

Comparison of the Expressiveness of Timed Automata and Time Petri Nets

B. Bérard¹, F. Cassez², S. Haddad¹, D. Lime³, O.H. Roux²

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) “à la Merlin” where time is associated with transitions, but we extend it with open intervals to specify the firing intervals of the 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 the following results: 1) bounded TPNs and TA are equally expressive w.r.t. timed language acceptance; 2) there exists a TA \mathcal{A} s.t. there is no TPN (even unbounded) that is (weakly) timed bisimilar to \mathcal{A} . Because of 2) above, it is natural to try and identify the (strict) subclass of TA that is equivalent to TPN w.r.t. weak timed bisimilarity. Thus we give some further results: 1) we characterize the subclass TA^- of TA that is equivalent to the original model of TPN as defined by Merlin, *i.e.* restricted to closed intervals, 2) we show that the associated membership problem for TA^- is *PSPACE*-complete and 3) we prove that the reachability problem for TA^- is also *PSPACE*-complete.

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

1 Introduction

Petri Nets with Time. The two main extensions of Petri Nets with time are Time Petri Nets (TPNs) [14] and Timed Petri Nets [16]. 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 considered relative to places or transitions [17,15]. The two corresponding subclasses namely P-Timed Petri Nets and T-Timed Petri Nets are expressively equivalent [17,15]. The same classes are defined for TPNs *i.e.* T-TPNs and P-TPNs, but both classes of Timed Petri Nets are included in both P-TPNs and T-TPNs [15]. P-TPNs and T-TPNs are proved to be incomparable in [12]. Finally TPNs form a subclass of Time Stream Petri Nets [9] which were introduced to model multimedia applications.

Timed Automata. Timed Automata (TA) were introduced by Alur & Dill [2] 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 [3] and form a strict subclass of TA.

Related Work. In a previous work [8] 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 [13] with a completely different approach. In another line of work in [11], 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). [7] also considers expressiveness problems but for a subclass of TPNs. It is claimed in [7] that 1-safe TPNs with large constraints are strictly less expressive than TA with arbitrary types of constraints.

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 previous results in this area in the following directions: *i)* we consider general types of constraints (strict, large, bounded, unbounded); *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; *iv)* to conclude we characterize the subclass of TA that is equally expressive to TPN without strict constraints w.r.t. Timed Bisimilarity, and show that the membership problem for this class is *PSPACE*-complete as well the reachability problem. The results of the paper are summarized in Table 1: all the results are new except the one on the first line obtained in [8]. We use the following notations: B-TPN_ε for the set of bounded TPNs with ε -transitions; $\text{1-B-TPN}_\varepsilon$ for the subset of B-TPN_ε with at most one token in each place (one safe TPN); $\text{B-TPN}(\leq, \geq)$ for the subset of B-TPN_ε where only closed intervals are used; \mathcal{TA}_ε for TA with ε -transitions; \mathcal{TA}^- for the class of TA that is equivalent to $\text{B-TPN}(\leq, \geq)$ (to be defined precisely in section 5).

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 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 2, and 3. Finally in section 5, we characterize the subclass of TA that is equivalent w.r.t. Timed Bisimilarity, to the original version of TPNs (with closed intervals) and prove that the associated membership problem and the reachability problem are *PSPACE*-complete.

	Timed Language Acceptance	Timed Bisimilarity
	$\leq_{\mathcal{L}} \mathcal{TA}_\varepsilon$ ([8])	$\leq_{\mathcal{W}} \mathcal{TA}_\varepsilon$ ([8])
B- $\mathcal{TPN}_\varepsilon$	$=_{\mathcal{L}} 1\text{-B-}\mathcal{TPN}_\varepsilon =_{\mathcal{L}} \mathcal{TA}_\varepsilon$	$<_{\mathcal{W}} \mathcal{TA}_\varepsilon$
B- $\mathcal{TPN}(\leq, \geq)$	—	$\approx_{\mathcal{W}} \mathcal{TA}^-$

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

	Membership Problem	Reachability Problem
\mathcal{TA}^-	$PSPACE$ -complete	

Table 1. Summary of the Results

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 $w = a_1 \cdots a_n$, $|w| = n$. 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 $\{\text{tt}, \text{ff}\}$, $\mathbb{R}_{\geq 0}$ denotes the set of positive reals and $\mathbb{R}_{> 0} = \mathbb{R}_{\geq 0} - \{0\}$. 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 $\nu(x)$. Accordingly each constraint $\varphi \in \mathcal{C}(X)$ defines a set of valuations $\llbracket \varphi \rrbracket$ defined by $\llbracket \varphi \rrbracket = \{\nu \in \mathbb{R}_{\geq 0}^X \mid \varphi(\nu) = \text{tt}\}$. I is a $\mathbb{Q}_{> 0}$ -*interval* of $\mathbb{R}_{\geq 0}$ if there is a constraint φ of the form $a \prec_1 x \prec_2 b$ with $a \in \mathbb{Q}_{> 0}$, $b \in \mathbb{Q}_{> 0} \cup \{\infty\}$ and $\prec_1, \prec_2 \in \{<, \leq\}$, such that $I = \llbracket \varphi \rrbracket$. We let $I^\downarrow = \llbracket 0 \leq x \prec_2 b \rrbracket$ be the *downward closure* of I and $I^\uparrow = \llbracket a \prec_1 x \rrbracket$ be the *upward closure* of I . We denote $\mathcal{I}(\mathbb{Q}_{> 0})$ the set of $\mathbb{Q}_{> 0}$ -intervals of $\mathbb{R}_{\geq 0}$. Let $g \in \mathbb{N}_{> 0}$, we write $\mathbb{N}_g = \{\frac{i}{g} \mid i \in \mathbb{N}\}$. A vector $\mathbf{v} \in \mathbb{Q}^n$ belongs to the g -*grid* if $\mathbf{v}(k) \in \mathbb{N}_g$ for all $1 \leq k \leq n$.

