Petri Nets with Time and Cost
(Tutorial)

Parosh Aziz Abdulla

Department of Information Technology
Uppsala University*
Sweden

parosh@it.uu.se

Richard Mayr

School of Informatics, LFCS
University of Edinburgh
United Kingdom

homepages.inf.ed.ac.uk/rmayr/

1 Introduction

Petri nets [13, 12] 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 (*TPN*s)* (see [10, 6] for surveys).

In parallel, there have been several works on extending the model of timed automata [4] with *prices* (*weights*) (see e.g., [5, 11, 8]). 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.

In this tutorial, we recall, through a sequence of examples, a very expressive model, introduced in [2], that subsumes the above models. *Priced Timed Petri Nets* (PTPN) are a generalization of classic Petri nets [13] with real-valued (i.e., continuous-time) clocks, real-time constraints, and prices for computations.

In a PTPN, 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. Like in timed automata, this is used to encode strict or non-strict inequalities that describe constraints on the real-valued clocks. When firing a transition, tokens which are removed from or added to places must have ages lying in the intervals of the corresponding transition arcs.

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. The total cost of a computation is given by the sum of all costs of fired transitions plus the storage costs for storing certain numbers of tokens in certain places for certain times during the computation. Like in priced timed automata, having integers as costs and time bounds is not a restriction, because the case of rational numbers can be reduced to the integer case.

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 (unlike timed automata). Secondly, every single clock value is a real number of which there are uncountably many.

In [2] 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 both continuous-time and discrete-time TPN [15], even without taking costs into account. Our goal is to compute the optimal cost for moving to a control state (equivalently for covering a set of markings). 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 $\leq \varepsilon$ for every $\varepsilon > 0$). We show that the *infimum* of the costs to reach a given control-state is computable, provided that all transition and place costs are non-negative.

Outline.   In the next section we introduce PTPNs. In Section 3 we describe a special type of computations that are sufficient to solve the cost-optimality problem. We introduce a symbolic encoding of infinite sets of markings in Section 4, and describe a symbolic algorithm for solving the cost-optimality problem in Section 5. Finally, in Section 6, we give conclusions and directions for future work.

2   Timed Petri Nets

In this section, we introduce Priced Timed Petri Nets, the set of markings, the transition relation it induces, and the coverability problem.

We use $\mathbb{N}$ and $\mathbb{R}_{\geq 0}$ to denote the sets of natural numbers (including 0) and nonnegative reals respectively. 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.

Model.   A *Priced Timed Petri Net (*PTPN*)* is a tuple $\mathcal{N} = (P, T, Cost)$ where P is a finite set of places. T is a finite set of transitions, where each transition $t \in T$ is of the form $t = (In, Out)$. We have that *In* and *Out* are finite multisets over $P \times Intrv$ which define the input-arcs and output-arcs of t, respectively. *Cost* : $P \cup T \to \mathbb{N}$ is the cost function assigning firing costs to transitions and storage costs to places. Figure 1 shows an example of a PTPN with five places: 🔴, ⚪, 🔵, 🟢, 🟠, and five transitions: t_1, t_2, t_3, t_4, t_5. The transition t_1 has an input arc from 🔴 labeled with the interval $[1..3]$, and two output arcs to ⚪ and 🔵, labeled with the intervals $(0..1)$ and $[2..5]$ respectively. The price (cost) associated with 🔴, is 3, while the price associated with t_1 is 2. We let *cmax* denote the maximum integer appearing on the arcs of a given PTPN. In Figure 1, we have *cmax* = 6.

Markings.   A marking is a multiset over $P \times \mathbb{R}_{\geq 0}$. The marking M defines the numbers and ages of tokens in each place in the net. In Figure 2, we show an example of a marking M. The marking assigns two tokens in 🔴, with ages 7.93 and 1.08, respectively. We will represent markings by lists of "colored balls" with real numbers inside. Each ball represents one token in the marking. The color describes the place in which the token resides, while the number represents the age of the token (see Figure 2).

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. A
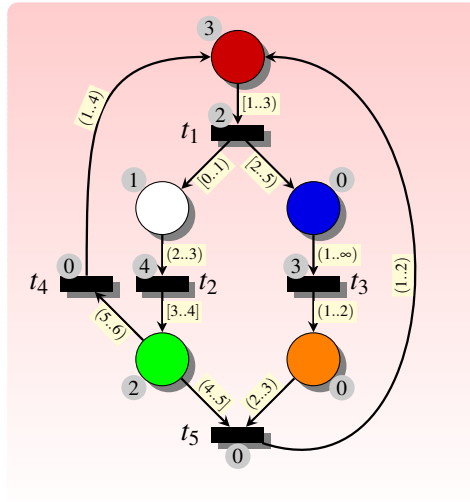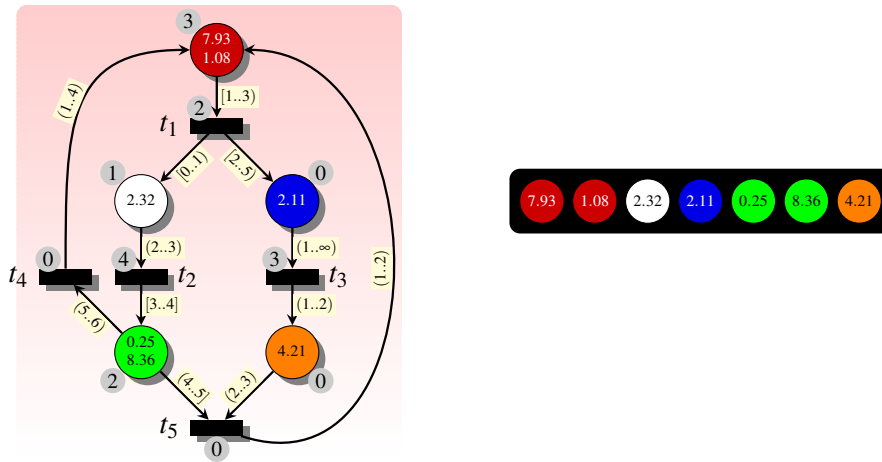
Figure 1: A Price Timed Petri Net.

