Theoretical Computer Science 474 (2013) 1–20

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

The expressive power of time Petri nets[☆,☆☆]

B. Bérard [a], F. Cassez [b], S. Haddad [c], D. Lime [d,*], O.H. Roux [d]

[a] Université Pierre et Marie Curie, LIP6, Paris, France
[b] National ICT Australia, Sydney, Australia
[c] École Normale Supérieure de Cachan, LSV, Cachan, France
[d] LUNAM Université, École Centrale de Nantes, IRCCyN, Nantes, France

ARTICLE INFO

Article history:
Received 23 March 2011
Received in revised form 16 November 2012
Accepted 4 December 2012
Communicated by J. Esparza

Keywords:
Time Petri nets
Timed automata
Expressiveness
Language equivalence
Bisimilarity

ABSTRACT

We investigate expressiveness questions for time Petri nets (TPNs) and some of their most useful extensions. We first introduce generalised time Petri nets (GTPNs) as an abstract model that encompasses variants of TPNs such as self modifications and read, reset and inhibitor arcs.

We give a syntactical translation from bounded GTPNs to timed automata (TA) that generates *isomorphic* transition systems. We prove that the class of bounded GTPNs is strictly less expressive than TA *w.r.t.* weak timed bisimilarity. We prove that bounded GTPNs, bounded TPNs and TA are equally expressive *w.r.t.* timed language acceptance. Finally, we characterise a syntactical subclass of TA that is equally expressive to bounded GTPNs "à la Merlin" *w.r.t.* weak timed bisimilarity. These results provide a unified comparison of the expressiveness of many variants of timed models often used in practice. It leads to new important results for TPNs. Among them are: 1-safe TPNs and bounded-TPNs are equally expressive; ε-transitions strictly increase the expressive power of TPNs; self modifying nets as well as read, inhibitor and reset arcs do not add expressiveness to bounded TPNs.

1. Introduction

In the last decade, a number of extensions of Petri Nets (PNs) with time have been proposed; among them are *Stochastic* Petri Nets, as well as several variants of the so-called *time* or *timed* Petri nets. Stochastic Petri nets are now well-known and a large body of work is devoted to this model whereas the theoretical properties of the other timed extensions have not been as thoroughly investigated.

Petri Nets with Time. Previous studies [2–4] consider 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 be delayed, even if this implies that some transitions are disabled because their input tokens become too old. The semantics for this class of Petri nets enjoys nice *monotonicity* properties and they 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) [5] and timed Petri nets [6], do not have such nice monotonicity properties although the number of *clocks* to be considered is

☆ Work supported by the French National Research Agency under grant DOTS ANR-06-SETI-003.
☆☆ A preliminary version of this paper appeared in Bérard et al. (2005) [1].
* Corresponding author. Tel.: +33 2 40 37 69 76; fax: +33 2 40 37 69 30.
E-mail addresses: beatrice.berard@lip6.fr (B. Bérard), Franck.Cassez@nicta.com.au (F. Cassez), haddad@lsv.ens-cachan.fr (S. Haddad), Didier.Lime@irccyn.ec-nantes.fr (D. Lime), Olivier-h.Roux@irccyn.ec-nantes.fr (O.H. Roux).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.12.005

finite (one per transition). Also those models are very popular in the Discrete Event Systems and industrial communities [7–9] 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 [10–12].

For TPNs, a transition can fire within a time interval whereas for timed Petri nets it fires as soon as possible. For timed Petri nets, time can be considered relative to places (P-timed Petri nets), arcs (A-timed Petri nets) or transitions (T-timed Petri nets) [13,14]. The same classes are defined for TPNs *i.e.* T-TPNs [5,15], A-TPNs [16] and P-TPNs [17]. A comparison of the expressiveness of these variants w.r.t. (weak) timed bisimilarity can be found in [18].

In this paper, we address the class of T-TPNs, which is the most commonly-used subclass of TPNs. It will henceforth be referred to as TPNs.

PNs with read, inhibitor and reset arcs, self-modifying nets. Petri nets can be extended by adding new types of arcs: *read arcs* enable to check the contents of a place without removing the tokens in it; *inhibitor arcs* prevent the firing of a transition if a place contains some tokens; *reset arcs* flush the input places. Petri nets with at least two inhibitor arcs (or "zero test") are Turing-powerful [19]. Moreover, in [20], the authors prove that the reachability problem is undecidable for PNs with reset arcs. In [21], it has been proved that for any PN $\mathcal{N}$ with reset arcs, there is a PN $\mathcal{N}'$ with inhibitor arcs s.t. $\mathcal{N}$ and $\mathcal{N}'$ are (weakly) bisimilar. Moreover read arcs do not add expressivity to PNs since a read arc between a place p and a transition t can be simulated by two arcs (p, t) and (t, p). This simulation does not hold for TPNs since reading the place p imposes to fire t and this will reset all clocks associated with transitions enabled by p. More broadly, the expressiveness of these arcs (read, reset and logical inhibitor) associated with TPNs is still an open problem.

Self-modifying nets [22] are yet another extensions of PNs in which the weights of the arcs can be specified either as constants or as the current marking of some place of the net. It has been shown that self-modifying nets are strictly more expressive w.r.t language acceptance than (standard) Petri nets [22]. As for the read/reset/inhibitor arcs above, the expressiveness of this extension for TPNs is also an open problem.

Timed automata. Timed automata (TA) were introduced by Alur & Dill [23,24] and have since been extensively studied. This model is an extension of finite automata with (dense time) *clocks* for the specification of real-time systems. Theoretical properties of various classes of TA have been considered in the last two decades. For instance, classes of determinizable TA such as *Event Clock Automata* are investigated in [25] and form a strict subclass of TA.

TA vs. TPNs. In a previous work [26] we have proved that TPNs form a subclass of TA in the sense that every TPN can be translated in a strongly timed bisimilar TA. This translation however needs a full state-space computation. A similar result can be found in [27], with a *syntactical* translation, but gives only a weak timed bisimulation. In another line of work, in [28], the authors compare timed state machines and time Petri nets. They give a translation from one model to another that preserves timed languages. However, they consider only constraints with closed intervals and do not deal with general timed languages (*i.e.* Büchi timed languages).

In the preliminary version of this paper [1] we showed that TA are strictly more expressive than TPNs w.r.t. weak timed bisimilarity and we proposed a translation from TA to TPNs, which preserves timed language acceptance. In [29], Berthomieu et al. extend the TPN model with specific priorities to establish an equivalence with TA w.r.t. weak timed bisimilarity. In [30], a strict subclass of TA is identified which is equivalent to bounded TPNs w.r.t. weak timed bisimilarity. In [31] the authors provide a translation from TA with diagonal constraints and general resets of clocks to TPNs, which preserves timed language acceptance. However, this translation does not include invariants in TA, introduces new deadlocks into the system and does not consider infinite timed words. Finally, [32] provides an overview of the known results about the relationships among these models.

Our contribution. In this article, we introduce generalised time Petri nets (GTPNs) as an abstract model that encompasses many of the variants of TPNs described previously. We then precisely compare the expressive power of TA vs. generalised TPNs using the notions of *timed language acceptance* and *timed bisimilarity*. This extends the previously mentioned results of [1,27] to GTPNs.

The results of the paper are summarised in Table 2: all the results are new, except the ones followed by [27] or [1] (which is the preliminary version of this paper). The names of the classes used in the sequel are defined in Table 1, where the following conventions apply: an ε subscript means that ε-transitions are allowed in the class (and not allowed otherwise), a B indicates a subclass of bounded Petri nets (and no boundedness assumption otherwise).

In the table, $\preceq_{\mathcal{L}}$, $\preceq_{\mathcal{W}}$ and $\preceq_{\mathcal{S}}$ with $\preceq \in \{<, \leq\}$ means "less expressive" for $\leq$ and "strictly less expressive" for $<$, w.r.t. respectively timed language acceptance, weak timed bisimilarity and strong timed bisimilarity (the relations can also be used the other way around: $>_{\mathcal{L}}$ means "strictly more expressive"); $=_{\mathcal{L}}$ means "equally expressive as" w.r.t. language acceptance and $=_{\mathcal{W}}$ "equally expressive as" w.r.t. weak timed bisimilarity. Fig. 1 gives a picture on how the different classes are intertwined with each other (2CM stands for 2-counter machines).

Outline of the paper. Section 2 gives notations and introduces timed languages and timed bisimulation. Section 3 introduces TA and generalised time Petri nets and show how they supersede all the extensions of TPNs (with self-modification and read/reset/inhibitor arcs). In Section 4, we extend the result of [27] to the generalised class B-*GTPN* and give a syntactical translation that preserves isomorphism of the underlying timed transitions systems. In Section 6 we focus on timed language acceptance and we propose a structural translation from TA_ε to 1-B-TPN_ε preserving timed language acceptance. We then prove that TA_ε and bounded $GTPN_\varepsilon$ are equally expressive w.r.t. timed language acceptance. This enables us to obtain new

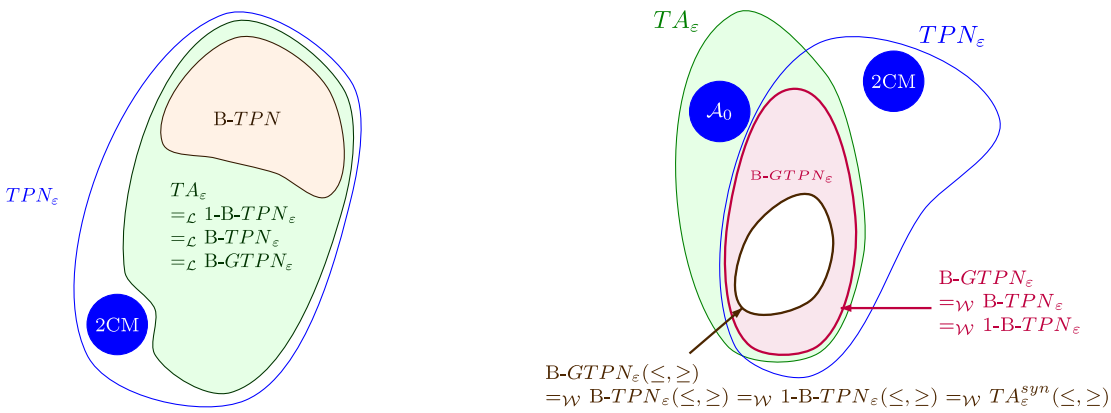
Table 1
Names of the different classes of TPNs and TA.

Name	Class
$GTPN_\varepsilon$	generalised labelled time Petri nets (with ε-transitions)
B-$GTPN_\varepsilon$	bounded $GTPN_\varepsilon$
TPN_ε	labelled time Petri nets (with ε-transitions)
B-TPN_ε	bounded TPN_ε
1-B-TPN_ε	subclass of B-TPN_ε with at most one token in each place (one safe TPN)
TA_ε	timed automata (with ε-transitions)
$\mathcal{C}lass$	for any class $\mathcal{C}lass_\varepsilon$ above, $\mathcal{C}lass$ is the subclass of $\mathcal{C}lass_\varepsilon$ without ε-transitions
$\mathcal{C}lass(\leq, \geq)$	for any class $\mathcal{C}lass$ above, $\mathcal{C}lass(\leq, \geq)$ is the subclass of $\mathcal{C}lass$ with only non-strict temporal constraints
$TA_\varepsilon^{syn}(\leq, \geq)$	syntactical subclass of TA_ε that is equivalent to B-$TPN_\varepsilon(\leq, \geq)$ (see Section 7)

Table 2
Summary of the results.

	Timed language acceptance	Timed bisimilarity
B-$GTPN$	$\leq_{\mathcal{L}}$ TA $<_{\mathcal{L}}$ 1-B-TPN_ε	$<_{\mathcal{S}}$ TA
B-$GTPN_\varepsilon$	$=_{\mathcal{L}}$ TA_ε $=_{\mathcal{L}}$ 1-B-TPN_ε $=_{\mathcal{L}}$ B-TPN_ε	$<_{\mathcal{W}}$ TA_ε
B-$GTPN_\varepsilon(\leq, \geq)$	$=_{\mathcal{L}}$ $TA_\varepsilon^{syn}(\leq, \geq)$ $=_{\mathcal{L}}$ 1-B-$TPN_\varepsilon(\leq, \geq)$	$=_{\mathcal{W}}$ $TA_\varepsilon^{syn}(\leq, \geq)$ $=_{\mathcal{W}}$ 1-B-$TPN_\varepsilon(\leq, \geq)$
TPN_ε $GTPN_\varepsilon$	$>_{\mathcal{L}}$ TA_ε	incomparable with TA_ε

	Emptiness Problem	Universality Problem
B-TPN_ε	Decidable [27]	Undecidable [1]
B-$GTPN_\varepsilon$	Decidable	Undecidable



(a) w.r.t. timed language acceptance. (b) w.r.t. timed bisimilarity.

Fig. 1. Expressiveness of TPN_ε vs. TA_ε.

results on TPNs given by Corollaries 2 to 3, page 16. Finally, in Section 7, we characterise a syntactical subclass $TA_\varepsilon^{syn}(\leq, \geq)$ of TA that is equivalent, *w.r.t.* timed bisimilarity, to the original version of bounded TPNs (with closed intervals) *i.e.* TPNs "à la Merlin"[5]. This enables us to obtain new results on TPNs "à la Merlin" given by Corollaries 7 to 10, page 19.

2. Basic notations and definitions

2.1. Notations

Let Σ be a finite alphabet, Σ^* (resp. Σ^ω) denotes the set of finite (resp. infinite) sequences of elements (or words) of Σ and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. For $w \in \Sigma^\infty$ we write $|w|$ for the *length* of w, which is ∞ if $w \in \Sigma^\omega$. We also use $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ with $\varepsilon \notin \Sigma$, where ε is both the empty word and the silent letter.

If A and B are two sets, B^A stands for the set of mappings from A to B. If A is finite and $|A| = n$, an element of B^A can be viewed as a vector of B^n. The usual operators $+, -, <$ and $=$ are used on vectors of A^n with $A \in \{\mathbb{N}, \mathbb{Q}, \mathbb{R}\}$ (which denote respectively the sets of natural, rational and real numbers) 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 $\mathbb{R}_{>0} = \mathbb{R}_{\geq 0} \setminus \{0\}$.