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 action a_k . We write $Untimed(w) = a_0 a_1 \cdots a_n \cdots$ for the untimed part of w , and $Duration(w) = \sup_{d_k \in \tau} d_k$ for the duration of the timed word w . A *timed language* L over Σ is a set of timed words.

Definition 2 (Timed Transition Systems). 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.

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 $first(\rho) = q_0$. We assume that a finite run ends with a time transition d_n . If ρ ends with d_n , we let $last(\rho) = q'_n$ 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. $first(\rho) = q_0$ and $last(\rho) = q'$. The *trace* of an infinite run ρ is the timed word $trace(\rho) = (a_{i_0}, d_{i_0}) \cdots (a_{i_k}, d_{i_k}) \cdots$ that consists of the sequence of letters of $A \setminus \{\varepsilon\}$. If ρ is a finite run and $d_n - d_{n-1} = 0$ we define the trace of ρ by $trace(\rho) = (a_{i_0}, d_{i_0}) \cdots (a_{i_k}, d_{i_k})$ where the a_{i_k} are in $A \setminus \{\varepsilon\}$. We define $Untimed(\rho) = Untimed(trace(\rho))$ and $Duration(\rho) = \sup_{d_k \in \mathbb{R}_{\geq 0}} d_k$. A run is *initial* if $first(\rho) \in Q_0$. A run ρ is *accepting* if *i*) either ρ is a finite initial run and $last(\rho) \in F$ 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 $s_1 \xrightarrow{d} s'_1$ with $d \in \mathbb{R}_{\geq 0}$ and $s_1 \preceq s_2$ then $s_2 \xrightarrow{d} s'_2$ for some s'_2 , and $s'_1 \preceq s'_2$; 4) if $s_1 \xrightarrow{a} s'_1$ with $a \in A$ and $s_1 \preceq s_2$ then $s_2 \xrightarrow{a} s'_2$ 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 $S_1 \preceq_S S_2$ 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 relation⁴ 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 $S_1 \approx_S S_2$ 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:

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

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 $S_1 \preceq_W S_2$ 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 $S_1 \approx_W S_2$ in this case. Note that if $S_1 \preceq_S S_2$ then $S_1 \preceq_W S_2$ and if $S_1 \preceq_W S_2$ then $\mathcal{L}(S_1) \subseteq \mathcal{L}(S_2)$.

2.2 Time Petri Nets

Time Petri Nets (TPN) were introduced in [14] 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 it was last 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 version⁵ of TPN with accepting and repeated markings and prove our results for this general model.

Definition 5 (Labelled Time Petri Net). A Labelled Time Petri Net \mathcal{N} is a tuple $(P, T, \Sigma_\varepsilon, \bullet(\cdot), (\cdot)^\bullet, M_0, \Lambda, I, F, R)$ where: P is a finite set of places and

⁴ $s_2 \preceq^{-1} s_1 \iff s_1 \preceq s_2$.

⁵ This is required to be able to define Büchi timed languages, which is not possible in the original version of TPN of [14].

T is a finite set of transitions and $P \cap T = \emptyset$; Σ is a finite set of actions $\bullet(\cdot) \in (\mathbb{N}^P)^T$ is the backward incidence mapping; $(\cdot)^\bullet \in (\mathbb{N}^P)^T$ is the forward incidence mapping; $M_0 \in \mathbb{N}^P$ is the initial marking; $\Lambda : T \rightarrow \Sigma_\varepsilon$ is the labeling function; $I : T \rightarrow \mathcal{I}(\mathbb{Q}_{\geq 0})$ 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. An unlabelled TPN is a TPN s.t. $\Sigma = T$ and $\Lambda(t) = t$ for each $t \in T$.

\mathcal{N} is a g -TPN if for all $t \in T$, $I(t)$ is an interval with bounds in \mathbb{N}_g . We also use $\bullet t$ (resp. t^\bullet) to denote the set of places $\bullet t = \{p \in P \mid \bullet t(p) > 0\}$ (resp. $t^\bullet = \{p \in P \mid t^\bullet(p) > 0\}$) as it is common in the literature⁶.

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 $En(M)$ the set of enabled transitions 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 markings.

In this paper, we consider the *intermediate semantics* for TPNs, based on [6,4], 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:

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

Definition 6 (Semantics of TPN). *The semantics of a TPN $\mathcal{N} = (P, T, \Sigma_\varepsilon, \bullet(\cdot), (\cdot)^\bullet, M_0, \Lambda, I, F, R)$ 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:*

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

⁶ Whether $\bullet t$ (resp. t^\bullet) stands for a vector of $(\mathbb{N}^P)^T$ or a subset of P will be unambiguously defined by the context.

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

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

A run ρ of \mathcal{N} is an initial run of $S_{\mathcal{N}}$. The timed language accepted by \mathcal{N} is $\mathcal{L}(\mathcal{N}) = \mathcal{L}(S_{\mathcal{N}})$. An unlabelled TPN is a TPN s.t. $\Lambda(t) = t$ for all $t \in T$. An unlabelled TPN accepts a timed language in $(T \times \mathbb{R}_{\geq 0})^\infty$.

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 $\text{Duration}(w) = 0$ we say that w is an *instantaneous firing sequence*. The set of *reachable configurations* of \mathcal{N} is $\text{Reach}(\mathcal{N}) = \{M \in \mathbb{N}^P \mid \exists (M, \nu) \mid (M_0, \mathbf{0}) \xrightarrow{*} (M, \nu)\}$.

