{\downarrow \mathbb{P}}$
8: **end if**
9: **if** for all $C' \in \textsc{Passed}, C_\sigma \not\preccurlyeq C'$ **then**
10: add C_σ to $\textsc{Passed}$
11: for all $t \in \mathsf{firable}(C_\sigma)$, add $C_{\sigma.t}$ to $\textsc{Waiting}$
12: **end if**
13: **end while**
14: **return** $\mathsf{PolyRes}$

The following result is a fairly direct consequence of Definition 3:

Lemma 3. *Let C_{σ_1} and C_{σ_2} be two state classes such that $C_{\sigma_1} \preccurlyeq C_{\sigma_2}$.*

If a transition sequence σ is firable from C_{σ_1}, it is also firable from C_{σ_2} and $\mathsf{cost}(C_{\sigma_1.\sigma}) \geq \mathsf{cost}(C_{\sigma_2.\sigma})$.

Proof. Let $C_{\sigma_1} = (m_1, D_1)$ and $C_{\sigma_2} = (m_2, D_2)$. From Definition 3, for any point $(\vec{\theta}, c_1, v) \in D_1$, there exists a point $(\vec{\theta}, c_2, v) \in D_2$ such that $c_2 \leq c_1$. This implies that: (i) $\mathsf{cost}(C_{\sigma_1}) \geq \mathsf{cost}(C_{\sigma_2})$; (ii) if transition t is firable from C_{σ_1}, then it is firable from C_{σ_2} and $\mathsf{Next}(C_{\sigma_1}, t) \preccurlyeq \mathsf{Next}(C_{\sigma_2}, t)$. And the result follows by a straightforward induction. □

While $\preccurlyeq$ can be checked using standard linear algebra techniques, we can also reduce it to standard inclusion on polyhedra by removing the upper bounds on cost (an operation called cost relaxation) [11].

Lemma 4. *The following invariant holds after each iteration of the while loop in Algorithm 2: for all $C_\sigma = (m, D) \in \textsc{Passed}$,*

1. *for all prefixes σ' of σ, $C_{\sigma'} \in \textsc{Passed}$;*
2. *if $m \in \mathsf{Goal}$ then $\big(D \cap (c \leq c_{\max})\big)_{\downarrow \mathbb{P}} \subseteq \mathsf{PolyRes}$;*
3. *if t is firable from C_σ*
   - *either $C_{\sigma.t} \in \textsc{Waiting}$,*
   - *or there exists $C' \in \textsc{Passed}$ such that $C_{\sigma.t} \preccurlyeq C'$.*

Proof. We prove this lemma by induction. Before the while loop starts, $\textsc{Passed}$ is empty so the invariant is true. Let us now assume that the invariant holds for all iterations up to the n-th one, with $n \geq 0$, and that $\textsc{Waiting} \neq \emptyset$. Let $C_\sigma \in \textsc{Waiting}$ be the selected class at line 5; to check whether the invariant still holds at the end of the $(n+1)$-th iteration, we only have to test the case where C_σ is added to $\textsc{Passed}$ (which means that the condition at line 9 is true). We can then check each part of the invariant:

1. C_σ was picked from WAITING (line 5); except for the initial class (for which σ is empty, and therefore has no prefix), it means that, in a previous iteration, there was a sequence σ' and a transition $t \in \mathsf{firable}(C_{\sigma'})$ such that $\sigma = \sigma'.t$ (line 11) and $C_{\sigma'} \in$ PASSED (line 10). Since we add at most one state class to PASSED at each iteration, $C_{\sigma'}$ was added in a previous iteration and we can apply to it the induction hypothesis, which allows us to prove the first part of the invariant;
2. lines 6 and 7 obviously imply the second part of the invariant;
3. if $C_\sigma \in$ PASSED, then the condition of the if on line 9 is true and then for any transition t that is firable from C_σ, $C_{\sigma.t}$ is added to WAITING (line 11) so the third part of the invariant holds for C_σ. Nevertheless, C_σ itself is no longer in WAITING, and it is (except for the initial state class) the successor of some state class in PASSED. But then we have only two possibilities: either C_σ has been added to PASSED in line 10 if the condition on line 9 was true, and certainly $C_\sigma \preccurlyeq C_\sigma$, or there exists $C' \in$ PASSED such that $C_\sigma \preccurlyeq C'$ if that condition was false. Therefore the third part of the invariant holds.