An *interval* is a convex subset of $\mathbb{R}_{\geq 0}$. In the sequel, we mainly use the set $\mathcal{I}(\mathbb{Q}_{\geq 0})$ of intervals with lower bound in $\mathbb{Q}_{\geq 0}$ and upper bound in $\mathbb{Q}_{\geq 0} \cup \{\infty\}$. Open intervals do not contain their bounds, closed intervals contain them and semi-open (or semi-closed) intervals contain only one of the bounds. For an interval I, we let $I^\downarrow = \{x \mid 0 \leq x \leq y \text{ for some } y \in I\}$ to be the (positive) *downward closure* of I and $I^\uparrow = \{x \mid x \geq y \text{ for some } y \in I\}$ to be the *upward closure* of I. As I is convex we have $I = I^\downarrow \cap I^\uparrow$.

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. $\mathbf{0}_X$ denotes the valuation s.t. $\forall x \in X, \nu(x) = 0$, and we omit the subscript X when it is clear from the context. An *atomic constraint* is a formula of the form $x \bowtie c$ for $x \in X$, $c \in \mathbb{Q}_{\geq 0}$ and $\bowtie \in \{<, \leq, \geq, >\}$. The constraint is said to be *non-strict* if $\bowtie \in \{\leq, \geq\}$ and *strict* if $\bowtie \in \{<, >\}$. The set of *constraints* over a set X of variables is denoted by $\mathcal{C}(X)$ and consists of 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 $\nu(x)$. We let $[\![\varphi]\!] = \{\nu \in \mathbb{R}_{\geq 0}^X \mid \varphi(\nu) = \mathsf{tt}\}$.

2.2. Timed languages and timed transition systems

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) at which the action a_k occurs. 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. We let $TW^*(\Sigma)$ (resp. $TW^\omega(\Sigma)$) for the set of finite (resp. infinite) timed words over Σ and define $TW^\infty(\Sigma) = TW^*(\Sigma) \cup TW^\omega(\Sigma)$. A *timed language* L over Σ is a subset of $TW^\infty(\Sigma)$.

Definition 1 (*Timed Transition System*)**.** A *timed transition system (TTS)* over the alphabet Σ is a tuple $S = (Q, Q_0, \Sigma_\varepsilon, \longrightarrow, \mathsf{F}, \mathsf{R})$ where:

- Q is a set of states,
- $Q_0 \subseteq Q$ is the set of *initial* states,
- Σ is disjoint from $\mathbb{R}_{\geq 0}$,
- $\longrightarrow \subseteq Q \times (\Sigma_\varepsilon \cup \mathbb{R}_{\geq 0}) \times Q$ is a set of edges. If $(q, e, q') \in \longrightarrow$, we also write $q \xrightarrow{e} q'$;
- $\mathsf{F} \subseteq Q$ and $\mathsf{R} \subseteq Q$ are respectively the set of *final* and *repeated* states. □

Notice that $q \xrightarrow{d} q'$ with $d \in \mathbb{R}_{\geq 0}$ denotes a delay transition and not an absolute time. Moreover, in the sequel, we assume that TTS satisfy the classical time-related conditions where $d, d' \in \mathbb{R}_{\geq 0}$:

- time determinism: if $q \xrightarrow{d} q'$ and $q \xrightarrow{d} q''$ then $q' = q''$;
- time additivity: if $q \xrightarrow{d} q'$ and $q' \xrightarrow{d'} q''$ then $q \xrightarrow{d+d'} q''$;
- null delay: $\forall q : q \xrightarrow{0} q$;
- time continuity: if $q \xrightarrow{d} q'$ then $\forall d' \leq d, \exists q'', q \xrightarrow{d'} q''$ and $q'' \xrightarrow{d-d'} q'$.

A *run* ρ from q_0 is a finite or infinite 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 delay transition d_n and in this case we let $last(\rho) = q_n'$ and write ρ as $q_0 \xrightarrow{d_0 a_0 \cdots d_n} q_n'$. We also write $q \xrightarrow{*} q'$ for any run ρ s.t. $first(\rho) = q$ and $last(\rho) = q'$. Given a run ρ, we can define the sequence $abs(\rho) = (a_0, D_0)(a_1, D_1) \cdots (a_n, D_n) \cdots$ with $D_i = \sum_{k=0}^{i} d_i$; notice that some actions a_i may be equal to ε. The *trace* of a run ρ, denoted by $trace(\rho)$, is the timed word obtained from $abs(\rho)$ by deleting the ε actions (thus it is a timed word over Σ). We define $Untimed(\rho) = Untimed(trace(\rho))$ and $Duration(\rho) = \sum_{d_k \in \mathbb{R}_{\geq 0}} d_k$ (this way the trace of ρ can be a finite word and at the same time the run ρ can have an infinite duration).

A run is *initial* if $first(\rho) \in Q_0$. An initial run ρ is *accepting* if:

- either ρ is a finite run and $last(\rho) \in \mathsf{F}$,
- or ρ is infinite and there exists $q \in \mathsf{R}$ that appears infinitely often on ρ.

A timed word w is *accepted* by S if there is an accepting run ρ in S of trace w. The *timed language* $\mathcal{L}(S)$ of S is the set of timed words accepted by S.

2.3. Simulation, bisimulation and isomorphism

In this section, we recall the definitions of isomorphism, similarity and bisimilarity for timed systems.

Let $S = (Q, q_0, A, \rightarrow, \mathsf{F}, \mathsf{R})$ be a TTS. Let $\rightarrow^*$ be the reflexive and transitive closure of $\rightarrow$. We denote $\mathsf{Reach}(q_0) = \{q \in Q \mid q_0 \rightarrow^* q\}$, the set of reachable states in S.

Definition 2 (*Isomorphism of TTS*)**.** Let $S_1 = (Q_1, q_0^1, A, \rightarrow_1, \mathsf{F}_1, \mathsf{R}_1)$ and $S_2 = (Q_2, q_0^2, A, \rightarrow_2, \mathsf{F}_2, \mathsf{R}_2)$ be two TTSs. S_1 and S_2 are isomorphic (we write $S_1 \cong S_2$) whenever there is a bijection $g : \mathsf{Reach}(q_0^1) \rightarrow \mathsf{Reach}(q_0^2)$ such that $\forall q \in \mathsf{Reach}(q_0^1)$ we have $q \in \mathsf{F}_1$ (Resp. R_1) iff $g(q) \in \mathsf{F}_2$ (Resp. R_2) and $\forall q, q' \in \mathsf{Reach}(q_0^1)$ we have: $q \xrightarrow{a \in A}_1 q'$ iff $g(q) \xrightarrow{a}_2 g(q')$ and $q \xrightarrow{d \in \mathbb{R}_{\geq 0}}_1 q'$ iff $g(q) \xrightarrow{d}_2 g(q')$. □

Definition 3 (*Strong Timed Similarity*)**.** Let $S_1 = (Q_1, q_0^1, \Sigma_\varepsilon, \longrightarrow_1, \mathsf{F}_1, \mathsf{R}_1)$ and $S_2 = (Q_2, q_0^2, \Sigma_\varepsilon, \longrightarrow_2, \mathsf{F}_2, \mathsf{R}_2)$ be two TTS and $\preceq$ be a binary relation over $Q_1 \times Q_2$. We write $s \preceq s'$ for $(s, s') \in \preceq$. The relation $\preceq$ is a *strong (timed) simulation relation* of S_1 by S_2 if:

1. (a) if $s_1 \in \mathsf{F}_1$ and $s_1 \preceq s_2$ then $s_2 \in \mathsf{F}_2$;
   (b) if $s_1 \in \mathsf{R}_1$ and $s_1 \preceq s_2$ then $s_2 \in \mathsf{R}_2$;
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 $s_1 \xrightarrow{a}_1 s_1'$ with $a \in \Sigma_\varepsilon \cup \mathbb{R}_{\geq 0}$ and $s_1 \preceq s_2$ then $s_2 \xrightarrow{a}_2 s_2'$ for some 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[1] of S_2 by S_1, we say that $\preceq$ is a *strong (timed) bisimulation relation* between S_1 and S_2. 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 $S_1 =_{\mathcal{S}} S_2$ in this case.

Let $S = (Q, Q_0, \Sigma_\varepsilon, \longrightarrow, \mathsf{F}, \mathsf{R})$ be a TTS. The relation $\longrightarrow_\varepsilon$ is defined by:

- $q \xrightarrow{d}_\varepsilon q'$ with $d \in \mathbb{R}_{\geq 0}$ iff there is a run ρ of the form $q \xrightarrow{*} q'$ with $Untimed(\rho) = \varepsilon$ and $Duration(\rho) = d$,
- $q \xrightarrow{a}_\varepsilon q'$ with $a \in \Sigma$ iff there is a run ρ of the form $q \xrightarrow{*} q'$ with $Untimed(\rho) = a$ and $Duration(\rho) = 0$.

Definition 4 (*Weak Timed Similarity*)**.** Let $S_1 = (Q_1, q_0^1, \Sigma_\varepsilon, \longrightarrow_1, \mathsf{F}_1, \mathsf{R}_1)$ and $S_2 = (Q_2, q_0^2, \Sigma_\varepsilon, \longrightarrow_2, \mathsf{F}_2, \mathsf{R}_2)$ be two TTS. A binary relation $\preceq$ over $Q_1 \times Q_2$ is a *weak (timed) simulation relation* of S_1 by S_2 if

1. (a) if $s_1 \in \mathsf{F}_1$ and $s_1 \preceq s_2$ then $s_2 \in \mathsf{F}_2$;
   (b) if $s_1 \in \mathsf{R}_1$ and $s_1 \preceq s_2$ then $s_2 \in \mathsf{R}_2$;
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 $s_1 \xrightarrow{a}_{1,\varepsilon} s_1'$ with $a \in \Sigma \cup \mathbb{R}_{\geq 0}$ and $s_1 \preceq s_2$ then $s_2 \xrightarrow{a}_{2,\varepsilon} s_2'$ for some s_2', and $s_1' \preceq s_2'$.

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. □

Note that $S_1 \cong S_2$ implies $S_1 \preceq_{\mathcal{S}} S_2$ implies $S_1 \preceq_{\mathcal{W}} S_2$ implies $\mathcal{L}(S_1) \subseteq \mathcal{L}(S_2)$.

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. 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 $S_1 =_{\mathcal{W}} S_2$ in this case.

3. Time Petri nets and timed automata

3.1. Time Petri nets

We consider here an extended version[2] of TPNs with accepting and repeated markings.

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

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

Time Petri nets (TPNs) were introduced by Merlin in [5] and extend Petri nets with timing constraints on the firings of transitions. In this model, a clock is associated with each *enabled* transition, and gives the elapsed time since the most 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 only progress if the enabling duration still belongs to the downward closure of the interval associated with any enabled transition.

There are different possible semantics for TPNs [33] and also various extensions of the original model (self-modification, read/inhibitor/reset arcs) and we introduce here generalised labelled TPNs which enable us to encompass the different semantics and variations in a single formalism. We then define classical TPNs and TPNs with self-modification, read/inhibitor/reset arcs as particular cases of generalised labelled TPNs.

3.1.1. Generalised labelled time Petri nets (GTPNs)

We denote by $GTPN_\varepsilon$ the class of generalised labelled time Petri nets.

Definition 5 (*Generalised Labelled Time Petri Net*)**.** A *generalised labelled time Petri net* $\mathcal{N} \in GTPN_\varepsilon$ is a tuple $(P, T, \Sigma_\varepsilon, En, Intermediate, Next, M_0, \Lambda, I, \mathsf{F}, \mathsf{R})$ where:

- $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of *places*. A *marking* of the net is an element of $\mathbb{N}^P$;
- $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of *transitions* with $P \cap T = \emptyset$;
- Σ is a finite set of *actions*;
- $En : \mathbb{N}^P \to 2^T$ is the *enabling function*. For a marking M, a transition in $En(M)$ is said to be *enabled* by M;
- $Intermediate : (\mathbb{N}^P \times T) \to \mathbb{N}^P$ is the *intermediate firing* function and we require that $Intermediate(M, t) \leq M$ for each $t \in T$;
- $Next : (\mathbb{N}^P \times T) \to \mathbb{N}^P$ is the *firing* function;
- $M_0 \in \mathbb{N}^P$ is the *initial* marking;
- $\Lambda : T \to \Sigma_\varepsilon$ is the *labelling function*;
- $I : T \to \mathcal{I}(\mathbb{Q}_{\geq 0})$ associates with each transition a *firing interval*;
- $\mathsf{F} \subseteq \mathbb{N}^P$ is the set of *final markings* and $\mathsf{R} \subseteq \mathbb{N}^P$ is the set of *repeated markings*. □

GTPNs are designed for an easy parametrisation of the key features of TPNs through the *En*, *Intermediate* and *Next* functions. *En* generalises the criteria for a transition to be enabled and *Next* generalises the computation of the marking resulting from the firing of a transition. *Intermediate* is more subtle and addresses the issue of the reset of the clock implicitly associated with an enabled transition: a transition enabled before and after the firing of some other transition, but not enabled by the intermediate marking, has its clock reset. Different variants of the semantics for TPNs, based on this notion of intermediate marking, are investigated in [33].

Under some timing constraints (see Definition 6), a transition t, enabled by marking M, can be *fired* leading to the new marking $M' = Next(M, t)$. A transition t_k is said to be *newly* enabled by the firing of the transition t_i from the marking M, (denoted by $\uparrow enabled(t_k, M, t_i)$), if the transition is enabled by the new marking $M' = Next(M, t_i)$ but was not by $Intermediate(M, t_i)$. Formally,

$$\uparrow enabled(t_k, M, t_i) = t_k \in En(Next(M, t_i)) \bigwedge \Big((t_k = t_i) \vee \neg\big(t_k \in En(Intermediate(M, t_i))\big)\Big). \tag{1}$$

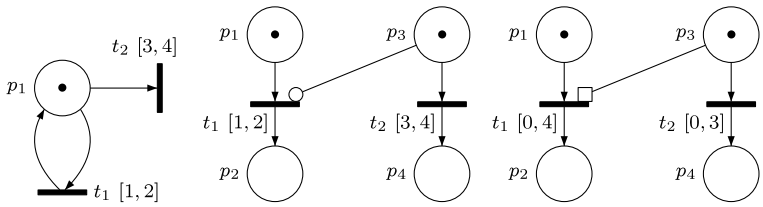
For a marking M in $\mathbb{N}^P$, $M(p_i)$ can be seen as a number of *tokens* in place p_i. To decide whether a transition t can be fired, we need to know for how long it has been continuously enabled: if this amount of time lies into the interval $I(t)$, t can actually be fired and we say that it is *firable*, 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.