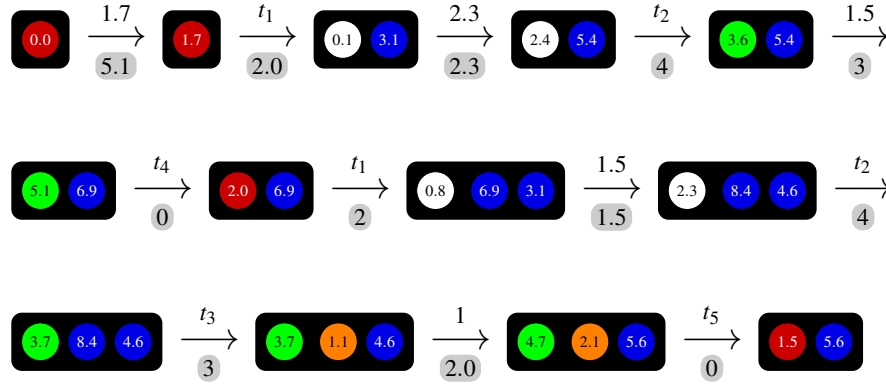Figure 2: A marking M and its representation.

Figure 3: A computation π. Above each $\longrightarrow$ in the computation we show the transition that has fired, and below each step we show the cost of the step.

discrete transition represents the effect of *firing* a transition t in the PTPN. More precisely, for each input arc to the transition, we remove a token from the corresponding input place, whose age lies in the relevant interval. Also, for each input arc to the transition, we add a new token to the corresponding place. The age of the newly generated token is chosen non-deterministically from the relevant interval. Performing a discrete transition implies paying a cost which is equal to the cost of the transition. When performing a timed transition, we pay a cost per each token and time unit that is equal to the cost of the place in which the token resides. A *computation* is a sequence of discrete and timed transitions. The cost of a computation is the accumulated cost of all the transitions in the computation. Figure 2 shows an example of a computation π. It starts from an initial marking where we have a single token in ⬤ with age 0. In the seventh step of π, transition t_1 fires removing one token from ⬤ with age 2. The age belongs to the interval $[1..3]$ (which is the interval on the arc from ⬤ to t_1). At the same time, it adds two new tokens with ages 0.8 and 3.1 to the places ⚪ resp. ⬤. The cost of this step is equal to 2. The eighth step is a timed transition of length 1.5, where the ages of all tokens are increased by 1.5. The cost of the step is determined by the number of tokens in each place and the cost of the place, i.e., $1.5 \times (1 \times 1 + 2 \times 0) = 1.5$ (the cost of ⚪ and ⬤ are 1 resp. 0). The total cost of π is given by $Cost(\pi) = 5.1 + 2 + 2.3 + 4 + 3 + 0 + 2 + 1.5 + 4 + 3 + 2 + 0 = 28.9$.

For a place p, we define M_p to be the set of markings which put at least one token in the place p (regardless of the ages of the tokens). For instance, if $p = $ ⬤ then M_p is the set of markings that have at least one token in ⬤.

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 (i) *initial* marking M_{init} that places a single token (with age 0) in a given initial place p_{init}, and (ii) a set of *final* markings $M_{p_{fin}}$ defined by a *final* place p_{fin}. In other words, we start from a marking where there is only one token with age 0 in p_{init} and where all the other places are empty, and then consider the cost of computations that takes us to $M_{p_{fin}}$.

In the *Cost-Threshold* problem we ask the question whether there is a computation starting from M_{init} and reaching a marking in $M_{p_{fin}}$ with a cost that is at most v for a given threshold $v \in \mathbb{N}$. In the *Cost-Optimality* problem, we want to compute the *optimal* (smallest) cost of reaching $M_{p_{fin}}$ staring from M_{init}. For given M_{init} and $M_{p_{fin}}$, the optimal cost of reaching $M_{p_{fin}}$ from M_{init} may not exist. However, in
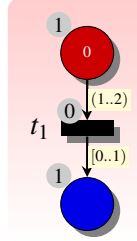
Figure 4: A Simple PTPN.

Figure 5: A marking in δ-form, $\delta = 0.2$.

[2], we show that the infimum of the costs of all computations is a natural number (or ∞ if $M_{p_{fin}}$ is not reachable from M_{init}). The situation is illustrated in Figure 4. The optimal cost for putting a token in ⬤ can be made arbitrarily close to 1 (but not equal to 1). In such a case, we simply define the optimal cost to be 1. In fact, the non-existence of an optimal cost has already been observed for timed automata [9].

3 Computations in δ-Form

In order to solve the Cost-Threshold and the Cost-Optimality problems, it is sufficient to consider computations of a certain form where the ages of all the tokens that appear in the computation are arbitrarily close to (within some small real number δ from) an integer. Below, we assume a real number $\delta : 0 < \delta < 0.2$.

δ-**Markings.** A marking M is said to be in δ-form (Figure 5) if any fractional part of the age of a token appearing in M is either smaller than δ or larger than $1 - \delta$. We decompose a δ-marking into submarkings such that in every submarking the fractional parts (but not necessarily the integer parts) of the token ages are identical. We then arrange these submarkings in a sequence $M_{-m}, \ldots, M_{-1}, M_0, M_1, \ldots, M_n$ such that $M_{-m}, \ldots, M_{-1}$ contain tokens with fractional parts $\geq \delta$ in increasing order, M_0 contains the tokens with fractional part zero, and $M_1, \ldots, M_n$ contain tokens with fractional parts $< \delta$ in increasing order. Figure 6 shows that partitioning of the marking M in Figure 5. More precisely, We start with the token with the high fractional parts, namely 0.91 (one token in ⬤), followed by 0.93 (one token in ⬤), followed by 0.97 (one token in ◯ and one token in ⬤). Furthermore, there are two tokens with zero fractional parts (one token in ◯ and one token in ⬤). Finally, we consider the tokens with low fractional parts, namely 0.02 (one token in ⬤, one token in ⬤, and one token in ⬤), followed by 0.03 (one token in ⬤), followed by 00.7 (one token in ⬤).

Computations in δ-form. The occurrence of a discrete transition t is said to be in δ-form if the ages of the newly generated tokens are close to an integer (i.e., within distance δ). This is not a property of the transition t as such, but a property of its occurrence. Figure 7 shows the result of an occurrence of t_1 in δ-form (with $\delta = 0.2$) on the marking of Figure 6.
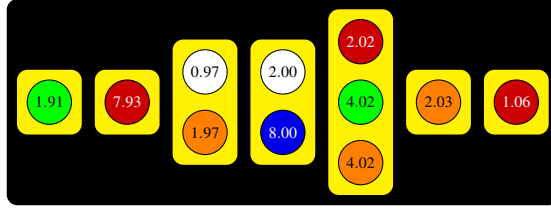   A computation is in δ-form if:

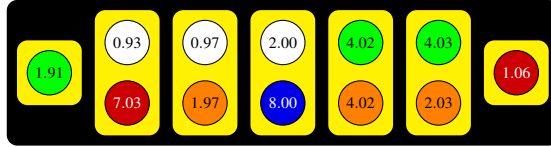Figure 6: The partitioning the marking in Figure 5.

