

Blending Timed Formal Models with Clock Transition Systems

Claude Jard, Didier Lime, Olivier Henri Roux

▶ **To cite this version:**

Claude Jard, Didier Lime, Olivier Henri Roux. Blending Timed Formal Models with Clock Transition Systems. Fundamenta Informaticae, 2014, 129 (1-2), pp.85-100. hal-00940970

HAL Id: hal-00940970
https://hal.science/hal-00940970

Submitted on 6 Feb 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Fundamenta Informaticae XXI (2001) 1001–1015
DOI 10.3233/FI-2012-0000
IOS Press

Blending Timed Formal Models with Clock Transition Systems

Claude Jard*
Université de Nantes, LINA, Nantes, France
Claude.Jard@univ-nantes.fr

Didier Lime*
École Centrale de Nantes, IRCCyN, Nantes, France
Didier.Lime@irccyn.ec-nantes.fr

Olivier H. Roux*
École Centrale de Nantes, IRCCyN, Nantes, France
Olivier-H.Roux@irccyn.ec-nantes.fr

Abstract. Networks of Timed Automata (NTA) and Time Petri Nets (TPNs) are well-established formalisms used to model, analyze and control industrial real-time systems. The underlying theories are usually developed in different scientific communities and both formalisms have distinct strong points: for instance, conciseness for TPNs and a more flexible notion of urgency for NTA. The objective of the paper is to introduce a new model allowing the joint use of both TPNs and NTA for the modeling of timed systems. We call it *Clock Transition System* (CTS). This new model incorporates the advantages of the structure of Petri nets, while introducing explicitly the concept of clocks. Transitions in the network can be guarded by an expression on the clocks and reset a subset of them as in timed automata. The urgency is introduced by a separate description of invariants. We show that CTS allow to express TPNs (even when unbounded) and NTA. For those two classical models, we identify subclasses of CTSs equivalent by isomorphism of their operational semantics and provide (syntactic) translations. The classical state-space computation developed for NTA and then adapted to TPNs can easily be defined for general CTSs. Armed with these merits, the CTS model seems a good candidate to serve as an intermediate theoretical and practical model to factor out the upcoming developments in the TPNs and the NTA scientific communities.

Keywords: Real-time systems, Timed models, Timed Automata, Time Petri nets

Address for correspondence: Didier Lime, École Centrale de Nantes, 1 rue de la Noë, BP92101, 44321 Nantes Cedex 3, France
*This work was partially funded by the ANR national research program **ImpRo** (ANR-2010-BLAN-0317).

1. Introduction

Mastering the development of correct distributed real-time systems remains a priority in light of the clear scientific issues that have to be overcome. One necessary line to follow, in our opinion, is the use of mathematically based models.

Low Level timed models. In [11], the authors introduce the abstract notion of *timed transition systems* allowing to give the formal semantics of a real-time system as a set of timed execution sequences. They incorporate time into classical transition systems by assuming that all discrete transitions happen instantaneously while real time constraints restrict the times at which discrete transition may occur. *Timed transition systems* (TTS) are defined in [15] as a basic semantical model for real-time systems which is a labelled transition system with two type of labels: atomics actions and delay actions (i.e. positive reals) representing discrete and continuous changes of real-time systems.

To avoid delay actions, the authors of [2, 3] advocate an alternative proposal, namely, to designate certain program variables as clock variables. It leads to higher level of specification, explicitly referring to clocks, which are just another kind of system variables. Thus, in [10], labeled transition systems are extended with clocks and both discrete or dense time domains are considered. Similarly, in [14], a computational model is proposed for real-time systems called Clocked Transition Systems. This model represents time by a set of timers (clocks) which increase whenever time progresses, but can be set to arbitrary values by system (program) transitions. A Clocked Transition System is also equipped with discrete variables of any type. Assertions associated with transitions allow the updates of variables, and assertions over system variables specify a global restriction of time progress.

TPNs and TA. For the class of critical systems that we aim at, in which the specification of permissible behaviors requires a description of fine temporal constraints, and for which verification must be performed by efficient tools, the scientific community has notably focused for many years on two timed models: Time Petri nets (TPNs for short) [19, 5] and timed automata (TA for short) or networks of timed automata (NTA for short) [1], and their different extensions. These models extend with time respectively Petri nets and finite automata. An overview of the theoretical known results about the relationships among these models is provided in [20].

Each class of models has distinct strong points. TPNs are particularly well-suited for having a compact representation of concurrent behaviors with causal dependencies induced by complex synchronizations between activities. The time constraints are described on transitions by intervals of firing.

Timed automata better clarify how time should change. This model is equipped with a set of temporal variables (clocks) used to form expressions guarding transitions. Transitions may reset clocks. Urgency is expressed by defining invariants on states, forcing the progress when possible. The introduction of concurrency is achieved by synchronously connecting a set of components. We believe it would be interesting to allow a hybrid modeling, in which some aspects could be described with NTA (once decomposition is decided) and others with TPNs (e.g. when there is a complex parallel control flow). To achieve this, the idea is to blend these models into a more general formalism for which the existing analysis methods could still be used as implemented currently in TINA [6] and UPPAAL [16].