We define *valuations* $\nu \in (\mathbb{R}_{\geq 0})^T$ over T so that the value $\nu(t)$ is the time elapsed since transition t was last enabled. A *state* of the GTPN $\mathcal{N}$ is a pair $(M, \nu) \in \mathbb{N}^P \times (\mathbb{R}_{\geq 0})^T$. An *admissible state* of a GTPN is a state (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 states of $\mathcal{N}$.

Definition 6 (*Semantics of a GTPN*)**.** The semantics of a generalised labelled time Petri nets $\mathcal{N} \in GTPN_\varepsilon$ with $\mathcal{N} = (P, T, \Sigma_\varepsilon, En, Intermediate, Next, M_0, \Lambda, I, \mathsf{F}, \mathsf{R})$ is the timed transition system $S_{\mathcal{N}} = (Q, \{q_0\}, \Sigma_\varepsilon, \to, \mathsf{F}', \mathsf{R}')$ where:

- $Q = ADM(\mathcal{N})$,
- $q_0 = (M_0, \mathbf{0})$, where $\mathbf{0}$ denotes the valuation with value 0 for all transitions enabled by M_0,
- $\mathsf{F}' = \{(M, \nu) \in Q \mid M \in \mathsf{F}\}$ and $\mathsf{R}' = \{(M, \nu) \in Q \mid M \in \mathsf{R}\}$,
- $\longrightarrow \in Q \times (\Sigma_\varepsilon \cup \mathbb{R}_{\geq 0}) \times Q$ consists of the discrete and continuous transition relations:
  1. the discrete transition relation is defined $\forall t \in T$ by:

$$(M, \nu) \xrightarrow{\Lambda(t)} (M', \nu') \quad \textit{iff} \quad \begin{cases} t \in En(M), \\ M' = Next(M, t), \\ \nu(t) \in I(t), \\ \forall t' \in En(M'),\ \nu'(t') = \begin{cases} 0 \ \textit{if} \ \uparrow enabled(t', M, t), \\ \nu(t') \ \textit{otherwise}. \end{cases} \end{cases}$$



(a) A simple TPN. (b) TPN with an inhibitor arc. (c) TPN with a read arc.

Fig. 2. Examples of TPN with read and inhibitor arcs.

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

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

A run of $\mathcal{N}$ is an initial run of $S_{\mathcal{N}}$ and the language accepted by $\mathcal{N}$ is $\mathcal{L}(\mathcal{N}) = \mathcal{L}(S_{\mathcal{N}})$. □

We simply write $(M, \nu) \xrightarrow{w}$ to emphasise that there is a sequence of transitions w that can be fired in $S_{\mathcal{N}}$ from (M, ν). The resulting state (M', ν') is said to be *reachable* from (M, ν). The duration of the corresponding run in the TTS is also denoted by $Duration(w)$. If $Duration(w) = 0$ we say that w is an *instantaneous firing sequence*.

Definition 7 (*Reachable State, Reachable Marking*)**.** The set of *reachable states* of $\mathcal{N} \in GTPN_{\varepsilon}$ is $\mathsf{Reach}(\mathcal{N}) = \{(M, \nu) \in \mathbb{N}^P \times (\mathbb{R}_{\geq 0})^T \mid (M_0, \mathbf{0}) \rightarrow^* (M, \nu)\}$. The set of *reachable markings* of $\mathcal{N}$ is $\mathsf{mReach}(\mathcal{N}) = \{M \in \mathbb{N}^P \mid \exists \nu \in (\mathbb{R}_{\geq 0})^T \mid (M, \nu) \in \mathsf{Reach}(\mathcal{N})\}$. □

Definition 8 (*Bounded Generalised Time Petri Nets (B-GTPN$_{\varepsilon}$)*)**.** Like for standard Petri nets, the GTPN $\mathcal{N}$ is said to be *K-bounded* if for any reachable marking M and for each place p, $M(p) \leq K$. It is *bounded* if there exists some K such that it is K-bounded. We denote by B-$GTPN_{\varepsilon}$, for the class of bounded generalised time Petri nets. □

These two previous definitions hold for all the subclasses of $GTPN_{\varepsilon}$ listed in Table 1.

3.1.2. Time Petri Nets (TPNs)

We denote by TPN_{ε} the class of time Petri nets.

Definition 9 (*Time Petri Nets*)**.** A *time Petri net* $\mathcal{N} \in TPN_{\varepsilon}$ is a *generalised labelled time Petri net* $(P, T, \Sigma_{\varepsilon}, En, Intermediate, Next, M_0, \Lambda, I, \mathsf{F}, \mathsf{R})$ for which there exist two mappings ${}^{\bullet}(.), (.)^{\bullet} : T \rightarrow \mathbb{N}^P$, called respectively the *backward* and *forward* incidence mappings, and such that $\forall M \in \mathbb{N}^P$ and $\forall t \in T$:

- $En(M) = \{t \mid (M \geq {}^{\bullet}t)\}$,
- $Next(M, t) = M - {}^{\bullet}t + t^{\bullet}$,
- $Intermediate(M, t) = M - {}^{\bullet}t$.

On the other hand, setting $\forall M \in \mathbb{N}^P$, $\forall t \in T$, $Intermediate(M, t) = M$ yields the so-called atomic semantics of TPNs [33].

The TPN Fig. 2(a) illustrates the notion of intermediate marking. For this net, the transition t_2 is never fired even if it is enabled in all reachable markings. Indeed, for each firing of the transition t_1, we have $Intermediate(M, t_1) < {}^{\bullet}t_2$ and the value $\nu(t_2)$ is reset to zero.

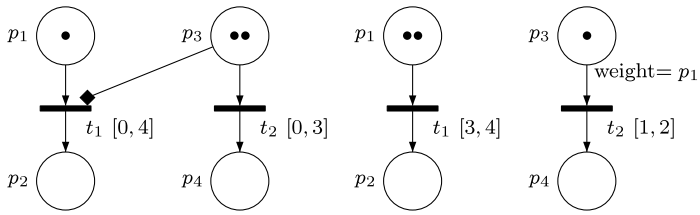
Finally, the original definition of TPN by Merlin [5] (*i.e.* TPN "à la Merlin") is the class B-$TPN_{\varepsilon}(\leq, \geq)$ of TPNs without strict constraints in the firing intervals I.

3.1.3. Time Petri nets with self-modification, read/logical inhibitor/reset arcs

A read arc functions as an input arc for the enabling of transitions. The tokens in the input place of such arcs are however not consumed when the transition fires. Inhibitor arcs are dual to read arcs in the sense that enough (w.r.t. the weight of the arc) tokens in the input place of such an arc prevents the transition to be enabled. As for read arcs, inhibitor arcs are only involved in the enabling of transitions, not in their firing. Conversely, reset arcs are ignored when deciding if a transition is enabled but empty their input places when the transition fires, regardless of their previous contents. Finally, in self-modifying nets [22], the *weight* of the input or output arcs is a function of the current marking: the weight of the arc can either be an integer, as usual, or a reference to some place of the net. In the latter case, the weight is the number of tokens that are currently in the referenced place.

Let us now consider some illustrative examples. The TPN Fig. 2(b) has an inhibitor arc between p_3 and t_1. Then, the transition t_1 cannot be fired before the firing of t_2 since it is inhibited by the token in p_3. A corresponding run is (each marking M is presented as $(M(p_1)M(p_2)M(p_3)M(p_4))$)

$$(1010) \xrightarrow{3.7} (1010) \xrightarrow{t_2} (1001) \xrightarrow{1.12} (1001) \xrightarrow{t_1} (0101).$$



(a) TPN with a reset arc. (b) A self modifying TPN.

Fig. 3. Examples of TPN with reset and self modifying TPN.

The TPN Fig. 2(c) has a read arc between p_3 and t_1. Then, if t_2 is fired first, after its firing, the firing of t_1 is not possible since the transition t_1 is not enabled anymore. The sequence t_1, t_2 is possible however since firing t_1 does not consume the token in p_3. The TPN Fig. 3(a) has a reset arc between p_3 and t_1. Then, if t_1 is fired first, after its firing, the firing of t_2 is not possible since there is no token anymore in p_3. The sequence t_2, t_2, t_1 is possible however since firing t_1 does not require any token in p_3. The self modifying TPN Fig. 3(b) has an arc between p_3 and t_2 with a weight equal to the marking of place p_1. Then t_2 cannot be fired first. After the firing of t_1 the transition t_2 can be fired whereas t_1 has to wait at least 3 time units. A corresponding run is:

$$(2010) \xrightarrow{3.7} (2010) \xrightarrow{t_1} (1110) \xrightarrow{1.8} (1110) \xrightarrow{t_2} (1101) \xrightarrow{1.3} (1101) \xrightarrow{t_1} (0201).$$

In [34], the authors showed that for Petri nets, inhibitor arcs can simulate reset arcs (and conversely). Thus reset arcs increase the expressiveness of Petri net (they are Turing-powerful) and reachability and boundedness problems are undecidable for Petri net with reset arcs. It is easy to show in the untimed setting that a read arc between a place p and a transition t is equivalent to having both an arc from p to t and an arc from t to p. In the timed setting however, this result obviously does not hold as the firing of t might disable other transitions enabled by p and thus reset their clocks. It has been shown that self-modifying nets are more expressive w.r.t language acceptance than (standard) Petri nets [22]. Here we will consider the more general setting where the weight of any arc is an arbitrary function of the current marking, i.e., a function in $\mathbb{N}^{(\mathbb{N}^P)}$.

We propose to study the expressiveness of self-modifying time Petri nets with reset, logical inhibitor, and read arcs which are classically used to extend time Petri nets. For this purpose, in this paragraph, we define these several specific types of arcs and we show how they can be seen as particular cases of GTPNs.

Definition 10 (*Self-modifying TPN with Read/Inhibitor/Reset Arcs*)**.** A *labelled self-modifying time Petri net with read/logical inhibitor/reset arcs* $\mathcal{N}$ is a tuple $(\mathcal{N}, {}^\circ(.), {}^*(.), {}^\triangledown(.))$ where:

- $\mathcal{N} = (P, T, \Sigma_\varepsilon, {}^\bullet(.), (.)^\bullet, M_0, \Lambda, I, \mathsf{F}, \mathsf{R})$ is a TPN
- ${}^\circ(.) : T \to (\mathbb{N}^{(\mathbb{N}^P)})^P$ is the *read* incidence mapping;
- ${}^*(.) : T \to (\mathbb{N}^{(\mathbb{N}^P)})^P$ is the *logical inhibitor* incidence mapping;
- ${}^\triangledown(.) : T \to \{0, 1\}^P$ is the *reset* incidence mapping; □

For any transition t, ${}^\bullet t$, $t^\bullet$, ${}^\circ t$ and ${}^* t$ are vectors of functions associating an integer to a marking. For any marking M, we denote by ${}^\bullet t(M)$, $t^\bullet(M)$, ${}^\circ t(M)$ and ${}^* t(M)$ the vectors of integers obtained by applying each component of the vectors to M.

Then, this can be easily defined in the framework of generalised TPNs. A *labelled self-modifying time Petri net with read/logical inhibitor/reset arcs* $\mathcal{N} = (P, T, \Sigma_\varepsilon, {}^\bullet(.), (.)^\bullet, {}^\circ(.), {}^*(.), {}^\triangledown(.), M_0, \Lambda, I, \mathsf{F}, \mathsf{R})$ is the *generalised labelled time Petri net* $(P, T, \Sigma_\varepsilon, En, Intermediate, Next, M_0, \Lambda, I, \mathsf{F}, \mathsf{R})$ such that $\forall M \in \mathbb{N}^P$,

- $En(M) = \{t \mid ((M \geq {}^\bullet t(M)) \text{ and } (M \geq {}^\circ t(M)) \text{ and } (M < {}^* t(M))\}$,
- $\forall t_i \in T, Next(M, t_i) = M - max({}^\triangledown t_i \times M^t, {}^\bullet t_i(M)) + t_i^\bullet(M)$ where M^t is the transposed matrix of M, $\times$ is the matrix multiplication between two vectors and $max({}^\triangledown t_i \times M^t, {}^\bullet t_i(M))$ is defined as follows : $\forall p_j \in P, max({}^\triangledown t_i \times M^t, {}^\bullet t_i(M))(p_j) = max({}^\triangledown t_i \times M^t(p_j), {}^\bullet t_i(M)(p_j))$.
- $\forall t_i \in T, Intermediate(M, t_i) = M - max({}^\triangledown t_i \times M^t, {}^\bullet t_i(M))$.

Notice that the requirement $Intermediate(M, t) \leq M$ for each $t \in T$ of Definition 5 is satisfied. Thus, labelled self-modifying time Petri nets with read, logical inhibitor and reset arcs belong to the class of generalised TPNs.

3.2. Timed automata

Timed automata were first introduced by Alur and Dill in [23,24] and extend finite automata with a finite number of clocks. We consider the model of [35] in which transitions and locations are decorated by constraints on clocks specifying respectively when the transition can be taken (guards) and when sojourn in the location is allowed (invariants).