Figure 7: An application in δ-form ($\delta = 0.2$) of t_1 on the marking of Figure 5. The two new tokens have fractional parts that are equal to 0.93 resp. 0.03.

1. Every occurrence of a discrete transition is in δ-form, and

2. For every timed transition, the delay is either in the interval $(0 : \delta)$ or in the interval $x \in (1 - \delta : 1)$.

Detailed Timed Transitions. We say that a timed transition (from a marking M) is *detailed* iff at most one fractional part of any token in M changes its status about reaching or exceeding the next integer value. Figures 8 and 9 show some steps in a detailed computation. In the first transition, time passes by a positive amount but not sufficiently long to make any tokens with positive fractional parts to increase to the next integer. More precisely, the time delay is 0.01 which means that two tokens in ◯ and ● that have zero fractional parts, will now have positive fractional parts (0.1). On the other hand, the two tokens in ◯ and ● that have the highest fractional parts (0.8) will not cross to the next integer (their ages will now be 0.98 and 1.98 respectively).

In the second step, the amount of delay is 0.02 which is exactly the amount needed to allow the tokens that currently have the highest fractional parts to become integers. These tokens are the ones with ages 0.98 and 1.98 in ◯ resp. ●. Their new ages are 1.00 resp. 2.00. In the last step, all tokens have small fractional parts. We let time pass sufficiently much (0.78 time units) so that the tokens will all have high fractional parts. Every computation of a PTPN can be transformed into an equivalent one (w.r.t. reachability and cost) where all timed transitions are detailed, by replacing long timed transitions with several detailed shorter ones where necessary. Thus we may assume w.l.o.g. that timed transitions are detailed.

Detailed Computations in δ-form. In [2], we show the following result. For any computation π starting from an initial marking M_{init} (defined by a *initial* place p_{init}), and reaching a give set $M_{p_{fin}}$ of final markings (defined by a *final* place p_{fin}), and for each $\delta : 0 < \delta < 0.2$, there is a detailed computation π' in δ-form where (i) π' starts from the same initial marking as π, (ii) π' is in δ-from, (iii) π' reaches $M_{p_{fin}}$, and (iv) if π is detailed then π' is detailed. This means that, to solve the Cost-Threshold and Cost-Optimality problems, it is sufficient to consider detailed computations in δ-form.

Figure 3 shows a detailed computation in δ-form for the PTPN of Figure 1.
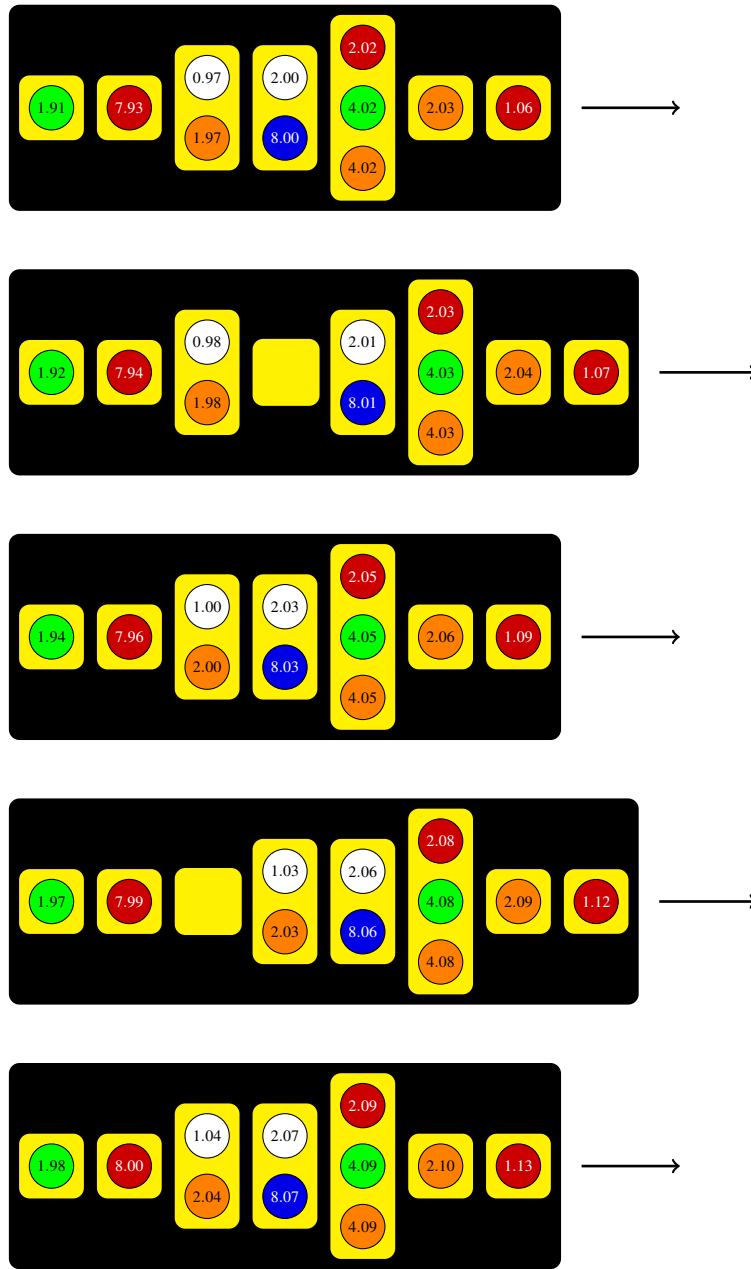
Figure 8: Detailed timed transitions for $\delta = 0.2$.