Both the basis case and the induction step are true: the result follows by induction. □

Proposition 2. *After any iteration of the while loop in Algorithm 2:*

1. *if $v \in$ PolyRes, then there exists a run ρ in $v(\mathcal{N})$ such that $\mathsf{cost}(\rho) \leq c_{\max}$ and $\mathsf{lastm}(\rho) \in$ Goal.*
2. *if WAITING $= \emptyset$ then, for all parameter valuations v such that there exists a run ρ in $v(\mathcal{N})$ such that $\mathsf{cost}(\rho) \leq c_{\max}$ and $\mathsf{lastm}(\rho) \in$ Goal, we have $v \in$ PolyRes.*

Proof. 1. By induction on the while loop: initially, PolyRes is empty so the result holds trivially. Suppose it holds after some iteration n, and consider iteration $n+1$. Let $v \in$ PolyRes after iteration $n+1$. If v was already in PolyRes after iteration n then we can apply the induction hypothesis. Otherwise it means that if $C_\sigma = (m, D)$ is the class examined at iteration $n+1$, then $m \in$ Goal and $v \in \big(D \cap (c \leq c_{\max})\big)_{|\mathbb{P}}$. This means that there exists some point $(\vec{\theta}, c, v) \in D$ with $c \leq c_{\max}$. By Lemma 1, this means that there exists a run ρ such that $(m, I, c, v) = \mathsf{last}(\rho)$, for some I such that $\vec{\theta} \in I$, and therefore $\mathsf{lastm}(\rho) \in$ Goal and $\mathsf{cost}(\rho) \leq c_{\max}$.

2. Let v be a parameter valuation such that there exists a run ρ in $v(\mathcal{N})$ such that $\mathsf{cost}(\rho) \leq c_{\max}$ and $\mathsf{lastm}(\rho) \in$ Goal. Let $\sigma = \mathsf{sequence}(\rho)$. We proceed by induction on the length n of the biggest suffix σ_2 of σ such that, either σ_2 is empty or, if we note $\sigma = \sigma_1\sigma_2$, with the first element of σ_2 being transition t, then $C_{\sigma_1 t} \notin$ PASSED.
   If $n = 0$, then $C_\sigma = (m, D) \in$ PASSED. By Lemma 1, $v \in D_{|\mathbb{P}}$ and $m \in$ Goal. From the latter, we have $\big(D \cap (c \leq c_{\max})\big)_{|\mathbb{P}} \subseteq$ PolyRes and therefore $v \in$ PolyRes because $v \in \big(D \cap (c \leq c_{\max})\big)_{|\mathbb{P}}$.

Consider now $n > 0$ and assume the property holds for $n-1$. Since $n > 0$, then there exists a transition t and a sequence σ_3 such that $\sigma_2 = t.\sigma_3$. By definition of σ_2, we have $C_{\sigma_1} \in \text{PASSED}$ but $C_{\sigma_1.t} \notin \text{PASSED}$. By Lemma 4, since $\text{WAITING} = \emptyset$, there must exists some class $C_{\sigma'}$ such that $C_{\sigma_1.t} \preccurlyeq C_{\sigma'}$. From Lemma 3, sequence σ_3 is also firable from $C_{\sigma'}$ and $C_{\sigma'.\sigma_3} = (m, D')$, with $\mathsf{cost}(C_{\sigma'.\sigma_3}) \leq \mathsf{cost}(C_\sigma) \leq c_{\max}$. By Lemma 1, there exists thus a run ρ' in $v(\mathcal{N})$, with $\mathsf{sequence}(\rho') = \sigma'.\sigma_3$, $\mathsf{lastm}(\rho') \in \mathsf{Goal}$ and $\mathsf{cost}(\rho') \leq c_{\max}$. Also, from Lemma 4 (item 1), we know that for all prefixes of σ', the corresponding state class is in PASSED, so the biggest suffix of $\sigma'.\sigma_3$ as defined above in the induction hypothesis has length less or equal to $n-1$, and the induction hypothesis applies to ρ', which allows to conclude. □

In particular, if the algorithm terminates, then the waiting list is empty and PolyRes is exactly the solution to the synthesis problem.

5 Restricting to Integer Parameters

Obviously, in general, (semi-)Algorithm 2 will not terminate, since the emptiness problem for the set it computes is undecidable.

To ensure termination, we can however follow the methodology of [15]: we require that parameters are bounded integers and, instead of just enumerating the possible parameter values, we propose a modification of the symbolic state computation to compute these integer parameters symbolically. For this we rely on the notion of integer hull.

We call *integer valuation* a $\mathbb{Z}$-valuation. Note that a $\mathbb{Z}$-valuation is also an $\mathbb{R}$-valuation, and given a set D of $\mathbb{R}$-valuations, we denote by $\mathsf{Ints}(D)$ the set of integer valuations in D.

The *convex hull* of a set D of valuations, denoted by $\mathsf{Conv}(D)$, is the intersection of all the convex sets of valuations that contain D.

The *integer hull* of a set D of valuations, denoted by $\mathsf{IH}(D)$, is defined as the convex hull of the integer valuations in D: $\mathsf{IH}(D) = \mathsf{Conv}(\mathsf{Ints}(D))$.

For a state class $C = (m, D)$, we write $\mathsf{IH}(C)$ for $(m, \mathsf{IH}(D))$.

Before we see how our result can be adapted for the restriction to integer parameter valuations, and from there how we can enforce termination of the symbolic computations when parameters are assumed to be bounded, we need some results on the structure of the polyhedra representing firing domains of cost TPNs.