Definition 11 (*Timed Automaton*)**.** A *timed automaton* $\mathcal{A}$ is a tuple $(L, l_0, X, \Sigma_\varepsilon, E, Inv, \mathsf{F}, \mathsf{R})$ where:

- L is a finite set of *locations*;
- $l_0 \in L$ is the *initial location*;
- X is a finite set of nonnegative real-valued *clocks*;
- Σ_ε is a finite set of *actions*;
- $E \subseteq L \times \mathcal{C}(X) \times \Sigma_\varepsilon \times 2^X \times L$ 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 $\gamma \in \mathcal{C}(X)$, 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 $\preceq \in \{<, \leq\}$;
- $\mathsf{F} \subseteq L$ is the set of *final locations* and $\mathsf{R} \subseteq L$ is the set of *repeated locations*. □

Definition 12 (*Semantics of a Timed Automaton*)**.** The semantics of the timed automaton $\mathcal{A} = (L, l_0, X, \Sigma_\varepsilon, E, Inv, \mathsf{F}, \mathsf{R})$ is the timed transition system $S_{\mathcal{A}} = (Q, q_0, \Sigma_\varepsilon, \rightarrow, \mathsf{F}', \mathsf{R}')$ with:

- $Q = \{(l, v) \in L \times (\mathbb{R}_{\geq 0})^X \mid Inv(l)(v) = \mathrm{tt}\}$,
- $q_0 = (l_0, \mathbf{0})$ is the initial state,
- $\mathsf{F}' = \{(\ell, \nu) \mid \ell \in \mathsf{F}\}$ and $\mathsf{R}' = \{(\ell, \nu) \mid \ell \in \mathsf{R}\}$,
- and $\rightarrow$ is defined by:
  1. the discrete transitions relation $(l, v) \xrightarrow{a} (l', v')$ iff $\exists\, (l, \gamma, a, R, l') \in E$ s.t. $v \in [\![\gamma]\!]$, $v' = v[R \mapsto 0]$ and $v' \in [\![Inv(l')]\!]$;
  2. the continuous transition relation $(l, v) \xrightarrow{t} (l', v')$ iff $l = l'$, $v' = v + t$ and $v' \in [\![Inv(l)]\!]$.

A run of $\mathcal{A}$ is an initial run of $S_{\mathcal{A}}$ and the language accepted by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \mathcal{L}(S_{\mathcal{A}})$. □

3.3. Expressiveness and equivalence problems

If B, B' are two timed models, TPNs or TA, we write $B =_{\mathcal{S}} B'$ (resp. $B =_{\mathcal{W}} B'$) for $S_B =_{\mathcal{S}} S_{B'}$ (resp. $S_B =_{\mathcal{W}} S_{B'}$) where S_B and $S_{B'}$ are the TTS semantics of B and B'. We write $B =_{\mathcal{L}} B'$ when $\mathcal{L}(B) = \mathcal{L}(B')$.

Let $\mathcal{C}$ and $\mathcal{C}'$ be two classes of timed model and $\bowtie \in \{\mathcal{L}, \mathcal{W}, \mathcal{S}, \mathcal{I}\}$ respectively for timed language, weak and strong timed bisimilarity and isomorphism of TTS.

Definition 13 (*Expressiveness w.r.t.* $\bowtie$)**.** $\mathcal{C}$ is *more expressive* than $\mathcal{C}'$ w.r.t. $\bowtie$, written $\mathcal{C}' \leq_{\bowtie} \mathcal{C}$, if for all $B' \in \mathcal{C}'$ there is a $B \in \mathcal{C}$ s.t. $B =_{\bowtie} B'$. If moreover there is some $B \in \mathcal{C}$ s.t. there is no $B' \in \mathcal{C}'$ with $B =_{\bowtie} B'$, then $\mathcal{C}' <_{\bowtie} \mathcal{C}$ (read "strictly more expressive"). If both $\mathcal{C}' \leq_{\bowtie} \mathcal{C}$ and $\mathcal{C} \leq_{\bowtie} \mathcal{C}'$ then $\mathcal{C}$ and $\mathcal{C}'$ are equally expressive w.r.t. $\bowtie$ and we write $\mathcal{C} =_{\bowtie} \mathcal{C}'$. □

4. From generalised time Petri nets to timed automata

We first recall the following theorem from [27]:

Theorem 1 (*[27]*)**.** *For any $\mathcal{N} \in$ B-TPN$_\varepsilon$ there is a TA $\mathcal{A} \in TA_\varepsilon$ s.t. $\mathcal{N} =_{\mathcal{W}} \mathcal{A}$, hence B-TPN$_\varepsilon \leq_{\mathcal{W}} TA_\varepsilon$. Moreover, we also have that B-TPN$_\varepsilon(\leq, \geq) \leq_{\mathcal{W}} TA_\varepsilon(\leq, \geq)$.*

This previous result was obtained by a structural translation from TPNs to TA preserving weak timed bisimilarity. In this paper, we extend and strengthen this previous result: we give a syntactical translation from B-*GTPN*$_\varepsilon$ to (products of) timed automata that preserves *isomorphism* of the semantics and thus strong timed bisimilarity.

We define our translation using products of timed automata with a finite number of shared bounded integer variables. They are equally expressive as timed automata, since each variable can be encoded by a finite automaton. A TA with shared variables has an additional set of integer variables V and we therefore extend its notation to $A = (L, \mathbf{l}_0, C, V, \Sigma_\varepsilon, E, Inv, \mathsf{F}, \mathsf{R})$. We classically allow tests and updates of integer variables on transitions. To synchronise transitions, we use a distinct synchronisation alphabet Σ_s. An edge of such a TA component in the product is therefore a tuple $\langle l, \gamma, s, a, U, R, l' \rangle \in E$ from the location l to the location l' with the guard $\gamma \in \mathcal{C}(X)$, the synchronisation action $s \in \{b!, b?\}$ with $b \in \Sigma_s$, the label $a \in \Sigma_\varepsilon$, the update of shared variables U (where U is either $\emptyset$ or the conjunction of atomic updates $v := k$ with $v \in V$ and $k \in \mathbb{N}$) and the reset set $R \subseteq X$. We also impose that all the transitions of the product are synchronised.

Definition 14 (*Synchronised Product of Timed Automata with Variables*)**.** Let $A_1, \ldots, A_n$ be n timed automata with $A_i = (L_i, l_{i,0}, C_i, V, \Sigma_\varepsilon \cup \Sigma_s, E_i, Inv_i, \mathsf{F}_i, \mathsf{R}_i)$, The synchronised product of TA $(A_1 | \ldots | A_n)$ is a timed automaton $A = (L, \mathbf{l}_0, C, V, \Sigma_\varepsilon, E, Inv, \mathsf{F}, \mathsf{R})$ where:

- $L = L_1 \times L_2 \times \cdots \times L_n$,
- the initial location of A is $\mathbf{l}_0 = (l_{1,0}, l_{2,0}, \ldots, l_{n,0})$,
- the set of clocks of A is $C = \cup_{i=1}^n C_i$,
- $\mathbf{l} \xrightarrow{\gamma, a, U, R} \mathbf{l}' \in E$ iff $\forall i, \exists (\mathbf{l}[i], \gamma_i, s_i, a_i, U_i, R_i, \mathbf{l}'[i]) \in E_i$ s.t.:

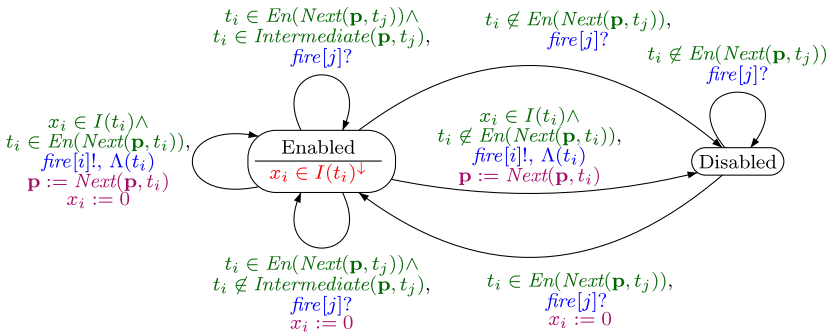


Fig. 4. Automaton $\mathcal{A}_i$ – Translation of a transition t_i.

- there exists a unique j such that $s_j = b!$ and $a_j = a$,
- $\forall i \neq j$, we have $s_i = b?$, $a_i = \varepsilon$ and $U_i = \emptyset$,
- $\gamma = \bigwedge_i \gamma_i$ and $R = \bigcup_i R_i$.

- for all $\mathbf{l} = (l_1, \ldots, l_n) \in L$, $Inv(\mathbf{l}) = \bigwedge_{i=1}^{n} Inv(l_i)$,
- $\mathsf{F} \subseteq L$ and $\mathsf{R} \subseteq L$ are arbitrary sets and they will be defined on a product when necessary. □

Assume we are given a $GTPN_\varepsilon$ $\mathcal{N} = (P, T, \Sigma_\varepsilon, En, Intermediate, Next, M_0, \Lambda, I, \mathsf{F}, \mathsf{R})$ with $P = \{p_1, \ldots, p_m\}$ and $T = \{t_1, \ldots, t_n\}$.

We build one timed automaton $\mathcal{A}_i$ for each transition t_i (Fig. 4) and synchronise them to faithfully simulate $\mathcal{N}$. The idea of the translation is as follows: the current marking of the net is given by a shared/global array variable $\mathbf{p}$ of size m: $\mathbf{p}[k]$, $1 \leq k \leq m$ gives the number of tokens in place p_k (note that we consider bounded TPN then $\mathbf{p}$ can be Each $\mathcal{A}_i$ has its own local clock x_i that records the time since transition t_i was last enabled (it holds the value of $\nu(t_i)$). Furthermore, each transition is either *enabled* or *disabled*, and $\mathcal{A}_i$ has two locations to represent this status.

We associate a synchronisation action $fire[i]$ with each transition t_i. To simulate the firing of t_i (and the update of $t_j, j \neq i$) we let $\mathcal{A}_i$ make the action $fire[i]!$ and force the other automata to make $fire[i]?$. To ensure this, we make sure that the automata $\mathcal{A}_j, j \neq i$, can all make $fire[i]?$ and thus are forced to synchronise: this is achieved by ensuring that $fire[i]?$ is always enabled in any location of $\mathcal{A}_j, j \neq i$.

We first explain what happens when transition t_i is fired. This corresponds to the two transitions labelled with $fire[j]!$. First, notice that clock x_i of $\mathcal{A}_i$ holds the value of $\nu(t_i)$ and thus t_i is firable iff $x_i \in I(t_i)$ which is enforced by the guard $x_i \in I(t_i)$ and the invariant $x_i \in I(t_i)^{\downarrow}$ in the *Enabled* location. The automaton is in location *Enabled* iff transition $t_i \in En(\mathbf{p})$ and we maintain this invariant for any transition.

There are two possible results when firing t_i: either the transition is disabled and in this case we reach location *Disabled* or it is still enabled after the firing and we stay in location *Enabled*. In the latter case, the transition is newly enabled (Eq. (1) evaluates to true) and we reset its clock. In the other case, x_i is not used in location *Disabled* and we may safely reset it or leave it unchanged.

Now consider that another transition $t_j \neq t_i$ is fired. This corresponds to the other transitions labelled with $fire[j]?$. As required earlier, in each location, $fire[j]?, j \neq i$ is enabled: the guards of all transitions are disjoint and their union is equivalent to tt. The target location when firing another transition $j \neq i$ is fully determined by the formal definitions of *En*, *Intermediate* and *Next*. Notice that there is one copy of each transition in Fig. 4 for each $j \neq i$.

The whole (synchronised) system is obtained by the synchronisation of the timed automata $\mathcal{A}_i$, $1 \leq i \leq n$. The final and repeated states are those for which the marking $\mathbf{p} \in \mathsf{F}$ and $\mathbf{p} \in \mathsf{R}$ respectively.

Let $\Delta(\mathcal{N}) = (\mathcal{A}_1 \mid \mathcal{A}_2 \mid \cdots \cdots \mid \mathcal{A}_n)$ be the product of timed automata with shared variables obtained by the translation of the $GTPN_\varepsilon$ $\mathcal{N}$. The size of each automaton $\mathcal{A}_i$ is linear w.r.t. the number of transitions of $\mathcal{N}$ and since the product is synchronised on all transitions, in each state of the product, at most one edge per transition of the net can be effectively taken. The size of the product is therefore exponential w.r.t. the size of the net. Moreover, since we consider bounded nets, $\mathbf{p}$ can be encoded by a finite automaton with one state per marking and the size of this finite automaton is then exponential w.r.t. the number of places of the net $\mathcal{N}$. Thus the size of $\Delta(\mathcal{N})$ is exponential w.r.t. the size of $\mathcal{N}$.

Let $S_{\mathcal{N}}$ be the TTS that gives the semantics of $\mathcal{N}$. Let $\Delta(\mathcal{N}) = (\mathcal{A}_1 \mid \mathcal{A}_2 \mid \cdots \cdots \mid \mathcal{A}_n)$ be the product of timed automata obtained by the translation above and $S_{\Delta(\mathcal{N})}$ its semantics.

Proposition 1. *$S_{\mathcal{N}}$ and $S_{\Delta(\mathcal{N})}$ are isomorphic.*

Proof. We relate the states of $\mathcal{N}$ to the states of $\Delta(\mathcal{N})$. Let (M, ν) and $(\mathbf{p}, \mathbf{q}, \mathbf{x})$ be, respectively, a state of $S_{\mathcal{N}}$ and a state of $S_{\Delta(\mathcal{N})}$ where $\mathbf{q}$ gives the product location of $(\mathcal{A}_1 | \cdots | \mathcal{A}_n)$ *i.e.* for $1 \leq i \leq n$, $\mathbf{q}[i]$ gives the location of $\mathcal{A}_i$, and $\mathbf{x}[i]$, $i \in [1..n]$ gives the value of the clock x_i.

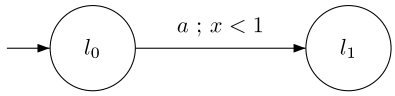


Fig. 5. The timed automaton $\mathcal{A}_0$.

Let g be the mapping defined by $g\big((M,\nu)\big) = (\mathbf{p},\mathbf{q},\mathbf{x})$ iff:

- $\forall 1 \le i \le m$, $\mathbf{p}[i] = M(p_i)$,
- $\forall 1 \le i \le n$, $\mathbf{q}[i] = \begin{cases} \text{Enabled } \textit{if } t_i \in En(M), \\ \text{Disabled } \textit{otherwise.} \end{cases}$
- $\forall 1 \le i \le m$, $\mathbf{x}[i] = \nu(t_i)$.

It is easy to see that g is a bijection. Let R_Δ and F_Δ be respectively the final and repeated states of $\Delta(\mathcal{N})$. As defined previously, these sets are respectively those for which the marking $\mathbf{p} \in \mathsf{F}$ and $\mathbf{p} \in \mathsf{R}$. Then for all states (M,ν) of $S_{\mathcal{N}}$ we have $(M,\nu) \in \mathsf{F}$ (Resp. R) iff $g((M,\nu)) \in \mathsf{R}_\Delta$ (Resp. F_Δ). To prove isomorphism (Definition 2), we have to prove that continuous transitions and discrete transitions can be executed by each of the transitions systems. Let (M,ν) be a state of $S_{\mathcal{N}}$ and $g\big((M,\nu)\big) = (\mathbf{p},\mathbf{q},\mathbf{x})$.