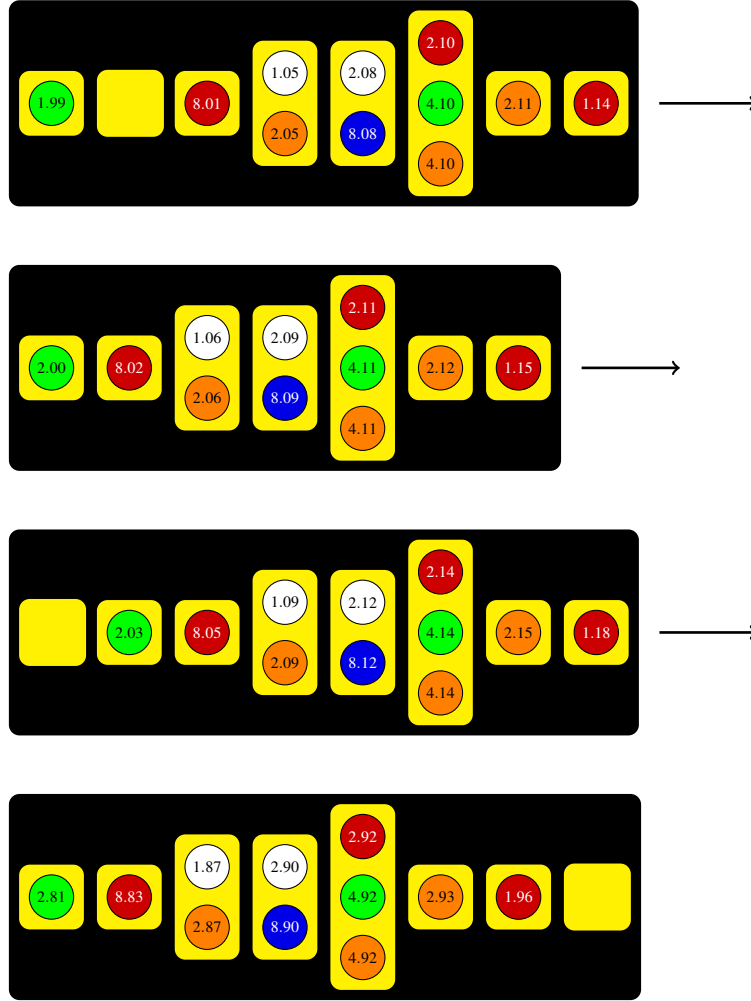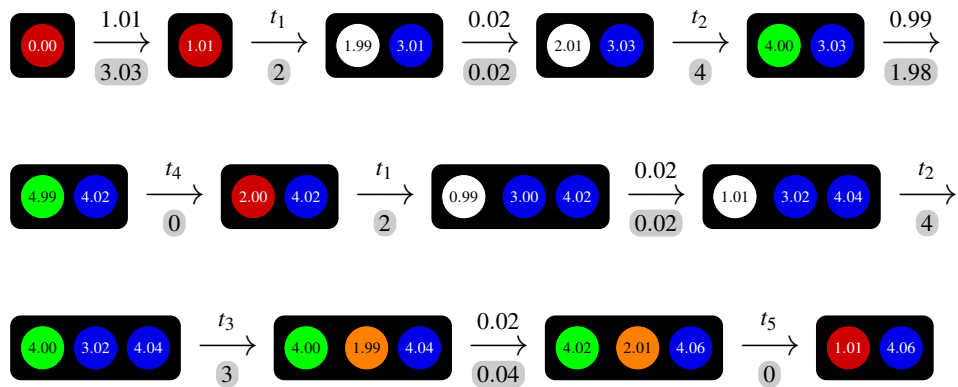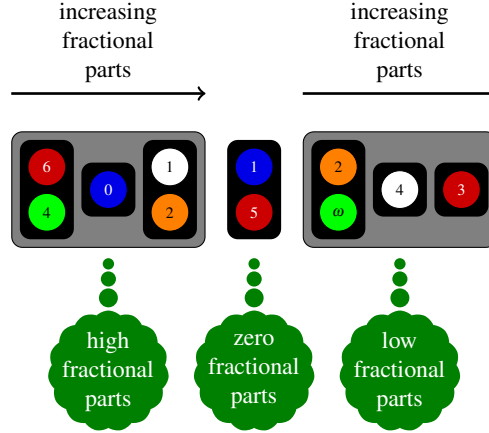Figure 9: Detailed timed transitions (cont.).

Figure 10: A detailed computation in δ-form.

Figure 11: A region r.

4 Regions

In this section, we introduce a symbolic encoding for infinite sets of markings. The encoding is a variant of the classical notion of *regions* [4]. The main difference is that we here need to deal with an unbounded number of clocks. It is an adaptation of the encoding introduced in [1]. More precisely, we change the encoding of [1] so that we can now deal with markings in δ-form. First, we give the definition of regions, and then we show how to simulate timed and discrete transitions on regions. For each type of transition, we define the cost of firing the transition from the region.

Regions. A region characterizes a set of marking in δ-form for some $\delta : 0 < \delta < 0.2$. An example of (our notion of) a region r is shown in Figure 11. The region consists of three parts, referred to as H (for high), Z (for zero), and L (for low). The part H is a word of multisets. Each element in a multiset is a colored ball with a natural number, representing one token. The color defines the place in which the token resides, while the number defines the integer part of the age of the token. Furthermore, tokens whose ages are larger than $cmax + 1$ are all represented by one element ω (ages $> cmax$ cannot be distinguished by the transitions of the PTPN). The ordering of the multisets reflects the ordering of the factional parts of the corresponding tokens: elements belonging to the same multiset represent tokens with identical fractional parts, and elements in successive multisets represent tokens with increasing fractional parts. The part Z consists of one multiset, and represents the tokens with zero fractional parts. Finally, the part L consists of a word of multisets. It has a similar interpretation to H, except that it represents tokens with low fractional parts. Figure 12 shows a marking M (of the Petri net of Figure 1) satisfying the region r of Figure 11 as follows:

- The left-most multiset in H contains a red ball with value 6 and a green ball with value 4. They represent the token with age 6.95 in the place 🔴, and the token with age 4.95 in 🟢. The fractional parts of the two tokens are equal (0.95) and high.

- The next multiset contains a blue ball with value 0. It represents the token with age 0.96 in 🔵. The fractional part of the token (0.96) is high and is larger than the fractional parts of the tokens in the previous multiset.

- The right-most multiset in H contains a white ball with value 1 and an orange ball with value 2. They represent the token with ages 1.97 in the place ⚪, and the token with age 2.97 in 🟠. The
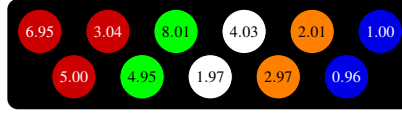
Figure 12: A marking *M* satisfying the region of Figure 11.

fractional parts of the two tokens are equal (0.97). The fractional parts of these tokens (0.97) are high and are larger than the fractional part of the token in the previous multiset.

- The part Z consists of a single multiset. It contains a blue ball with value 1 and a red ball with value 5. They represent the token with age 1.00 in the place ●, and the token with age 5 in ●. The fractional parts of the two tokens are zero.

- The left-most multiset in L contains an orange ball with value 2 and a green ball with value ω. They represent the token with age 2.01 in the place ●, and the token with age 8.01 in ●. The fractional parts of the two tokens are equal (0.01) and low. The age of the token in ● is $8.01 \geq cmax + 1$ which means that it is represented by ω in r.

