Béatrice Bérard, Franck Cassez, Serge Haddad, O. H. Roux, and Didier Lime.
Comparison of the Expressiveness of Timed Automata and Time Petri Nets.
In *Proceedings of the 3rd International Conference on Formal Modelling and Analysis of Timed Systems* **(FORMATS'05)**, Lecture Notes in Computer Science, Uppsala, Sweden, September 2005. Springer. Extended version with proofs.

Comparison of the Expressiveness of Timed Automata and Time Petri Nets

B. Bérard[1], F. Cassez[2,⋆], S. Haddad[1], D. Lime[3], O.H. Roux[2,⋆]

[1] LAMSADE, Paris, France
E-mail: {berard | haddad}@lamsade.dauphine.fr

[2] IRCCyN, Nantes, France
{Franck.Cassez | Olivier-h.Roux}@irccyn.ec-nantes.fr

[3] CISS, Aalborg, Denmark
didier@cs.aau.dk

Abstract. In this paper we consider the model of Time Petri Nets (TPN) where time is associated with transitions. We also consider Timed Automata (TA) as defined by Alur & Dill, and compare the expressiveness of the two models w.r.t. timed language acceptance and (weak) timed bisimilarity. We first prove that there exists a TA $\mathcal{A}$ s.t. there is no TPN (even unbounded) that is (weakly) timed bisimilar to $\mathcal{A}$. We then propose a structural translation from TA to (1-safe) TPNs preserving timed language acceptance. Further on, we prove that the previous (slightly extended) translation also preserves weak timed bisimilarity for a syntactical subclass $\mathcal{TA}_{syn}(\leq,\geq)$ of TA. For the theory of TPNs, the consequences are: 1) TA, bounded TPNs and 1-safe TPNs are equally expressive w.r.t. timed language acceptance; 2) TA are strictly more expressive than bounded TPNs w.r.t. timed bisimilarity; 3) The subclass $\mathcal{TA}_{syn}(\leq,\geq)$, bounded and 1-safe TPNs "à la Merlin" are equally expressive w.r.t. timed bisimilarity.

Keywords: Timed Language, Timed Bisimilarity, Time Petri Nets, Timed Automata, Expressiveness.

1 Introduction

In the last decade a number of extensions of Petri Nets with time have been proposed: among them are *Stochastic* Petri Nets, and different flavors of so-called *Time* or *Timed* Petri nets. Stochastic Petri Nets are now well known and a lot of literature is devoted to this model whereas the theoretical properties of the other timed extensions have not been investigated much.

Petri Nets with Time. Recent work [1,11] considers Timed Arc Petri Nets where each token has a clock representing its "age" but a lazy (non-urgent) semantics of the net is assumed: this means that the firing of transitions may

⋆ Work supported by the ACI CORTOS, a program of the French government.

be delayed, even if this implies that some transitions are disabled because their input tokens become too old. Thus the semantics used for this class of Petri nets is such that they enjoy nice *monotonic* properties and fall into a class of systems for which many problems are decidable.

In comparison, the other timed extensions of Petri Nets (apart from Stochastic Petri Nets), *i.e.* Time Petri Nets (TPNs) [18] and Timed Petri Nets [20], do not have such nice monotonic features although the number of *clocks* to be considered is finite (one per transition). Also those models are very popular in the Discrete Event Systems and industrial communities as they allow to model real-time systems in a simple and elegant way and there are tools to check properties of Time Petri Nets [6,14].

For TPNs a transition can fire within a time interval whereas for Timed Petri Nets it fires as soon as possible. Among Timed Petri Nets, time can be assigned to places or transitions [21,19]. The two corresponding subclasses namely P-Timed Petri Nets and T-Timed Petri Nets are expressively equivalent [21,19]. The same classes are defined for TPNs i.e. T-TPNs and P-TPNs, and both classes of Timed Petri Nets are included in both P-TPNs and T-TPNs [19]. P-TPNs and T-TPNs are proved to be incomparable in [16].

The class T-TPNs is the most commonly-used subclass of TPNs and in this paper we focus on this subclass that will be henceforth referred to as TPN.

Timed Automata. Timed Automata (TA) were introduced by Alur & Dill [3] and have since been extensively studied. This model is an extension of finite automata with (dense time) *clocks* and enables one to specify real-time systems. Theoretical properties of various classes of TA have been considered in the last decade. For instance, classes of determinizable TA such as *Event Clock Automata* are investigated in [4] and form a strict subclass of TA.

TA and TPNs. TPNs and TA are very similar and until now it is often assumed that TA have more features or are more expressive than TPNs because they seem to be a lower level formalism. Anyway the expressiveness of the two models have not been compared so far. This is an important direction to investigate as not much is known on the complexity or decidability of common problems on TPNs *e.g.* "is the universal language decidable on TPNs ?". Moreover it is also crucial for deciding which specification language one is going to use. If it turns out that TPNs are strictly less expressive (w.r.t. some criterion) than TA, it is important to know what the differences are.

Related Work. In a previous work [10] we have proved that TPN forms a subclass of TA in the sense that every TPN can be simulated by a TA (weak timed bisimilarity). A similar result can be found in [17] with a completely different approach. In another line of work in [15], the authors compare Timed State Machines and Time Petri Nets. They give a translation from one model to another that preserves timed languages. Nevertheless, they consider only the constraints with closed intervals and do not deal with general timed languages (*i.e.* Büchi timed languages). [9] also considers expressiveness problems but for a subclass of TPNs. Finally it is claimed in [9] that 1-safe TPNs with weak[4] constraints are strictly less expressive than TA with arbitrary types of constraints but a fair comparison should allow the same type of constraints in both models.

Our Contribution. In this article, we compare precisely the expressive power of TA vs. TPN using the notions of *Timed Language Acceptance* and *Timed Bisimilarity*. This extends the previous results above in the following directions: *i)* we consider general types of constraints (strict, weak); *ii)* we then show that there is a TA $\mathcal{A}_0$ s.t. no TPN is (even weakly) timed bisimilar to $\mathcal{A}_0$; *iii)* this leads us to consider weaker notions of equivalence and we focus on Timed Language Acceptance. We prove that TA (with general types of constraints) and TPN are equally expressive w.r.t. Timed Language Acceptance which is a new and somewhat surprising result; for instance it implies (using a result from [10]) that 1-safe TPNs and bounded TPNs are equally expressive w.r.t. Timed Language Acceptance; *iv)* to conclude we characterize a syntactical subclass of TA that is equally expressive to TPN without strict constraints w.r.t. Timed Bisimilarity. The results of the paper are summarized in Table 1: all the results are new except the one followed by [10]. We use the following notations: B-$\mathcal{TPN}_\varepsilon$ for the set of bounded TPNs with ε-transitions; 1-B-$\mathcal{TPN}_\varepsilon$ for the subset of B-$\mathcal{TPN}_\varepsilon$ with at most one token in each place (one safe TPN); B-$\mathcal{TPN}(\leq,\geq)$ for the subset of B-$\mathcal{TPN}_\varepsilon$ where only closed intervals are used; $\mathcal{TA}_\varepsilon$ for TA with ε-transitions; $\mathcal{TA}_{syn}(\leq,\geq)$ for the syntactical subclass of TA that is equivalent to B-$\mathcal{TPN}(\leq,\geq)$ (to be defined precisely in section 5). In the table $\preceq_\mathcal{L}$ or $\preceq_\mathcal{W}$ with $\preceq \in \{<,\leq\}$, respectively means "less expressive" w.r.t. Timed Language Acceptance and Weak Timed Bisimilarity; $=_\mathcal{L}$ means "equally expressive as" w.r.t. language acceptance and $\approx_\mathcal{W}$ "equally expressive as" w.r.t. weak timed bisimilarity.

Outline of the paper. Section 2 introduces the semantics of TPNs and TA, Timed Languages and Timed Bisimilarity. In section 3 we prove our first result: there is a TA $\mathcal{A}_0$ s.t. there is no TPN that is (weakly) timed bisimilar to $\mathcal{A}_0$. In section 4 we focus on Timed Language Acceptance and we propose a structural translation from TA to 1-B-$\mathcal{TPN}_\varepsilon$ preserving timed language acceptance. We then prove that TA and bounded TPNs are equally expressive w.r.t. Timed Language Acceptance. This enables us to obtain new results for TPNs given by corollaries 3 and 4. Finally, in section 5, we characterize a syntactical subclass of TA ($\mathcal{TA}_{syn}(\leq,\geq)$) that is equivalent, w.r.t. Timed Bisimilarity, to the original version of TPNs (with closed intervals).This enables us to obtain new results for TPNs given by corollary 6.

2 Time Petri Nets and Timed Automata

Notations. Let Σ be a set (or alphabet). Σ^* (resp. Σ^ω) denotes the set of finite (resp. infinite) sequences of elements (or words) of Σ and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. By convention if $w \in \Sigma^\omega$ then the *length* of w denoted $|w|$ is ω; otherwise if

[4] Constraints using only $\leq$ and $\geq$.

	Timed Language Acceptance	Timed Bisimilarity
B-$\mathcal{TPN}_\varepsilon$	$\leq_\mathcal{L}$ $\mathcal{TA}_\varepsilon$ [10]	$\leq_\mathcal{W}$ $\mathcal{TA}_\varepsilon$ [10]
	$=_\mathcal{L}$ 1-B-$\mathcal{TPN}_\varepsilon$ $=_\mathcal{L}$ $\mathcal{TA}_\varepsilon$	$<_\mathcal{W}$ $\mathcal{TA}_\varepsilon$
B-$\mathcal{TPN}(\leq,\geq)$	$=_\mathcal{L}$ $\mathcal{TA}_{syn}(\leq,\geq)$	$\approx_\mathcal{W}$ 1-B-$\mathcal{TPN}(\leq,\geq)$ $\approx_\mathcal{W}$ $\mathcal{TA}_{syn}(\leq,\geq)$

	Emptiness Problem	Universal Problem
B-$\mathcal{TPN}_\varepsilon$	Decidable [10]	Undecidable

Table 1. Summary of the Results

$w = a_1 \cdots a_n$, $|w| = n$. We also use $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ with $\varepsilon \notin \Sigma$, where ε is the empty word. B^A stands for the set of mappings from A to B. If A is finite and $|A| = n$, an element of B^A is also a vector in B^n. The usual operators $+, -, <$ and $=$ are used on vectors of A^n with $A = \mathbb{N}, \mathbb{Q}, \mathbb{R}$ and are the point-wise extensions of their counterparts in A. The set $\mathbb{B}$ denotes the boolean values $\{\mathsf{tt}, \mathsf{ff}\}$, $\mathbb{R}_{\geq 0}$ denotes the set of non-negative reals and $\mathbb{R}_{>0} = \mathbb{R}_{\geq 0} \setminus \{0\}$. A *valuation* ν over a set of variables X is an element of $\mathbb{R}_{\geq 0}^X$. For $\nu \in \mathbb{R}_{\geq 0}^X$ and $d \in \mathbb{R}_{\geq 0}$, $\nu + d$ denotes the valuation defined by $(\nu + d)(x) = \nu(x) + d$, and for $X' \subseteq X$, $\nu[X' \mapsto 0]$ denotes the valuation ν' with $\nu'(x) = 0$ for $x \in X'$ and $\nu'(x) = \nu(x)$ otherwise. $\mathbf{0}$ denotes the valuation s.t. $\forall x \in X, \nu(x) = 0$. An *atomic constraint* is a formula of the form $x \bowtie c$ for $x \in X$, $c \in \mathbb{Q}_{\geq 0}$ and $\bowtie \in \{<, \leq, \geq, >\}$. We denote $\mathcal{C}(X)$ the set of *constraints* over a set of variables X which consists of the conjunctions of atomic constraints. Given a constraint $\varphi \in \mathcal{C}(X)$ and a valuation $\nu \in \mathbb{R}_{\geq 0}^X$, we denote $\varphi(\nu) \in \mathbb{B}$ the truth value obtained by substituting each occurrence of x in φ by $\nu(x)$.