Continuous transitions. First notice that if a continuous transition of duration d is allowed in $S_{\Delta(\mathcal{N})}$ from $(\mathbf{p},\mathbf{q},\mathbf{x})$, $(\mathbf{p},\mathbf{q},\mathbf{x}) \xrightarrow{d} (\mathbf{p},\mathbf{q},\mathbf{x}')$ and allowed in $S_{\mathcal{N}}$ from (M,ν), $(M,\nu) \xrightarrow{d} (M,\nu')$, we have $(M,\nu') = (M,\nu+d)$ and $(\mathbf{p},\mathbf{q},\mathbf{x}+d) = (\mathbf{p},\mathbf{q},\mathbf{x}')$ and thus $g\big((M,\nu')\big) = (\mathbf{p},\mathbf{q},\mathbf{x}')$.

It remains to prove that $S_{\mathcal{N}}$ can make a transition of duration d from (M,ν) iff $S_{\Delta(\mathcal{N})}$ can do the same from $(\mathbf{p},\mathbf{q},\mathbf{x})$. $S_{\mathcal{N}}$ can make a transition of duration d from (M,ν) iff $\forall 1 \le i \le n$:

$$\begin{aligned} t_i \in En(M) &\implies \nu(t_i) + d \in I(t_i)^{\downarrow} \\ [t_i \in En(M) \iff \mathbf{q}[i] = \textit{Enabled}] &\iff \mathbf{q}[i] = \textit{Enabled} \implies \nu(t_i) + d \in I(t_i)^{\downarrow} \\ [\mathbf{x}[i] = \nu(t_i)] &\iff \mathbf{q}[i] = \textit{Enabled} \implies \mathbf{x}[i] + d \in I(t_i)^{\downarrow} \end{aligned}$$

which is equivalent to $S_{\Delta(\mathcal{N})}$ can make a transition of duration d.

Discrete transitions. Let us now consider the discrete transition $(M,\nu) \xrightarrow{\Lambda(t_i)} (M',\nu')$. As t_i can be fired we must have: (1) $t_i \in En(M)$, (2) $\nu(t_i) \in I(t_i)$. As $g\big((M,\nu)\big) = (\mathbf{p},\mathbf{q},\mathbf{x})$, this is equivalent to (1) $\mathbf{q}[i] = \textit{Enabled}$ and (2) $\mathbf{x}[i] \in I(t_i)$ and thus *fire*$[i]$! can be triggered in $S_{\Delta(\mathcal{N})}$ and we have $(\mathbf{p},\mathbf{q},\mathbf{v}) \xrightarrow{\Lambda(t_i)} (\mathbf{p}',\mathbf{q}',\mathbf{x}')$. The updates in $\Delta(\mathcal{N})$ are directly computed using the functions *En*, *Intermediate* and *Next* defined for $GTPN_\varepsilon$, then we have: (1) $\mathbf{p}' = \textit{Next}(\mathbf{p},t_i)$ (*i.e.* $M' = \textit{Next}(M,t_i)$) and $\forall t_j \in \textit{En}(\textit{Next}(\mathbf{p},t_i))$, (2) $\mathbf{x}'[j] = 0$ if $t_j \notin \textit{Intermediate}(\mathbf{p},t_i)$ (*i.e.* $\uparrow\textit{enabled}(t_j,M,t_i)$), and $\mathbf{x}'[j] = \mathbf{x}[j]$ otherwise. Thus $g\big((M',\nu')\big) = (\mathbf{p}',\mathbf{q}',\mathbf{x}')$. $\square$

This enables us to obtain the following results:

Theorem 2. *For any $\mathcal{N} \in$ B-GTPN (resp. B-GTPN$_\varepsilon$) there is a TA $\Delta(\mathcal{N}) \in$ TA (resp. TA$_\varepsilon$) s.t. $S_{\mathcal{N}} \cong S_{\mathcal{A}}$.*

Theorem 2 implies:

Theorem 3. *B-GTPN $\le_{\mathcal{I}}$ TA and B-GTPN$_\varepsilon$ $\le_{\mathcal{I}}$ TA$_\varepsilon$.*

Remark 1. Isomorphism of TTS implies all the other equivalences, and thus Theorem 2 also implies the same order for all other equivalences. Notice also that the order applies to subclasses of B-*GTPN* where the constraints are restricted: for instance B-*GTPN*$(\le,\ge)$ $\le_{\mathcal{I}}$ *TA*$(\le,\ge)$ as the same comparison operators are used in $\mathcal{N}$ and $\Delta(\mathcal{N})$.

Remark 2. Starting from a GTPN $\mathcal{N} \in$ B-*GTPN*$_\varepsilon(\le,\ge)$ the translation from GTPN to TA gives a TA $\mathcal{A}$ in the subclass $TA_\varepsilon^{syn}(\le,\ge)$ defined in Section 7 (Definition 15). Thus B-*GTPN*$(\le,\ge)$ $\le_{\mathcal{I}}$ $TA_\varepsilon^{syn}(\le,\ge)$.

5. Strict ordering results

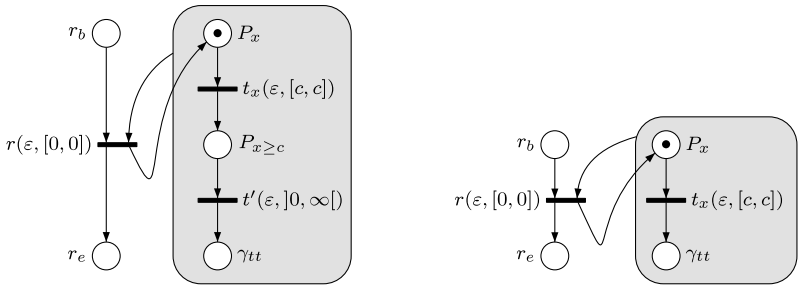
In this section, we recall some results from [1] and extend them to bounded GTPNs proving that they are strictly less expressive w.r.t. weak timed bisimilarity than timed automata.

Consider the timed automata $\mathcal{A}_0$ of Fig. 5 and $\mathcal{A}_1$ which is as $\mathcal{A}_0$ with the guards $x < 1$ replaced by $x \le 1$.

Theorem 4 as been proved in [1] for TPNs and easily extends to GTPNs:

Theorem 4. *There is no GTPN$_\varepsilon$ weakly timed bisimilar to $\mathcal{A}_0 \in$ TA (Fig. 5) or to $\mathcal{A}_1 \in$ TA$(\le,\ge)$.*

We can deduce several new interesting results from the previous theorems. These new results are expressed by the following corollaries:



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

Fig. 6. Widgets for $\mathcal{N}_{x>c}$ and $\mathcal{N}_{x\geq c}$.

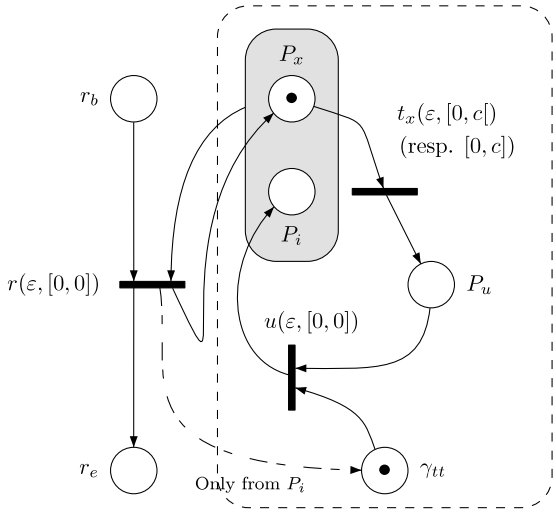


Fig. 7. Widget $\mathcal{N}_{x<c}$ (resp. $\mathcal{N}_{x\leq c}$).

Corollary 1. *B-GTPN* $<_{\mathcal{S}}$ *TA, B-GTPN*$_\varepsilon$ $<_{\mathcal{W}}$ *TA*$_\varepsilon$ *and B-GTPN*$_\varepsilon(\leq, \geq) <_{\mathcal{W}}$ *TA*$_\varepsilon(\leq, \geq)$.

Proof. Theorem 3 states that B-*GTPN* $\leq_{\mathcal{S}}$ *TA* and B-*GTPN*$_\varepsilon$ $\leq_{\mathcal{W}}$ *TA*$_\varepsilon$ and Theorem 4 implies the strict relation. By Remark 1 B-*GTPN*$_\varepsilon(\leq, \geq) \leq_{\mathcal{W}}$ *TA*$_\varepsilon(\leq, \geq)$ and Theorem 4 implies the strict relation. □

Following these "negative" results, we compare the expressiveness of bounded TPNs and TA w.r.t. to timed language acceptance and then characterise a subclass of TA that admits bisimilar bounded TPNs.

6. Equivalence w.r.t. timed language acceptance

In this section, we prove that TA, safe TPNs and bounded GTPNs are equally expressive w.r.t. timed language acceptance, and give an effective syntactical translation from TA to a subclass of GTPNs (1-safe TPNs). The result of Proposition 2, page 13 already appeared in [1], and in this paper we improve the translation, give the full proof, and some new consequences of this result.

Let $\mathcal{A} = (L, l_0, X, \Sigma_\varepsilon, E, Act, Inv, \mathsf{F}, \mathsf{R})$ be a TA. Since we are only concerned in this section with the language accepted by $\mathcal{A}$ we assume the invariant function *Inv* is uniformly true and the original constraints of the invariants are instead added to the guards of transitions. Let $\mathcal{C}_x$ be the set of atomic constraints on clock x that are used in $\mathcal{A}$. The TPN resulting from our translation is built from "elementary blocks" modelling 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 Figs. 6 and 7. 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 notation: 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. 6(b) has 2 copies of the 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. 7 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).

We also 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. 6(b) only appears with $c > 0$. Each of these TPNs basically consists of a "constraint" sub-part (in

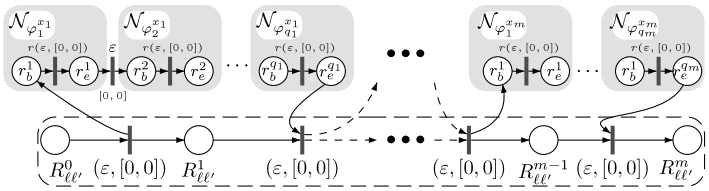


Fig. 8. Widget $\mathcal{N}_{Reset(R)}$ to reset the widgets of the constraints of clocks x_i, $1 \leq i \leq m$.

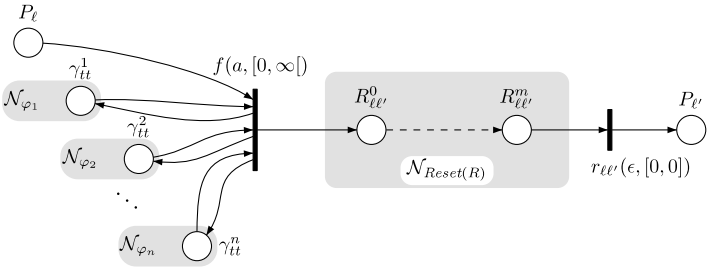


Fig. 9. Widget $\mathcal{N}_e$ of an edge $e = (\ell, \gamma, a, R, \ell')$.

the grey boxes for Fig. 6 and in the dashed box for Fig. 7) that models the truth value of the atomic constraint, and another "reset" sub-part that will be used to update the truth value of the constraint when the clock x is reset.
The "constraint" sub-part 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 modelling a constraint on x must be set to their *initial* marking which has one token in P_x for Fig. 6 and one token in P_x and γ_{tt} for Fig. 7. Our strategy to reset a block modelling a constraint is to put a token in the place r_b (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 modelling constraints on a clock x, we chain all of them in some arbitrary order, the r_e place of the ith block is linked to the r_b place of the $i+1$th block, *via* a 0 time unit transition ε. Assume $R \subseteq X$ is a non empty set of clocks. To reset all the widgets in the scope of R, we connect the reset chains in some arbitrary order as illustrated in Fig. 8. For an edge $(\ell, \gamma, a, R, \ell')$, we denote by $R^0_{\ell\ell'}$ the first place of this widget and $R^m_{\ell\ell'}$ the last one. The picture inside the dashed box in Fig. 8 denotes the widget $\mathcal{N}_{Reset(R)}$. To update the (truth value of the) widgets of $D(R)$ it then suffices to put a token in $R^0_{\ell\ell'}$. In null duration it will go to $R^n_{\ell\ell'}$ and have the effect of updating each widget of $D(R)$ on its way.

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. create a transition $f(a, [0, \infty[)$ and if $m \geq 1$ (i.e. $R \neq \emptyset$) another one $r_{\ell\ell'}(\epsilon, [0, 0])$,
2. connect them to the places of the widgets $\mathcal{N}_{\varphi_i}$ and $\mathcal{N}_{Reset(R)}$ as described on Fig. 9. In case $\gamma = \mathrm{tt}$ (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 $r_{\ell\ell'}(\epsilon, [0, 0])$ 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 $\ell \in \mathsf{F}$ and the set of repeated markings by the set of markings M s.t. $M(P_\ell) = 1$ for $\ell \in \mathsf{R}$.

We note $\Delta(\mathcal{A})$ 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 constant 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 [28]. 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 $\mathcal{A} \preceq_2 \Delta(\mathcal{A})$.

Proposition 2. $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\Delta(\mathcal{A}))$.

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 $\mathcal{A}$ weakly simulates $\Delta(\mathcal{A})$ which entails $\mathcal{L}(\Delta(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$ and thus $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\Delta(\mathcal{A}))$. It is sufficient to give the proof for the case $\mathcal{A}$ has no ε transitions. In case $\mathcal{A}$ has ε transitions, ε is treated as an ordinary action.

Let $\mathcal{A} = (L, l_0, C, A, E, Act, Inv, \mathsf{F}, \mathsf{R})$ and $\Delta(\mathcal{A}) = (P, T, A_\varepsilon, {}^\bullet(.), (.)^\bullet, M_0, \Lambda, \Gamma, F_\Delta, R_\Delta)$. Assume $C = \{x_1, \ldots, x_k\}$, $P = \{p_1, \ldots, p_m\}$ and $T = \{t_1, \ldots, t_n\}$. We denote the set of atomic constraints of $\mathcal{A}$ by $\mathcal{C}_\mathcal{A}$ and the set of atomic constraints of $\mathcal{A}$ on clock x by $\mathcal{C}_\mathcal{A}(x)$.