By the Minkowski-Weyl Theorem (see e.g. [20]), every convex polyhedron can be either described as a set of linear inequalities, as seen above, or by a set of *generators*. More precisely, for the latter: if d is the dimension of polyhedron P, there exists $v_1, \ldots, v_p, r_1, \ldots, r_s \in \mathbb{R}^d$, such that for all points $x \in P$, there exists $\lambda_1, \ldots, \lambda_p \in \mathbb{R}, \mu_1, \ldots, \mu_s \in \mathbb{R}_{\geq 0}$ such that $\sum_i \lambda_i = 1$ and $x = \sum_i \lambda_i v_i + \sum_i \mu_i r_i$. The v_i's are called the *vertices* of P and the r_i's are the *extremal rays* of P. The latter correspond to the directions in which the polyhedron is infinite. In our case, they correspond to transitions with a (right-)infinite static interval, and possibly the cost.

A classic property of vertices, which can also be used as a definition, is as follows: $\vec{v}$ is a vertex of P iff for all non-null vectors $\vec{x} \in \mathbb{R}^d$, either $\vec{v} + \vec{x} \notin P$ or $\vec{v} - \vec{x} \notin P$ (or both), $+$ and $-$ being understood component-wise.

Proposition 3. *Let $\mathcal{N}$ be a (non-parametric) cost TPN and let $C = (m, D)$ be one of its state classes, then D has integer vertices.*

Proof. We have proved in [11] that the domain D of a state class of a cost TPNs, with removed upper bounds on cost (so-called relaxed classes), can be partitioned into a union of simpler polyhedra $\bigcup_{i=1}^{n} D_i$ that have the following key properties: (1) by projecting the cost out we obtain a convex polyhedron $D_{i|\Theta}$ with integer vertices (actually a *zone*, as in [17,9]), and (2) these simpler polyhedra all have exactly one constraint on the cost variable, i.e., of the form $c \geq \ell(\vec{\theta})$, with integer coefficients. Note that the same result can be obtained, with the same technique, if we consider non-relaxed state classes, except that, we also have an upper bound on cost that is always greater or equal to the lower bound. We prove in Lemma 5 that each of these simpler polyhedra also has integer vertices. Since D and each of the D_i's are convex and since $D = \bigcup_i D_i$, D is equal to the convex hull of the vertices of the D_i's and therefore D also has integer vertices.

Lemma 5. *Let D be a convex polyhedron on variables $\theta_1, \ldots, \theta_n, c$ such that the projection of P on the θ variables has integer vertices, and there are two constraints on c of the form $c \geq \ell(\theta_1, \ldots, \theta_n)$ and $c \leq \ell'(\theta_1, \ldots, \theta_n)$, with ℓ and ℓ' linear terms with integer coefficients, such that $\ell(\theta_1, \ldots, \theta_n) \leq \ell'(\theta_1, \ldots, \theta_n)$, for all values of the θ_i's.*

Then, the vertices of D are the points $(\theta_1, \ldots, \theta_n, \ell(\theta_1, \ldots, \theta_n))$ and $(\theta_1, \ldots, \theta_n, \ell'(\theta_1, \ldots, \theta_n))$ such that $(\theta_1, \ldots, \theta_n)$ is a vertex of $D_{|\Theta}$, and they are integer points.

Proof. Recall here that we consider all constraints in D to be non-strict so all polyhedra are topologically closed. The reasoning extends with no difficulty to non-necessarily-closed polyhedra by considering so-called *closure points* in addition to vertices [6].

Consider a non-vertex point $\vec{\theta}$ in $D_{|\Theta}$ and let $(\vec{\theta}, c)$ be a point of D. Then using the form of the unique cost constraint, we have $c \geq \ell(\vec{\theta})$. Now since $\vec{\theta}$ is not a vertex, there exists a vector $\vec{x}$ such that both $\vec{\theta} + \vec{x}$ and $\vec{\theta} - \vec{x}$ belong to $D_{|\Theta}$. Then, for sure, $(\vec{\theta} + \vec{x}, \ell(\vec{\theta} + \vec{x})) \in D$ and $(\vec{\theta} - \vec{x}, \ell(\vec{\theta} - \vec{x})) \in D$. And since ℓ is linear, $(\vec{\theta} + \vec{x}, \ell(\vec{\theta}) + \ell(\vec{x}))) \in D$, i.e., $(\vec{\theta}, \ell(\vec{\theta})) + (\vec{x}, \ell(\vec{x})) \in D$. And similarly, $(\vec{\theta}, \ell(\vec{\theta})) - (\vec{x}, \ell(\vec{x})) \in D$. Using again the form of the unique cost constraint, and the fact that $c \geq \ell(\vec{\theta})$, we finally have $(\vec{\theta}, c) + (\vec{x}, \ell(\vec{x})) \in D$ and $(\vec{\theta}, c) - (\vec{x}, \ell(\vec{x})) \in D$, that is, $(\vec{\theta}, c)$ is not a vertex of D.