2.1 Timed languages and Timed Transition Systems

Let Σ be a fixed finite alphabet s.t. $\varepsilon \notin \Sigma$. A is a finite set that can contain ε.

Definition 1 (Timed Words). *A* timed word w *over Σ is a finite or infinite sequence $w = (a_0, d_0)(a_1, d_1) \cdots (a_n, d_n) \cdots$ s.t. for each $i \geq 0$, $a_i \in \Sigma$, $d_i \in \mathbb{R}_{\geq 0}$ and $d_{i+1} \geq d_i$.*

A timed word $w = (a_0, d_0)(a_1, d_1) \cdots (a_n, d_n) \cdots$ over Σ can be viewed as a pair $(v, \tau) \in \Sigma^\infty \times \mathbb{R}_{\geq 0}^\infty$ s.t. $|v| = |\tau|$. The value d_k gives the absolute time (considering the initial instant is 0) of the action a_k.

We write $\mathit{Untimed}(w) = a_0 a_1 \cdots a_n \cdots$ for the untimed part of w, and $\mathit{Duration}(w) = \sup_{d_k \in \tau} d_k$ for the duration of the timed word w.

A *timed language* L over Σ is a set of timed words.

Definition 2 (Timed Transition System). *A* timed transition system (TTS) *over the set of actions A is a tuple $S = (Q, Q_0, A, \longrightarrow, F, R)$ where Q is a set of states, $Q_0 \subseteq Q$ is the set of initial states, A is a finite set of actions disjoint* *from* $\mathbb{R}_{\geq 0}$*,* $\longrightarrow \subseteq Q \times (A \cup \mathbb{R}_{\geq 0}) \times Q$ *is a set of edges. If* $(q, e, q') \in \longrightarrow$*, we also write* $q \xrightarrow{e} q'$*.* $F \subseteq Q$ *and* $R \subseteq Q$ *are respectively the set of* final *and* repeated *states.*

In the case of $q \xrightarrow{d} q'$ with $d \in \mathbb{R}_{\geq 0}$, d denotes a delay and not an absolute time. We assume that in any TTS there is a transition $q \xrightarrow{0} q'$ and in this case $q = q'$. A *run* ρ of length $n \geq 0$ is a finite ($n < \omega$) or infinite ($n = \omega$) sequence of alternating time and discrete transitions of the form

$$\rho = q_0 \xrightarrow{d_0} q_0' \xrightarrow{a_0} q_1 \xrightarrow{d_1} q_1' \xrightarrow{a_1} \cdots q_n \xrightarrow{d_n} q_n' \cdots$$

We write $\mathit{first}(\rho) = q_0$. We assume that a finite run ends with a time transition d_n. If ρ ends with d_n, we let $\mathit{last}(\rho) = q_n'$ and write $q_0 \xrightarrow{d_0 a_0 \cdots d_n} q_n'$. We write $q \xrightarrow{*} q'$ if there is run ρ s.t. $\mathit{first}(\rho) = q_0$ and $\mathit{last}(\rho) = q'$. The *trace* of an infinite run ρ is the timed word $\mathit{trace}(\rho) = (a_{i_0}, d_0 + \cdots + d_{i_0}) \cdots (a_{i_k}, d_0 + \cdots + d_{i_k}) \cdots$ that consists of the sequence of letters of $A \setminus \{\varepsilon\}$. If ρ is a finite run, we define the trace of ρ by $\mathit{trace}(\rho) = (a_{i_0}, d_0 + \cdots + d_{i_0}) \cdots (a_{i_k}, d_0 + \cdots + d_{i_k})$ where the a_{i_k} are in $A \setminus \{\varepsilon\}$.

We define $\mathit{Untimed}(\rho) = \mathit{Untimed}(\mathit{trace}(\rho))$ and $\mathit{Duration}(\rho) = \sum_{d_k \in \mathbb{R}_{\geq 0}} d_k$.

A run is *initial* if $\mathit{first}(\rho) \in Q_0$. A run ρ is *accepting* if *i)* either ρ is a finite initial run and $\mathit{last}(\rho) \in F$ or *ii)* ρ is infinite and there is a state $q \in R$ that appears infinitely often on ρ.

A timed word $w = (a_i, d_i)_{0 \leq i \leq n}$ is *accepted* by S if there is an accepting run of trace w. The *timed language* $\mathcal{L}(S)$ accepted by S is the set of timed words accepted by S.

Definition 3 (Strong Timed Similarity). *Let* $S_1 = (Q_1, Q_0^1, A, \longrightarrow_1, F_1, R_1)$ *and* $S_2 = (Q_2, Q_0^2, A, \longrightarrow_2, F_2, R_2)$ *be two TTS and* $\preceq$ *be a binary relation over* $Q_1 \times Q_2$*. We write* $s \preceq s'$ *for* $(s, s') \in \preceq$*.* $\preceq$ *is a* strong (timed) simulation relation *of* S_1 *by* S_2 *if: 1) if* $s_1 \in F_1$ *(resp.* $s_1 \in R_1$*) and* $s_1 \preceq s_2$ *then* $s_2 \in F_2$ *(resp.* $s_2 \in R_2$*); 2) if* $s_1 \in Q_0^1$ *there is some* $s_2 \in Q_0^2$ *s.t.* $s_1 \preceq s_2$*; 3) if* $s_1 \xrightarrow{d}_1 s_1'$ *with* $d \in \mathbb{R}_{\geq 0}$ *and* $s_1 \preceq s_2$ *then* $s_2 \xrightarrow{d}_2 s_2'$ *for some* s_2'*, and* $s_1' \preceq s_2'$*; 4) if* $s_1 \xrightarrow{a}_1 s_1'$ *with* $a \in A$ *and* $s_1 \preceq s_2$ *then* $s_2 \xrightarrow{a}_2 s_2'$ *and* $s_1' \preceq s_2'$*.*
A TTS S_2 strongly simulates S_1 *if there is a strong (timed) simulation relation of* S_1 *by* S_2*. We write* $S_1 \preceq_{\mathcal{S}} S_2$ *in this case.*

When there is a strong simulation relation $\preceq$ of S_1 by S_2 and $\preceq^{-1}$ is also a strong simulation relation[5] of S_2 by S_1, we say that $\preceq$ is a *strong (timed) bisimulation relation* between S_1 and S_2 and use $\approx$ instead of $\preceq$. Two TTS S_1 and S_2 are *strongly (timed) bisimilar* if there exists a strong (timed) bisimulation relation between S_1 and S_2. We write $S_1 \approx_{\mathcal{S}} S_2$ in this case.

Let $S = (Q, Q_0, \Sigma_\varepsilon, \longrightarrow, F, R)$ be a TTS. We define the ε-abstract TTS $S^\varepsilon = (Q, Q_0^\varepsilon, \Sigma, \longrightarrow_\varepsilon, F, R)$ (with no ε-transitions) by:

[5] $s_2 \preceq^{-1} s_1 \iff s_1 \preceq s_2$.

– $q \xrightarrow{d}_\varepsilon q'$ with $d \in \mathbb{R}_{\geq 0}$ iff there is a run $\rho = q \xrightarrow{*} q'$ with $\mathit{Untimed}(\rho) = \varepsilon$ and $\mathit{Duration}(\rho) = d$,
– $q \xrightarrow{a}_\varepsilon q'$ with $a \in \Sigma$ iff there is a run $\rho = q \xrightarrow{*} q'$ with $\mathit{Untimed}(\rho) = a$ and $\mathit{Duration}(\rho) = 0$,
– $Q_0^\varepsilon = \{q \mid \exists q' \in Q_0 \mid \; q' \xrightarrow{*} q \textit{ and } \mathit{Duration}(\rho) = 0 \wedge \mathit{Untimed}(\rho) = \varepsilon\}$.

Definition 4 (Weak Time Similarity). *Let* $S_1 = (Q_1, Q_0^1, \Sigma_\varepsilon, \longrightarrow_1, F_1, R_1)$ *and* $S_2 = (Q_2, Q_0^2, \Sigma_\varepsilon, \longrightarrow_2, F_2, R_2)$ *be two TTS and* $\preceq$ *be a binary relation over* $Q_1 \times Q_2$. $\preceq$ *is a* weak (timed) simulation relation *of* S_1 *by* S_2 *if it is a strong timed simulation relation of* S_1^ε *by* S_2^ε. *A TTS* S_2 weakly simulates S_1 *if there is a weak (timed) simulation relation of* S_1 *by* S_2. *We write* $S_1 \preceq_{\mathcal{W}} S_2$ *in this case.*

When there is a weak simulation relation $\preceq$ of S_1 by S_2 and $\preceq^{-1}$ is also a weak simulation relation of S_2 by S_1, we say that $\preceq$ is a *weak (timed) bisimulation relation* between S_1 and S_2 and use $\approx$ instead of $\preceq$. Two TTS S_1 and S_2 are *weakly (timed) bisimilar* if there exists a weak (timed) bisimulation relation between S_1 and S_2. We write $S_1 \approx_{\mathcal{W}} S_2$ in this case. Note that if $S_1 \preceq_{\mathcal{S}} S_2$ then $S_1 \preceq_{\mathcal{W}} S_2$ and if $S_1 \preceq_{\mathcal{W}} S_2$ then $\mathcal{L}(S_1) \subseteq \mathcal{L}(S_2)$.

2.2 Time Petri Nets

Time Petri Nets (TPN) were introduced in [18] and extend Petri Nets with timing constraints on the firings of transitions. In such a model, a clock is associated with each enabled transition, and gives the elapsed time since the more recent date at which it became enabled. An enabled transition can be fired if the value of its clock belongs to the interval associated with the transition. Furthermore, time can progress only if the enabling duration still belongs to the downward closure of the interval associated with any enabled transition. We consider here a generalized version[6] of TPN with accepting and repeated markings and prove our results for this general model.

Definition 5 (Labeled Time Petri Net). *A* Labeled Time Petri Net $\mathcal{N}$ *is a tuple* $(P, T, \Sigma_\varepsilon, {}^\bullet(.), (.)^\bullet, M_0, \Lambda, I, F, R)$ *where:* P *is a finite set of* places *and* T *is a finite set of* transitions *and* $P \cap T = \emptyset$; Σ *is a finite set of* actions ${}^\bullet(.) \in (\mathbb{N}^P)^T$ *is the* backward *incidence mapping;* $(.)^\bullet \in (\mathbb{N}^P)^T$ *is the* forward *incidence mapping;* $M_0 \in \mathbb{N}^P$ *is the* initial *marking;* $\Lambda : T \to \Sigma_\varepsilon$ *is the* labeling function*;* $I : T \to \mathcal{I}(\mathbb{Q}_{\geq 0})$ *associates with each transition a* firing interval*;* $R \subseteq \mathbb{N}^P$ *is the set of* final markings *and* $F \subseteq \mathbb{N}^P$ *is the set of* repeated markings.