In the sequel, the name of places and transitions of a widget $\mathcal{N}_\varphi$ are superscripted by φ. For example, for a constraint $\varphi = x \geq c$, the places γ_{tt} and P_u of a widget $\mathcal{N}_\varphi$ are respectively written γ_{tt}^φ and P_u^φ.

Proof of $\Delta(\mathcal{A})$ 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:

$$(\ell, v) \preceq (M, \nu) \iff \begin{cases} (1)\, M(P_\ell) = 1 \\ (2)\, \forall \varphi \in \{x < c, x \leq c\},\ M(P_u^\varphi) = 0 \\ (3)\, \forall \varphi \in \mathcal{C}_\mathcal{A},\ v \in [\![\varphi]\!] \iff M(\gamma_{tt}^\varphi) = 1 \\ (4)\, \forall \varphi \in \mathcal{C}_\mathcal{A}(x),\ \nu(t_x^\varphi) = v(x). \end{cases} \tag{I}$$

We can now 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 $\rho = (M, \nu) \xrightarrow{*} (M', \nu')$ with $Duration(trace(\rho)) = d$ and $Untimed(trace(\rho)) = \varepsilon$. 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 $(\ell, v + d) \preceq (M', \nu')$ it remains to prove items (2) and (3) of Eq. (I).
   Let $\varphi = x \bowtie c$ with $\bowtie \in \{<, \leq\}$.
   - if $v \in [\![\varphi]\!]$ and $v + d \notin [\![\varphi]\!]$, then, from item (4), there is some $d' \leq d$ s.t. transition t_x^φ of widget $\mathcal{N}_\varphi$ is enabled and it must be fired before φ becomes false. Thus t_x^φ is fired at d' (which is possible as there is no token in P_u^φ and thus the token is in P_x^φ) and subsequently u^φ in the same widget, thus transferring the tokens from P_x^φ, γ_{tt}^φ to P_i^φ.
   - if $v \in [\![\varphi]\!]$ and $v + d \in [\![\varphi]\!]$, it is possible to do nothing in widget $\mathcal{N}_\varphi$ and let the token in P_x^φ and γ_{tt}^φ.
   - if $v \notin [\![\varphi]\!]$ then $v + d \notin [\![\varphi]\!]$, then there must be a token in P_i^φ and we let time elapse without firing any transition.
   Let $\varphi = x \bowtie c$ with $\bowtie \in \{>, \geq\}$.
   - if $v \in [\![\varphi]\!]$ then $v + d \in [\![\varphi]\!]$ and $M(\gamma_{tt}^\varphi) = 1$. We just let time elapse in $\mathcal{N}_\varphi$.
   - if $v \notin [\![\varphi]\!]$ and $v + d \in [\![\varphi]\!]$, there is $d' \leq d$ s.t. transitions t_x^φ must be fired (and t'^φ 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 $v \notin [\![\varphi]\!]$ and $v + d \notin [\![\varphi]\!]$ we also let time elapse and leave a token in P_x^φ.

   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 Eq. (I). Taken separately we have for each constraint $(\ell, v) \preceq (M_\varphi, \nu_\varphi)$. It is not difficult[3] to build a run ρ with an interleaving of the previous runs ρ_φ s.t. $\rho = (M, \nu) \xrightarrow{d}_\varepsilon (M', \nu')$ and (M', ν') satisfies requirements (2) and (3) of Eq. (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 12, $v \in [\![\varphi_i]\!]$ for $1 \leq i \leq n$. This implies $M(\gamma_{tt}^{\varphi_i}) = 1$ (definition of $\preceq$). Thus the transition $f(a, [0, \infty[)$ is firable in the widget $\mathcal{N}_e$ leading to (M', ν'). From there on we do not change the markings 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 $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\Delta(\mathcal{A}))$.

Proof of $\mathcal{L}(\Delta(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$. We can now build a simulation relation of $\Delta(\mathcal{A})$ by $\mathcal{A}$. We first define the following boolean conditions for a clock $x \in X$, for all the widgets of $\Delta(\mathcal{A})$ involving the clock x (and associated with a constraint $\varphi \in \mathcal{C}_\mathcal{A}(x)$), and given a state $(M, \nu) \in (\mathbb{N}^p \times \mathbb{R}^m_{\geq 0})$ of $\Delta(\mathcal{A})$ and a state $(\ell, v) \in (L \times \mathbb{R}^n_{\geq 0})$ of $\mathcal{A}$:

$$C1(x) = \Big(\forall \varphi = (x > c),\ v \in [\![\varphi]\!] \iff M(\gamma_{tt}^\varphi) = 1 \vee (M(P_{x\geq c}^\varphi) = 1 \wedge \nu(t'^\varphi) > 0)\Big)$$

$$C2(x) = \Big(\forall \varphi = (x \geq c),\ v \in [\![\varphi]\!] \iff M(\gamma_{tt}^\varphi) = 1 \vee (M(P_x^\varphi) = 1 \wedge \nu(t_x^\varphi) = c)\Big)$$

$$C3(x) = \Big(\forall \varphi \in \{x < c, x \leq c\}, v \notin [\![\varphi]\!] \Rightarrow M(P_i^\varphi) = 1\Big).$$

Note that these conditions imply for all widgets: $M(\gamma_{tt}^\varphi) = 1 \Rightarrow v \in [\![\varphi]\!]$.

[3] 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.

We define the relation $\preceq \subseteq (\mathbb{N}^p \times \mathbb{R}^m_{\geq 0}) \times (L \times \mathbb{R}^n_{\geq 0})$ by:

$$(M, \nu) \preceq (\ell, v) \iff \begin{cases} M(P_\ell) = 1 \Rightarrow \forall x \in X, \begin{cases} C1(x) \wedge C2(x) \wedge C3(x) \\ \forall \varphi \in \mathcal{C}_{\mathcal{A}}(x), \nu(t_x^\varphi) = v(x) \end{cases} \\ M(P_\ell) = 0 \Rightarrow \begin{cases} \exists e = (\ell_\bullet, \gamma, a, R, \ell) \in E \\ st \begin{cases} \sum_{i=0}^{m} M(R^i_{\ell_\bullet \ell}) = 1 \\ \forall x \notin R, \begin{cases} C1(x) \wedge C2(x) \wedge C3(x) \\ \forall \varphi \in \mathcal{C}_{\mathcal{A}}(x), \nu(t_x^\varphi) = v(x). \end{cases} \end{cases} \end{cases} \end{cases} \tag{II}$$

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

- the property on final and repeated states is satisfied by definition of $\mathcal{A}$,
- for the initial configuration, it is clear that $(M_0, \mathbf{0}) \preceq (l_0, \mathbf{0})$,
- continuous time transitions: let $(M, \nu) \xrightarrow{d} (M', \nu')$ with $d \geq 0$ and $M = M'$. Let $(M, \nu) \preceq (\ell, v)$. Since the traversal of a reset widget is in null duration, and since time can elapse from (M, ν), we have $M(P_\ell) = 1$. As there are no invariants in $\mathcal{A}$, time d can elapse from (ℓ, v). For all widgets associated to a constraint $\varphi \in \mathcal{C}_{\mathcal{A}}(x)$, we have $\nu'(t_x^\varphi) = \nu(t_x^\varphi) + d = v(x) + d$ (even if t_x^φ is not enabled). If no ε transition fires in the TPN, then either the truth values of the constraints stay unchanged or three cases can occur:
  - there is some clock x and constraint $\varphi = (x > c)$ such that $v(x) = c$ and $M(P^\varphi_{x \geq c}) = 1$ and $\nu(t'^\varphi) = 0$. Then $v + d \in [\![\varphi]\!]$ and $\nu(t'^\varphi) = d$ and $\forall x \in X$ the condition $C1(x)$ is respected.
  - there is some clock x and constraint $\varphi = (x \geq c)$ such that $v(x) < c$ and $v(x) + d = c$ and $M(P^\varphi_x) = 1$. Then we have $v \notin [\![\varphi]\!]$ and $\nu(t_x{}^\varphi) = v(x) < c$. Moreover we have $\nu'(t_x{}^\varphi) = \nu(t_x{}^\varphi) + d = c$, $v'(x) = c$ and $v' \in [\![\varphi]\!]$ and the condition $C2(x)$ remains true.
  - there is some clocks x and constraints $\varphi \in \{(x < c), (x \leq c)\}$ such that $v(x) < c$, $v(x) + d > c$ and $M(P^\varphi_i) = 1$ (it means that the guard was true in $\mathcal{A}$ whereas the widget of $\Delta(\mathcal{A})$ considered the guard false and the guard becomes false for both). Then $v + d \notin [\![\varphi]\!]$ and since $M(P^\varphi_i) = 1$, the condition $C3(x)$ remains true.

  Then, the condition $C1(x) \wedge C2(x) \wedge C3(x)$ remains true.

  Thus $(\ell, v) \xrightarrow{d} (\ell, v + d)$ in $\mathcal{A}$ s.t. $(M', \nu') \preceq (\ell, v + d)$.
- discrete transitions: let $(M, \nu) \xrightarrow{a} (M', \nu')$. We distinguish the cases $a = \varepsilon$ and $a \in \Sigma$ and when $a = \varepsilon$, we distinguish the cases $M(P_\ell) = 0$ and $M(P_\ell) = 1$.
  If $a = \varepsilon$ and $M(P_\ell) = 1$ then we are updating some widget $\mathcal{N}_\varphi$ (ε transition is not a reset transition because a reset can only occur 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^φ or t'^φ is fired. If t_x^φ is fired then the time elapsed since the x was last reset is equal to c. Thus $M(\gamma^\varphi_{tt}) = 0$ 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 $(M', \nu') \preceq (\ell, v)$.
  - update of a widget $\mathcal{N}_{x \geq c}$: the same reasoning as before can be used and leads to $(M', \nu') \preceq (\ell, v)$.
  - update of a widget $\mathcal{N}_{x<c}$: In this case either t_x^φ or u^φ is fired. Assume t_x^φ is fired. Thus $M'(P^\varphi_i) = 0$. The time elapsed since x was last reset is strictly less than c and $v \in [\![\varphi]\!]$. Thus $(M', \nu') \preceq (\ell, v)$. Now assume u^φ is fired. Again $M(P^\varphi_i) = 0$ and thus $v(x) < c$. This time $M'(P^\varphi_i) = 1$ and $C3(x)$ is true. From this state, the automaton $\mathcal{A}$ has more behaviours than $\Delta(\mathcal{A})$ but we have $(M', \nu') \preceq (\ell, v)$. The same reasoning applies for $\mathcal{N}_{x \leq c}$.

  If $a \in \Sigma$ then the transition is $f(a, [0, \infty[)$ for some widget $\mathcal{N}_e$ for $e = (\ell, \gamma, a, R, \ell')$. Since the transition $f(a, [0, \infty[)$ is firable from (M, ν), all the widgets $\mathcal{N}_\varphi$ of atomic constraints φ that appears in the guard γ have a token in the place γ^φ_{tt}. By Eq. (II), the conditions $C1(x)$, $C2(x)$ and $C3(x)$ are true for (M, ν) and we have $M(\gamma^\varphi_{tt}) = 1 \Rightarrow v \in [\![\varphi]\!]$. It means that the guard γ is true and we can fire the matching transition in $\mathcal{A}$ leading to a state (ℓ', v'). The firing of f has left the input places γ_{tt} unchanged. Since, $\forall x \in X, x \notin R$, the truth values of the constraints involving x stay unchanged and conditions $C1(x)$, $C2(x)$ and $C3(x)$ remain true for these clocks. Thus $(M', \nu') \preceq (\ell', v')$.

  We can now consider the case $a = \varepsilon$ and $M(P_\ell) = 0$. It means that the last transition fired in $\mathcal{A}$ corresponds to an edge $e = (\ell_\bullet, \gamma, a, R, \ell)$. The ϵ transition is either an update of a widget or the transition $r_{\ell\ell'}(\epsilon, [0, 0])$ of the widget $\mathcal{N}_e$ (Fig. 9) or a transition of the widget $\mathcal{N}_{Reset(R)}$ of Fig. 8 (either $(\epsilon, [0, 0])$ or r).
  - If the ϵ transition is an update of a widget, then we can go back to the case $a = \varepsilon$ and $M(P_\ell) = 1$ and apply the same reasoning for all clocks $x \notin R$ and, for these clocks we have $C1(x)$, $C2(x)$ and $C3(x)$ and thus $(M', \nu') \preceq (\ell, v)$.
  - If the ϵ transition is a transition $r(\epsilon, [0, 0])$ or $(\epsilon, [0, 0])$ of the widget $\mathcal{N}_{Reset(R)}$ (Fig. 8) then it is the reset of a widget corresponding to a clock $x \in R$. There exists $i \in [0, m]$ such that $M(R^i_{\ell_\bullet \ell}) = 1$ and after the firing of the transition we have either $M'(R^i_{\ell_\bullet \ell}) = 1$ or $M'(R^{i+1}_{\ell_\bullet \ell}) = 1$. Then for all the widgets in the scope of a clock $x \in X$ such that $x \notin R$, the firing of this transition has left the input places γ_{tt} unchanged and conditions $C1(x)$, $C2(x)$ and $C3(x)$ remain true for these clocks. Thus $(M', \nu') \preceq (\ell, v)$.

– If the ϵ transition is the transition $r_{\ell\ell'}(\epsilon, [0, 0])$ of the widget $\mathcal{N}_e$ (Fig. 9), then $M(R^m_{\ell_\bullet\ell}) = 1$, $M'(P_{\ell\ell'}) = 1$ and $C1(x)$, $C2(x)$ and $C3(x)$ are true $\forall x \notin R$. Moreover, $\forall x \in R$, all the widgets of the constraints involving the clock x are in the state $M'(P_x) = 1$ and then we have $C1(x)$, $C2(x)$ and $C3(x)$ for all $x \in R$. Since these conditions remain true $\forall x \notin R$, we have $(M', \nu') \preceq (\ell, \upsilon)$.