By contraposition, any vertex of D extends a vertex of $D_{|\Theta}$, and using a last time the form of the cost constraint, any vertex of D, is of the form $(\vec{\theta}, \ell(\vec{\theta}))$, with $\vec{\theta}$ a vertex of $D_{|\Theta}$: suppose $(\vec{\theta}, c)$ is a vertex of D, with $c > \ell(\vec{\theta})$, then for

$\vec{x}$ defined with $c - \ell(\vec{\theta})$ on the cost variable, and 0 on all other dimensions, we clearly have both $(\vec{\theta}, c) + \vec{x}$ and $(\vec{\theta}, c) - \vec{x}$ in D, which is a contradiction.

We conclude by remarking that, since $D_{|\Theta}$ has integer vertices, all the coordinates of $\vec{\theta}$ are integers, and since ℓ has integer coefficients then $\ell(\vec{\theta})$ is an integer.

We can deal with the upper bound defined by ℓ' in exactly the same way. □

From Proposition 3, we can prove the following lemma that will be very useful in the subsequent proofs.

Lemma 6. *Let (m, D) be a state class of a pcTPN and let $(\vec{\theta}, c, v)$ be a point in D.*

If v is an integer valuation, then $(\vec{\theta}, c, v) \in \mathsf{IH}(D)$.

Proof. Since $(\vec{\theta}, c, v) \in D$ then $(\vec{\theta}, c) \in v(D)$. By Proposition 3, $v(D)$ being the firing domain of a state class in a (non-parametric) cost TPN, it has integer vertices, and therefore $v(D) = \mathsf{IH}(v(D))$. Point $(\vec{\theta}, c)$ is therefore a convex combination of integer points in $v(D)$. Clearly, for all integer points $(\vec{\theta'}, c')$ in $v(D)$, we have that $(\vec{\theta'}, c', v)$ is an integer point of D. Since D is convex, this implies that $(\vec{\theta}, c, v) \in \mathsf{IH}(D)$. □

When we restrict ourselves to integer parameter but continue to work symbolically, we need to adjust the definitions of the firability of a transition from a class and of the cost of a class.

First, a transition t_f is firable for integer parameter valuations from a class (m, D), call this $\mathbb{N}^{\mathbb{P}}$-firable, if there exists an *integer* parameter valuation v and a point $(\vec{\theta}, c, v)$ in D such that for all transitions $t_i \in \mathsf{en}(m), \theta_i \geq \theta_f$.

Lemma 7. *Let $C = (m, D)$ be a state class. Transition $t_f \in \mathsf{en}(m)$ is $\mathbb{N}^{\mathbb{P}}$-firable from C if and only if it is firable (not necessarily $\mathbb{N}^{\mathbb{P}}$-firable) from $(m, \mathsf{IH}(D))$.*

Proof. $\Leftarrow$: trivial because $\mathsf{IH}(D) \subseteq D$.

$\Rightarrow$: since t_f is $\mathbb{N}^{\mathbb{P}}$-firable from C, there exists an integer parameter valuation v, and $(\vec{\theta}, c, v) \in D$ such that for all transitions $t_i \in \mathsf{en}(m), \theta_i \geq \theta_f$. And the result follows from Lemma 6 because v is an integer valuation. □

Second, the cost of a class $C = (m, D)$, for integer parameters, is $\mathsf{cost}_{\mathbb{N}}(C) = \inf_{(\vec{\theta}, c, v) \in D, v \in \mathbb{N}^{\mathbb{P}}} c$.

Lemma 8 is a direct consequence of Lemma 6:

Lemma 8. *Let (m, D) be a state class. We have: $\mathsf{cost}_{\mathbb{N}}((m, D)) = \mathsf{cost}((m, \mathsf{IH}(D))$.*

Lemma 9. *If v is an integer parameter valuation, then for all classes $C_\sigma = (m, D)$, $(\vec{\theta}, c, v) \in \mathsf{IH}(D)$ if and only if there exists a run ρ in $v(\mathcal{N})$, and $I : \mathsf{en}(m) \to \mathcal{I}(\mathbb{Q}_{\geq 0})$, such that $\mathsf{sequence}(\rho) = \sigma$, $(m, I, c) = \mathsf{last}(\rho)$, and $\vec{\theta} \in I$.*

Proof. $\Rightarrow$: if $(\vec{\theta}, c, v) \in \mathsf{IH}(D)$ then it is also in D and the result follows from Lemma 1.

$\Leftarrow$: by Lemma 1, we know that there exists some $(\vec{\theta}, c, v) \in D$, and since v is an integer valuation, by Lemma 6, $(\vec{\theta}, c, v) \in \mathsf{IH}(D)$. □

Lemma 10. *Let C_{σ_1} and C_{σ_2} be two state classes such that $\mathsf{IH}(C_{\sigma_1}) \preccurlyeq \mathsf{IH}(C_{\sigma_2})$.*

If a transition sequence σ is $\mathbb{N}^{\mathbb{P}}$-firable from C_{σ_1} it is also $\mathbb{N}^{\mathbb{P}}$-firable from C_{σ_2} and $\mathsf{cost}_{\mathbb{N}}(C_{\sigma_1.\sigma}) \geq \mathsf{cost}_{\mathbb{N}}(C_{\sigma_2.\sigma})$.