Our contribution. A whole set of theories, methods and tools of analysis has been separately developed for TPNs and NTA. Yet we know that these models are very close, but nevertheless have subtle differences that have until now prevented an actual factorization of research and development of associated technologies and their joint use in the modeling phase. The objective of the paper is to introduce a new model, Clock Transition Systems (CTSs), bridging this gap. This new model incorporates the

advantages of the structure of Petri nets, while introducing explicitly the concept of clocks. Transitions in the network can be guarded by an expression on the clocks and reset a subset of them as in timed automata. Urgency is introduced by a separate description of invariants. We show that CTS allow to express TPNs (even when unbounded) and NTA. For those two classical models, we identify subclasses of CTSs equivalent by isomorphism of their operational semantics and provide (syntactic) translations. The classical state-space computation developed for NTA and then adapted to TPNs can easily be defined for general CTSs. Armed with these merits, the CTS model seems a good candidate to serve as an intermediate theoretical and practical model to factor out the upcoming developments in the TPNs and the NTA scientific communities.

Outline of the paper. We first introduce in Section 2 the Clock Transition System model giving its syntax and its operational sequential semantics as usual. We then show in Sections 2.2 and 2.3 how TPNs and NTA can be easily represented by a Clock Transition System. Finally, Section 4 discusses the model and the techniques for its analysis.

2. Definitions

2.1. Basic Notations and Definitions

$\mathbb{N}$ is the set of natural numbers and $\mathbb{Z}$ is the set of integers. $\mathbb{B} = \{\mathsf{true}, \mathsf{false}\}$ is the set of booleans. For a finite set E, we denote its size by $|E|$ and by 2^E the set of all its subsets. For any two sets E and F, we denote by E^F the set of mappings from F into E.

Let $\mathbb{R}$ (resp. $\mathbb{Q}$) be the set of real (resp. rational) numbers. $\mathbb{R}_{\geq 0}$ (resp. $\mathbb{Q}_{\geq 0}$ is the set of non-negative real (resp. rational) numbers. Let X be a finite set of *clocks*. A *valuation* v of X is a mapping from X into $\mathbb{R}_{\geq 0}$. We denote by $\mathbf{0}$ the null valuation such that $\forall x \in X, \mathbf{0}(x) = 0$. For a valuation v and $R \subseteq X$, we write $v[R \leftarrow 0]$ the valuation such that $\forall x \in R, v[R \leftarrow 0](x) = 0$ and $\forall x \notin R, v[R \leftarrow 0](x) = v(x)$. Finally, for $d \in \mathbb{R}_{\geq 0}$, $v + d$ is the valuation such that $\forall x \in X, (v + d)(x) = v(x) + d$. Similarly a valuation on a set of integer variables V is a mapping from V to $\mathbb{N}$.

We denote by $\mathcal{C}(X)$ the set of constraints generated by the grammar $\phi ::= \mathsf{true} | x \leq k | x < k | \neg\phi | \phi \wedge \phi$, where x is a clock in X, $k \in \mathbb{Q}_{\geq 0}$, $\neg$ is the logical negation and $\wedge$ is the logical conjunction. We denote by $\mathcal{B}(X)$ the subset of $\mathcal{C}(X)$ without the use of negation. We say that a valuation v satisfies a simple constraint γ if the expression obtained by replacing all clocks x by their valuation $v(x)$ logically evaluates to true. We then write $v \models \gamma$.

For two finite sets A and B, $\mathcal{F}(A, B)$ denotes the set of computable functions from A to B.

Definition 2.1. (Timed Transition System)
A *timed transition system (TTS)* over the alphabet A is a tuple $S = (Q, q_0, A, \rightarrow)$ where Q is a set of states, $q_0 \in Q$ is the *initial* state, A is a finite set of actions disjoint from $\mathbb{R}_{\geq 0}$, $\rightarrow \subseteq Q \times (A \cup \mathbb{R}_{\geq 0}) \times Q$ is a set of edges. If $(q, e, q') \in \longrightarrow$, we also write $q \xrightarrow{e} q'$. Moreover, TTS should satisfy the classical time-related conditions where $d, d' \in \mathbb{R}_{\geq 0}$: i) time determinism: $(q \xrightarrow{d} q') \wedge (q \xrightarrow{d} q'') \Rightarrow (q' = q'')$, ii) time additivity: $(q \xrightarrow{d} q') \wedge (q' \xrightarrow{d'} q'') \Rightarrow (q \xrightarrow{d+d'} q'')$, iii) null delay: $\forall q : q \xrightarrow{0} q$, and iv) time continuity: $(q \xrightarrow{d} q') \Rightarrow (\forall d' \leq d, \exists q'', q \xrightarrow{d'} q'')$.

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

Definition 2.2. (Isomorphism)
Let $S_1 = (Q_1, q_{0_1}, A, \rightarrow_1)$ and $S_2 = (Q_2, q_{0_2}, A, \rightarrow_2)$ be two TTSs. S_1 and S_2 are isomorphic (we write $S_1 \cong S_2$) whenever there is a bijection $f : Reach(q_{0_1}) \rightarrow Reach(q_{0_2})$ such that $\forall q, \forall q' \in Reach(q_{0_1})$ we have: $q \xrightarrow{a \in A}_1 q'$ iff $f(q) \xrightarrow{a}_2 f(q')$ and $q \xrightarrow{d \in \mathbb{R}_{\geq 0}}_1 q'$ iff $f(q) \xrightarrow{d}_2 f(q')$.