2.3 Timed Automata

Definition 7 (Timed Automaton). A Timed Automaton \mathcal{A} is a tuple $(L, l_0, X, \Sigma_\varepsilon, E, \text{Inv}, F, R)$ 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 $R \subseteq X$; $\text{Inv} \in \mathcal{C}(X)^L$ 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 $\mathcal{A} = (L, l_0, C, \Sigma_\varepsilon, E, \text{Act}, \text{Inv}, F, R)$ 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 $\exists \langle l, \gamma, a, R, l' \rangle \in E$ s.t. $\gamma(v) = \text{tt}$, $v' = v[R \mapsto 0]$ and $\text{Inv}(l')(v') = \text{tt}$; ii) the continuous transition relation $(l, v) \xrightarrow{t} (l', v')$ iff $l = l'$, $v' = v + t$ and $\forall 0 \leq t' \leq t$, $\text{Inv}(l)(v + t') = \text{tt}$. 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 $B \approx_S B'$ (resp. $B \approx_W B'$) for $S_B \approx_S S_{B'}$ (resp. $S_B \approx_W S_{B'}$). 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. $B \approx_S B'$ (resp. $B \approx_W B'$). We write $\mathcal{C}' \leq_S \mathcal{C}$ (resp. $\mathcal{C}' \leq_W \mathcal{C}$) in this case. If moreover there is a $B \in \mathcal{C}$ s.t. there is no $B' \in \mathcal{C}'$ with $B \approx_S B'$ (resp. $B \approx_W B'$), then $\mathcal{C}' <_S \mathcal{C}$ (resp. $\mathcal{C}' <_W \mathcal{C}$). If both $\mathcal{C}' <_S \mathcal{C}$ and $\mathcal{C} <_S \mathcal{C}'$ (resp. $<_W$) then \mathcal{C} and \mathcal{C}' are equally expressive w.r.t. strong (resp. weak) timed bisimilarity, and we write $\mathcal{C} \approx_S \mathcal{C}'$ (resp. $\mathcal{C} \approx_W \mathcal{C}'$).*

In the sequel we will compare various classes of TPNs and TAs. We recall the following theorem adapted from [8]:

Theorem 1 ([8]). *For any $\mathcal{N} \in B\text{-TPN}_\varepsilon$ there is a TA \mathcal{A} s.t. $\mathcal{N} \approx_W \mathcal{A}$, hence $B\text{-TPN}_\varepsilon \leq_W \text{TA}_\varepsilon$.*

3 Strict Ordering Results

In this section, we establish some results proving that TPN are strictly less expressive w.r.t. weak timed bisimilarity than various classes of TA. We first state a lemma whose the omitted proof does not present any difficulty.

Lemma 1 (Waiting Cannot Disable Transitions). *Let (M, ν) be a marking of a TPN. If $(M, \nu) \xrightarrow{t_1 t_2 \dots t_k}$ with $t_1 t_2 \dots t_k$ an instantaneous firing sequence and $(M, \nu) \xrightarrow{d} (M_d, \nu_d)$ for some $d \geq 0$, then $(M_d, \nu_d) \xrightarrow{t_1 t_2 \dots t_k}$.*

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

Proof. The proof is based on the previous lemma.

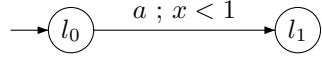


Fig. 1. The Timed Automaton \mathcal{A}_0

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 $(M_0, \mathbf{0}) \xrightarrow{\varepsilon^{i_0} d_1 \varepsilon^{i_1} d_2 \varepsilon^{i_2} \dots d_n \varepsilon^{i_n}} (M_1, \nu_1)$ in \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 $(M_0, \mathbf{0}) \xrightarrow{\varepsilon^{i_0} d_1 \varepsilon^{i_1} d_2 \varepsilon^{i_2} \dots d_{n-1} \varepsilon^{i_{n-1}}} (M', \nu')$. (M', ν') is bisimilar to a configuration $(l_0, \nu(x) = d')$ 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 $(M_1, \nu_1) \approx (l_0, \nu(x) = 1)$ from which no a can be fired. \square

A similar theorem holds for a TA \mathcal{A}_1 with large constraints:

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

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 a transition 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 ??, some 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.

Corollary 1. $B\text{-TPN}_\varepsilon <_{\mathcal{W}} \mathcal{TA}(<)$ and $B\text{-TPN}_\varepsilon <_{\mathcal{W}} \mathcal{TA}(\leq)$.

Following this negative results, we compare the expressiveness of TPNs and TA w.r.t. to Timed Language Acceptance and then characterize the 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 languages 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 of Atomic Constraints. Let $\varphi \in \mathcal{C}_x$ be an atomic constraint on x . From φ , we define the TPN \mathcal{N}_φ , given by the widgets of Fig. 2 ((a) and (b)) and Fig. 3. In the figures, a transition is written $t(\ell, I)$ where t is the name of the transition, $\ell \in A_\varepsilon$ 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). 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 $a > 0$.

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.

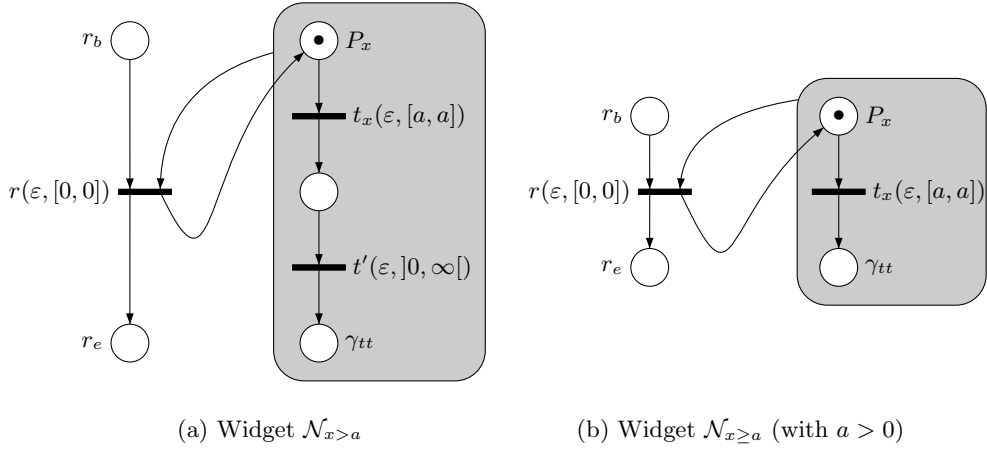


Fig. 2. Widgets for $\mathcal{N}_{x>a}$ and $\mathcal{N}_{x\geq a}$

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.