Semantics of Time Petri Nets. A *marking* M of a TPN is a mapping in $\mathbb{N}^P$ and $M(p_i)$ is the number of tokens in place p_i. A transition t is *enabled* in a marking M iff $M \geq {}^\bullet t$. We denote $\mathit{En}(M)$ the set of enabled transitions

[6] This is required to be able to define Büchi timed languages, which is not possible in the original version of TPN of [18].

in M. To decide whether a transition t can be fired we need to know for how long it has been enabled: if this amount of time lies into the interval $I(t)$, t can actually be fired, otherwise it cannot. On the other hand, time can progress only if the enabling duration still belongs to the downward closure of the interval associated with any enabled transition. Let $\nu \in (\mathbb{R}_{\geq 0})^{En(M)}$ be a *valuation* such that each value $\nu(t)$ is the time elapsed since transition t was last enabled. A *configuration* of the TPN $\mathcal{N}$ is a pair (M, ν). An *admissible configuration* of a TPN is a configuration (M, ν) s.t. $\forall t \in En(M), \nu(t) \in I(t)^{\downarrow}$. We let $ADM(\mathcal{N})$ be the set of admissible configurations.
In this paper, we consider the *intermediate semantics* for TPNs, based on [8,5], which is the most common one. The key point in the semantics is to define when a transition is *newly enabled* and one has to reset its clock. Let $\uparrow enabled(t', M, t) \in \mathbb{B}$ be true if t' is *newly enabled* by the firing of transition t from marking M, and false otherwise. The firing of t leads to a new marking $M' = M - {}^\bullet t + t^\bullet$. The fact that a transition t' is newly enabled on the firing of a transition $t \neq t'$ is determined w.r.t. the intermediate marking $M - {}^\bullet t$. When a transition t is fired it is newly enabled whatever the intermediate marking is. Formally this gives:

$$\uparrow enabled(t', M, t) = \big(t' \in En(M - {}^\bullet t + t^\bullet)\big) \wedge \big(t' \notin En(M - {}^\bullet t) \vee (t = t')\big) \quad (1)$$

Definition 6 (Semantics of TPN). *The semantics of a TPN $\mathcal{N} = (P, T, \Sigma_\varepsilon, {}^\bullet(.), (.)^\bullet, M_0, \Lambda, I, F, R)$ is a timed transition system $S_{\mathcal{N}} = (Q, \{q_0\}, T, \rightarrow, F', R')$ where: $Q = ADM(\mathcal{N})$, $q_0 = (M_0, \mathbf{0})$, $F' = \{(M, \nu) \mid M \in F\}$ and $R = \{(M, \nu) \mid M \in R\}$, and $\longrightarrow \in Q \times (T \cup \mathbb{R}_{\geq 0}) \times Q$ consists of the discrete and continuous transition relations:* i) *the discrete transition relation is defined $\forall t \in T$ by:*

$$(M, \nu) \xrightarrow{\Lambda(t)} (M', \nu') \text{ iff } \begin{cases} t \in En(M) \wedge M' = M - {}^\bullet t + t^\bullet \\ \nu(t) \in I(t), \\ \forall t \in \mathbb{R}_{\geq 0}^{En(M')}, \nu'(t) = \begin{cases} 0 \text{ if } \uparrow enabled(t', M, t), \\ \nu(t) \text{ otherwise.} \end{cases} \end{cases}$$

and ii) *the continuous transition relation is defined $\forall d \in \mathbb{R}_{\geq 0}$ by:*

$$(M, \nu) \xrightarrow{d} (M, \nu') \text{ iff } \begin{cases} \nu' = \nu + d \\ \forall t \in En(M), \nu'(t) \in I(t)^{\downarrow} \end{cases}$$

A run ρ of $\mathcal{N}$ is an initial run of $S_{\mathcal{N}}$. The timed language accepted by $\mathcal{N}$ is $\mathcal{L}(\mathcal{N}) = \mathcal{L}(S_{\mathcal{N}})$.

We simply write $(M, \nu) \xrightarrow{w}$ to emphasize that there is a sequence of transitions w that can be fired in $S_{\mathcal{N}}$ from (M, ν). If $Duration(w) = 0$ we say that w is an *instantaneous firing sequence*. The set of *reachable configurations* of $\mathcal{N}$ is $\mathsf{Reach}(\mathcal{N}) = \{M \in \mathbb{N}^P \mid \exists (M, \nu) \mid (M_0, \mathbf{0}) \xrightarrow{*} (M, \nu)\}$.

2.3 Timed Automata

Definition 7 (Timed Automaton). *A* Timed Automaton $\mathcal{A}$ *is a tuple* $(L, l_0, X, \Sigma_\varepsilon, E, Inv, F, R)$ *where:* L *is a finite set of* locations*;* $l_0 \in L$ *is the* initial location*;* X *is a finite set of positive real-valued* clocks*;* $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ *is a finite set of* actions *and* ε *is the* silent *action;* $E \subseteq L \times \mathcal{C}(X) \times \Sigma_\varepsilon \times 2^X \times L$ *is a finite set of* edges, $e = \langle l, \gamma, a, R, l' \rangle \in E$ *represents an edge from the location* l *to the location* l' *with the guard* γ*, the label* a *and the reset set* $R \subseteq X$*;* $Inv \in \mathcal{C}(X)^L$ *assigns an* invariant *to any location. We restrict the invariants to conjuncts of terms of the form* $x \preceq r$ *for* $x \in X$ *and* $r \in \mathbb{N}$ *and* $\preceq \in \{<, \leq\}$*.* $F \subseteq L$ *is the set of* final locations *and* $R \subseteq L$ *is the set of* repeated locations*.*

Definition 8 (Semantics of a Timed Automaton). *The semantics of a timed automaton* $\mathcal{A} = (L, l_0, C, \Sigma_\varepsilon, E, Act, Inv, F, R)$ *is a timed transition system* $S_\mathcal{A} = (Q, q_0, \Sigma_\varepsilon, \rightarrow, F', R')$ *with* $Q = L \times (\mathbb{R}_{\leq 0})^X$*,* $q_0 = (l_0, \mathbf{0})$ *is the initial state,* $F' = \{(\ell, \nu) \mid \ \ell \in F\}$ *and* $R' = \{(\ell, \nu) \mid \ \ell \in R\}$*, and* $\rightarrow$ *is defined by:* i) *the discrete transitions relation* $(l, v) \xrightarrow{a} (l', v')$ *iff* $\exists (l, \gamma, a, R, l') \in E$ *s.t.* $\gamma(v) = \mathtt{tt}$*,* $v' = v[R \mapsto 0]$ *and* $Inv(l')(v') = \mathtt{tt}$*;* ii) *the continuous transition relation* $(l, v) \xrightarrow{t} (l', v')$ *iff* $l = l'$*,* $v' = v + t$ *and* $\forall\, 0 \leq t' \leq t$*,* $Inv(l)(v + t') = \mathtt{tt}$*.*

A run ρ *of* $\mathcal{A}$ *is an initial run of* $S_\mathcal{A}$*. The timed language accepted by* $\mathcal{A}$ *is* $\mathcal{L}(\mathcal{A}) = \mathcal{L}(S_\mathcal{A})$*.*

2.4 Expressiveness and Equivalence Problems

If B, B' are either TPN or TA, we write $B \approx_\mathcal{S} B'$ (resp. $B \approx_\mathcal{W} B'$) for $S_B \approx_\mathcal{S} S_{B'}$ (resp. $S_B \approx_\mathcal{W} S_{B'}$). Let $\mathcal{C}$ and $\mathcal{C}'$ be two classes of TPNs or TA.

Definition 9 (Expressiveness w.r.t. Timed Language Acceptance). *The class* $\mathcal{C}$ *is* more expressive *than* $\mathcal{C}'$ *w.r.t. timed language acceptance if for all* $B' \in \mathcal{C}'$ *there is a* $B \in \mathcal{C}$ *s.t.* $\mathcal{L}(B) = \mathcal{L}(B')$*. We write* $\mathcal{C}' \leq_\mathcal{L} \mathcal{C}$ *in this case. If moreover there is some* $B \in \mathcal{C}$ *s.t. there is no* $B' \in \mathcal{C}'$ *with* $\mathcal{L}(B) = \mathcal{L}(B')$*, then* $\mathcal{C}' <_\mathcal{L} \mathcal{C}$ *(read "strictly more expressive"). If both* $\mathcal{C}' \leq_\mathcal{L} \mathcal{C}$ *and* $\mathcal{C} \leq_\mathcal{L} \mathcal{C}'$ *then* $\mathcal{C}$ *and* $\mathcal{C}'$ *are equally expressive w.r.t. timed language acceptance, and we write* $\mathcal{C} =_\mathcal{L} \mathcal{C}'$*.*

Definition 10 (Expressiveness w.r.t. Timed Bisimilarity). *The class* $\mathcal{C}$ *is* more expressive *than* $\mathcal{C}'$ *w.r.t. strong (resp. weak) timed bisimilarity if for all* $B' \in \mathcal{C}'$ *there is a* $B \in \mathcal{C}$ *s.t.* $B \approx_\mathcal{S} B'$ *(resp.* $B \approx_\mathcal{W} B'$*). We write* $\mathcal{C}' \leq_\mathcal{S} \mathcal{C}$ *(resp.* $\mathcal{C}' \leq_\mathcal{W} \mathcal{C}$*) in this case. If moreover there is a* $B \in \mathcal{C}$ *s.t. there is no* $B' \in \mathcal{C}'$ *with* $B \approx_\mathcal{S} B'$ *(resp.* $B \approx_\mathcal{W} B'$*), then* $\mathcal{C}' <_\mathcal{S} \mathcal{C}$ *(resp.* $\mathcal{C}' <_\mathcal{W} \mathcal{C}$*). If both* $\mathcal{C}' <_\mathcal{S} \mathcal{C}$ *and* $\mathcal{C} <_\mathcal{S} \mathcal{C}'$ *(resp.* $<_\mathcal{W}$*) then* $\mathcal{C}$ *and* $\mathcal{C}'$ *are equally expressive w.r.t. strong (resp. weak) timed bisimilarity, and we write* $\mathcal{C} \approx_\mathcal{S} \mathcal{C}'$ *(resp.* $\mathcal{C} \approx_\mathcal{W} \mathcal{C}'$*).*

In the sequel we will compare various classes of TPNs and TAs. We recall the following theorem adapted from [10]:

Theorem 1 ([10]). *For any* $\mathcal{N} \in$ *B-*$\mathcal{TPN}_\varepsilon$ *there is a TA* $\mathcal{A}$ *s.t.* $\mathcal{N} \approx_\mathcal{W} \mathcal{A}$*, hence B-*$\mathcal{TPN}_\varepsilon \leq_\mathcal{W} \mathcal{TA}_\varepsilon$*.*

Moreover if $\mathcal{TA}(\leq, \geq)$ is the set of TA with only large constraints, we even have that B-$\mathcal{TPN}(\leq, \geq) \leq_{\mathcal{W}} \mathcal{TA}(\leq, \geq)$.

3 Strict Ordering Results

In this section, we establish some results proving that TPNs are strictly less expressive w.r.t. weak timed bisimilarity than various classes of TA: $\mathcal{TA}(<)$ only including strict constraints and $\mathcal{TA}(\leq)$ only including large constraints.