Definition 2.3. (Equivalence up to isomorphism)
Let $\mathcal{A}$ and $\mathcal{A}'$ be two models whose semantics are expressed as TTSs $\mathcal{S}_{\mathcal{A}}$ and $\mathcal{S}_{\mathcal{A}'}$. $\mathcal{A}$ and $\mathcal{A}'$ are equivalent up to isomorphism, which we denote by $\mathcal{A} \cong \mathcal{A}'$, iff $\mathcal{S}_{\mathcal{A}} \cong \mathcal{S}_{\mathcal{A}'}$.

2.2. Time Petri Nets

Definition 2.4. (Petri Net)
A (labeled) *Petri Net* $\mathcal{N}$ is a tuple $\langle P, T, \mathsf{Pre}, \mathsf{Post}, m_0, A, \lambda \rangle$ such that: P is a finite non-empty set of *places*; T is a finite non-empty set of *transitions*; $\mathsf{Pre} : P \times T \rightarrow \mathbb{N}$ is the *backward incidence* function; $\mathsf{Post} : P \times T \rightarrow \mathbb{N}$ is the *forward incidence* function; $m_0 : P \rightarrow \mathbb{N}$ is the *initial marking* of the net; A is finite non-empty *alphabet*; $\lambda : T \rightarrow A$ is a *labeling* function of the transitions.

A marking of $\mathcal{N}$ is an application from P to $\mathbb{N}$. Let m be a marking of $\mathcal{N}$. Then, for any place $p \in P$, we say that p contains $m(p)$ *tokens*. For any transition t we denote by ${}^\bullet t$ the set of places p such that $\mathsf{Pre}(p,t) \neq 0$ and by $t^\bullet$ the set of places p such that $\mathsf{Post}(p,t) \neq 0$.

A transition $t \in T$ is said to be enabled by the marking m if $\forall p \in {}^\bullet t, m(p) \geq \mathsf{Pre}(p,t)$. This is denoted by $t \in \mathsf{en}(m)$. The operational semantics of the Petri Net $\mathcal{N} = \langle P, T, \mathsf{Pre}, \mathsf{Post}, m_0 \rangle$ is defined by the transition system $\mathcal{S}_{\mathcal{N}} = (\mathbb{N}^{|P|}, m_0, A, \rightarrow)$ such that: $m \xrightarrow{a} m'$ iff there exists $t \in \mathsf{en}(m)$ such that $\lambda(t) = a$ and $\forall p \in P, m'(p) = m(p) - \mathsf{Pre}(p,t) + \mathsf{Post}(p,t)$.

We then say that m' is obtained from m by *firing* the enabled transition t.

Petri nets can be extended with timing information in many ways. We focus here on Time Petri Nets [19] in which time intervals are attached to transitions, defining the durations during which they will be enabled.

We note $\mathcal{I}$ the set of rational intervals $\{x \in \mathbb{R} | a \sim_1 x \sim_2 b, a \in \mathbb{Q}_{\geq 0}, b \in \mathbb{Q}_{\geq 0}, \sim_1, \sim_2 \in \{<, \leq\}\} \cup \{x \in \mathbb{Q} | a \sim x < +\infty, a \in \mathbb{Q}_{\geq 0}, \sim \in \{<, \leq\}\}$. For any interval I, we denote by $I^\downarrow$ the smallest left-closed interval with lower bound 0 that contains I.

Definition 2.5. (Time Petri Net)
A *time Petri net* (TPN) is a tuple $\mathcal{T} = \langle \mathcal{N}, I_s \rangle$ where $\mathcal{N} = \langle P, T, \mathsf{Pre}, \mathsf{Post}, m_0, A, \lambda \rangle$ is a Petri Net and $I_s : T \rightarrow \mathcal{I}$ assigns a *static time interval* to each transition.

For each transition t there is an associated clock x_t. We consider valuations on the set of clocks $\{x_t | t \in T\}$ and we will slightly abuse the notations by writing $v(t)$ instead of $v(x_t)$.

Let m be a marking of the net and t a transition in $\mathsf{en}(m)$. Let m' be the marking obtained from m by firing t. Let m'' be the *intermediate marking* defined by $\forall p, m''(p) = m(p) - \mathsf{Pre}(p,t)$. A transition t' is *newly enabled* by the firing of t from m, and we note $t \in \mathsf{new}(m,t)$ if $t' \in \mathsf{en}(m') \setminus \mathsf{en}(m'') \cup \{t\}$

The operational semantics of the TPN $\mathcal{T} = \langle \mathcal{N}, I_s \rangle$ is defined by the time transition system $\mathcal{S}_{\mathcal{T}} = (\mathbb{N}^P \times \mathbb{R}^T_{\geq 0}, (m_0, \mathbf{0}), A, \rightarrow)$ such that:

- $(m,v) \xrightarrow{a \in A} (m',v')$ iff $\exists t \in \mathsf{en}(m)$ s.t.: $\begin{cases} \lambda(t) = a, \\ \forall p \in P, m'(p) = m(p) - \mathsf{Pre}(p,t) + \mathsf{Post}(p,t), \\ v(t) \in I_s(t), \\ v' = v[\mathsf{new}(m,t) \leftarrow 0], \end{cases}$

- $(m,v) \xrightarrow{d \in \mathbb{R}_{\geq 0}} (m, v+d)$ iff $\forall t' \in \mathsf{en}(m), \forall 0 < d' \leq d, (v+d')(t') \in I_s^{\downarrow}(t')$.