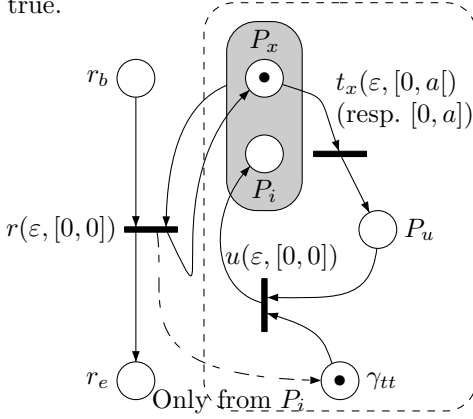


Fig. 3. Widget $\mathcal{N}_{x<a}$ (resp. $\mathcal{N}_{x\leq a}$)

When a clock x is reset, all the grey blocks modeling an x -constraint must be set to their *initial* marking with 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 $D(R) = \{\varphi_1^{x_1}, \varphi_2^{x_1}, \dots, \varphi_{q_1}^{x_1}, \dots, \varphi_{q_n}^{x_n}\}$ 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.

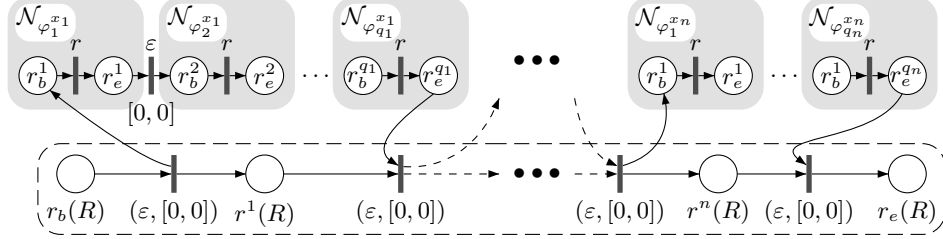


Fig. 4. Widget $\mathcal{N}_{Reset(R)}$ to reset the widgets of the constraints of clocks x_i , $1 \leq i \leq n$

The Complete Construction. First we create fresh places P_ℓ for each $\ell \in L$, and another place $Firing$ just for convenience: it will allow us to define a simulation relation more succinctly. Then we build the widgets \mathcal{N}_φ , 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 $r(\epsilon, [0, 0])$,
3. connect them to the places of the widgets \mathcal{N}_{φ_i} and $\mathcal{N}_{Reset(R)}$ as described on Fig. 5. In case $\gamma = \text{tt}$ (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 $r(\epsilon, [0, 0])$ and the output place of $f(a, [0, \infty[)$ is $P_{\ell'}$ instead of $Firing$.

The place $Firing$ is just added for convenience: it has a token only during the reset phase of the TPN \mathcal{N}_e and thus means “we are firing transitions in the reset widget $\mathcal{N}_{Reset(R)}$ ”. 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}_φ 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 $\Delta(\mathcal{A})$ $(\sum_{\ell \in L} M(P_\ell)) + M(Firing) = 1$. 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 [11]. 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.

New Results for TPNs.

Corollary 2. $1\text{-}B\text{-}TPN_{\mathcal{N}_e} =_{\mathcal{L}} B\text{-}TPN_{\mathcal{N}_e} =_{\mathcal{L}} \mathcal{TA}_e$.

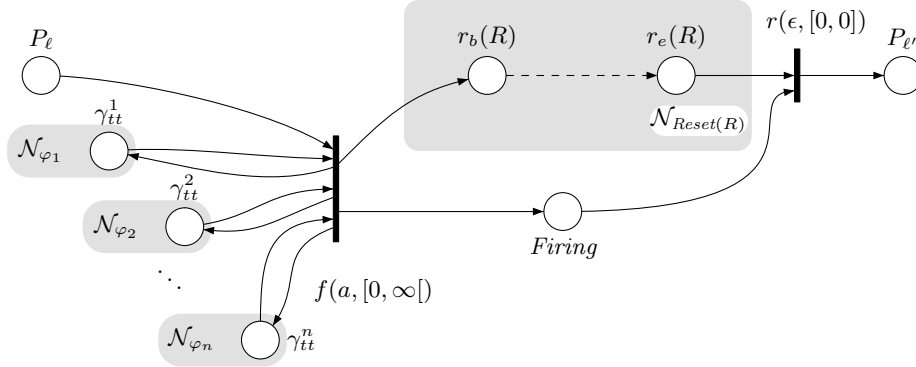


Fig. 5. Widget \mathcal{N}_e of an edge $e = (\ell, \gamma, a, R, \ell')$

Proof. Let $T \in \text{B-TPN}_{\mathcal{N}_e}$. We use Theorem 1 and thus there is a TA A_T s.t. $\mathcal{L}(T) = \mathcal{L}(A_T)$. From A_T we use Proposition 2 and obtain $\Delta(A_T)$ which is a 1-safe TPN. The second equality results from the construction and theorem 1. \square

Corollary 3. *The language emptiness problem is decidable for $\text{B-TPN}_{\mathcal{N}_e}$ whereas the universal language problem is undecidable for $1\text{-B-TPN}_{\mathcal{N}_e}$.*

5 Bisimulation of TA by TPNs

In this section, we consider TPN originally defined by Merlin i.e. without strict constraints and labelled-free TA i.e. where two different edges have different labels and no label is ϵ . We develop here the main result of the paper: a characterization of the TA which admit a bisimilar TPN. From this characterization, we will deduce that given a TA, the problem of deciding whether there is a TPN bisimilar to it, is *PSPACE*-complete. Furthermore, we will provide two effective constructions for such a TPN: the first one with rational constants has a size linear w.r.t. the TA and the other using only integer constraints but with an exponential size.

Since our proofs are based on the regions, we detail their definition. First, a *region* is a pair composed by a location and an *elementary time zone* of the grid defined by the clocks. In the sequel, the topology of the regions is implicitly derived from the one of its associated zone. We now formally define the particular case of regions for a maximal constant $K = \infty$. Obviously it may lead to an infinite region automaton but will be an helpful tool for proving our characterization. Note also that the following definition is equivalent to the original one but is more appropriate for our theoretical developments.