This completes the proof that $\mathcal{A}$ simulates $\Delta(\mathcal{A})$. It follows that $\mathcal{L}(\Delta(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$. We can thus conclude that $\mathcal{L}(\Delta(\mathcal{A})) = \mathcal{L}(\mathcal{A})$, which ends the proof of Proposition 2. □

Proposition 2 implies many new interesting results expressed by the following corollaries:

Corollary 2. *B-GTPN$_\varepsilon$ $=_{\mathcal{L}}$ B-TPN$_\varepsilon$ $=_{\mathcal{L}}$ 1-B-TPN$_\varepsilon$ $=_{\mathcal{L}}$ TA$_\varepsilon$.*

Proof. Let $\mathcal{N} \in$ B-*GTPN*$_\varepsilon$, thanks to the translation of Section 4 and to Theorem 2, there is a TA $A_{\mathcal{N}} \in TA_\varepsilon$ s.t. $\mathcal{L}(N) = \mathcal{L}(A_{\mathcal{N}})$ which can effectively be built. From $A_{\mathcal{N}}$ we use Proposition 2 and obtain $\Delta(A_{\mathcal{N}})$ (again effective) which is 1-safe ($\Delta(A_{\mathcal{N}}) \in$ 1-B-*TPN*$_\varepsilon$) and then in B-*GTPN*$_\varepsilon$. □

It follows that Self-modification, read, logical inhibitor and reset arcs do not add expressiveness to bounded TPNs w.r.t. timed language acceptance: as shown in Section 3.1.3, bounded self-modifying TPNs with read, logical inhibitor and reset arcs forms a subclass of B-*GTPN*$_\varepsilon$ which is equally expressive to 1-B-*TPN*$_\varepsilon$ and as B-*TPN*$_\varepsilon$.

Some counterparts of important Theorems for TA can be obtained for TPNs:

Corollary 3 (*ε-transitions add Expressiveness to Bounded TPNs*)**.**
B-GTPN $<_{\mathcal{L}}$ 1-B-TPN$_\varepsilon$ (and thus B-TPN $<_{\mathcal{L}}$ 1-B-TPN$_\varepsilon$).

Proof. From Theorem 3, we have B-*GTPN* $\leq_{\mathcal{L}}$ *TA*. A main result of [36] states that *TA* $<_{\mathcal{L}}$ *TA*$_\varepsilon$ and thus we have B-*GTPN* $<_{\mathcal{L}}$ *TA*$_\varepsilon$. Using Corollary 2 we get B-*GTPN* $<_{\mathcal{L}}$ 1-B-*TPN*$_\varepsilon$. □

Given a TTS $\mathcal{S} = (Q, Q_0, \Sigma_\varepsilon, \rightarrow, \mathsf{F}, \mathsf{R})$, the *universal language problem* asks whether $\mathcal{L}(\mathcal{S}) = TW^\infty(\Sigma)$, *i.e.* whether $\mathcal{S}$ accepts every timed word.

Corollary 4. *The universal language problem is undecidable for* 1-*B-TPN*$_\varepsilon$.

Proof. From the well-known result of Alur & Dill [24] the universal language problem is undecidable fro TA. By Corollary 2, we can reduce the language universal problem for 1-B-*TPN*$_\varepsilon$ to the universality problem on the equivalent automaton. The construction of the equivalent automaton is effective. □

Finally we recall the following theorem from [19]:

Theorem 5 (*[19]*)**.** *TPNs can simulate 2-counter-machines (2CM) and they are Turing powerful.*

This implies that unbounded TPNs can generate non regular (untimed) languages. This does not hold for timed automata [24] and thus:

Corollary 5. *TA$_\varepsilon$ $<_{\mathcal{L}}$ TPN$_\varepsilon$.*

Proof. *TPN*$_\varepsilon$ can simulate 2-counter-machines (Theorem 5) but not *TA*$_\varepsilon$ and Corollary 2 states that *TA*$_\varepsilon$ $=_{\mathcal{L}}$ B-*TPN*$_\varepsilon$ $<_{\mathcal{L}}$ *TPN*$_\varepsilon$. □

On the other hand, since there is no TPN (and no GTPN) weakly timed bisimilar to $\mathcal{A}_0$ (Fig. 5):

Corollary 6. *The classes GTPN$_\varepsilon$ and TA$_\varepsilon$ (as well as TPN$_\varepsilon$ and TA$_\varepsilon$) are incomparable w.r.t. timed bisimilarity.*

7. Equivalence w.r.t. timed bisimilarity

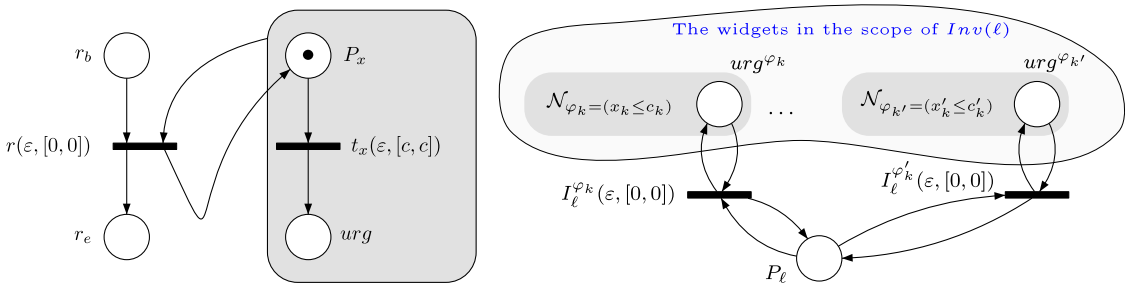
From Theorem 4, we know that there is no translation from TA to TPN preserving timed bisimilarity. This can be illustrated by considering the widget $\mathcal{N}_{x<c}$ of Fig. 7: the firing of transitions t_x then u indeed leads to a state where γ_{tt} is not marked, while $x < c$ and the corresponding guard in the TA is therefore true.

Thus, in this section, we consider the original definition of TPN by Merlin [5] *i.e.* the class B-*TPN*$_\varepsilon(\leq, \geq)$ of TPNs without strict constraints.

First recall (Remark 2) that starting from a GTPN $\mathcal{N} \in$ B-*GTPN*$_\varepsilon(\leq, \geq)$ (and in particular from a TPN "à la Merlin"), the translation proposed in Section 4 gives a TA $\mathcal{A}$ with a particular form, belonging to the following subclass $TA^{syn}_\varepsilon(\leq, \geq)$:

Definition 15. The subclass $TA^{syn}_\varepsilon(\leq, \geq)$ of TA is defined by the set of TA of the form $(L, l_0, X, \Sigma_\varepsilon, E, Inv, \mathsf{F}, \mathsf{R})$ where:

- guards are conjunctions of atomic constraints of the form $x \geq c$ and invariants are conjunctions of atomic constraints of the form $x \leq c$.
- 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 either $Inv(\ell')$ does not constrain x or the constraint on x is of the form $x \leq c'$ with $c' \geq c$. □



(a) Widget $\mathcal{N}_{x \leq c}$. (b) Widgets for $Inv(\ell) = \varphi_k \wedge \cdots \wedge \varphi_{k'}$.

Fig. 10. Widgets for an invariant.

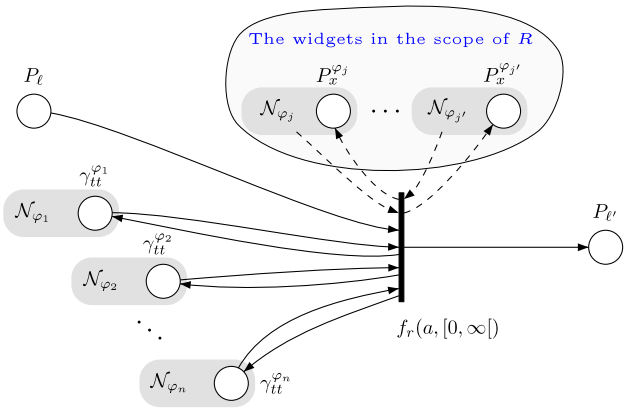


Fig. 11. New widget $\mathcal{N}_e$ for an edge $e = (\ell, \gamma, a, R, \ell')$.

We now adapt the construction of Section 6 to define a translation from $TA_\varepsilon^{syn}(\leq, \geq)$ to B-$TPN_\varepsilon(\leq, \geq)$ preserving timed bisimilarity. The widget $\mathcal{N}_{x \leq c}$ is modified as depicted in Fig. 10(a). The widget $\mathcal{N}_{x \geq c}$ is the one of Section 6 in Fig. 6(b). Moreover, for each invariant, a widget depicted in Fig. 10(b) prevents timed from elapsing when the border value of the invariant is reached.

In the sequel, 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. Moreover, for a constraint $\varphi = (x \geq c)$, 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.

Edges and resetting clocks. The reset of all the blocks in the scope of a set of clocks R can be done in one step by merging[4] all the transitions $r(\varepsilon, [0, 0])$ and $(\varepsilon, [0, 0])$ of the widget $\mathcal{N}_{Reset(R)}$ of Fig. 8. Since the marking of a widget, when the reset occurs, is not unique, and since a clock x of an automaton can be reset in more than one transition, we create copies of these $(\varepsilon, [0, 0])$ transitions in such a way that, given an edge $e = (\ell, \gamma, a, R, \ell')$ and given a marking of widgets in scope of R, the reset of these widgets can be done by the firing of one transition which we denote $R_{\ell\ell'}(\varepsilon, [0, 0])$. Then there is one copy of $R_{\ell\ell'}$ per possible marking of the set of blocks in the scope of R. Finally, for each edge $(\ell, \gamma, a, R, \ell')$ we create, as in Section 6, a transition $f(a, [0, \infty[)$ and we merge it with $R_{\ell\ell'}(\varepsilon, [0, 0])$ leading to a transition $f_r(a, [0, \infty[)$ as depicted in Fig. 11. We obtain one copy of $f_r(a, [0, \infty[)$ per possible marking of the set of blocks in the scope of R. It is represented in Fig. 11 by starting the input arcs of $f_r(a, [0, \infty[)$ from the border of boxes $\mathcal{N}_{\varphi_j}$ and $\mathcal{N}_{\varphi_{j'}}$.

The construction. As in Section 6, we create a place P_ℓ for each location $\ell \in L$. Then we build the block $\mathcal{N}_\varphi$ for each atomic constraint $\varphi = (x \geq c)$ (Fig. 6(b)) that appears in the guards of $\mathcal{A}$ and for each atomic constraint $\varphi = (x \leq c)$ (Fig. 10(a)) that appears in an invariant of $\mathcal{A}$.

For each edge $(\ell, \gamma, a, R, \ell') \in E$, we create the transition $f_r(a, [0, \infty[)$ and we connect it to the widgets in the scope of R as described in the paragraph above (see Fig. 11). Now, assume $\gamma = \wedge_{i=1,n}\varphi_i$ and $n \geq 0$, we connect $f_r(a, [0, \infty[)$ to the places $\gamma_{tt}^{\varphi_i}$ of the widgets $\mathcal{N}_{\varphi_i}$ as described on Fig. 11. In case $\gamma = \mathrm{tt}$ (or $n = 0$) there is only one input place to $f_r(a, [0, \infty[)$ which is P_ℓ.

Finally, for each location $\ell \in L$ with $Inv(\ell) = \varphi_k \wedge \cdots \wedge \varphi_{k'}$, (as a shorthand we denote $\varphi_k \in Inv(\ell)$ when $Inv(\ell) = \cdots \wedge \varphi_k \wedge \cdots$.) we proceed as follows (see Fig. 10(b)):

1. create a transition $I_\ell^{\varphi_k}(\varepsilon, [0, 0])$ for each $\varphi_k \in Inv(\ell)$;
2. connect $I_\ell^{\varphi_k}(\varepsilon, [0, 0])$ to P_ℓ and to the place urg^{φ_k} of block $\mathcal{N}_{\varphi_k}$.

[4] The merging of two transitions t_1 and t_2 gives one transition $t_{1,2}$ such that ${}^\bullet t_{1,2} = {}^\bullet t_1 + {}^\bullet t_2$ and $t_{1,2}{}^\bullet = t_1{}^\bullet + t_2{}^\bullet$. Here, temporal intervals are $[0, 0]$ and labels are ε both for the merged transitions and for the result of the merging.

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

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.

As for the language preserving translation, we have, by construction: (1) $\Delta^+(\mathcal{A})$ is 1-safe and moreover (2) in each reachable marking M of $\Delta^+(\mathcal{A})$ $\left(\sum_{\ell \in L} M(P_\ell)\right) \leq 1$.

Each widget related to an atomic constraint has a constant size. However each widget associated with an edge has an exponential number of copies of f_r w.r.t. the number of atomic constraints of $\mathcal{A}$. Thus the size of $\Delta^+(\mathcal{A})$ is exponential w.r.t. the size of $\mathcal{A}$.

We can now build a bisimulation relation $\approx$ between $\mathcal{A}$ and $\Delta^+(\mathcal{A})$.

Let (ℓ, v) be a state of $\mathcal{A}$ and (M, ν) be a state 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:

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

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

Proposition 3. *The relation $\approx$ of Eq. (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 $v \in [\![\varphi]\!]$, and $v + d \in [\![\varphi]\!]$. 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 $\varphi = (x \leq c) \in Inv(\ell)$,
   - 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_ℓ^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 $\varphi = (x \geq c)$,
   - $v \in [\![\varphi]\!]$ and $v + d \in [\![\varphi]\!]$. 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.
   - $v \notin [\![\varphi]\!]$ and $v + d \in [\![\varphi]\!]$, iff there is $d' \leq d$ s.t. transition t_x must be fired at d'. Transition t_x is fired and let $d - d'$ elapse.
   - $v \notin [\![\varphi]\!]$ and $v + d \notin [\![\varphi]\!]$ iff time d elapse and leave a token in P_x.
   For $\varphi = (x \leq c) \notin Inv(\ell)$, 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.
   - $v \notin [\![\varphi]\!]$ and then $v + d \notin [\![\varphi]\!]$. Then $M(urg^\varphi) = 1$ but there is no transition I_ℓ^φ and for all possible transition $I_{\ell'}^\varphi$, we have $M(P_{\ell'}) = 0$ and a time d can elapsed.
   - $v + d \notin [\![\varphi]\!]$. If $M(urg^\varphi) = 1$ then a time d can elapsed. If $M(urg^\varphi) = 0$ then there is $d' \leq d$ s.t. transition t_x^φ must be fired at d'. Transition t_x^φ is fired and let $d - d'$ elapse.
   - $v \in [\![\varphi]\!]$ and $v + d \in [\![\varphi]\!]$. This case is similar to $\varphi \in Inv(\ell)$ but there is no transition I_ℓ^φ.