Proof. Let $C_{\sigma_1} = (m_1, D_1)$ and $C_{\sigma_2} = (m_2, D_2)$. From Definition 3, for any point $(\vec{\theta}, c_1, v) \in \mathsf{IH}(D_1)$, there exists a point $(\vec{\theta}, c_2, v) \in \mathsf{IH}(D_2)$ such that $c_2 \leq c_1$. With Lemma 7 and Lemma 8, this implies that: (i) $\mathsf{cost}_{\mathbb{N}}(C_{\sigma_1}) \geq \mathsf{cost}_{\mathbb{N}}(C_{\sigma_2})$; (ii) if transition t is $\mathbb{N}^{\mathbb{P}}$-firable from C_{σ_1}, then it is $\mathbb{N}^{\mathbb{P}}$-firable from C_{σ_2} and $\mathsf{Next}(C_{\sigma_1}, t) \preccurlyeq \mathsf{Next}(C_{\sigma_2}, t)$. And, as before, the result follows by a straightforward induction. □

Algorithm 3 Restriction of (semi-)Algorithm 2 to integer parameter valuations.

```
1: PolyRes ← ∅
2: PASSED ← ∅
3: WAITING ← {(m_0, D_0)}
4: while WAITING ≠ ∅ do
5:    select C_σ = (m, D) from WAITING
6:    if m ∈ Goal then
7:       PolyRes ← PolyRes ∪ (IH(D) ∩ (c ≤ c_max))_|ℙ
8:    end if
9:    if for all C' ∈ PASSED, IH(C_σ) ⋠ IH(C') then
10:      add C_σ to PASSED
11:      for all t ∈ firable(IH(C_σ)), add C_σ.t to WAITING
12:   end if
13: end while
14: return PolyRes
```

Using Lemma 9 instead of Lemma 1, and Lemma 10 instead of Lemma 3 in the proof of Proposition 2, we get the following proposition, stating the completeness and soundness of Algorithm 3.

Proposition 4. *After any iteration of the while loop in Algorithm 3:*

1. *if $v \in \mathit{PolyRes}$ and v is an integer parameter valuation then there exists a run ρ in $v(\mathcal{N})$ such that $\mathsf{cost}(\rho) \leq c_{\max}$ and $\mathsf{lastm}(\rho) \in \mathit{Goal}$.*
2. *if $\text{WAITING} = \emptyset$ then for all integer parameter valuations v such that there exists a run ρ in $v(\mathcal{N})$ such that $\mathsf{cost}(\rho) \leq c_{\max}$ and $\mathsf{lastm}(\rho) \in \mathit{Goal}$, we have $v \in \mathit{PolyRes}$.*

In Algorithm 3, we compute state classes as usual then handle them via their integer hulls. We can actually simply integrate integer hulls at the end of Algorithm 1 and use Algorithm 2 with this updated successor computation as proved by Lemma 11.

Lemma 11. *Let (m, D) be a state class of a pcTPN $\mathcal{N}$, and t a transition firable from C. Let $(m', D') = \mathsf{Next}((m, D), t)$ and $(m'', D'') = \mathsf{Next}((m, \mathsf{IH}(D)), t)$. Then $m'' = m'$ and $\mathsf{IH}(D'') = \mathsf{IH}(D')$.*

Proof. The equality of markings is trivial so we focus on firing domains.

By definition of the integer hull, we have $\mathsf{IH}(D) \subseteq D$. Since the computation of the next class domain is non-decreasing with respect to inclusion, we then have $D'' \subseteq D'$. Taking the integer hull is also non-decreasing wrt. inclusion, so $\mathsf{IH}(D'') \subseteq \mathsf{IH}(D')$.

Consider now an integer point $(\vec{\theta'}, c', v)$ in D'. Then $(\vec{\theta'}, c') \in v(D')$. Consider state class computations in the (non-parametric) cost TPN $v(\mathcal{N})$: there exists some point $(\vec{\theta}, c)$ in $v(D)$ such that $(m', \vec{\theta'}, c') \in \mathsf{Next}((m, \{(\vec{\theta}, c)\}), t)$. Since $(\vec{\theta}, c, v)$ thus belongs to D and since v is an integer parameter valuation, by Lemma 6, we have that $(\vec{\theta}, c, v) \in \mathsf{IH}(D)$. Thus $(\vec{\theta'}, c', v) \in D''$ and since it is an integer point, it is in $\mathsf{IH}(D'')$. □

6 Termination of Algorithm 3

We now consider that parameter valuations are bounded by some value $M_1 \in \mathbb{N}$ (and that they still have integer values). We also assume that, for all integer parameter valuations, there exists $M_2 \in \mathbb{Z}$ such that for all runs ρ in $v(\mathcal{N})$, $\mathsf{cost}(\rho) \geq M_2$: this allows us, as in [11,12], to keep Algorithm 3 simple by doing away with negative cost loop-checking. Finally, we assume the net itself is bounded: there exists $M_3 \in \mathbb{N}$ such that for all reachable markings m, for all places p, $m(p) \leq M_3$.