We first give a lemma stating that "Waiting Cannot Disable Transitions" in TPNs. The proof follows directly from the definitions.

Lemma 1. *Let (M, ν) be a marking of a TPN. If $(M, \nu) \xrightarrow{t_1 t_2 \cdots t_k}$ with $t_1 t_2 \cdots t_k$ an instantaneous firing sequence and $(M, \nu) \xrightarrow{d} (M_d, \nu_d)$ for some $d \geq 0$, then $(M_d, \nu_d) \xrightarrow{t_1 t_2 \cdots t_k}$.*

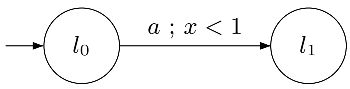


Fig. 1. The Timed Automaton $\mathcal{A}_0$

Theorem 2. *There is no TPN weakly timed bisimilar to $\mathcal{A}_0 \in \mathcal{TA}(<)$ (Fig. 1).*

Proof. Assume there is a TPN $\mathcal{N}$ that is weakly timed bisimilar to $\mathcal{A}_0$ and let $\approx$ be a weak timed bisimulation between $S_{\mathcal{N}}$ and $S_{\mathcal{A}_0}$. Let $(M_0, \mathbf{0})$ be the initial state of $S_{\mathcal{N}}$ and $(l_0, \mathbf{0})$ the initial state of $S_{\mathcal{A}_0}$. In $S_{\mathcal{A}_0}$ there is a run of duration 1 and thus there is a run $(M_0, \mathbf{0}) \xrightarrow{\varepsilon^{i_0} d_1 \varepsilon^{i_1} d_2 \varepsilon^{i_2} \cdots d_n \varepsilon^{i_n}} (M_1, \nu_1)$ in $S_{\mathcal{N}}$, with $i_k \geq 1$ for $1 \leq k \leq n-1$, $i_0 \geq 0$, $i_n \geq 0$ and $\sum_{1 \leq k \leq n} d_k = 1$. We can further assume $d_k > 0$ for all k, and equally $i_n = 0$ because the configuration reached after d_n is also bisimilar to $(l_0, \nu(x) = 1)$. Then $(M_0, \mathbf{0}) \xrightarrow{\varepsilon^{i_0} d_1 \varepsilon^{i_1} d_2 \varepsilon^{i_2} \cdots d_{n-1} \varepsilon^{i_{n-1}}} (M', \nu')$. (M', ν') is bisimilar to a configuration $(\ell_0, \nu(x) = d')$ with $d' < 1$. This entails that $(M', \nu') \xrightarrow{\varepsilon^* a}$. As $(M', \nu') \xrightarrow{d_n} (M_1, \nu_1)$, it follows that $(M_1, \nu_1) \xrightarrow{\varepsilon^* a}$ contradicting the fact that $(M_1, \nu_1) \approx (\ell_0, \nu(x) = 1)$ from which no a can be fired. □

A similar theorem holds for a TA $\mathcal{A}_1$ with large constraints. Let $\mathcal{A}_1$ be the automaton $\mathcal{A}_0$ with the strict constraint $x < 1$ replaced by $x \leq 1$.

Theorem 3. *There is no TPN weakly timed bisimilar to $\mathcal{A}_1 \in \mathcal{TA}(\leq)$.*

Proof. Let $\mathcal{A}_1$ be the automaton $\mathcal{A}_0$ with the strict constraint $x < 1$ replaced by $x \leq 1$. It is clear that $(\ell_0, \mathbf{0}) \xrightarrow{1} (\ell_0, 1)$ and thus $(M_0, \mathbf{0}) \xrightarrow{1}_\varepsilon (M_1, \nu_1)$ and $(\ell_0, 1)$ and (M_1, ν_1) are weakly timed bisimilar. As a can be fired from $(\ell_0, 1)$ all the configurations (M_1', ν_1') reachable from (M_1, ν_1) in null duration (ε transitions) can fire an instantaneous sequence labelled a. Also there must be one such configuration (M', ν') s.t. some duration $d > 0$ can elapse from (M', ν') reaching (M'', ν''). By lemma 1, some instantaneous sequence labelled by a can be fired from (M'', ν''). But (M'', ν'') is weakly timed bisimilar to the configuration $(\ell_0, 1 + d)$ which prevents an a to be fired. Hence a contradiction. □

The previous theorems entail B-$\mathcal{TPN}_\varepsilon <_\mathcal{W} \mathcal{TA}(<)$ and B-$\mathcal{TPN}_\varepsilon <_\mathcal{W} \mathcal{TA}(\leq)$ and as a consequence:

Corollary 1. *B-$\mathcal{TPN}_\varepsilon <_\mathcal{W} \mathcal{TA}_\varepsilon$.*

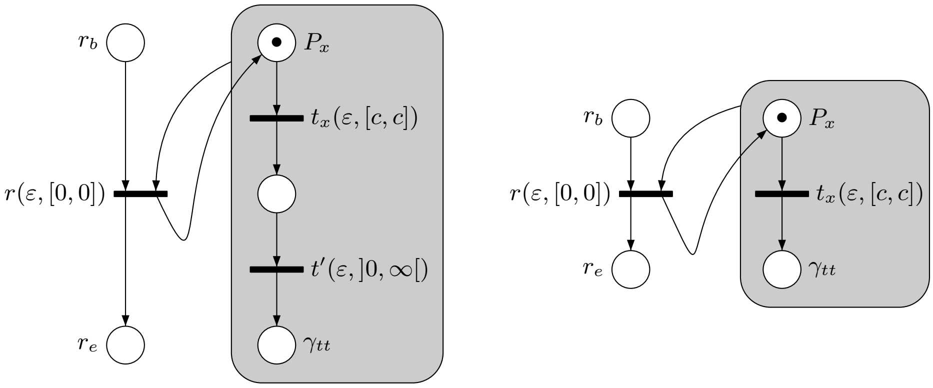
To be fair, one should notice that actually the class of bounded TPNs is strictly less expressive than $\mathcal{TA}(\leq)$ and $\mathcal{TA}(<)$ but also that, obviously unbounded TPNs are more expressive than TA (because they are Turing powerful). Anyway the interesting question is the comparison between bounded TPNs and TA.
Following these negative results, we compare the expressiveness of TPNs and TA w.r.t. to Timed Language Acceptance and then characterize a subclass of TA that admits bisimilar TPNs without strict constraints.

4 Equivalence w.r.t. Timed Language Acceptance

In this section, we prove that TA and labeled TPNs are equally expressive w.r.t. timed language acceptance, and give an effective syntactical translation from TA to TPNs. Let $\mathcal{A} = (L, l_0, X, \Sigma_\varepsilon, E, Act, Inv, F, R)$ be a TA. As we are concerned in this section with the language accepted by $\mathcal{A}$ we assume the invariant function Inv is uniformly true. Let $\mathcal{C}_x$ be the set of atomic constraints on clock x that are used in $\mathcal{A}$. The Time Petri Net resulting from our translation is built from "elementary blocks" modeling the truth value of the constraints of $\mathcal{C}_x$. Then we link them with other blocks for resetting clocks.

Encoding Atomic Constraints. Let $\varphi \in \mathcal{C}_x$ be an atomic constraint on x. From φ, we define the TPN $\mathcal{N}_\varphi$, given by the widgets of Fig. 2 ((a) and (b)) and Fig. 3. In the figures, a transition is written $t(\sigma, I)$ where t is the name of the transition, $\sigma \in \Sigma_\varepsilon$ and $I \in \mathcal{I}(\mathbb{Q}_{\geq 0})$.

To avoid drawing too many arcs, we have adopted the following semantics: the grey box is seen as a macro place; an arc from this grey box means that there are as many copies of the transition as places in the grey box. For instance the TPN of Fig. 2.(b) has 2 copies of the target transition r: one with input places P_x and r_b and output places r_e and P_x and another fresh copy of r with input places r_b and γ_{tt} and output places r_e and P_x. Note that in the widgets of Fig. 3 we put a token in γ_{tt} when firing r only on the copy of r with input place P_i (otherwise the number of tokens in place γ_{tt} could be unbounded).



(a) Widget $\mathcal{N}_{x>c}$ (b) Widget $\mathcal{N}_{x \geq c}$ (with $c > 0$)

Fig. 2. Widgets for $\mathcal{N}_{x>c}$ and $\mathcal{N}_{x \geq c}$

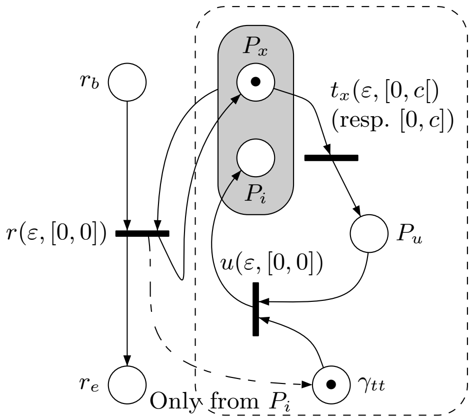


Fig. 3. Widget $\mathcal{N}_{x<c}$ (resp. $\mathcal{N}_{x \leq c}$)

Also we assume that the automaton $\mathcal{A}$ has no constraint $x \geq 0$ (as it evaluates to true they can be safely removed) and thus that the widget of Fig. 2.(b) only appears with $c > 0$. Each of these TPNs basically consists of a "constraint" subpart (in the grey boxes for Fig. 2 and in the dashed box for Fig. 3) that models the truth value of the atomic constraint, and another "reset" subpart that will be used to update the truth value of the constraint when the clock x is reset.

The "constraint" subpart features the place γ_{tt}: the intended meaning is that when a token is available in this place, the corresponding atomic constraint φ is true.

When a clock x is reset, all the grey blocks modeling an x-constraint must be set to their *initial* marking which has one token in P_x for Fig. 2 and one token in P_x and γ_{tt} for Fig. 3. Our strategy to reset a block modeling a constraint is to put a token in the r_b place (r_b stands for "reset begin"). Time cannot elapse from there on (strong semantics for TPNs), as there will be a token in one of the places of the grey block and thus transition r will be enabled.

Resetting Clocks. In order to reset all the blocks modeling constraints on a clock x, we chain all of them in some arbitrary order, the r_e place of the i^{th} block is linked to the r_b place of the $i+1^{th}$ block, *via* a 0 time unit transition ε. This is illustrated in Fig. 4 for clocks x_1 and x_n. Assume $R \subseteq X$ is a non empty set of clocks. Let $D(R)$ be the set of atomic constraints that are in the scope of R (the clock of the constraint is in R). We write $D(R) = \{\varphi_1^{x_1}, \varphi_2^{x_1}, \cdots, \varphi_{q_1}^{x_1}, \cdots, \varphi_{q_n}^{x_n}\}$ where $\varphi_i^{x_j}$ is the i^{th} constraints of the clock x_j. To update all the widgets of $D(R)$, we connect the reset chains as described on Fig. 4. The picture inside the dashed box denotes the widget $\mathcal{N}_{Reset(R)}$. We denote by $r_b(R)$ the first place of this widget and $r_e(R)$ the last one. To update the (truth value of the) widgets of $D(R)$ it then suffices to put a token in $r_b(R)$. In null duration it will go to $r_e(R)$ and have the effect of updating each widget of $D(R)$ on its way.