Definition 11 (Regions of an automaton w.r.t. the g -grid and constant ∞). *A time-closed region r is defined by:*

- l_r the location of r ,
- $\min_r \in \mathbb{N}_g^X$ the minimal vector of the topological closure of r ,

- $1 \leq \text{size}_r \leq |X|$ the number of different fractional parts of the values of clocks in the \mathbb{N}_g^X grid and $\text{ord}_r : X \rightarrow \{1, \dots, \text{size}_r\}$ an onto mapping related to the relative positions of these fractional parts,

- $r = \{(l_r, \min_r + \delta) \mid \delta \in \mathbb{R}_{\geq 0}^X \wedge \forall x, y \in X [\text{ord}_r(x) = 1 \Leftrightarrow \delta(x) = 0] \wedge \delta(x) < 1/g \wedge [\text{ord}_r(x) < \text{ord}_r(y) \Leftrightarrow \delta(x) < \delta(y)]\}$.

A time-open region r is defined with the same attributes as the time-closed region by: $r = \{(l_r, \min_r + \delta + d) \mid d \in \mathbb{R}_{> 0} \wedge \forall x \in X, \delta(x) + d < 1/g\}$.

$[X]_r$ is the set of equivalence classes of clocks w.r.t. their fractional parts, i.e. x and y are equivalent iff $\text{ord}_r(x) = \text{ord}_r(y)$.

Note that letting time elapse leads to an alternation of time-open regions where time can elapse and time-closed ones where time cannot elapse. Let us also remark that $\min_r \notin r$ except if r is a singleton. More generally, whatever be the grid and the maximal constant, we note \bar{r} , the topological closure of r ; \min_r denotes the minimum vector of \bar{r} and we say that a region is *reachable* if it belongs to the region automaton.

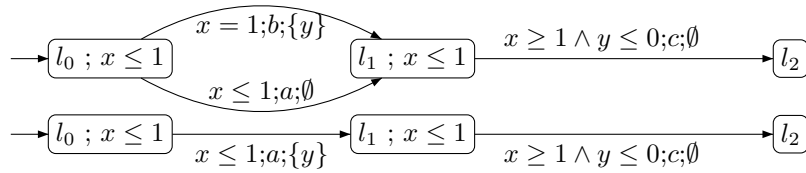
The following useful lemma points out the ‘‘granularity’’ of the effect of time on the behaviour of TPN when strict constraints are excluded.

Lemma 2. *Let (M, ν) and $(M, \nu + \delta)$ be two admissible configurations of a g -TPN with $\nu, \delta \in \mathbb{R}_{\geq 0}^{En(M)}$. Let σ be an instantaneous firing sequence, then:*

(a) $(M, \nu) \xrightarrow{\sigma} (M, \nu + \delta) \xrightarrow{\sigma}$

(b) *If $\nu \in \mathbb{N}_g^{En(M)}$ and $\delta \in [0, 1/g]^{En(M)}$ then $(M, \nu + \delta) \xrightarrow{\sigma} (M, \nu) \xrightarrow{\sigma}$*

The characterization of TA bisimilar to some TPN is closely related to the topological closure of reachable regions: it states that any region intersecting the topological closure of a reachable region is also reachable and that a discrete step either from a region or from the minimal vector of its topological closure is possible in the whole topological closure. The two following automata will illustrate our results. In the sequel, we suppose that any atomic constraint related to a clock x occurring in the invariant of a location is added to the guard of each incoming transition which does not reset x . The first automaton presented below admits a bisimilar TPN whereas the second does not.



Theorem 4 (Characterization of TA bisimilar to some TPN). *Let \mathcal{A} be an (labelled-free) timed automaton, let $R(\mathcal{A})$ its region automaton w.r.t. the 1-grid and a constant K strictly greater than any constant occurring in the automaton, then \mathcal{A} is weakly timed bisimilar to a time Petri net iff:*

$\forall r \in R(\mathcal{A}), \forall e$ an edge of \mathcal{A} ,

(a) *Every region r' s.t. $r' \cap \bar{r} \neq \emptyset$ is reachable*

- (b) $\forall (l_r, v) \in r, (l_r, v) \xrightarrow{e} (l_r, \min_r) \xrightarrow{e}$
(c) $\forall (l_r, v) \in \bar{r}, (l_r, \min_r) \xrightarrow{e} (l_r, v) \xrightarrow{e}$

Furthermore, if these conditions are satisfied then one can build a 1-bounded 2-TPN bisimilar to \mathcal{A} whose size is linear w.r.t. the size of \mathcal{A} and a 1-bounded 1-TPN bisimilar to \mathcal{A} whose size is exponential w.r.t. the size of \mathcal{A} .

We note \mathcal{TA}^- this class of automata. Using the theorem, we justify why the second TA does not admit a bisimilar TPN. The region $r = \{(l_1, x = 1 \wedge 0 < y < 1)\}$ is reachable. The guard of edge c is true in $\min_r = (l_1, (1, 0))$ whereas it is false in r .

Proof (Sketch).

The reader may refer to [5] for the detailed developments omitted here.

Necessity. Let us suppose that \mathcal{A} is bisimilar to some g -TPN \mathcal{N} via the relation \mathcal{R} . We consider the region automaton of \mathcal{A} w.r.t. the \mathbb{N}_g^X grid and the constant $K = \infty$. Let us remark that \bar{r} is then a finite union of regions. Then we prove by induction on the reachability relation between regions the following uniform version of bisimilarity:

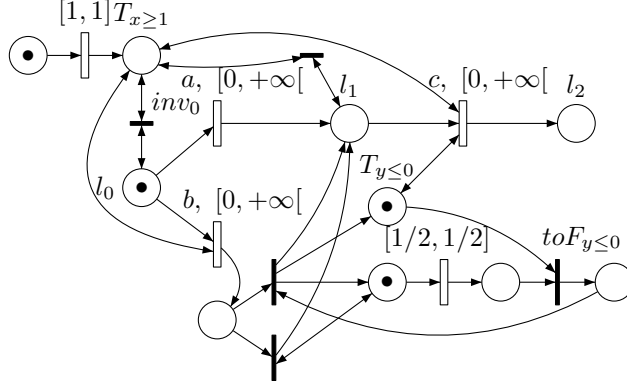
- if a region r belongs to $R(\mathcal{A})$ then any region in \bar{r} also belongs to $R(\mathcal{A})$;
- with each reachable region r is associated a configuration of the net (M_r, ν_r) with $\nu_r \in \mathbb{N}_g^{En(M_r)}$ and a mapping $\phi_r : En(M_r) \rightarrow [X]_r$ which fulfill: if r is time-closed then $\forall (l_r, \min_r + \delta) \in \bar{r}, (l_r, \min_r + \delta) \mathcal{R}(M_r, \nu_r + proj_r(\delta))$ with $proj_r(\delta)(t) = \delta(\phi_r(t))$
; if r is time-open then $\forall (l_r, \min_r + \delta + d) \in \bar{r}, (l_r, \min_r + \delta + d) \mathcal{R}(M_r, \nu_r + proj_r(\delta) + d)$

This induction is the difficult part of the proof and depends both on the kind of step (discrete or time) and of the kind of region (time-open or time closed). Then, the truth of conditions (a), (b), and (c) relative to the g -grid and the constant ∞ is straightforward:

- (a) This assertion is included in the inductive assertions.
(b) Let r be a reachable region, let $(l_r, \min_r + \delta) \in r$ be a configuration with $\delta \in [0, 1/g]^X$, then $\exists (M, \nu) \nu \in \mathbb{N}_g^{En(M)}$ bisimilar to (l_r, \min_r) and $(M, \nu + \delta')$ with $\delta' \in [0, 1/g]^{En(M)}$ bisimilar to $(l_r, \nu + \delta)$. Suppose that $(l_r, \min_r + \delta) \xrightarrow{e}$, then $(M, \nu + \delta') \xrightarrow{\sigma}$ with σ an instantaneous firing sequence and $label(\sigma) = e$. Now by lemma 2-b, $(M, \nu) \xrightarrow{\sigma}$, thus $(l_r, \min_r) \xrightarrow{e}$.
(c) Let r be a region, and $(l_r, \min_r + \delta) \in \bar{r}$ with $\delta \in [0, 1/g]^X$ thus $\exists (M, \nu)$ bisimilar to (l_r, \min_r) and $(M, \nu + \delta')$ with $\delta' \in [0, 1/g]^{En(M)}$ bisimilar to $(l_r, \min_r + \delta)$. Suppose that $(l_r, \min_r) \xrightarrow{e}$, then $(M, \nu) \xrightarrow{\sigma}$ with σ an instantaneous firing sequence and $label(\sigma) = e$. Now by lemma 2-a, $(M, \nu + \delta') \xrightarrow{\sigma}$, thus $(l_r, \min_r + \delta) \xrightarrow{e}$. In order to complete the proof, we successively show that if the conditions are satisfied w.r.t. the g -grid and infinite constant, they are satisfied w.r.t. the 1-grid and infinite constant and when satisfied w.r.t the 1-grid and infinite constant, they are satisfied w.r.t the 1-grid and the usual finite constant.

Sufficiency: a first construction. The principles of the construction are similar to the ones used for the language equivalence. We build a subnet per elementary condition (including the part associated with the clock resetting). However

except for the conditions $x \geq c$ and the resetting part, all the constructions are different. Instead of giving the whole construction process, we illustrate it on the first TA presented above and its translation given below (with some simplifications related to this particular TA). For readability, immediate transitions (with interval $[0, 0]$) are represented in black and ϵ labels are not shown.



First, note that the subnet associated to the constraint $y \leq 0$ switches the condition to false (firing of $toF_{y \leq 0}$) when the implicit value of y maintained in the net reaches $1/2$. Seemingly, this translation appears to be less constrained than the original condition. We explain how we prove that this translation is nevertheless sound. Let r be the region corresponding to the current configuration (l, v) of the automaton simulated by the net, if the net is able to simulate a discrete step of the automaton, we prove that in the configuration (l, \min_r) of the automaton this step is also possible. Thus by condition (c), the step is also possible from (l, v) . On the other hand, if a discrete step is possible for (l, v) in the automaton, we show that this step is also simulatable in the net using both conditions (b) and (c) and the following fact: $\forall x \in X, \exists (l_r, v'), (l_r, v'') \in \bar{r}$ such that $v'(x) = \lfloor v(x) \rfloor$ and $v''(x) = \lceil v(x) \rceil$. We also need to handle the invariants. First it is straightforward to observe that due to condition (a), an atomic constraint $x < c$ occurring in an invariant may be safely deleted since its effectiveness leads to the existence of a region r whose time-successor (which intersects \bar{r}) would not be reachable. The subnet associated to the atomic constraint $x \leq 1$ occurring in the invariant of l_0 leads to transition inv_0 (not modifying the marking) which is fireable as soon as the simulated value of x reaches 1 and the place l_0 is marked. Thus time cannot progress except if the location is left. The second construction is given in appendix B.

This characterization leads to the the following complexity results whose proof is given in appendix B.

Proposition 1 (Complexity results). *Given a (labelled-free) timed automaton \mathcal{A} , deciding whether there is a TPN weakly time bisimilar to \mathcal{A} is PSPACE-complete. The reachability problem for the class \mathcal{TA}^- is PSPACE-complete.*

At last, we complete these results by adapting them to other models of TA. The previous characterization holds for TA with diagonal constraints and when

satisfied a bisimilar 1-bounded 1-TPN whose size is exponential w.r.t. the TA may be built. A simpler characterization holds for TA without strict (and diagonal) constraints. Nevertheless, for these two models, the complexity of the membership and reachability problems is still $PSPACE$ -complete.