To prove the termination of Algorithm 3 under these assumptions, we consider $\succcurlyeq$ the symmetric relation to $\preccurlyeq$, such that $x \succcurlyeq y$ iff $y \preccurlyeq x$. We prove that it is a well quasi-order (wqo), i.e., that for every infinite sequence of state classes, there exist C and C' in the sequence, with C strictly preceding C' such that $C \succcurlyeq C'$. This implies that the exploration of children in Algorithm 3 will always eventually stop.

Proposition 5. *Let $\mathcal{N}$ be a bounded pcTPN, with bounded integer parameters and such that the cost of all runs is uniformly lower-bounded for all integer parameter valuations.*

Relation $\succcurlyeq$ is well-quasiorder on the set of state classes of $\mathcal{N}$.

Proof. Consider an infinite sequence $C_0, C_1, C_2, \ldots$ of state classes. Let $C_i = (m_i, D_i)$.

From [11], we know that $\succcurlyeq$ is a wqo for the state classes of bounded (non parametric) cost TPNs. So for each integer parameter valuation v, and using a

classic property of wqo we can extract a subsequence of $v(C_0), v(C_1), \ldots$ that is completely ordered by $\succcurlyeq$. And since, we have a finite number of such parameter valuations, we can extract an infinite subsequence $C_{i_0}, C_{i_1}, \ldots$ such that for all integer parameter valuations v, $v(C_{i_0}) \succcurlyeq v(C_{i_1}) \succcurlyeq \cdots$.

Let us consider two of those: C_{i_r} and C_{i_s}, with $r < s$.

Since $\mathsf{IH}(D_{i_s})$ has integer vertices, and for any integer parameter valuation, $v(C_{i_r}) \succcurlyeq v(C_{i_s})$, which implies that $v(D_{i_s}) \subseteq v(D_{i_r})$, then all the vertices of D_{i_s} are also in D_{i_r}. Now assume that some extremal ray $\vec{r}$ of D_{i_s} is not in D_{i_r}. Then starting from some vertex $\vec{x}$ of D_{i_r}, there must be some $\lambda \leq 0$ such that $\vec{x} + \lambda r \notin D_{i_s}$ and the same holds for any $\lambda' \geq \lambda$ (by convexity). But since r has rational coordinates for some value of λ', $\lambda' r$ is an integer vector and so is $\vec{x} + \lambda' r$, which contradicts the fact that $v(D_{i_s}) \subseteq v(D_{i_r})$, for all integer parameter valuations v, and in particular $(\vec{x} + \lambda' r)_{|\mathbb{P}}$. We can therefore conclude that $D_{i_r} \subseteq D_{i_s}$ and we now proceed to proving that D_{i_s} is also "cheaper" than D_{i_r}.

We use another property of the vertices of convex polyhedra: vertices of a convex polyhedron of dimension n defined by m inequalities $\sum_{k=1}^{n} a_{kl} x_k \leq b_l$, for $j \in [1..m]$ are solutions of a system of n linearly independent equations $\sum_{k=1}^{n} a_{kl} x_k = b_l$, with l in a subset of size n of $[1..m]$.

Now consider the polyhedron D obtained from $\mathsf{IH}(D_{i_r})$, with its cost variable c, by adding one variable c' constrained by the cost inequalities of $\mathsf{IH}(D_{i_s})$. Clearly, since c and c' are not constrained together, the vertices of D are those of $\mathsf{IH}(D_{i_r})$, extended with the corresponding minimal and maximal values of c', and symmetrically those of $\mathsf{IH}(D_{i_s})$, extended with the corresponding minimal and maximal values of c'. Since the inequalities constraining c and c' have integer coefficients, and $\mathsf{IH}(D_{i_s})$ and $\mathsf{IH}(D_{i_r})$ have integer vertices, D also has integer vertices.

For the i-th lower-bound inequality on c, and the j-th lower-bound inequality on c', we define E_{ij} as D in which we transform both constraints into equalities. Clearly, from the property above, this does not add any new vertex, but it may remove some. Second, by construction, we have $\bigcup_{ij} E_{ij} = \{(\vec{\theta}, \min_{(\vec{\theta},c,v)\in\mathsf{IH}(D_{i_r})} c, \min_{(\vec{\theta},c,v)\in\mathsf{IH}(D_{i_s})} c) | \vec{\theta} \in \mathsf{IH}(D_r)_{|\Theta}\}$. If we minimize $c - c'$ over E_i, we know from the theory of linear programming that the minimum is obtained at a vertex of E_{ij}, and therefore, in particular, for an integer valuation v of the parameters, and an integer vector $\vec{\theta}$ of D_{i_r}. Since we have $v(C_{i_r}) \succcurlyeq v(C_{i_s})$, we then know that for these values of the theta variables and parameters, $c \leq c'$. This means that this holds for the whole of E_{ij}, and finally that $C_{i_r} \succcurlyeq C_{i_s}$. □

7 Case Study

We now consider a scheduling problem where some tasks include *runnables*, a key concept of the AUTomotive Open System ARchitecture (AUTOSAR), the open standard for designing the architecture of vehicle software [5]. Runnables represent the functional view of the system and are executed by the runtime of the software component [19]. For their execution they are mapped to tasks and

a given runnable can be split across different tasks to introduce parallelism, for instance. In industrial practice, runnables that interact a lot are mapped to the same task, in particular when they perform functions with the same period.