- The next multiset contains a white ball with value 4. It represents that token with age 4.03 in ○. The fractional part of the token (0.03) is low and is larger than the fractional parts of the tokens in the previous multiset.

- The next multiset contains a red ball with value 3. It represents that token with age 3.04 in ●. The fractional part of the token (0.04) is low and is larger than the fractional part of the token in the previous multiset.

We use $[\![r]\!]$ to denote the set of markings satisfying r.

Timed Transitions.   We will describe how to encode the effect of detailed timed transitions on regions. To do that, we define 4 different types of transitions on regions.

Type I This simulates a small delay where the tokens of integer age now have a positive fractional part, but no tokens reach an integer age. An example of such a transition is shown in Figure 13. Here, the delay is 0.01 which is not sufficient to make the tokens with the highest fractional parts (the token with age 1.97 in ○, and the token with age 2.97 in ●) to become integers. Notice that the tokens with zero fractional parts (the token with age 1.00 in ●, and the token with age 5.00 in ●) will now have have low fractional parts (in fact, they will have the smallest fractional parts, namely 0.01, among all tokens in the marking). At the region level, the two elements in Z will move to L, forming the left-most multiset in L (reflecting the fact that they have the lowest fractional parts).

Type II Transition. This simulates a small delay in the case where there were no tokens of integer age and the tokens with the highest fractional parts just reach the next integer age. An example of such a transition is shown in Figure 14. Here, the delay is 0.02, which is sufficient to make the tokens with the highest fractional parts (the token with age 1.98 in ○, and the token with age 2.98 in ●) to become integers, i.e., 2 and 3 respectively. At the region level, the right-most multiset in H will move to Z, and the value of each element in the multiset is incremented by one to reflect the fact that the ages of the token moves to the next integer.

Type III Transition. This simulates a delay close to (but smaller than) 1 where the tokens with low fractional parts will now either have high fractional parts, or they have reached (and passed) the
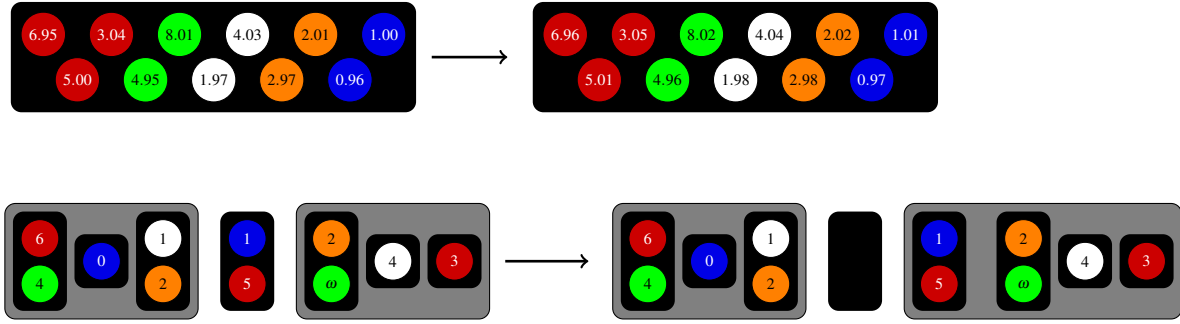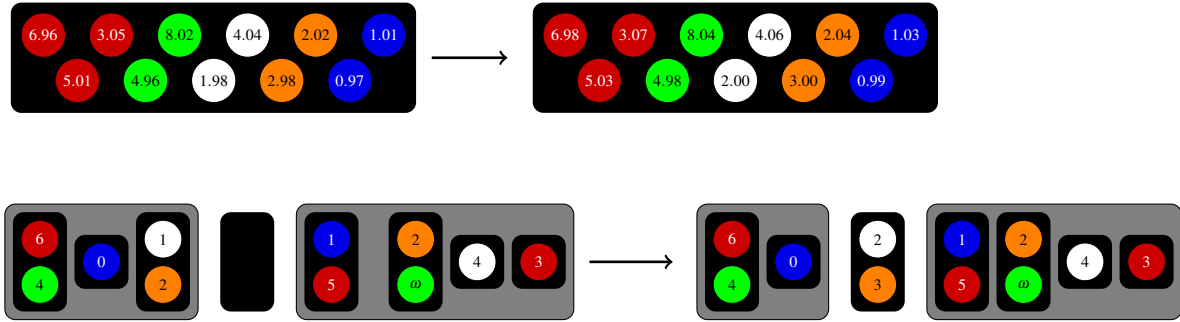
Figure 13: Type I Transition.

Figure 14: Type II Transition.

next integer and thus have low fractional parts again. The tokens that already had high fractional parts will all have passed the next integer and will now have high fractional parts again. No token will have an integer value after the transition (the case where some tokens have integer ages is covered in Type IV transitions, see below). Here, the delay is 0.95. We have three types of tokens:

- Tokens that have low fractional parts both before and after the transition (the token with age 4.06 in ◯, and the token with age 3.07 in ●). The ages of these tokens are 5.01 and 4.02 after the transition. Thus, the delay is sufficient to make their ages go beyond the next integer. After the transition, these tokens will be the only ones with low fractional parts. The relative ordering of their fractional parts will not be changed. The integer part of their ages will have increased by one. At the region level, these two tokens are represented by the two right-most multisets in L. After the transition, they will be the only multisets in L, and their values are incremented by 1 each. Notice that the relative ordering of these tokens inside the region will be preserved.

- Tokens that have low fractional parts before the transition and high fractional parts after the transition (the token with age 1.03 in ●, the token with age 5.03 in ●, the token with age 2.04 in ●, and the token with age 8.04 in ●). The ages of these tokens are 1.98, 5.98, 2.99, resp. 8.99 after the transition. These tokens have the highest fractional parts among all tokens in the marking. The relative ordering of the fractional parts of these tokens will not be changed. Also, the delay is not sufficiently long to make their values reach (or pass) to the next integer. At the region level, the corresponding multisets move from L to H, and will now be the right-most multisets in H. The ordering of these multisets is preserved.