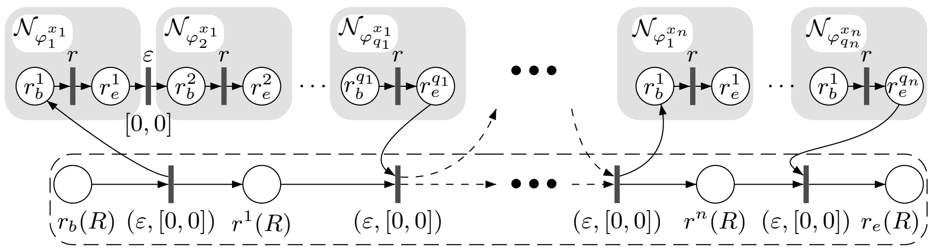


Fig. 4. Widget $\mathcal{N}_{Reset(R)}$ to reset the widgets of the constraints of clocks x_i, $1 \leq i \leq n$

The Complete Construction. First we create fresh places P_ℓ for each $\ell \in L$. Then we build the widgets $\mathcal{N}_\varphi$, for each atomic constraint φ that appears in $\mathcal{A}$. Finally for each $R \subseteq X$ s.t. there is an edge $e = (\ell, \gamma, a, R, \ell') \in E$ we build a reset widget $\mathcal{N}_{Reset(R)}$. Then for each edge $(\ell, \gamma, a, R, \ell') \in E$ with $\gamma = \wedge_{i=1,n}\varphi_i$ and $n \geq 0$ we proceed as follows:

1. assume $\gamma = \wedge_{i=1,n}\varphi_i$ and $n \geq 0$,
2. create a transition $f(a, [0, \infty[)$ and if $n \geq 1$ another one $r(\varepsilon, [0, 0])$,
3. connect them to the places of the widgets $\mathcal{N}_{\varphi_i}$ and $\mathcal{N}_{Reset(R)}$ as described on Fig. 5. In case $\gamma = \mathtt{tt}$ (or $n = 0$) there is only one input place to $f(a, [0, \infty[)$ which is P_ℓ. In case $R = \emptyset$ there is no transition $r(\varepsilon, [0, 0])$ and the output place of $f(a, [0, \infty[)$ is $P_{\ell'}$.

To complete the construction we just need to put a token in the place P_{ℓ_0} if ℓ_0 is the initial location of the automaton, and set each widget $\mathcal{N}_\varphi$ to its initial marking, for each atomic constraint φ that appears in $\mathcal{A}$, and this defines the initial marking M_0. The set of final markings is defined by the set of markings M s.t. $M(P_\ell) = 1$ for $\ell \in F$ and the set of repeated markings by the set of markings M s.t. $M(P_\ell) = 1$ for $\ell \in R$. We denote $\Delta(\mathcal{A})$ the TPN obtained as described previously. Notice that by construction 1) $\Delta(\mathcal{A})$ is 1-safe and moreover 2) in each reachable marking M of $\Delta(\mathcal{A})$ $\left(\sum_{\ell \in L} M(P_\ell)\right) \leq 1$. A widget related to an atomic constraint has a linear size w.r.t. its size, a clock resetting widget has a linear size w.r.t. the number of atomic constraints of the clock and a widget associated with an edge has a linear size w.r.t. its description size. Thus the size of $\Delta(\mathcal{A})$ is linear w.r.t. the size of $\mathcal{A}$ improving the quadratic complexity of the (restricted) translation in [15]. Finally, to prove $\mathcal{L}(\Delta(\mathcal{A})) = \mathcal{L}(\mathcal{A})$ we build two simulation relations $\preceq_1$ and $\preceq_2$ s.t. $\Delta(\mathcal{A}) \preceq_1 \mathcal{A}$ and $\mathcal{A} \preceq_2 \Delta(\mathcal{A})$. The complete proof is given in appendix A.

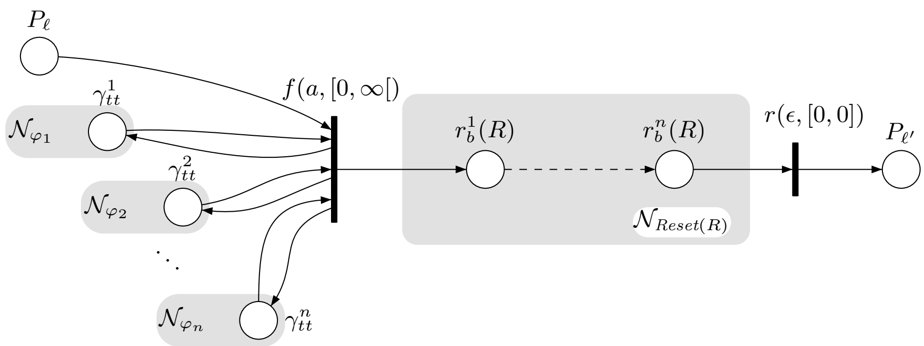


Fig. 5. Widget $\mathcal{N}_e$ of an edge $e = (\ell, \gamma, a, R, \ell')$

New Results for TPNs.

Corollary 2. *The classes B-$\mathcal{TPN}_\varepsilon$ and $\mathcal{TA}_\varepsilon$ are equally expressive w.r.t. timed language acceptance, i.e. B-$\mathcal{TPN}_\varepsilon =_{\mathcal{L}} \mathcal{TA}_\varepsilon$.*

Proof. From Theorem 1, we know that B-$\mathcal{TPN}_\varepsilon \leq_\mathcal{L} \mathcal{TA}_\varepsilon$. Proposition 1 proves that $\mathcal{TA}_\varepsilon \leq_\mathcal{L}$ B-$\mathcal{TPN}_\varepsilon$ and hence B-$\mathcal{TPN}_\varepsilon =_\mathcal{L} \mathcal{TA}$. □

Corollary 3. 1-*B*-$\mathcal{TPN}_\varepsilon =_\mathcal{L}$ *B*-$\mathcal{TPN}_\varepsilon$.

Proof. Let $T \in$ B-$\mathcal{TPN}_\varepsilon$. We use Theorem 1 and thus there is a TA A_T s.t. $\mathcal{L}(T) = \mathcal{L}(A_T)$ which can effectively be built. From A_T we use Proposition 1 and obtain $\Delta(A_T)$ (again effective) which is a 1-safe TPN. □

From the well-known result of Alur & Dill [3] and as our construction is effective, it follows that:

Corollary 4. *The universal language problem is undecidable for B-$\mathcal{TPN}_\varepsilon$ (and already for* 1*-B-$\mathcal{TPN}_\varepsilon$).*

5 Equivalence w.r.t. Timed Bisimilarity

In this section, we consider the class B-$\mathcal{TPN}(\leq,\geq)$ of TPNs without strict constraints, *i.e.* the original version of Merlin [18]. First recall that starting with a TPN $\mathcal{N} \in$ B-$\mathcal{TPN}(\leq,\geq)$, the translation from TPN to TA proposed in [10] gives a TA $\mathcal{A}$ with the following features:

– guards are of the form $x \geq c$ and invariants have the form $x \leq c$;
– between two resets of a clock x, the atomic constraints of the invariants over x are *increasing i.e.* the sequence of invariants encountered from any location is of the form $x \leq c_1$ and later on $x \leq c_2$ with $c_2 \geq c_1$ etc.

Let us now consider the syntactical subclass $\mathcal{TA}_{syn}(\leq,\geq)$ of TA defined by:

Definition 11. *The subclass $\mathcal{TA}_{syn}(\leq,\geq)$ of TA is defined by the set of TA of the form $(L, l_0, X, \Sigma_\varepsilon, E, Inv, F, R)$ where :*

– *guards are conjunctions of atomic constraints of the form $x \geq c$ and invariants are conjunction of atomic constraints $x \leq c$.*
– *the invariants satisfy the following property; $\forall e = (\ell, \gamma, a, R, \ell') \in E$, if $x \notin R$ and $x \leq c$ is an atomic constraint in $Inv(\ell)$, then if $x \leq c'$ is $Inv(\ell')$ for some c' then $c' \geq c$.*

We now adapt the construction of section 4 to define a translation from $\mathcal{TA}_{syn}(\leq,\geq)$ to B-$\mathcal{TPN}(\leq,\geq)$ preserving timed bisimulation. The widget $\mathcal{N}_{x\leq c}$ is modified as depicted in figure Fig. 6.(a). The widgets $\mathcal{N}_{x\geq c}$ and $\mathcal{N}_{reset}(R)$ are those of section 4 respectively in figures Fig. 2.(b) and Fig. 4.

The construction. As in section 4, we create a place P_ℓ for each location $\ell \in L$. Then we build the blocks $\mathcal{N}_\varphi$ for each atomic constraints $\varphi = x \geq c$ (Fig. 2.(b)) that appears in guards of $\mathcal{A}$ and we build the blocks $\mathcal{N}_\mathcal{I}$ for each atomic constraints $\mathcal{I} = x \leq c$ (Fig.6.(a)) that appears in an invariant of $\mathcal{A}$. Finally for each $R \subseteq X$ s.t. there is an edge $e = (\ell, \gamma, a, R, \ell') \in E$ we build a reset widget $\mathcal{N}_{Reset(R)}$ (Fig. 4). Then for each edge $(\ell, \gamma, a, R, \ell') \in E$ with $\gamma = \wedge_{i=1,n}\varphi_i$ and $n \geq 0$, we proceed exactly as in section 4 (Fig. 5). For each location $\ell \in L$ with $Inv(\ell) = \wedge_{k=1,n}\mathcal{I}_k$, we proceed as follows:

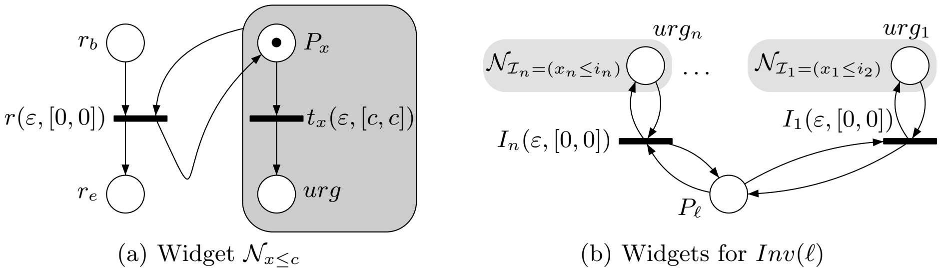


Fig. 6. Widget $\mathcal{N}_e$ of an edge $e = (\ell, \gamma, a, R, \ell')$

1. if $n \ge 1$, create a transition $I_k(\varepsilon, [0, 0])$ for $1 \le k \le n$;
2. for $1 \le k \le n$ connect $I_k(\varepsilon, [0, 0])$ to P_ℓ and to the place urg of block $\mathcal{N}_{\mathcal{I}_k}$, as depicted in figure Fig. 6.(b).

Let $\mathcal{A} = (L, \ell_0, X, \Sigma_\varepsilon, E, Inv, F, R)$ and assume that the set of atomic constraints of $\mathcal{A}$ is $\mathcal{C}_\mathcal{A} = \mathcal{C}_\mathcal{A}(\ge) \cup \mathcal{C}_\mathcal{A}(\le)$ where $\mathcal{C}_\mathcal{A}(\bowtie)$ is the set of atomic constraints $x \bowtie c$, $\bowtie \in \{\le, \ge\}$, of $\mathcal{A}$ and $X = \{x_1, \cdots, x_k\}$.