$\mathcal{T}$ is said to be *k-bounded* if for any (m,v) reachable from $(m_0, \mathbf{0})$ in $\mathcal{S}_{\mathcal{T}}$, we have $\forall p \in P, m(p) \leq k$. $\mathcal{T}$ is said to be *bounded* if there exists k such that $\mathcal{T}$ is k-bounded.

2.3. Networks of Timed Automata

Timed Automata [1] are used to model systems which combine *discrete* and *continuous* evolutions.

Definition 2.6. (Timed Automaton)
A *Timed Automaton* (TA) is a tuple $\mathcal{A} = \langle L, \ell_0, E, A, \lambda, X, \mathsf{Guard}, \mathsf{Resets}, \mathsf{Inv} \rangle$ where: L is a finite non-empty set of *locations*; $\ell_0 \in L$ is the *initial* location; $E \subseteq L \times L$ is a finite set of directed *edges*; A is finite non-empty *alphabet*; $\lambda : E \to A$ is the *edge labelling* function; $X$is a finite set of positive real-valued *clocks*; $\mathsf{Guard} : E \to \mathcal{C}(X)$ gives a *guard* for each edge; $\mathsf{Resets} : E \to 2^X$ gives a set of *clocks to reset* for each edge; $\mathsf{Inv} : L \to \mathcal{B}(X)$ defines a set of *invariants*;

Definition 2.7. (Semantics of TA)
The semantics of a timed automaton $\mathcal{A} = \langle L, \ell_0, E, A, \lambda, X, \mathsf{Guard}, \mathsf{Resets}, \mathsf{Inv} \rangle$ is a timed transition system $S_{\mathcal{A}} = (Q, q_0, A, \to)$ with $Q = L \times (\mathbb{R}_{\leq 0})^X$, $q_0 = (l_0, \mathbf{0})$ is the initial state and $\to$ consists of the discrete and continuous transition relations:

- $(l,v) \xrightarrow{a \in A} (l',v')$ iff $\exists e = (l,l') \in E$ such that: $\begin{cases} \lambda(e) = a, \\ v \models \mathsf{Guard}(e), \\ v' = v[\mathsf{Resets}(e) \leftarrow 0], \\ v' \models \mathsf{Inv}(l') \end{cases}$

- $(l,v) \xrightarrow{d \in \mathbb{R}_{\geq 0}} (l, v+d)$ iff $\forall d'\ 0 < d' \leq d, v + d' \models \mathsf{Inv}(l)$

A *run* of a timed automaton $\mathcal{A}$ is a path in $S_{\mathcal{A}}$ starting in q_0.

It is convenient to describe a system as a parallel composition of timed automata. To this end, we use the classical composition notion based on a *synchronization function* à la Arnold-Nivat.

Definition 2.8. (Networks of Timed Automata)
Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be n timed automata with $\mathcal{A}_i = \langle L_i, \ell_{0_i}, E_i, A, \lambda_i, X, \mathsf{Guard}_i, \mathsf{Resets}_i, \mathsf{Inv}_i \rangle$. A *synchronization function* f is a partial function from $(A \cup \{\bullet\})^n$ to A where $\bullet$ is a special symbol used when an automaton is not involved in a step of the global system. A Network of Timed Automata $(\mathcal{A}_1 | \ldots | \mathcal{A}_n)_f$ is the parallel composition of the $\mathcal{A}_i$'s w.r.t. f.

The configurations of $(\mathcal{A}_1|\ldots|\mathcal{A}_n)_f$ are pairs $(\vec{l}, v)$ with $\vec{l} = (l_1, \ldots, l_n) \in L_1 \times \ldots \times L_n$, the i^{th} component $l_i \in L_i$ of $\vec{l}$ is denoted by $\vec{l}[i]$, v is a valuation on the set of clocks X and $v(x)$ is the value of the clock $x \in X$. The network can do a discrete transition if all the components agree to and time can progress in the network also if all the components agree to. This is formalized by the following definition:

Definition 2.9. (Semantics of NTA)
Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be n TA with $\mathcal{A}_i = \langle L_i, \ell_{0_i}, E_i, A, \lambda_i, X, \mathsf{Guard}_i, \mathsf{Resets}_i, \mathsf{Inv}_i\rangle$, $S_{\mathcal{A}_1}, \ldots, S_{\mathcal{A}_n}$ their semantics with $S_{\mathcal{A}_i} = (Q_i, q_{0_i}, A, \rightarrow_i)$. Let f be a (partial) synchronization function $(A \cup \{\bullet\})^n \rightarrow A$. The semantics of $(\mathcal{A}_1|\ldots|\mathcal{A}_n)_f$ is a timed transition system $S = (Q, q_0, A, \rightarrow)$ with $Q = L_1 \times \ldots \times L_n \times (\mathbb{R}_{\geq 0})^X$, q_0 is the initial state $((\ell_{0_1}, \ldots, \ell_{0_n}), \mathbf{0})$ and $\rightarrow$ is defined by:

- $(\vec{l}, v) \xrightarrow{b \in A} (\vec{l'}, v')$ iff
  - Let $\mathsf{R} = \bigcup_{i \in [1..n], (\vec{l}[i], \vec{l'}[i]) \in E_i} \mathsf{Resets}_i((\vec{l}[i], \vec{l'}[i]))$. Then $v' = v[\mathsf{R} \leftarrow 0]$,
  - all $\mathcal{A}_i$ agree on synchronization *i.e.* $\exists (a_1, \ldots, a_n) \in (A \cup \{\bullet\})^n$ s.t. $f(a_1, \ldots, a_n) = b$ and for any $i \in [1..n]$ we have:
    - If $a_i = \bullet$, then $\vec{l'}[i] = \vec{l}[i]$,
    - If $a_i \in A$, then $(\vec{l}[i], v) \xrightarrow{a_i}_i (\vec{l'}[i], v'_i)$. Note that $\forall x \in X \backslash \mathsf{Resets}$, $v'(x) = v'_i(x)$
- $(\vec{l}, v) \xrightarrow{d \in \mathbb{R}_{\geq 0}} (\vec{l}, v + d)$ iff for all $i \in [1..n]$, every $\mathcal{A}_i$ agrees on time elapsing *i.e.* $(\vec{l}[i], v) \xrightarrow{d}_i (\vec{l}[i], v + d)$

3. Clock Transition Systems

Definition 3.1. (Clock Transition System)
A (labeled) *Clock Transition System* is a tuple $\langle V, T, \mathsf{Pre}, \mathsf{Post}, m_0, A, \lambda, X, \mathsf{Guard}, \mathsf{Resets}, \mathsf{Inv}\rangle$ such that:

- V is a finite non-empty set of integer variables;
- T is a finite non-empty set of transitions;
- $\mathsf{Pre} : T \rightarrow \mathcal{F}(\mathbb{N}^V, \mathbb{B})$ gives a *discrete guard* for each transition;
- $\mathsf{Post} : T \rightarrow \mathcal{F}(\mathbb{N}^V, \mathbb{N}^V)$ gives a *discrete assignment* for each transition;
- m_0 is the initial valuation of V;
- A is a finite non-empty *alphabet*;
- $\lambda : T \rightarrow A$ is a *labeling function* of the transitions; X is a finite set of *clocks*;
- $\mathsf{Guard} : T \rightarrow \mathcal{C}(X)$ gives a *time guard* for each transition;
- $\mathsf{Resets} : T \rightarrow 2^{\mathcal{F}(\mathbb{N}^V, \mathbb{B}) \times X}$ defines a conditional *reset* of clocks on transitions;
- $\mathsf{Inv} \subseteq \mathcal{F}(\mathbb{N}^V, \mathbb{B}) \times \mathcal{B}(X)$ defines a finite set of *invariants*.

The semantics of the CTS $\mathcal{T} = \langle V, T, \mathsf{Pre}, \mathsf{Post}, m_0, A, \lambda, X, \mathsf{Guard}, \mathsf{Resets}, \mathsf{Inv}\rangle$ is defined by the timed transition system $\mathcal{S}_\mathcal{T} = (\mathbb{N}^V \times \mathbb{R}^X_{\geq 0}, (m_0, \mathbf{0}), A, \rightarrow)$ such that

- $(m,v) \xrightarrow{a\in A} (m',v')$ iff $\exists t \in T$ s.t. $\begin{cases} \mathsf{Pre}(t)(m) \text{ is true} \\ \lambda(t) = a, \\ m' = \mathsf{Post}(t)(m), \\ v \models \mathsf{Guard}(t), \\ v' = v\big[\{x|(f,x) \in \mathsf{Resets}(t) \text{ and } f(m)\} \leftarrow 0\big], \\ \forall (f,J) \in \mathsf{Inv}, f(m') \text{ implies } v' \models J. \end{cases}$
- $(m,v) \xrightarrow{d\in\mathbb{R}_{\geq 0}} (m,v+d)$ iff $\forall (f,J) \in \mathsf{Inv}$, $f(m) \Rightarrow \forall 0 < d' \leq d, v+d' \models J$.

Boundedness of CTSs is defined exactly as for TPNs.

State space and main properties. Clock Transition Systems only feature explicit clocks and integer variables. We have shown in [17] how the classical simulation/zone graph construction, used in the tool Uppaal [16], can be easily extended to CTSs. For *bounded CTS* this abstraction gives a finite representation of the infinite state-space and many analysis techniques can be constructed to decide safety, reachability, liveness, etc. We then obtain the following theorems.

Theorem 3.2. k-boundedness is decidable for CTSs. Reachability is decidable for bounded CTSs.

3.1. TPNs and CTSs

We now prove that possibly unbounded TPNs form a subclass of CTSs. First, the following theorem holds:

Theorem 3.3. Every TPN $\mathcal{N}$ can be translated into a CTS $\mathcal{T}(\mathcal{N})$ s.t. $\mathcal{N} \cong \mathcal{T}(\mathcal{N})$.