- Tokens that have high fractional parts both before and after the transition (the token with
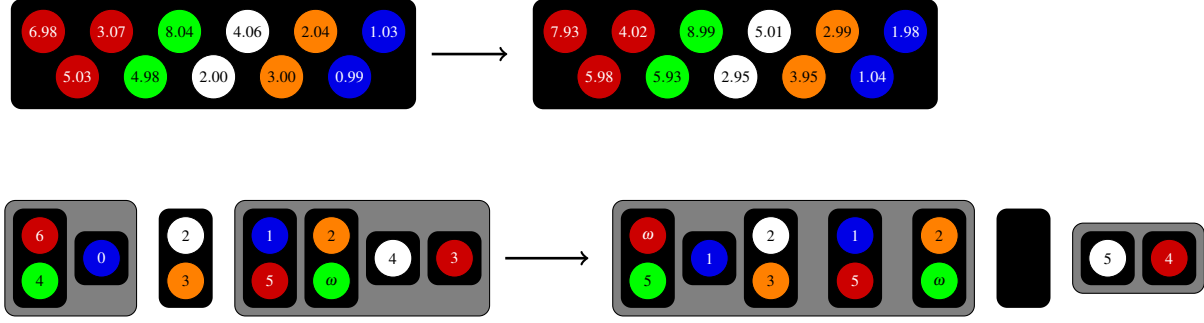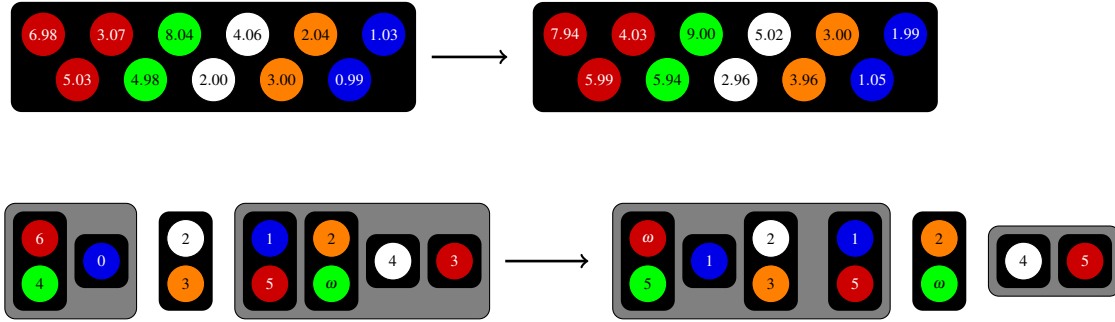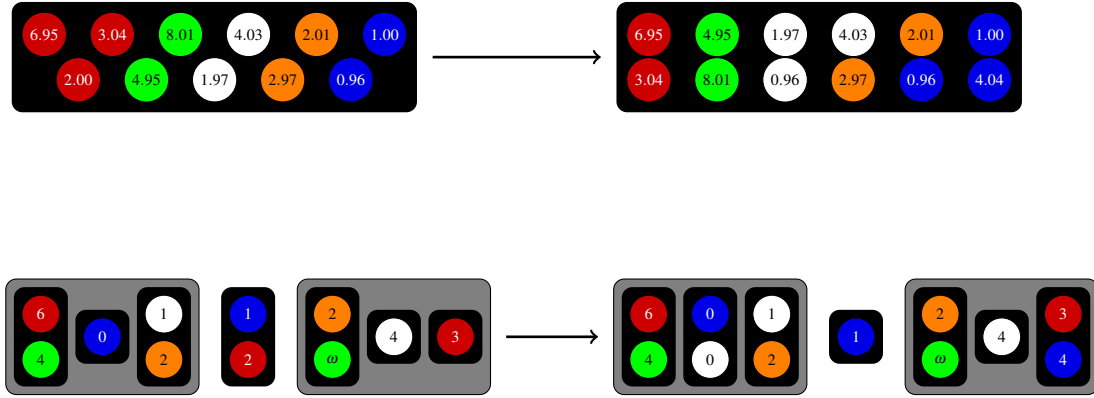
Figure 15: Type III Transition.

Figure 16: Type IV Transition.

age 6.98 in 🔴, the token with age 4.98 in 🟢, and the token with age 0.99 in 🔵). The ages of these tokens are 7.93, 5.93, resp. 1.94 after the transition. The delay is sufficiently long both to make their values pass the next integer integer, and to make their fractional parts high again. However, these tokens have now the lowest fractional parts among all tokens with high fractional parts. The relative ordering of the fractional parts of the tokens will not be changed. At the region level, the corresponding multisets will be the left-most multisets in H. The ordering of these multisets is preserved. Their values are incremented by one (to reflect that they have reached the next integer). Notice that the new value of the token in 🔴 is represented by ω since the value is $\geq cmax + 1$.

Type IV Transition. This is similar to a Type III transition, except that some of the tokens that have low fractional parts will have integer values after the transition (see Figure 16).

Discrete Transitions. Figure 17 shows the firing of transition t_1 (Figure 1), and describes how the firing of the transition may be simulated at the region level. We remove a token from 🔴 whose age is in the interval [1..3]. This is done at the region level by removing the red ball with value 2 from Z (the ball represents a token in 🔴 whose age is exactly 2). We add one to token to ⚪ whose age is in the interval (0..1), and one to token to 🔵 whose age is in the interval [2..5]. In Figure 17, this is done at the region level by adding a white ball to a multiset in H with value 0 (the ball represents a token in ⚪ whose age is in the interval (0..1)), and adding a blue ball to a multiset in L with value 4 (the ball represents a token in 🔵 whose age is in the interval (4..5)).

Figure 17: Firing the transition t_1.

Costs. At the region level, the cost of performing a type I or type II transition is 0, since we can assume the time delay to be arbitrarily small. The cost of performing a type III or type IV transition is equal to the cost of performing a timed transition of 1 time unit, since we can make the delay arbitrarily close to 1. Thus, the cost of performing the transition in Figure 15 or Figure 16 is 15. The cost of performing a discrete transition at the region level is the same as the cost of performing the transition on concrete markings. Thus, the cost of performing the transition in Figure 17 is 2.

5 Solving the Cost-Optimality Problem

In this section we explain our solution for the Cost-Optimality problem. Here, we give an informal overview of the main ideas. The (quite complicated) technical details can be found in [2]. First, we show that the Cost-Optimality problem can be reduced to the Cost-Threshold problem. Then, we introduce a general framework of ordered transition systems, which we then instantiate to the case of regions. Finally, we present an algorithm that allows to solve the Cost-Threshold problem.

From Cost-Optimality to Cost-Threshold. Consider an instance the Cost-Optimality problem, defined by M_{init} and $M_{p_{fin}}$ (see Section 2). The task is to compute the optimal cost of reaching $M_{p_{fin}}$ from M_{init}, i.e., the infimum of the costs of all computations reaching $M_{p_{fin}}$ from M_{init}. To compute this value, it suffices to solve the Cost-Threshold problem for any given threshold $v \in \mathbb{N}$, i.e., to decide whether there is any computation from M_{init} to $M_{p_{fin}}$ with cost $\leq v$. To see this, we first decide whether $M_{p_{fin}}$ is reachable from M_{init} in the underlying timed Petri net (without considering costs). 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 the answer is no, then we can define the optimal cost to be ∞ ($M_{p_{fin}}$ is not reachable form M_{init}). 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. We solve the Cost-Threshold problem using regions as symbolic encodings of sets of markings.