In this example, we consider 3 non-preemptive, periodic tasks T1, T2 and T3, on which have already been mapped some runnables that interact together; we add another independent runnable whose code must be split between tasks T1 and T2:

- the period of task T1 is 100 time units; T1 includes a "fixed part", independent from the new runnable and whose execution lasts 22 t.u.;
- the period of T2 is 200 t.u.; T2 also has a fixed part lasting 28 t.u.;
- the period of T3 is 400 t.u.; its execution lasts 11 t.u.;
- the period of the runnable is 200 t.u.; its execution lasts 76 t.u.; parameter a denotes the duration of the section that is executed in T1[3].

The processing unit consists of 2 cores C0 and C1; T3 can only execute on C0 whereas both T1 and T2 can execute on either core. When both cores are idle, the cost is null; when only one core is busy, the cost is equal to 2/t.u.; when both cores are busy, the cost is equal to 3/t.u. Any optimised strategy to divide the runnable over T1 and T2 and to allocate these tasks to C0 or C1 must therefore favour the cases where both cores are in the same state.

Figure 3 presents the model for this problem[4]. The associated cost function is: $2*(C0 \neq C1)+3*C0*C1+1000*\big(W1*(R1C0+R1C1)+W2*(R2C0+R2C1)+W3*R3C0\big)$, where the name of a place (e.g. $R1C0$) represents its marking[5].

We limit the study of the system to the first 400 t.u., at the end of which T1 has been executed 4 times, T2 twice and T3 once. A preliminary analysis (not detailed here for the sake of concision) showed that the lowest cost is 466. By setting our maximal cost to this value, we then check the following property with our Romeo tool: `EF four==4 and two==2 and one==1 and cost≤466`. The answer provided by Romeo is that the property is true iff $a \in [13, 17]$. We then set a to 17; Romeo yields the following timed trace, in which the notation T1@t1 means that transition T1 is fired at date t1: T1C0@61, T2C1@69, T1@100, end1_C0@100, T1C0@100, end1_C0@139, end2_C1@139, T1@200, T2@200, T2C0@261, T1C1@261, T1@300, end1_C1@300, T1C1@303, end2_C0@331, T3C0@331, end3_C0@342, end1_C1@342

From this trace, we obtain the Gantt chart in Figure 4 (above). Setting a to 13 yields another timed trace, resulting in the Gantt chart in Figure 4 (below). In both cases, we can see that both cores are busy during 148 t.u. (and for 11 t.u., only one is idle), which confirms our analysis on the optimised strategy above.

[3] Every 200 t.u., since T1 is executed twice as often as T2, T1 is running during $(22+a)*2 = 44+2a$ t.u. whereas T2 is running during $28+(76-2a) = 104-2a$ t.u.

[4] To ensure a correct access to the cores, we could have added one place for each core and some arcs on each task to capture and release them but the resulting net would have been quite unreadable. Instead, we chose to add 2 integer variables C0 and C1 (both initialised to 0); a variable equal to 0 (resp. 1) obviously means the corresponding core is idle (resp. busy).

[5] The last term ensures that such cases where an instance of a task is activated while a previous one is running are heavily penalised.