References

1. 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.
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science B*, 126:183–235, 1994.
3. 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.
4. T. Aura and J. Lilius. A causal semantics for time Petri nets. *Theoretical Computer Science*, 243(1–2):409–447, 2000.
5. 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.
6. 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.
7. 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.
8. Franck Cassez and Olivier 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.
9. 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.
10. 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.
11. 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.
12. W. Khansa, J.P. Denat, and S. Collart-Dutilleul. P-Time Petri Nets for manufacturing systems. In *WODES'96*, Scotland, pages 94–102, 1996.
13. D. Lime and O. H. Roux. State class timed automaton of a time Petri net. In *PNPM'03*. IEEE Computer Society, September 2003.
14. P. M. Merlin. *A study of the recoverability of computing systems*. PhD thesis, University of California, Irvine, CA, 1974.
15. 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.
16. C. Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1974.

17. 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 2. *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}_μ with no ε transitions. We apply our construction to \mathcal{A}_μ 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 $\Delta(\mathcal{A}) = (P, T, A_\varepsilon, \bullet(\cdot), (\cdot)^\bullet, M_0, \Lambda, \Gamma, F_\Delta, R_\Delta)$. Assume $C = \{x_1, \dots, x_k\}$, $P = \{p_1, \dots, p_m\}$ and $T = \{t_1, \dots, t_n\}$. 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 $\preceq \subseteq (L \times \mathbb{R}_{\geq 0}^n) \times (\mathbb{N}^p \times \mathbb{R}_{\geq 0}^m)$ by:

$$(\ell, v) \preceq (M, \nu) \iff \begin{cases} (1) M(P_\ell) = 1 \vee M(\text{Firing}) = 1 \\ (2) \text{for each } \varphi = x \bowtie c, \bowtie \in \{<, \leq\}, M(P_u) = 0 \\ (3) \text{for each } \varphi \in \mathcal{C}_\mathcal{A}, v \in \llbracket \varphi \rrbracket \iff M(\gamma_{tt}^\varphi) = 1 \end{cases} \quad (\text{I})$$

Now we can prove that \preceq is a weak simulation relation of \mathcal{A} by $\Delta(\mathcal{A})$, and this by checking the 4 conditions of Def. 3.

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 immediatly 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}' .

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

We start with $\mathcal{A}' = \mathcal{A}$. The construction of the new features of \mathcal{A}' is depicted on Fig. 6. Let $(\ell, \gamma \wedge \psi, a, R, \ell')$ be an edge of \mathcal{A}' with $\gamma = \bigwedge_{x \preceq c \in K_\preceq} x \preceq c$ and

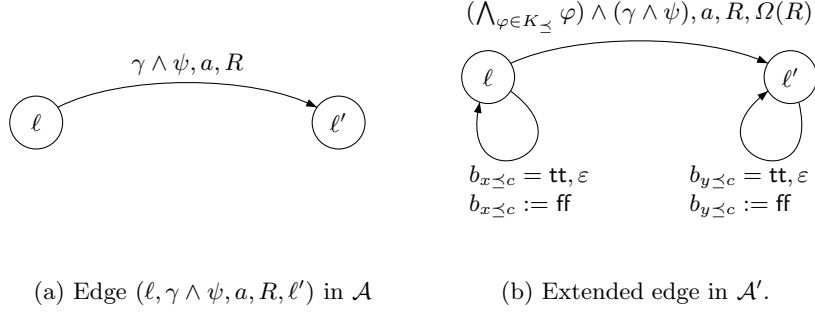


Fig. 6. From \mathcal{A} to \mathcal{A}' .

$\psi = \bigwedge_{x \succeq c \in K_{\succeq}} x \succeq c$. For such an edge we strengthen⁷ the guard $\gamma \wedge \psi$ to obtain γ' as follows: $\gamma' = \gamma \wedge \psi \wedge \bigwedge_{x \preceq c \in K_{\succeq}} b_{x \preceq c}$. 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 $b_{x \preceq c}$ is true as well. We also reset to true all the variables $b_{x \preceq c}$ s.t. $x \in R$ on a transition $(\ell, \gamma \wedge \psi, a, R, \ell')$ and $\Omega(R)$ corresponds to the reset of all $b_{x \preceq c}$ s.t. $x \in R$, $\Omega(R) = \bigwedge_{x \in R} b_{x \preceq c} := \text{tt}$.

Now let ℓ be location of \mathcal{A}' . For each variable $b_{x \preceq c}$ we add a loop edge $(\ell, b_{x \preceq c} = \text{tt}, \varepsilon, b_{x \preceq c} := \text{ff}, \ell)$ in \mathcal{A}' , *i.e.* the automaton \mathcal{A}' can decide non deterministically⁸ to set $b_{x \preceq c}$ to false if it is true (see Fig. 6). There are as many loops on each location as the number of variables $b_{x \preceq c}$. The new non deterministic TA \mathcal{A}' accepts exactly the same language as \mathcal{A} *i.e.* $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.

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 $\preceq \subseteq (\mathbb{N}^p \times \mathbb{R}_{\geq 0}^m) \times (L \times \mathbb{R}_{\geq 0}^n \times \mathbb{B}^k)$ by:

$$(M, \nu) \preceq (\ell, v, b) \iff \begin{cases} (1) M(P_\ell) = 1 \vee M(\text{Firing}) = 1 \\ (2) \forall \varphi = x > c \in K_{>}, v \in \llbracket \varphi \rrbracket \iff M(\gamma_{tt}^\varphi) = 1 \\ (3) \forall \varphi = x \geq c \in K_{\geq}, v \in \llbracket \varphi \rrbracket \iff M(\gamma_{tt}^\varphi) = 1 \vee \\ \quad (M(P_x^\varphi) = 1 \wedge \nu(t_x^\varphi) = c) \\ (4) \forall \varphi \in K_{\preceq}, M(P_i^\varphi) = 1 \iff (b_\varphi = \text{ff} \vee v \notin \llbracket \varphi \rrbracket) \end{cases} \quad (\text{II})$$

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

From the previous proofs we conclude that $\mathcal{L}(\Delta(\mathcal{A})) = \mathcal{L}(\mathcal{A})$ which ends the proof of Proposition 2. \square

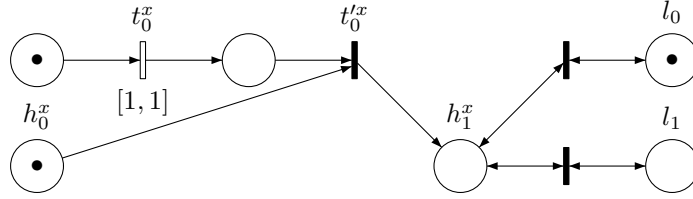
⁷ We need an extended type of TA with boolean variables; this does not add any expressive power to the model.