Proof:
Let $\mathcal{N} = \langle P, T, \mathsf{Pre}, \mathsf{Post}, m_0, A, \lambda, I_s\rangle$ be a TPN. And let $\mathcal{T}(\mathcal{N}) = \langle P, T, \mathsf{Pre}', \mathsf{Post}', m_0, A, \lambda, X, \mathsf{Guard}, \mathsf{Resets}, \mathsf{Inv}\rangle$ be the Clock Transition System defined by:

- for each $t \in T$, $\mathsf{Pre}'(t) = f_t$ with $f_t(m)$ iff $t \in \mathsf{en}(m)$;
- for each $t \in T$, $\mathsf{Post}'(t) = g$ with $\forall p \in P, g(m)(p) = m(p) - \mathsf{Pre}(p,t) + \mathsf{Post}(p,t)$;
- $X = T$;
- $\forall t \in T, \mathsf{Guard}(t) = t \in I_s(t)$, with a slight abuse on the expression of constraints;
- $\forall t \in T, \mathsf{Resets}(t) = \{(h_{t'}, t') | t' \in T \text{ s.t. } t^\bullet \cap {}^\bullet t' \neq \emptyset\}$, with $h_{t'}(m)$ is true iff $t' \in \mathsf{new}(m,t)$;
- $\mathsf{Inv} = \{(f_t, t \in I_s^{\downarrow}(t)) | t \in T\}$.

The operational semantics of $\mathcal{T}(\mathcal{N})$ is defined by the time transition system $\mathcal{S}_{\mathcal{T}(\mathcal{N})} = (\mathbb{N}^V \times \mathbb{R}_{\geq 0}^T, (m_0, \mathbf{0}), A, \rightarrow)$ such that:

- $(m,v) \xrightarrow{a\in A} (m',v')$ iff $\exists t \in T$ such that: $\begin{cases} t \in \mathsf{en}(m), \\ \lambda(t) = a, \\ \forall p \in P, m'(p) = m(p) - \mathsf{Pre}(p,t) + \mathsf{Post}(p,t), \\ v \models t \in I_s(t), \\ v' = v[\{t' | t' \in \mathsf{new}(m,t)\} \leftarrow 0], \\ \forall t' \in \mathsf{en}(m), v' \models t' \in I_s^{\downarrow}(t') \end{cases}$

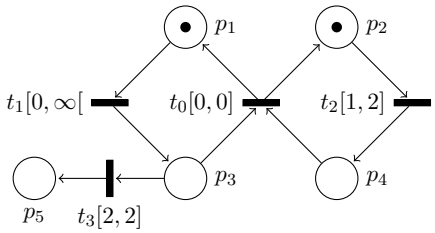


Figure 1. A Time Petri Net.

- $(m, v) \xrightarrow{d \in \mathbb{R}_{\geq 0}} (m, v + d)$ iff $\forall t' \in \mathsf{en}(m), \forall d',\ 0 < d' \leq d, v + d' \models t' \in I_s^{\downarrow}(t')$.

This is exactly the semantics of $\mathcal{N}$. This proves the theorem. □

$V = \{p_1, p_2, p_3, p_4, p_5\}, T = \{t_0, t_1, t_2, t_3\}$	$m_0 = (1, 1, 0, 0, 0), X = T$
$\mathsf{Guard}(t_0) = (t_0 = 0), \mathsf{Guard}(t_1) = (t_1 \geq 0)$	$\mathsf{Guard}(t_2) = (1 \leq t_2 \leq 2), \mathsf{Guard}(t_3) = (t_3 = 2)$
$\mathsf{Pre}(t_0) = (p_3 \geq 1) \wedge (p_4 \geq 1)$	$\mathsf{Resets}(t_0) = \{((p_1 = 0), t_1), ((p_2 = 0), t_2)\}$
$\mathsf{Pre}(t_1) = (p_1 \geq 1), \mathsf{Pre}(t_2) = (p_2 \geq 1)$	$\mathsf{Resets}(t_1) = \{((p_3 = 0), t_0), ((p_3 = 0), t_3)\}$
$\mathsf{Pre}(t_3) = (p_3 \geq 1)$	$\mathsf{Resets}(t_2) = \{((p_4 = 0), t_0)\}, \mathsf{Resets}(t_3) = \emptyset$
$\mathsf{Post}(t_1) = (p_1 := p_1 - 1, p_3 := p_3 + 1)$	$\mathsf{Post}(t_2) = (p_2 := p_2 - 1, p_4 := p_4 + 1)$
$\mathsf{Post}(t_3) = (p_3 := p_3 - 1, p_5 := p_5 + 1)$	
$\mathsf{Post}(t_0) = (p_1 := p_1 + 1, p_2 := p_2 + 1, p_3 := p_3 - 1, p_4 := p_4 - 1)$	
$\mathsf{Inv} = \{(\mathsf{Pre}(t_0), (t_0 = 0)), (\mathsf{Pre}(t_1), \mathsf{true}), (\mathsf{Pre}(t_2), (t_2 \leq 2)), (\mathsf{Pre}(t_3), (t_3 \leq 2))\}$	

Table 1. CTS coding the TPN of Fig. 1.