Ordered Transition Systems. An *ordered transition system* is a triple $\mathscr{T} = (S, \longrightarrow_A, \sqsubseteq)$ where S is a (potentially) infinite set of *configurations* (or *states*), $\longrightarrow$ is a transition relation on S, and $\sqsubseteq$ is an ordering on S. We say that $\longrightarrow$ is *monotone* wrt. $\sqsubseteq$ if the following holds for all configurations $c_1, c_2, c_3 \in S$: if $c_1 \longrightarrow c_2$ and $c_1 \sqsubseteq c_3$ then there is a c_4 such that $c_3 \longrightarrow c_4$ and $c_2 \sqsubseteq c_4$.
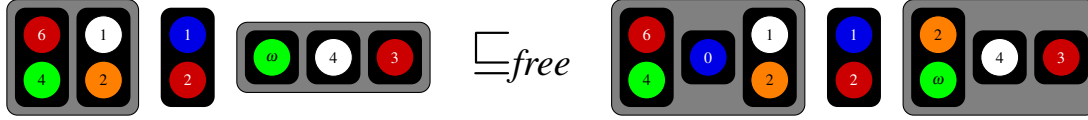
Figure 18: Ordering on Regions.

For a set $\mathsf{S} \subseteq S$ of configurations, we define $\mathit{Pre}(\mathsf{S})$ to be the set of predecessors of S wrt. $\longrightarrow$, i.e., the set of configurations from which we can reach a configuration in S through a single application (a single step) of $\longrightarrow$. We define Pre^* to be the reflexive transitive closure of Pre, i.e., $\mathit{Pre}^*(\mathsf{S})$ is the set of configurations from which we can reach a configuration in S through any number of steps of $\longrightarrow$.

A set $\mathsf{S} \subseteq S$ is said to be *upward-closed* if for any two configurations with $c_1 \sqsubseteq c_2$, it is the case that $c_1 \in \mathsf{S}$ implies $c_2 \in \mathsf{S}$. The *upward closure* $\mathsf{S}\!\uparrow$ of a set S of configurations is the set of configurations that are larger than or equal to some configuration in S wrt. $\sqsubseteq$, i.e., $\mathsf{S}\!\uparrow := \{c' \in S | \exists c \in \mathsf{S}. c \sqsubseteq c'\}$. Below, we will consider different transition systems that are induced by different sets of configurations and different transition relations.

Instantiation.   Consider an instance of the Cost-Threshold problem, defined by M_{init}, $M_{p_{fin}}$, and a threshold v. Define a configuration c to be a pair (r, u) where r is a region, and $u \leq v$. Intuitively, u denotes the maximal allowed cost of the remainder of a computation that passes through r. Let S be the set of all configurations. Let C be the set of configurations of the form (r, u) where r contains only tokens in the costs places (places whose costs are larger than 0), and where the number of tokens in r is smaller than u. Notice that C is finite. Consider regions r_1, r_2. We write $r_1 \sqsubseteq_{all} r_2$ if we can obtain r_2 from r_1 by adding a number of tokens to r_1. We write $r_1 \sqsubseteq_{free} r_2$ if we can obtain r_2 from r_1 by adding a number of tokens to the free places (places whose costs are 0). Notice that $\sqsubseteq_{free} \subseteq \sqsubseteq_{all}$. Figure 18 shows an example of two regions (interpreted over the PTPN of Figure 1) related by $\sqsubseteq_{free}$. For configurations $c_1 = (r_1, u_1)$ and $c_2 = (r_2, u_2)$, we use $c_1 \sqsubseteq_{all} c_2$ resp. $c_1 \sqsubseteq_{free} c_2$ to denote that $u_1 = u_2$ and that $r_1 \sqsubseteq_{all} r_2$ resp. $r_1 \sqsubseteq_{free} r_2$. For a set $\mathsf{S} \subseteq S$ of configurations, we use $\mathsf{S}\!\uparrow\! free$ to be the *upward closure* of S with respect to $\sqsubseteq_{free}$, i.e., it contains all configurations that are larger than or equal to some configuration in S wrt. $\sqsubseteq_{free}$. We define $\mathsf{S}\!\uparrow\! all$ in a similar manner.

Let $\longrightarrow_i$ denote the timed transition relation of type $i \in \{I, II, III, IV\}$, and let $\longrightarrow_{Disc}$ be the discrete transition relation. Define $\longrightarrow_A := \longrightarrow_1 \cup \longrightarrow_2 \cup \longrightarrow_{Disc}$, i.e., a transition of type A is either a timed transition of type I or II, or a discrete transition. Define $\longrightarrow_B := \longrightarrow_3 \cup \longrightarrow_4$, i.e., a transition of type B is a timed transition of type III or IV. For a set M, we define $\mathit{Pre}_A(\mathsf{M})$ to be the set of markings from which we can reach a marking in M through a single application of a transition of type A. We define $\mathit{Pre}_B(\mathsf{M})$ analogously.

Algorithm.   We give an overview of an algorithm to solve the reachability problem. We notice that $M_{p_{fin}}$ is reachable from M_{init} with a cost $\leq v$ iff $M_{init} \overset{*}{\longrightarrow}_A \cdot \left(\longrightarrow_B \cdot \overset{*}{\longrightarrow}_A \right)^+ M_{p_{fin}}$ and the accumulated cost of all involved transitions is $\leq v$. Furthermore, we observe that $M_{p_{fin}}$ can be characterized by the upward closure (wrt. $\sqsubseteq_{all}$) of a finite set of regions. Therefore, it is sufficient to give an algorithm that, given a region r_{fin} and threshold v, checks whether there is a region r_{init} where M_{init} is included in the denotation of r_{init} such that $(r_{init}, 0) \overset{*}{\longrightarrow}_A \cdot \left(\longrightarrow_B \cdot \overset{*}{\longrightarrow}_A \right)^+ (r_{fin}, v)\!\uparrow\! all$. To do that, we generate a sequence of sets of configurations $V_1, U_1, V_2, U_2, \ldots$, as follows:

- $V_1 := min_{free} \left(Pre_A^* \left((r_{fin}, v) \uparrow all \right) \cap (C \uparrow free) \right)$. This set is possible to compute as follows. The set $(r_{fin}, v) \uparrow all$ is (obviously) upward-closed wrt. $\sqsubseteq_{all}$. The relation $\longrightarrow_A$ is monotone wrt. $\sqsubseteq_{all}$. We can then use the backward reachability algorithm (introduced in [3]) for well quasi-ordered systems to compute $min_{all} \left(Pre_A^* \left((r_{fin}, v) \uparrow all \right) \right)$. The result follows from the fact that both $min_{all} \left(Pre_A^* \left((r_{fin}, v) \uparrow all \right) \right)$ and C are finite.