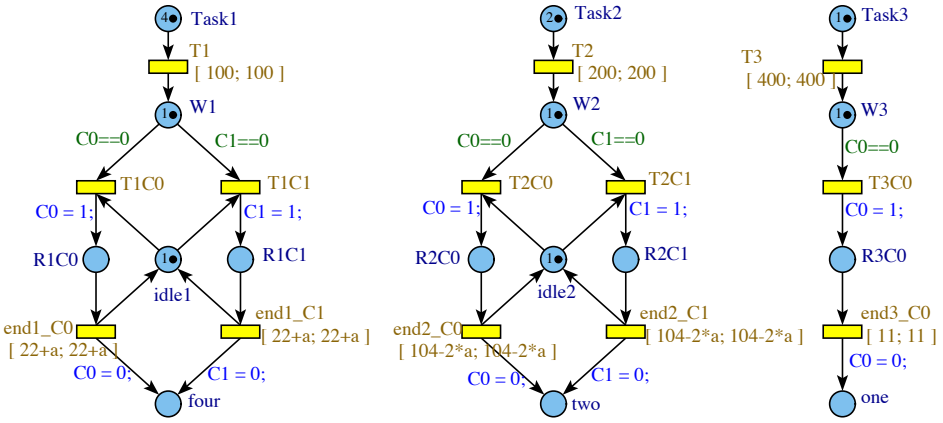

Fig. 3. Offline non preemptive scheduling problem

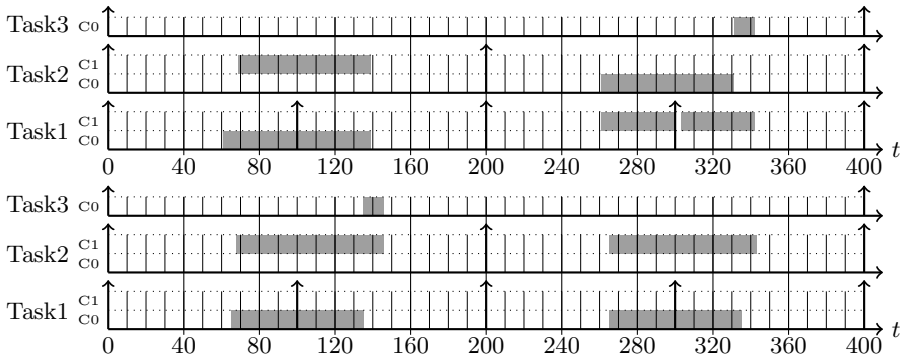

Fig. 4. Gantt charts for $a = 17$ (above) and $a = 13$ (below)

8 Conclusion

We have proposed a new Petri net-based formalism with parametric timing and cost features, thus merging two classic lines of work. For this formalism, we define an existential problem and a synthesis problem for parametric reachability within a bounded cost. We prove that the former is undecidable but we nonetheless give and prove a symbolic semi-algorithm for the latter. We finally propose a variant of the synthesis algorithm suitable for integer parameter valuations and prove its termination when those parameter valuations are bounded, and some other classic assumptions. This symbolic algorithm avoids the explicit enumeration of all possible parameter valuations. It is implemented in our tool Romeo and we have reported on a case-study addressing a scheduling problem, and inspired by the AUTOSAR standard.

Further work includes computing the optimal cost as a function of parameters and investigating the case of costs (discrete and rates) as parameters.

References

1. P. A. Abdulla and R. Mayr. Priced timed Petri nets. *Logical Methods in Computer Science*, 9(4), 2013.
2. R. Alur and D. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
3. R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *ACM Symposium on Theory of Computing*, pages 592–601, 1993.
4. R. Alur, S. L. Torre, and G. J. Pappas. Optimal paths in weighted timed automata. *Theoretical Computer Science*, 318(3):297 – 322, 2004.
5. AUTOSAR. Specification of RTE Software. Technical Report 4.4.0, october 2018.
6. R. Bagnara, P. Hill, and E. Zaffanella. Not necessarily closed polyhedra and the double description method. *Formal Aspects of Computing*, 17:222–257, 2005.
7. G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *HSCC 2001 Rome, Italy*, pages 147–161. Springer, 2001.
8. G. Behrmann, K. G. Larsen, and J. I. Rasmussen. Optimal scheduling using priced timed automata. *SIGMETRICS Perform. Eval. Rev.*, 32(4):34–40, Mar. 2005.
9. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE trans. on soft. eng.*, 17(3):259–273, 1991.
10. B. Berthomieu and M. Menasche. An enumerative approach for analyzing time petri nets. In *IFIP*, pages 41–46. Elsevier Science Publishers, 1983.
11. H. Boucheneb, D. Lime, B. Parquier, O. H. Roux, and C. Seidner. Optimal reachability in cost time Petri nets. In *FORMATS'17, Berlin, Germany*, LNCS, 2017.
12. P. Bouyer, M. Colange, and N. Markey. Symbolic optimal reachability in weighted timed automata. In *CAV'16*, volume 9779 of *LNCS*, Toronto, Canada, 2016.
13. T. Hune, J. Romijn, M. Stoelinga, and F. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 52-53:183–220, 2002.
14. A. Jovanović. *Parametric Verification of Timed Systems.* PhD thesis, École Centrale Nantes, Nantes, France, 2013.
15. A. Jovanović, D. Lime, and O. H. Roux. Integer parameter synthesis for real-time systems. *IEEE Transactions on Software Engineering (TSE)*, 41(5):445–461, 2015.
16. K. 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 *CAV'01*, volume 2102 of *LNCS*, pages 493–505, 2001.
17. K. G. Larsen, P. Pettersson, and W. Yi. Model-Checking for Real-Time Systems. In *Fundamentals of Computation Theory*, volume 965 of *LNCS*, pages 62–88, 1995.
18. M. Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall, 1967.
19. N. Naumann. AUTOSAR runtime environment and virtual function bus. Technical report, Hasso-Plattner-Institut, 2009.
20. A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, Inc., New York, NY, USA, 1986.
21. L.-M. Traonouez, D. Lime, and O. H. Roux. Parametric model-checking of stopwatch Petri nets. *Journal of Universal Computer Science*, 15(17):3273–3304, 2009.