To illustrate the encoding, consider the TPN in Fig. 1. Its equivalent in CTS is given in Table 1. We now define a syntactic subclass of CTSs that is equivalent to TPNs:

Definition 3.4. The syntactic subclass CTS-TPN of CTS is defined by the following restrictions:

- $\forall t \in T, \forall p \in V$, there exists $k(p,t) \in \mathbb{N}, k'(p,t) \in \mathbb{Z}$ s.t.:
  - $k'(p,t) \geq -k(p,t)$;
  - $\mathsf{Pre}(t) = \bigwedge_{p \in P} p \geq k(p,t)$ and $\mathsf{Post}(t)$ is a list of assignments $\forall p, p := p + k'(p,t)$;
  - For a valuation m, we define m'_t by $\forall p \in P, m'_t(p) = m(p) - k(p,t) + k'(p,t)$ and m''_t by $\forall p \in P, m''_t(p) = m(p) - k(p,t)$.
    Then $\mathsf{Resets}(t) = \{(g_{t'}, x_{t'}) | t' \in T\}$ and $g_{t'}(m)$ holds iff $t = t'$ or $(\mathsf{Pre}(t')(m'_t)$ and not $\mathsf{Pre}(t')(m''_t))$;

- $\forall t \in T$, $\mathsf{Guard}(t)$ refers to at most one clock x_t and $x_t = x_{t'}$ implies $t = t'$;
- $\mathsf{Inv} = \{(\mathsf{Pre}(t), J_t) | t \in T\}$ (note that J_t may be true);
- $\forall t \in T$, J_t refers only to x_t and is not equal to $x_t < 0$. Furthermore, if $J_t = x_t \leq k$ or $J_t = x_t < k$, then the set of valuations satisfying $\mathsf{Guard}(t) \wedge x_t = a$ is non-empty;
- $\forall t \in T$, if $J_t = \mathsf{true}$ then $\mathsf{Guard}(t)$ has no finite upper bound.

Theorem 3.5. Every CTS-TPN $\mathcal{T}$ can be translated into a TPN $\mathcal{N}(\mathcal{T})$ such that $\mathcal{T} \cong \mathcal{N}(\mathcal{T})$.

Proof:
Let $\mathcal{T} = \langle P, T, \mathsf{Pre}, \mathsf{Post}, m_0, A, \lambda, X, \mathsf{Guard}, \mathsf{Resets}, \mathsf{Inv} \rangle$ be a CTS-TPN with the above restrictions. Let $\mathcal{N}(\mathcal{T}) = \langle P, T, \mathsf{Pre}', \mathsf{Post}', m_0, A, \lambda, I_s \rangle$ be the TPN defined by:

- for $p \in P$ and $t \in T$, $\mathsf{Pre}'(p,t) = k(p,t)$ and $\mathsf{Post}'(p,t) = k(p,t) + k'(p,t)$ (as in definition 3.4);
- for $t \in T$, $\mathsf{Guard}(t)$ defines a right-unbounded interval I_g. Then $I_s(t) = I_g \cap \bigcap_{t' \in T \text{ s.t. } \mathsf{Pre}(t')} J_{t'}$.

Let us write the semantics of $\mathcal{T}$ taking the restrictions of CTS-TPN into account. The operational semantics of $\mathcal{T}$ is defined by the time transition system $\mathcal{S}_{\mathcal{T}} = (\mathbb{N}^V \times \mathbb{R}^X_{\geq 0}, (m_0, \mathbf{0}), A, \rightarrow)$ such that:

- $(m, v) \xrightarrow{a \in A} (m', v')$ iff there exists $t \in T$ such that:
  - $\mathsf{Pre}(t)$;
  - $\lambda(t) = a$;
  - $\forall p \in P, m'(p) = m(p) - k(p,t) + k'(p,t)$;
  - $v \models x_t \in I_g$: since x_t is the only variable used, the constraint is an inteval I_g;
  - $v' = v\big[\{x_{t'} | (t = t') \vee (\mathsf{Pre}(t')(m') \wedge \neg \mathsf{Pre}(t')(m''_t))\} \leftarrow 0\big]$;
  - $\forall t \in T$, $\mathsf{Pre}(t)(m')$ implies $v' \models J_t$.
- $(m, v) \xrightarrow{d \in \mathbb{R}_{\geq 0}} (m, v + d)$ iff $\forall 0 < d' \leq d, v + d' \models \bigcap_{t \in T \text{ s.t. } \mathsf{Pre}(t)} J_t$.

It is clear that we have $\mathsf{Pre}(t)(m)$ in $\mathcal{T}$ iff $t \in \mathsf{en}(m)$ in $\mathcal{N}(\mathcal{T})$. And therefore, $\{x_{t'} | (t = t') \vee (\mathsf{Pre}(t')(m') \wedge \neg \mathsf{Pre}(t')(m''_t))\}$ is indeed the set of clocks associated to transitions that are newly enabled in $\mathcal{N}$ when firing transition t from m.