- $U_1 := min_{free} \left(Pre_B(V_1 \uparrow free) \right)$. This set can be computed by a straightforward application of $\longrightarrow_B$ on the elements of V_1. Notice that $U_1 \subseteq C \uparrow free$, and that it is a finite set.

- For $k > 1$, given the finite set U_k, we compute $V_k := min_{free} \left(Pre_A^*(U_k \uparrow free) \cap (C \uparrow free) \right)$. Notice that we here are solving a *reachability* problem rather than *coverability* problem, since $U_k \uparrow free$ is not upward-closed wrt. $\sqsubseteq_{all}$. In fact, this problem has an extremely complicated solution (described in [2]). The construction to compute it uses many calls to a subroutine which relies on the decidability of the reachability problem for Petri nets with one inhibitor arc [14, 7]. In a sense, this is unavoidable, since the reverse reduction also holds. The reachability problem for Petri nets with one inhibitor arc can be reduced to the zero-cost coverability problem for PTPN, i.e., Cost-Threshold with threshold 0.

- For $k > 1$, we compute $U_k := min_{free} \left(Pre_B(V_k \uparrow free) \right)$ in a similar manner to U_1. Notice that $U_k \subseteq C \uparrow free$, and that it is a finite set.

The sequence $U_1 \uparrow free, U_2 \uparrow free, \ldots$ is a monotone-increasing sequence of upward-closed (wrt. $\sqsubseteq_{free}$) subsets of $C \uparrow free$. This sequence converges, because $\sqsubseteq_{free}$ is a well-quasi-ordering on $C \uparrow free$. Therefore, we get $U_n = U_{n+1}$ for some finite index n and $U_n \uparrow free = \left\{ c \mid c(\longrightarrow_B \stackrel{*}{\longrightarrow}_A)^* r_{fin} \right\}$, because the transition $\rightarrow_B$ is only enabled in $C \uparrow free$. Finally, we compute the (finite) set of configurations, $min_{all} \left(Pre_A^*(U_n \uparrow free) \right)$, and check whether the set contains a configuration of the form (r_{init}, u) such that M_{init} belongs to the denotation of r_{init}.

6 Conclusions and Future Work

We have given an informal description of a method for computing the infimum of the costs of placing a token in given place of a timed Petri net, starting from a given initial marking. Interesting directions for future work include augmenting time with other infinite-state discrete models such as push-down systems and asynchronously communicating processes, and to add other quantitative parameters such as probabilistic behaviors.

References

[1] P.A. Abdulla & B. Jonsson (2003): *Model checking of systems with many identical timed processes.* Theoretical Computer Science 290(1), pp. 241–264, doi:10.1016/S0304-3975(01)00330-9.

[2] P.A. Abdulla & R. Mayr (2011): *Computing optimal coverability costs in priced timed Petri nets.* In: *Logic in Computer Science (LICS), 2011 26th Annual IEEE Symposium on*, IEEE, pp. 399–408, doi:10.1109/LICS.2011.40.

[3] Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson & Yih-Kuen Tsay (1996): *General Decidability Theorems for Infinite-State Systems.* In: *LICS*, pp. 313–321, doi:10.1109/LICS.1996.561359.

[4] R. Alur & D. Dill (1994): *A Theory of Timed Automata.* TCS 126, pp. 183–235, doi:10.1016/0304-3975(94)90010-8.

[5] R. Alur, S. La Torre & G. J. Pappas (2001): *Optimal Paths in Weighted Timed Automata*. In: *HSCC*, pp. 49–62, doi:10.1007/3-540-45351-2_8.

[6] B. Bérard, F. Cassez, S. Haddad, O. Roux & D. Lime (2005): *Comparison of Different Semantics for Time Petri Nets*. In: *Automated Technology for Verification and Analysis*, LNCS 3707, Springer Berlin Heidelberg, pp. 293–307, doi:10.1007/11562948_23.

[7] R. Bonnet (2011): *The reachability problem for Vector Addition Systems with one zero-test*. In Filip Murlak & Piotr Sankowski, editors: *Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS'11)*, LNCS 6907, Springer, pp. 145–157, doi:10.1007/978-3-642-22993-0_16.

[8] P. Bouyer, T. Brihaye, V. Bruyère & J. Raskin (2007): *On the optimal reachability problem of weighted timed automata*. *Formal Methods in System Design* 31(2), pp. 135–175, doi:10.1007/s10703-007-0035-4.

[9] P. Bouyer, F. Cassez, E. Fleury & K. G. Larsen (2005): *Optimal Strategies in Priced Timed Game Automata*. In: *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science*, Lecture Notes in Computer Science 3328, Springer Berlin Heidelberg, pp. 148–160, doi:10.1007/978-3-540-30538-5_13.

[10] F. D. J. Bowden (1996): *Modelling Time in Petri nets*. In: *Proc. Second Australian-Japan Workshop on Stochastic Models*.

[11] K.G. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson & J. Romijn (2001): *As Cheap as Possible: Efficient Cost-Optimal Reachability for Priced Timed Automata*. In: *Proc. 13th Int. Conf. on Computer Aided Verification*, Lecture Notes in Computer Science 2102, Springer Berlin Heidelberg, pp. 493–505, doi:10.1007/3-540-44585-4_47.

[12] J.L. Peterson (1977): *Petri Nets*. *Computing Surveys* 9(3), pp. 221–252, doi:10.1145/356698.356702.

[13] C.A. Petri (1962): *Kommunikation mit Automaten*. Ph.D. thesis, University of Bonn.

[14] K. Reinhardt (2008): *Reachability in Petri Nets with Inhibitor Arcs*. *Electronic Notes in Theoretical Computuer Science* 223, pp. 239–264, doi:10.1016/j.entcs.2008.12.042.

[15] V. Valero Ruiz, F. Cuartero Gomez & D. de Frutos Escrig (1999): *On non-decidability of reachability for timed-arc Petri nets*. In: *Proceedings of the The 8th International Workshop on Petri Nets and Performance Models*, PNPM '99, IEEE Computer Society, pp. 188–196, doi:10.1109/PNPM.1999.796565.