We denote $\Delta^+(\mathcal{A}) = (P, T, \Sigma_\varepsilon, {}^\bullet(.), (.)^\bullet, M_0, \Lambda, I, F_\Delta, R_\Delta)$ the TPN built as described previously. The place P_x and the transition t_x of a widget $\mathcal{N}_\varphi$ for $\varphi \in \mathcal{C}_\mathcal{A}$ are respectively written P_x^φ and t_x^φ in the sequel. Moreover, for a constraint $\varphi = x \ge c$, the place γ_{tt} of a widget $\mathcal{N}_\varphi$ is written γ_{tt}^φ and the place urg of a widget $\mathcal{N}_\varphi$ is written urg^φ. We can now build a bisimulation relation $\approx$ between $\mathcal{A}$ and $\Delta^+(\mathcal{A})$: the proof is given in appendix B.

New Results for TPNs. From the previous result of appendix B we can state the following corollaries:

Corollary 5. *The classes B-$\mathcal{TPN}(\le, \ge)$ and $\mathcal{TA}_{syn}(\le, \ge)$ are equally expressive w.r.t. weak timed bisimulation, i.e. B-$\mathcal{TPN}(\le, \ge) \approx_\mathcal{W} \mathcal{TA}_{syn}(\le, \ge)$.*

Proof. Let $\mathcal{A} \in \mathcal{TA}_{syn}(\le, \ge)$. From the previous construction and proposition 2, page 21, we obtain $\Delta_1^+(\mathcal{A}) \in$ B-$\mathcal{TPN}(\le, \ge)$ with $\mathcal{A} \approx_\mathcal{W} \Delta_1^+(\mathcal{A})$. Let $\mathcal{N} \in$ B-$\mathcal{TPN}(\le, \ge)$, from Theorem 1 (in [10]), we obtain $\Delta_2^+(\mathcal{N}) \in \mathcal{TA}_{syn}(\le, \ge)$ with $\mathcal{N} \approx_\mathcal{W} \Delta_2^+(\mathcal{N})$.

Corollary 6. *The classes* 1*-B-$\mathcal{TPN}(\le, \ge)$ and B-$\mathcal{TPN}(\le, \ge)$ are equally expressive w.r.t. timed bisimulation* i.e. 1*-B-$\mathcal{TPN}(\le, \ge) \approx_\mathcal{W}$ B-$\mathcal{TPN}(\le, \ge)$.*

Proof. Let $\mathcal{N} \in$ B-$\mathcal{TPN}(\le, \ge)$. From Theorem 1, there exists a TA $A_\mathcal{N} \in \mathcal{TA}_{syn}(\le, \ge)$ s.t. $\mathcal{N} \approx_\mathcal{W} A_\mathcal{N}$. From the previous construction and proposition 2, we obtain $\Delta^+(A_\mathcal{N}) \in$ 1-B-$\mathcal{TPN}_\varepsilon$ s.t. $A_\mathcal{N} \approx_\mathcal{W} \Delta^+(A_\mathcal{N})$ then $\mathcal{N} \approx_\mathcal{W} \Delta^+(A_\mathcal{N})$.

6 Conclusion

In this paper, we have investigated different questions relative to the expressiveness of TPNs. First, we have shown that TA and bounded TPNs (strict constraints are permitted) are equivalent w.r.t. timed language equivalence. We have also provided an effective construction of a TPN equivalent to a TA. This enables us to prove that the universal language problem is undecidable for TPNs. Then we have addressed the expressiveness problem for weak time bisimilarity. We have proved that TA are strictly more expressive than bounded TPNs and given a subclass of TA expressively equivalent to TPN "à la Merlin".

Further work will consist in characterizing exactly the subclass of TA equivalent to TPN w.r.t. timed bisimilarity.

References

1. P.A. Abdulla and A. Nylén. Timed Petri nets and BQOs. In *ICATPN'01*, volume 2075 of *LNCS*, pages 53–72. Springer-Verlag, june 2001.
2. L. Aceto and F. Laroussinie. Is Your Model Checker on Time? On the Complexity of Model Checking for Timed Modal Logics. *Journal of Logic and Algebraic Programming*, volume 52-53, pages 7-51. Elsevier Science Publishers, august 2002.
3. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science B*, 126:183–235, 1994.
4. R. Alur and L. Fix and T.A. Henzinger. Event-Clock Automata: A Determinizable Class of Timed Automata. *Theoretical Computer Science*, 211:253–273, 1999.
5. T. Aura and J. Lilius. A causal semantics for time Petri nets. *Theoretical Computer Science*, 243(1–2):409–447, 2000.
6. B. Berthomieu, P.-O. Ribet and F. Vernadat. The tool TINA – Construction of Abstract State Spaces for Petri Nets and Time Petri Nets. *International Journal of Production Research*, 4(12), July 2004.
7. B. Bérard, F. Cassez, S. Haddad, D. Lime and O.H. Roux. Comparison of the Expressiveness of Timed Automata and Time Petri Nets. Research Report IRCCyN R2005-2 available at http://www.lamsade.dauphine.fr/~haddad/publis.html 2005.
8. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Transactions on Software Engineering*, 17(3):259–273, March 1991.
9. M. Boyer and M. Diaz. Non equivalence between time Petri nets and time stream Petri nets. In *Proceedings of 8th International Workshop on Petri Nets and Performance Modeling (PNPM'99)*, Zaragoza, Spain, pages 198–207.
10. F. Cassez and O.H. Roux. Structural Translation of Time Petri Nets into Timed Automata. In Michael Huth, editor, *Workshop on Automated Verification of Critical Systems (AVoCS'04)*, Electronic Notes in Computer Science. Elsevier, August 2004.
11. D. de Frutos Escrig, V. Valero Ruiz, and O. Marroquín Alonso. Decidability of properties of timed-arc Petri nets. In *ICATPN'00*, Aarhus, Denmark, volume 1825 of *LNCS*, pages 187–206, June 2000.
12. M. Diaz and P. Senac. Time stream Petri nets: a model for timed multimedia information. In *ATPN'94*, volume 815 of *LNCS*, pages 219–238, 1994.

13. D.L. Dill. Timing assumptions and verification of finite-state concurrent systems. In Proc. Workshop on Automatic Verification Methods for Finite State Systems, Grenoble, volume 407 of *LNCS*, 1989.
14. G. Gardey, D. Lime, M. Magin and O.H. Roux. ROMÉO: A Tool for Analyzing time Petri nets In *CAV '05*, Edinburgh, Scotland, UK, volume 3576 of *LNCS*, 2005, pages 418–423.
15. S. Haar, F. Simonot-Lion, L. Kaiser, and J. Toussaint. Equivalence of Timed State Machines and safe Time Petri Nets. In *Proceedings of WODES 2002*, Zaragoza, Spain, pages 119–126.
16. W. Khansa, J.P. Denat, and S. Collart-Dutilleul. P-Time Petri Nets for manufacturing systems. In *WODES'96*, Scotland, pages 94–102, 1996.
17. D. Lime and O.H. Roux. State class timed automaton of a time Petri net. In *PNPM'03*. IEEE Computer Society, September 2003.
18. P.M. Merlin. *A study of the recoverability of computing systems*. PhD thesis, University of California, Irvine, CA, 1974.
19. M. Pezzé and M. Young. Time Petri Nets: A Primer Introduction. Tutorial presented at the Multi-Workshop on Formal Methods in Performance Evaluation and Applications, Zaragoza, Spain, September 1999.
20. C. Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1974.
21. J. Sifakis. Performance Evaluation of Systems using Nets. In *Net Theory and Applications, Advanced Course on General Net Theory of Processes and Systems, Hamburg*, volume 84 of *LNCS*, pages 307–319, 1980.

A Proof of $\mathcal{L}(\Delta(\mathcal{A})) = \mathcal{L}(\mathcal{A})$

Proposition 1. *If $\Delta(\mathcal{A})$ is defined as in section 4, then $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\Delta(\mathcal{A}))$.*

Proof. The proof works as follows: we first show that $\Delta(\mathcal{A})$ weakly simulates $\mathcal{A}$ which implies $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\Delta(\mathcal{A}))$. Then we show that we can define a TA $\mathcal{A}'$ s.t. $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$ and $\mathcal{A}'$ weakly simulates $\Delta(\mathcal{A})$ which entails $\mathcal{L}(\Delta(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. It is sufficient to give the proof for the case $\mathcal{A}$ has no ε transitions. In case $\mathcal{A}$ has ε transitions we rename them with a fresh letter $\mu \notin \Sigma_\varepsilon$ and obtain an automaton $\mathcal{A}_\mu$ with no ε transitions. We apply our construction to $\mathcal{A}_\mu$ and obtain a TPN in which we replace every label μ by ε.

Let $\mathcal{A} = (L, l_0, C, A, E, Act, Inv, F, R)$ and $\Delta(\mathcal{A}) = (P, T, A_\varepsilon, {}^\bullet(.), (.)^\bullet, M_0, \Lambda, \Gamma, F_\Delta, R_\Delta)$. Assume $C = \{x_1, \cdots, x_k\}$, $P = \{p_1, \cdots, p_m\}$ and $T = \{t_1, \cdots, t_n\}$. We assume that the set of atomic constraints of $\mathcal{A}$ is $\mathcal{C}_\mathcal{A}$. Each place γ_{tt} of a widget $\mathcal{N}_{x \bowtie c}$ (for $x \bowtie c$ an atomic constraint of $\mathcal{A}$) is denoted $\gamma_{tt}^{x \bowtie c}$.

Proof that $\Delta(\mathcal{A})$ simulates $\mathcal{A}$. We define the relation $\preceq \subseteq (L \times \mathbb{R}^n_{\geq 0}) \times (\mathbb{N}^p \times \mathbb{R}^m_{\geq 0})$ by:

$$(\ell, v) \preceq (M, \nu) \iff \begin{cases} (1)\, M(P_\ell) = 1 \\ (2)\, \textit{for each } \varphi = x \bowtie c, \bowtie \in \{<, \leq\},\ M(P_u) = 0 \\ (3)\, \textit{for each } \varphi \in \mathcal{C}_\mathcal{A},\ v \in [\![\varphi]\!] \iff M(\gamma_{tt}^{\varphi}) = 1 \end{cases} \tag{I}$$

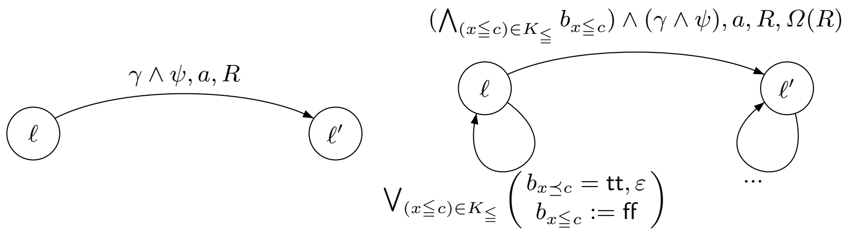
Now we can prove that $\preceq$ is a weak simulation relation of $\mathcal{A}$ by $\Delta(\mathcal{A})$:

1. final and repeated states: by definition of $\Delta(\mathcal{A})$ and the definition of $\preceq$;
2. initial states: it is clear that $(l_0, \mathbf{0}) \preceq (M_0, \mathbf{0})$;
3. continuous transitions: let $(\ell, v) \xrightarrow{d} (\ell, v + d)$. Take (M, ν) s.t. $(\ell, v) \preceq (M, \nu)$. As the widgets $\mathcal{N}_{\varphi_i}$ are non-blocking, time d can elapse from (M, ν), and there is a run $(M, \nu) \xrightarrow{\rho} (M', \nu')$ with $\mathit{Duration}(\mathit{trace}(\rho)) = d$ and $\mathit{Untimed}(\mathit{trace}(\rho)) = \varepsilon$. We can choose ρ without any transitions $f(a, [0, \infty[)$ so that a token remains in P_ℓ and $M'(P_\ell) = 1$. Thus to prove $(\ell, v + t) \preceq (M', \nu')$ it remains to prove items (2) and (3) of equation (I).
Let $\varphi = x \bowtie c$ with $\bowtie \in \{<, \leq\}$.
   – if $\varphi(v) = \mathtt{tt}$ and $\varphi(v + d) = \mathsf{ff}$, then there is some $d' \leq d$ s.t. transition t_x of widget $\mathcal{N}_\varphi$ is enabled and it must be fired before φ becomes false. Thus t_x is fired at d' (which is possible as there is no token in P_u and thus the token is in P_x) and subsequently u in the same widget, thus transferring the tokens from $P_x, \gamma^{\varphi}_{tt}$ to P_i.
   – if $\varphi(v) = \mathtt{tt}$ and $\varphi(v + d) = \mathtt{tt}$, it is possible to do nothing in widget $\mathcal{N}_\varphi$ and let the token in P_x and γ^{φ}_{tt}.
   – if $\varphi(v) = \mathsf{ff}$ then $\varphi(v + d) = \mathsf{ff}$, then there must be a token in P_i and we let time elapse without firing any transition.
Let $\varphi = x \bowtie c$ with $\bowtie \in \{>, \geq\}$.
   – if $\varphi(v) = \mathtt{tt}$ then $\varphi(v + d) = \mathtt{tt}$ and $M(\gamma^{\varphi}_{tt}) = 1$. We just let time elapse in $\mathcal{N}_\varphi$.
   – if $\varphi(v) = \mathsf{ff}$ and $\varphi(v + d) = \mathtt{tt}$, there is $d' \leq d$ s.t. transitions t_x must be fired (and t' can be fired at $d' + \xi$ with $\xi > 0$ for $\mathcal{N}_{x>c}$). We fire those transitions at d' and let $d - d'$ elapse.
   – if $\varphi(v) = \mathsf{ff}$ and $\varphi(v + d) = \mathsf{ff}$ we also let time elapse and leave a token in P_x.
This way for each constraint $\varphi = x \bowtie c$, there is a run $\rho_\varphi = (M, \nu) \xrightarrow{d}_{\varepsilon} (M_\varphi, \nu_\varphi)$ s.t. (M_φ, ν_φ) satisfies requirements (2) and (3) of equation (I). Taken separately we have for each constraint $(\ell, v) \preceq (M_\varphi, \nu_\varphi)$. It is not difficult[7] to build a run ρ with an interleaving of the previous runs ρ_φ s.t. $\rho = (M, \nu) \xrightarrow{t}_{\varepsilon} (M', \nu')$ and (M', ν') satisfies requirements (2) and (3) of equation (I) for each constraint φ, and thus $(\ell, v) \preceq (M', \nu')$.
4. discrete transitions: Let $(\ell, v) \xrightarrow{a} (\ell', v')$ and $(\ell, v) \preceq (M, \nu)$. Then there is an edge $e = (\ell, \gamma, a, R, \ell') \in E$ s.t. $\gamma = \wedge_{i=1,n} \varphi_i$, $n \geq 0$ and φ_i is an atomic constraint. By definition 8, $v \in [\![\varphi_i]\!]$ for $1 \leq i \leq n$. This implies $M(\gamma^{\varphi_i}_{tt}) = 1$ (definition of $\preceq$). Thus the transition $f(a, [0, \infty[)$ is fireable in the widget $\mathcal{N}_e$ leading to (M', ν'). From there on we do not change the marking of widgets $\mathcal{N}_{\varphi_i}$ for the constraints φ_i that do not need to be reset (the clock of φ_i is not in R). We also use the widget $\mathcal{N}_{Reset(R)}$ to reset the constraints φ_i with a clock in R and finally put a token in $P_{\ell'}$. The new state (M'', ν'') obtained this way satisfies $(\ell', v') \preceq (M'', \nu'')$.

This completes the proof that $\Delta(\mathcal{A})$ simulates $\mathcal{A}$ and thus $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\Delta(\mathcal{A}))$.

[7] Just find an ordering for all the date d' at which a transition must be fired and fire those transitions in this order with time elapsing between them.

Proof of $\mathcal{L}(\Delta(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$. To prove this, we cannot easily exhibit a simulation of $\Delta(\mathcal{A})$ by $\mathcal{A}$. Indeed, $\Delta(\mathcal{A})$, because of the widgets $\mathcal{N}_{x \bowtie c}$ with $\bowtie \in \{<, \leq\}$, has to make a decision at some point to fire transition t_x and immediately after u, *i.e.* it is as if it decides that $x \bowtie c$ is now false and the transitions with this guard cannot be fired anymore (until they are reset). To use the simulation framework, we build first a TA $\mathcal{A}'$ that accepts the same language as $\mathcal{A}$ but has the capability to sometimes (non deterministically) decide it will not use a transition with a guard $x \bowtie c$ until it is reset. It is then possible to build a simulation relation of $\Delta(\mathcal{A})$ by $\mathcal{A}'$.



(a) Edge $(\ell, \gamma \wedge \psi, a, R, \ell')$ in $\mathcal{A}$ (b) Extended edge in $\mathcal{A}'$.

Fig. 7. From $\mathcal{A}$ to $\mathcal{A}'$.

We denote $\leqq$ for either $\{<, \leq\}$ and $\geqq$ for $\{>, \geq\}$. Let $K_{\leqq}$ be the set of constraints $x \leqq c$ in $\mathcal{A}$. For each $x \leqq c \in K_{\leqq}$ we introduce a boolean variable $b_{x \leqq c}$. Each $b_{x \leqq c}$ is initially true.

We start with $\mathcal{A}' = \mathcal{A}$. The construction of the new features of $\mathcal{A}'$ is depicted on Fig. 7. Let $(\ell, \gamma \wedge \psi, a, R, \ell')$ be an edge of $\mathcal{A}'$ with $\gamma = \wedge_{x \leqq c \in K_{\leqq}} x \leqq c$ and $\psi = \wedge_{x \geqq c \in K_{\geqq}} x \geqq c$. For such an edge we strengthen[8] the guard $\gamma \wedge \psi$ to obtain γ' as follows: $\gamma' = \gamma \wedge \psi \wedge \bigwedge_{x \leqq c \in K_{\leqq}} b_{x \leqq c}$. This way the transition $(\ell, \gamma \wedge \psi, a, R, \ell')$ can be fired in $\mathcal{A}'$ only if the corresponding guard in $\mathcal{A}$ and the conjunction of the $b_{x \leqq c}$ is true as well. We also reset to true all the variables $b_{x \leqq c}$ s.t. $x \in R$ on a transition $(\ell, \gamma \wedge \psi, a, R, \ell')$ and $\Omega(R)$ corresponds to the reset of all $b_{x \leqq c}$ s.t. $x \in R$, $\Omega(R) = \wedge_{x \in R} b_{x \leqq c} := \mathsf{tt}$.

Now let ℓ be location of $\mathcal{A}'$. For each variable $b_{x \leqq c}$ we add a loop edge $(\ell, b_{x \leqq c} = \mathsf{tt}, \varepsilon, b_{x \leqq c} := \mathsf{ff}, \ell)$ in $\mathcal{A}'$, *i.e.* the automaton $\mathcal{A}'$ can decide non deterministically[9] to set $b_{x \leqq c}$ to false if it is true (see Fig. 7). There are as many loops on each location as the number of variables $b_{x \leqq c}$. The new non deterministic TA $\mathcal{A}'$ accepts exactly the same language as $\mathcal{A}$ *i.e.* $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.

---

8 We need an extended type of TA with boolean variables; this does not add any expressive power to the model.

9 This means we add ε transitions to $\mathcal{A}'$; nevertheless the restriction we made at the beginning that $\mathcal{A}$ has no ε transitions is useful when proving that $\Delta(\mathcal{A})$ simulates $\mathcal{A}$ and not required to prove that $\mathcal{A}'$ weakly simulates $\Delta(\mathcal{A})$.

We can now build a simulation relation of $\Delta(\mathcal{A})$ by $\mathcal{A}'$. We denote (ℓ, v, b) a configuration of $\mathcal{A}'$ with b the vector of b_φ variables. We define the relation $\preceq \subseteq (\mathbb{N}^p \times \mathbb{R}^m_{\geq 0}) \times (L \ \times \mathbb{R}^n_{\geq 0} \times \mathbb{B}^k)$ by:

$$(M, \nu) \preceq (\ell, v, b) \iff \begin{cases} (1)\, M(P_\ell) = 1 \\ (2)\, \forall \varphi = x > c \in K_>,\, v \in [\![\varphi]\!] \iff M(\gamma^\varphi_{tt}) = 1 \\ (3)\, \forall \varphi = x \geq c \in K_\geq,\, v \in [\![\varphi]\!] \iff M(\gamma^\varphi_{tt}) = 1 \vee \\ \qquad\qquad (M(P^\varphi_x) = 1 \wedge \nu(t^\varphi_x) = c) \\ (4) \forall \varphi \in K_\preceq, M(P^\varphi_i) = 1 \iff (b_\varphi = \mathsf{ff} \vee v \notin [\![\varphi]\!]) \end{cases} \tag{II}$$

Now we prove that $\preceq$ is a weak simulation relation of $\Delta(\mathcal{A})$ by $\mathcal{A}$.

- property on final and repeated states is satisfied by definition of $\mathcal{A}'$,
- for the initial configuration, it is clear that $(M_0, \mathbf{0}) \preceq (l_0, \mathbf{0}, b_0)$ (in b_0 all the variables b are true),
- continuous time transitions: let $(M, \nu) \xrightarrow{d} (M', \nu')$ with $d \geq 0$. Let $(M, \nu) \preceq (\ell, v, b)$. As there are no invariant in $\mathcal{A}'$ time d can elapse from (ℓ, v, b). If no ε transition fires in the TPN, then all the truth values of the constraints stay unchanged. Thus $(\ell, v, b) \xrightarrow{d} (\ell, v + d, b)$ in $\mathcal{A}'$ s.t. $(M', \nu') \preceq (\ell, v + d, b)$.
- discrete transitions: let $(M, \nu) \xrightarrow{a} (M', \nu')$. We distinguish the cases $a = \varepsilon$ and $a \in \Sigma$.
  If $a = \varepsilon$ then we are updating some widgets $\mathcal{N}_\varphi$ (ε transition is not a reset transition because reset can occur only when $M(P\ell) = 0$)). We split the cases according to the different types of widgets:
  - update of a widget $\mathcal{N}_{x>c}$: either t_x or t' is fired. If t_x is fired then the time elapsed since the x was last reset is equal to c. Thus $M(\gamma_{tt}) = 0$ and $v(x) \leq c$ and $v \notin [\![x > c]\!]$. This implies $(M', \nu') \preceq (\ell, v)$.
    If t' is fired on the contrary, $v'(x) > c$ but again $(M', \nu') \preceq (\ell, v, b)$.
  - update of a widget $\mathcal{N}_{x \geq c}$: the same reasoning as before can be used and leads to $(M', \nu') \preceq (\ell, v, b)$.
  - update of a widget $\mathcal{N}_{x<c}$: In this case either t_x or u is fired. Assume t_x is fired. Thus $M'(P_i) = 0$. The time elapsed since x was last reset is strictly less than c and $v \in [\![\varphi]\!]$. b_φ is true in (ℓ, v, b) as $M(P_i) = 0$. Thus $(M', \nu') \preceq (\ell, v, b)$. Now assume u is fired. Again $M(P_i) = 0$ and thus $v(x) < c$ and b_φ is true. This time $M'(P_i) = 1$. In the automaton $\mathcal{A}'$ we fire the transition setting b_φ to false and we end up in a state (ℓ, v, b') s.t. $(M', \nu') \preceq (\ell, v, b')$. The same reasoning applies for $\mathcal{N}_{x \geq c}$.