   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 Eq. (III). For all interleavings of previous runs ρ_φ we obtain a run $\rho = (M, \nu) \xrightarrow{d}_\varepsilon (M', \nu')$ s.t. $(\ell, v) \approx (M', \nu')$.
1. 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. $\gamma = \wedge_{i=1,n} \varphi_i$, $n \geq 0$ where φ_i is an atomic constraint. According to the subclass of TA we consider, invariants of ℓ' can be ignored for allowing the firing of a as (by definition) they are true if invariants of ℓ are true. From the semantics of timed automata (Definition 12), $v \in [\![\varphi_i]\!]$ for $1 \leq i \leq n$. From the definition of the 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 firable leading to $M(\gamma_{tt}^{\varphi_i}) = 1$. Thus, transition $f_r(a, [0, \infty[)$ is fired in widget $\mathcal{N}_e$ leading to (M', ν'). We have then $M'(P_\ell) = 0$ and $M'(P'_\ell) = 1$. Then for all the widgets in the scope of R, we have $M'(P_x^\varphi) = 1$ and $\nu'(t_x^\varphi) = v'(x) = 0$. Moreover, for all clocks $x \notin R$, the truth values of the constraints involving x, as well as the corresponding widgets, stay unchanged. Thus we have $(\ell', v') \approx (M', \nu')$.

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

From the previous results we can state the following corollaries:

Corollary 7. *$TA_\varepsilon^{syn}(\leq,\geq)$ is a syntactical subclass of TA_ε equally expressive to B-$TPN_\varepsilon(\leq,\geq)$ w.r.t. weak timed bisimilarity, i.e. $TA_\varepsilon^{syn}(\leq,\geq) =_{\mathcal{W}}$ B-$TPN_\varepsilon(\leq,\geq)$.*

Proof. Let $\mathcal{A} \in TA_\varepsilon^{syn}(\leq,\geq)$. From the previous construction and Proposition 3, page 18, we obtain $\Delta_1^+(\mathcal{A}) \in$ B-$TPN_\varepsilon(\leq,\geq)$ with $\mathcal{A} =_{\mathcal{W}} \Delta_1^+(\mathcal{A})$. Let $\mathcal{N} \in$ B-$TPN_\varepsilon(\leq,\geq)$, from Theorem 1 or Theorem 2, we obtain $\Delta_2^+(\mathcal{N}) \in TA_\varepsilon^{syn}(\leq,\geq)$ with $\mathcal{N} =_{\mathcal{W}} \Delta_2^+(\mathcal{N})$. □

Corollary 8. *The classes 1-B-$TPN_\varepsilon(\leq,\geq)$, B-$TPN_\varepsilon(\leq,\geq)$ and B-$GTPN_\varepsilon(\leq,\geq)$ are equally expressive w.r.t. weak timed bisimilarity* i.e. 1-B-$TPN_\varepsilon(\leq,\geq) =_{\mathcal{W}}$ B-$TPN_\varepsilon(\leq,\geq) =_{\mathcal{W}}$ B-$GTPN_\varepsilon(\leq,\geq)$.

Proof. Let $\mathcal{N} \in$ B-$GTPN_\varepsilon(\leq,\geq)$, thanks to the translation of Section 4 and to Theorem 2, there exists a TA $A_{\mathcal{N}} \in TA_\varepsilon^{syn}(\leq,\geq)$ s.t. $\mathcal{N} =_{\mathcal{W}} A_{\mathcal{N}}$ which can effectively be built. From the previous construction and Proposition 3, we obtain $\Delta^+(A_{\mathcal{N}}) \in$ 1-B-TPN_ε s.t. $A_{\mathcal{N}} =_{\mathcal{W}} \Delta^+(A_{\mathcal{N}})$ then $\mathcal{N} =_{\mathcal{W}} \Delta^+(A_{\mathcal{N}})$. □

Corollary 9. *Self-modification, read, logical inhibitor and reset arcs do not add expressiveness to bounded TPNs "à la Merlin" w.r.t. weak timed bisimilarity.*

Proof. As shown in Section 3.1.3, bounded self-modifying TPNs "à la Merlin" with read, logical inhibitor and reset arcs is a subclass of B-$GTPN_\varepsilon(\leq,\geq)$ which is equally expressive as 1-B-$TPN_\varepsilon(\leq,\geq)$ and as B-$TPN_\varepsilon(\leq,\geq)$ (i.e. bounded TPNs "à la Merlin"). □

Corollary 10. *Atomic and intermediate semantics are equally expressive for bounded TPNs "à la Merlin" w.r.t. weak timed bisimilarity.*

Proof. In [33] there is a translation from TPNs with intermediate semantics to TPNs with atomic semantics which preserves bisimilarity when the net is 1-safe. Moreover, as shown in Section 3.1.2, GTPNs allow to express intermediate as well as atomic semantics of TPNs. Thus, a bounded TPN "à la Merlin" with atomic semantics is in B-$GTPN_\varepsilon(\leq,\geq)$ and then, thanks to Corollary 8, can be translated into a bisimilar net in 1-B-$GTPN_\varepsilon(\leq,\geq)$ *i.e.* a 1-safe (and then bounded) TPN "à la Merlin" with intermediate semantics. □

8. Conclusion

In this paper, we have investigated different questions related to the expressiveness of TPNs and GTPNs.

We have first presented a structural translation from bounded generalised TPNs (encompassing read, logical inhibitor and reset arcs, self-modification and strict constraints) to TA preserving isomorphism of the underlying timed transitions systems. We have shown that TA, bounded TPNs and bounded GTPNs are equivalent w.r.t. timed language acceptance. We have also provided an effective construction of a 1-safe TPN equivalent to a TA. Finally, we have given a syntactic subclass of TA expressively equivalent to TPNs "à la Merlin" w.r.t. timed bisimilarity. This enables us to obtain new results for TPNs summarised in Table 2, page 3. Moreover these results lead to a classification of the expressiveness of different subclasses of GTPNs and TA given in Fig. 1.

References

[1] B. Bérard, F. Cassez, S. Haddad, D. Lime, O.H. Roux, Comparison of the expressiveness of timed automata and time Petri nets, in: 3rd Int. Conf. on Formal Modelling and Analysis of Timed Systems, FORMATS 2005, in: Lecture Notes in Computer Science, vol. 3829, Springer-Verlag, Uppsala, Sweden, 2005, pp. 211–225.

[2] P.A. Abdulla, A. Nylén, Timed Petri nets and BQOs, in: 22nd Int. Conf. on Application and Theory of Petri Nets, ICATPN'01, in: LNCS, vol. 2075, Springer-Verlag, United Kingdom, 2001, pp. 53–70.

[3] D. de Frutos Escrig, V.V. Ruiz, O.M. Alonso, Decidability of properties of timed-arc Petri nets, in: 21st Int. Conf. on Application and Theory of Petri Nets, ICATPN'00, in: LNCS, vol. 1825, Springer-Verlag, Aarhus, Denmark, 2000, pp. 187–206.

[4] J. Byg, K. Jørgensen, J. Srba, An efficient translation of timed-arc Petri nets to networks of timed automata, in: 11th Int. Conf. on Formal Engineering Methods, ICFEM'09, in: LNCS, vol. 5885, Springer-Verlag, 2009, pp. 698–716.

[5] P.M. Merlin, A study of the recoverability of computing systems, Ph.D. Thesis, Dep. of Information and Computer Science, University of California, Irvine, CA, 1974.

[6] C. Ramchandani, Analysis of asynchronous concurrent systems by timed Petri nets, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, project MAC Report MAC-TR-120, 1974.

[7] L. Popova-Zeugmann, Quantitative evaluation of time-dependent Petri nets and applications to biochemical networks, Natural Computing 10 (3) (2011) 1017–1043.

[8] S. Bernardi, J. Campos, Computation of performance bounds for real-time systems using time Petri nets, IEEE Transactions on Industrial Informatics 5 (2) (2009) 168–180.

[9] A. Aybar, A. Iftar, Supervisory controller design to enforce some basic properties in timed-transition Petri nets using stretching, Nonlinear Analysis: Hybrid Systems 6 (1) (2012) 712–729.

[10] B. Berthomieu, P.-O. Ribet, F. Vernadat, The tool tina – construction of abstract state spaces for Petri nets and time Petri nets, International Journal of Production Research 42 (4), tool available at http://www.laas.fr/tina/.

[11] G. Gardey, D. Lime, M. Magnin, O.H. Roux, Roméo: a tool for analyzing time Petri nets, in: 17th Int. Conf. on Computer Aided Verification, CAV 2005, in: Lecture Notes in Computer Science, vol. 3576, Springer-Verlag, Edinburgh, Scotland, UK, 2005, pp. 418–423.

[12] D. Lime, O.H. Roux, C. Seidner, L.-M. Traonouez, Romeo: A parametric model-checker for Petri nets with stopwatches, in: 15th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2009, in: Lecture Notes in Computer Science, vol. 5505, Springer, York, United Kingdom, 2009, pp. 54–57.

[13] J. Sifakis, Performance Evaluation of Systems using Nets, in: Net theory and applications: Proc. of the advanced course on general net theory, processes and systems (Hamburg, 1979), in: LNCS, vol. 84, Springer-Verlag, 1980, pp. 307–319.
[14] M. Pezzè, Time Petri nets: a primer introduction, Tutorial presented at the Multi-Workshop on Formal Methods in Performance Evaluation and Applications, Zaragoza, Spain, September 1999.
[15] B. Berthomieu, M. Diaz, Modeling and verification of time dependent systems using time Petri nets, IEEE Transactions on Software Engineering 17 (3) (1991) 259–273.
[16] H. Hanisch, Analysis of place/transition nets with timed-arcs and its application to batch process control, in: 14th Int. Conf. on Application and Theory of Petri Nets, ICATPN'93, in: LNCS, vol. 691, 1993, pp. 282–299.
[17] W. Khansa, J.-P. Denat, S. Collart-Dutilleul, P-time Petri nets for manufacturing systems, in: Int. Workshop on Discrete Event Systems, WODES'96, Edinburgh (U.K.), 1996, pp. 94–102.
[18] M. Boyer, O.H. Roux, On the compared expressiveness of arc, place and transition time Petri nets, Fundamenta Informaticae 88 (3) (2008) 225–249.
[19] N.D. Jones, L.H. Landweber, Y.E. Lien, Complexity of some problems in Petri nets, Theoretical Computer Science 4 (1977) 277–299.
[20] T. Araki, T. Kasami, Some decision problems related to the reachability problem for Petri nets, Theoretical Computer Science 3 (1) (1976) 85–104.
[21] C. Dufourd, Réseaux de Petri avec reset/transfert: Décidabilité et indécidabilité, Ph.D. Thesis, ENS de Cachan, 1998.
[22] R. Valk, Self-modifying nets, a natural extension of Petri nets, in: Fifth Colloquium on Automata, Languages and Programming, in: Lecture Notes in Computer Science, vol. 62, Springer, Udine, Italy, 1978, pp. 464–476.
[23] R. Alur, D.L. Dill, Automata for modeling real-time systems, in: Proc. 17th Int. Coll. Automata, Languages, and Programming, ICALP'90, Warwick University, England, July 1990, in: Lecture Notes in Computer Science, vol. 443, Springer, 1990, pp. 322–335.
[24] R. Alur, D.L. Dill, A theory of timed automata, Theoretical Computer Science 126 (2) (1994) 183–235.
[25] R. Alur, L. Fix, T.A. Henzinger, Event-clock automata: a determinizable class of timed automata, Theoretical Computer Science 211 (1999).
[26] D. Lime, O.H. Roux, Model checking of time Petri nets using the state class timed automaton, Journal of Discrete Events Dynamic Systems - Theory and Applications (DEDS) 16 (2) (2006) 179–205.
[27] F. Cassez, O.H. Roux, Structural translation from time Petri nets to timed automata – model-checking time Petri nets via timed automata, The Journal of Systems and Software 79 (10) (2006) 1456–1468.
[28] S. Haar, F. Simonot-Lion, L. Kaiser, J. Toussaint, Equivalence of timed state machines and safe time Petri nets., in: WODES 2002, Zaragoza, 2002, pp. 119–126.
[29] B. Berthomieu, F. Peres, F. Vernadat, Bridging the gap between timed automata and bounded time Petri nets, in: Formal Modeling and Analysis of Timed Systems, in: LNCS, vol. 4202, 2006, pp. 82–97.
[30] B. Bérard, F. Cassez, S. Haddad, D. Lime, O.H. Roux, When are timed automata weakly timed bisimilar to time Petri nets? Theoretical Computer Science 403 (2–3) (2008) 202–220.
[31] P. Bouyer, S. Haddad, P.-A. Reynier, Extended timed automata and time Petri nets, in: Proceedings of the 6th Int. Conf. on Application of Concurrency to System Design, ACSD'06, IEEE Computer Society Press, Turku, Finland, 2006, pp. 91–100.
[32] J. Srba, Comparing the expressiveness of timed automata and timed extensions of Petri nets, in: 6th Int. Conf. on Formal Modelling and Analysis of Timed Systems, FORMATS'08, in: LNCS, vol. 5215, Springer-Verlag, 2008, pp. 15–32.
[33] B. Bérard, F. Cassez, S. Haddad, D. Lime, O.H. Roux, Comparison of different semantics for time Petri nets, in: 3rd Int. Symposium on Automated Technology for Verification and Analysis, ATVA 2005, in: Lecture Notes in Computer Science, vol. 3707, Springer-Verlag, Taipei, Taiwan, 2005, pp. 293–307.
[34] C. Dufourd, A. Finkel, Ph. Schnoebelen, Reset nets between decidability and undecidability, in: 25th Int. Coll. on Automata, Languages and Programming, ICALP'98, in: Lecture Notes in Computer Science, vol. 1443, Springer, Aalborg, Denmark, 1998, pp. 103–115.
[35] T.A. Henzinger, X. Nicollin, J. Sifakis, S. Yovine, Symbolic model checking for real-time systems, Information and Computation 111 (2) (1994) 193–244.
[36] B. Bérard, V. Diekert, P. Gastin, A. Petit, Characterization of the expressive power of silent transitions in timed automata, Fundamenta Informaticae 36 (2) (1998) 145–182.