⁸ 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 $\Delta(\mathcal{A})$.

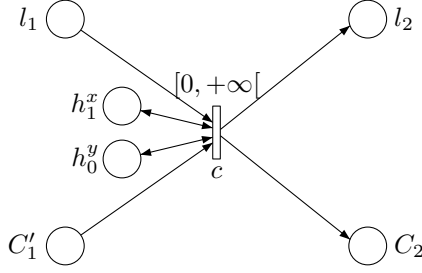
B Proofs related to bisimulation

We need to consider another automaton, called *class automaton*, in which the states, called *classes*, are of the form $(l, fut(Z) \cap Inv(l))$, where Z is a zone. In this case, the second component is not an elementary zone anymore (but a general zone) and the automaton is build from the initial class $(l_0, fut(\mathbf{0}) \cap Inv(l_0))$ by the following transitions: $(l, Z_1) \xrightarrow{a} (l', Z_2)$ if there is a transition $(l, \gamma, a, R, l') \in E$ such that $Z_1 \cap \llbracket \gamma \rrbracket \neq \emptyset$, and $Z_2 = fut((Z_1 \cap \llbracket \gamma \rrbracket)[R \mapsto 0]) \cap Inv(l')$. Note that this automaton also accepts $Uptime(L(\mathcal{A}))$. Moreover, since a class can be represented by a Difference Bounded Matrix [10], its size is at most $(4K + 2)^{(|X|+1)^2}$, which is exponential in the size of \mathcal{A} , as for the region automaton.

Proof (Sufficiency: the second construction of theorem 4). This construction is based on the class automaton and the integer points of the region automaton. In order to track the integer points, we build a subnet per clock x , in which each possible value c for the “integral part” of x corresponds to a token in a specific place h_c^x . The “fractional part” corresponds to the valuation of transition t_c^x and thus can reach 1. This simulation goes up to the maximal significant value of x . The invariants are handled in the same way as the previous construction. Below we give the subnet corresponding to the clock x with the invariant conditions of l_0 and l_1 for the TA illustrating our first construction. The subnet corresponding to y is similar.



The modelling of discrete transitions is more subtle and involves the class automaton of \mathcal{A} . Here we have four classes: $C_0 = \{l_0, 0 \leq x = y \leq 1\}$, $C_1 = \{l_1, 0 \leq x = y \leq 1\}$, $C'_1 = \{l_1, x = 1 \wedge y = 0\}$ and $C_2 = \{l_2, 0 \leq y = x - 1\}$. The current class is modelled by a token in an eponymous place. For a transition e possible in an integer point (l, v) of the region automaton and a class C to which this point belongs (or more exactly to which belongs a point equivalent to (l, v) w.r.t. the significant clocks), we define a transition labelled by e . The resetting part is similarly handled as in the previous constructions. The input places of this transition are the places modelling the integer point, the location and the class. The output places are those modelling the new integer point, location and class. Below, we present the subnet corresponding to the transition e at point $(l_1, (1, 0))$ and class C'_1 .



The soundness of this construction is omitted but we justify on this example why the modelling of classes is necessary. Let us consider the following sequence in \mathcal{A} : $(l_0, (0, 0)) \xrightarrow{a} (l_1, (0, 0)) \xrightarrow{1} (l_1, (1, 1))$. The simulation of this sequence by \mathcal{N} may lead to the following configuration: l_1, h_0^x, h_0^y and C_1 marked with t_0^x and t_0^y enabled since 1 t.u. Suppose that the sequence $t_0^x t_0^y$ is fired, marking the place t_1^x , then without the input place C_1' the transition labelled c could be erroneously fired. Since C_1' is unmarked this firing is disabled.

Proof (of proposition 1). The reachability problem for regions is in *PSPACE*. In order to check whether the condition (a) is false we *non deterministically* pick a region r and a region r' which intersects \bar{r} and check whether r is reachable and r' is not reachable. In order to check whether the condition (b) is false we *non deterministically* pick a region r and a edge e and check whether r is reachable and e is fireable from r and not fireable from (l_r, \min_r) . In order to check whether the condition (c) is false we *non deterministically* pick a region r , a region r' which intersects \bar{r} and a edge e and check whether r is reachable and e is not fireable from r or r' and fireable from (l_r, \min_r) . By Savitch construction, we obtain a deterministic algorithm in *PSPACE*.

In order to show the *PSPACE*-hardness, we use the construction given in [1] (in appendix D) which reduces the acceptance problem for linear bounded Turing machine (LBTM) to the reachability problem for TA with restricted guards.

The computed TA (called $A_{\mathcal{M}, w_0}$) satisfies the conditions (a) and (b) but does not satisfy the condition (c). However it can be safely transformed in order to satisfy this condition by adding the invariant $t \leq 1$ to any state (q, i) and the invariant $t \leq 0$ to any state (i, θ, j) . This intermediate automaton is now bisimilar to a TPN.

Then we transform the edges entering the *end* state by resetting t and at last we add an edge $(end, t = 0, e, \emptyset, end)$. Let us call $A'_{\mathcal{M}, w_0}$ this final automaton.

If the LBTM \mathcal{M} does not accept the word w_0 , then the state *end* is not reachable and $A'_{\mathcal{M}, w_0}$ satisfies the conditions (a),(b),(c).

If the LBTM \mathcal{M} accepts the word w_0 , then the state *end* is reachable and $A'_{\mathcal{M}, w_0}$ does not satisfy the condition (c) (the additional edge is fireable when entering *end* but not after letting the time elapse).

Note that we have implicitly proved that the reachability for \mathcal{TA}^- is *PSPACE*-hard when building the intermediate automaton.

□