  If $a \in \Sigma$ then the transition is $f(a, [0, \infty[)$ of some widget $\mathcal{N}_e$ for $e = (\ell, \gamma, a, R, \ell')$. The firing of f have left the input places γ_{tt} unchanged. By equation II and the definition of $\mathcal{A}'$ we can fire a matching transition in $\mathcal{A}'$ leading to a state (ℓ', v', b'). We have $M'(P_\ell) = M'(P_{\ell'}) = 0$ and this state is not in the simulation relation. We then fire in the TPN a run $(M', \nu') \xrightarrow{0}_\varepsilon (M'', \nu'')$ of duration 0 carrying out the reset of the clocks $x \in R$ and leading to (M'', ν'') s.t. $M''(P_{\ell'}) = 1$. Two cases can occur:

- This run is only made up of epsilon transitions corresponding to the reset of widgets over $x \in R$ which then return in their initial state. For widgets $\mathcal{N}_{x\leq c}$ and $\mathcal{N}_{x<c}$, we obtain token in P_x and γ_{tt}. As corresponding variables b'_φ are true in state (ℓ', v', b'), we have $(M'', \nu'') \preceq (\ell', v', b')$.
- the previous run is also composed of update transitions of widgets $\mathcal{N}_\varphi$ i.e. firing of t_x^φ of $\mathcal{N}_\varphi$. In this case :
  - ∗ if $x \in R$ then t_x^φ is fired before the reset of $\mathcal{N}_\varphi$. Then after the reset of $\mathcal{N}_\varphi$, we have $M''(P_x^\varphi) = 1$ and $(M'', \nu'') \preceq (\ell', v', b')$,
  - ∗ if $x \notin R$ then $\nu''(t_x^\varphi) = v'(x) = c$. In in $\mathcal{N}_\varphi$ we have $M''(\gamma_{tt}^\varphi) = 1$ and it satisfies requirements (3) of equation II. For the update of blocks $\mathcal{N}_{x\leq c}$ and $\mathcal{N}_{x<c}$, we then fire in $\mathcal{A}'$, the loop transitions setting to false the corresponding variables b_φ leading to (ℓ', v', b'') such that $(M'', \nu'') \preceq (\ell', v', b'')$.

This completes the proof that $\mathcal{A}'$ simulates $\Delta(\mathcal{A})$ and thus $\mathcal{L}(\Delta(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A}')$ and $\mathcal{L}(\Delta(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$.

We can thus conclude that $\mathcal{L}(\Delta(\mathcal{A})) = \mathcal{L}(\mathcal{A})$, which ends the proof of Proposition 1. □

B Proof of B-$\mathcal{TPN}_\varepsilon(\leq,\geq) \approx_{\mathcal{W}} \mathcal{TA}_{syn}(\leq,\geq)$

Let (ℓ, v) be a configuration of $\mathcal{A}$ and (M, ν) be a configuration of $\Delta^+(\mathcal{A})$. We define the relation $\approx \subseteq (\mathbb{N}^p \times \mathbb{R}^m_{\geq 0}) \times (L \times \mathbb{R}^n_{\geq 0})$ by :

$$(M,\nu) \approx (\ell, v) \iff \begin{cases} (1)\, M(P_\ell) = 1 \\ (2)\, \forall \varphi \in \mathcal{C}_\mathcal{A}, \nu(t_x^\varphi) = v(x) \\ (3)\, \forall \varphi = (x \geq c) \in \mathcal{C}_\mathcal{A}(\geq),\ v \in [\![\varphi]\!] \iff \\ \qquad M(\gamma_{tt}^\varphi) = 1 \vee (M(P_x^\varphi) = 1 \wedge \nu(t_x^\varphi) = c) \\ (4)\, \forall \varphi = (x \leq c) \in Inv(\ell),\ v \in [\![\varphi]\!] \iff \\ \qquad\qquad M(P_x^\varphi) = 1 \vee \\ \qquad\qquad (M(urg^\varphi) = 1 \wedge \nu(t_x^\varphi) = c) \end{cases} \tag{III}$$

Let us notice that item 2 of this equation is true even when the transition t_x^φ is not enabled.

Proposition 2. *The relation $\approx$ of equation (III) is a weak timed bisimulation relation.*

Proof. We prove that $\approx$ is a weak timed bisimulation between $\mathcal{A}$ and $\Delta(\mathcal{A})$:

1. final and repeated states: by definition of $\Delta^+(\mathcal{A})$ and the definition of $\approx$;
2. initial states: it is clear that $(M_0, \mathbf{0}) \approx (l_0, \mathbf{0})$,
3. continuous transitions: let $(\ell, v) \xrightarrow{d} (\ell, v + d)$. Take (M, ν)such that $(\ell, v) \approx (M, \nu)$. For $\varphi = (x \leq c) \in Inv(\ell)$, we have $\varphi(v) = \mathtt{tt}$, $\varphi(v+d) = \mathtt{tt}$. According to $\nu(t_x) = v(x)$, we have $\nu(t_x) + d = v(x) + d \leq c$ then $M(urg^\varphi) = 0$ and

time d can elapse in $\mathcal{N}_\varphi$. In $\Delta^+(\mathcal{A})$, from (M,ν), there is a run : $(M,\nu) \xrightarrow{d}_\varepsilon (M',\nu')$ with $M(P_\ell) = M'(P_\ell) = 1$ and the following evolutions of widgets : For $\varphi = (x \le c) \in Inv(\ell)$,

- If $v(x) + d = \nu(t_x^\varphi) + d < c$ then $M'(urg^\varphi) = 0$.
- If $v(x) + d = \nu(t_x^\varphi) + d = c$ then we obtain either $M(urg^\varphi) = 1$ or $M(urg^\varphi) = 0$ and $v'(x) = \nu'(t_x^\varphi) = c$. The transition I^φ is enabled or will be enabled after the immediate firing of t_x, thus blocking time as long as $M(P_\ell) = 1$.

For $\varphi = x \ge c$,

- $\varphi(v) = \mathsf{tt}$ and $\varphi(v+d) = \mathsf{tt}$. If $M(\gamma_{tt}^\varphi) = 1$ time d can elapse in $\mathcal{N}_\varphi$. If $M(\gamma_{tt}^\varphi) = 0$ then $M(P_x^\varphi) = 1$ and (as $d > 0$) t_x^φ is fired before the total elapsing of d.
- $\varphi(v) = \mathsf{ff}$ and $\varphi(v+d) = \mathsf{tt}$, iff there is $d' \le d$ s.t. transition t_x must be fired at d'. Transition t_x is fired and let $d - d'$ elapse.
- $\varphi(v) = \mathsf{ff}$ and $\varphi(v+d) = \mathsf{ff}$ iff time d elapse and leave a token in P_x.

For $\varphi = (x \le c) \notin Inv(\ell)$, according to the subclass of TA we consider, φ is a constraint which will not be used any more before the next reset of x.

- $\varphi(v+d) = \mathsf{ff}$. If $M(urg^\varphi) = 1$ then a time d can elapsed. If $M(urg^\varphi) = 0$ then there is $d' \le d$ s.t. transition t_x must be fired at d'. Transition t_x is fired and let $d - d'$ elapse.
- $\varphi(v) = \mathsf{tt}$ and $\varphi(v+d) = \mathsf{tt}$. This case is similar to $\varphi \in Inv(\ell)$ but no transition I^φ is enabled as (P_ℓ) is not an input place.

This way for each constraint, there is a run $\rho_\varphi = (M,\nu) \xrightarrow{d}_\varepsilon (M_\varphi,\nu_\varphi)$ s.t. (M_φ,ν_φ) satisfies requirements (2) and (3) of equation (III). For all interleaving of previous runs ρ_φ we obtain a run $\rho = (M,\nu) \xrightarrow{d}_\varepsilon (M',\nu')$ s.t. $(\ell, v) \approx (M',\nu')$.

4. discrete transitions : Let $(\ell,v) \xrightarrow{a} (\ell',v')$ and $(\ell,v) \approx (M,\nu)$. There is an edge $e = (\ell,\gamma,a,R,\ell') \in E$ s.t. $\gamma = \wedge_{i=1,n}\varphi_i$, $n \ge 0$ where φ_i is an atomic constraint. According to the subclass of TA we consider, invariants of ℓ' can be ignored for allowing the fire of a as (by definition) they are true if invariants of ℓ are true. From semantics of timed automata (definition 8), $v \in [\![\varphi_i]\!]$ for $1 \le i \le n$. From definition of bisimulation relation $\approx$ we have then, either $M(\gamma_{tt}^{\varphi_i}) = 1$, or $M(\gamma_{tt}^{\varphi_i}) = 0$ and transition $t_x^{\varphi_i}$ is immediately fireable leading to $M(\gamma_{tt}^{\varphi_i}) = 1$. Thus, transition $f(a,[0,\infty[)$ is fired in widget $\mathcal{N}_e$ leading to (M',ν'). We have then $M'(P_\ell) = M'(P'_\ell) = 0$ and $\Delta^+(\mathcal{A})$ must fire epsilon transition in null duration : $(M',\nu') \xrightarrow{0}_\varepsilon (M'',\nu'')$ where $M''(P_\ell) = 1$. The widget $\mathcal{N}_{Reset(R)}$ reset widgets of constraints φ whose clock $x \in R$ then $M''(P_x^\varphi) = 1$ and $\nu''(t_x^\varphi) = v'(x) = 0$. During the reset phase, if a transition t_x^φ is fired :
   - if $x \in R$ then t_x^φ has been fired before the reset of the widget $\mathcal{N}_\varphi$. After the reset phase, we have $M''(P_x^\varphi) = 1$,
   - if $x \notin R$ then $\nu''(t_x^\varphi) = v'(x) = c$. We obtain widget $\mathcal{N}_\varphi$ $M''(\gamma_{tt}^\varphi) = 1$ that satisfy requirement (3) of equation (III),

   The new state (M'',ν'') satisfy $(\ell',v') \approx (M'',\nu'')$. From (M'',ν''), a firing of an epsilon transition $(M'',\nu'') \xrightarrow{0}_\varepsilon (M''',\nu'')$ is a transition t_x^φ which corresponds to the last previous case : "$x \notin R$" and then $(\ell',v') \approx (M'',\nu'')$

This completes the proof that $\Delta^+(\mathcal{A}) \approx \mathcal{A}$.