Consequently, for $t' \in T$, if $\mathsf{Pre}(t')(m')$ we have two possibilities:

- either, $v'(x_{t'}) = 0$ and then $v' \models J_t$ since J_t is not $x_t < 0$;
- or $v'(x_{t'}) \neq 0$. But then $\mathsf{Pre}(t')(m)$ and $v'(x_{t'}) = v(x_{t'})$. We thus had $v \models J_t$ and we have $v' \models J_t$.

We have $v \models \mathsf{Guard}(t) = x_t \in I_g$ but we know that v must also satisfy all the enabled invariants so we could actually write this as: $v \models x_t \in I_g \cap \bigcap_{t' \in T \text{ s.t. } \mathsf{Pre}(t')} J_{t'}$. Finally, it easy to see that $I_g \cap \bigcap_{t' \in T \text{ s.t. } \mathsf{Pre}(t')} J_{t'}$ is an interval I and $I^{\downarrow} = \bigcap_{t' \in T \text{ s.t. } \mathsf{Pre}(t')} J_{t'}$.

With all this, it is clear that the semantics of $\mathcal{T}$ is exactly the semantics of $\mathcal{N}(\mathcal{T})$, and the theorem follows. □

Corollary 3.6. The class CTS-TPN is equivalent to the class of TPNs up to isomorphism of TTS.

Proof:
The CTS exhibited in the proof of theorem 3.3 clearly belongs to CTS-TPN. □

3.2. NTA and CTSs

Now we prove that NTA form a subclass of CTSs and are indeed equivalent to *bounded* CTSs.

Theorem 3.7. Every NTA $\mathcal{A}$ can be translated into a CTS $\mathcal{T}(\mathcal{A})$ s.t. $\mathcal{A} \cong \mathcal{T}(\mathcal{A})$.

Proof:
Let $\mathcal{A} = (\mathcal{A}_1 | \dots | \mathcal{A}_n)_f$ be a NTA, with for all i, $\mathcal{A}_i = \langle L_i, \ell_{0_i}, E_i, A, \lambda_i, X, \mathsf{Guard}_i, \mathsf{Resets}_i, \mathsf{Inv}_i \rangle$.
Let us define the CTS $\mathcal{T}(\mathcal{A}) = \langle P, T, \mathsf{Pre}, \mathsf{Post}, m_0, A, \lambda', X, \mathsf{Guard}', \mathsf{Resets}', \mathsf{Inv}' \rangle$ such that:

- $P = \{p_1, \dots, p_n\}$ contains one variable p_i for each TA $\mathcal{A}_i$. Suppose w.l.o.g. that locations are actually integers so that we can write, for instance, $m(p_1) = \ell_{0_1}$;
- Let $\perp$ be a special symbol belonging to none of the E_i. For all i, we extend λ_i to a function of $E_i \cup \{\perp\}$ to $A \cup \{\bullet\}$.
  T is the set of elements $(e_1, \dots, e_n)$ in $E_1 \cup \{\perp\} \times \dots \times E_n \cup \{\perp\}$ such $f(\lambda_1(e_1), \dots, \lambda_n(e_n))$ is defined;
- For $t = (e_1, \dots, e_n) \in T$, we have:
  - $\mathsf{Pre}(t)(m)$ is true iff for all i such that $e_i \neq \perp$, writing $e_i = (l_i, l_i')$, $m(p_i) = l_i$;
  - $\mathsf{Post}(t)(m) = m'$ with for all i such that $e_i \neq \perp$, writing $e_i = (l_i, l_i')$, $m'(p_i) = l_i'$ and for all i such that $e_i = \perp$, $m'(p_i) = l_i$;
  - $\lambda'(t) = f(\lambda_1(e_1), \dots, \lambda_n(e_n))$;
  - $\mathsf{Guard}'(t) = \bigwedge_{e_i \neq \perp} \mathsf{Guard}_i(e_i)$;
  - $\mathsf{Resets}'(t) = \{(\mathsf{true}, x) | x \in \bigcup_{e_i \neq \perp} \mathsf{Resets}_i(e_i)\}$, true being the function that returns always true;
- $\forall i \in [1..n]$, $m_0(p_i) = \ell_{0_i}$;
- $\mathsf{Inv}' = \{(g_{l_i}, \mathsf{Inv}_i(l)) | l_i \in L_i, i \in [1..n]\}$, with $g_{l_i}(m)$ iff $m(p_i) = l_i$.

Let us explicit the semantics of $\mathcal{T}(\mathcal{A})$: it is the TTS $\mathcal{S}_{\mathcal{T}} = (\mathbb{N}^V \times \mathbb{R}_{\geq 0}^X, (m_0, \mathbf{0}), A, \rightarrow)$ such that:

- $(m, v) \xrightarrow{b \in A} (m', v')$ iff there exists $t = (e_1, \dots, e_n) \in T$ such that:

$$\begin{cases} \forall e_i \neq \perp, m(p_i) = l_i, \text{ by writing } e_i = (l_i, l_i'), \\ b = f(\lambda(e_1), \dots, \lambda(e_n)), \\ m'(p_i) = l_i' \text{ if } \lambda(e_i) \neq \bullet \text{ and } m'(p_i) = l_i \text{ otherwise }, \\ \forall i, v \models \mathsf{Guard}_i(e_i), \\ v' = v\big[R \leftarrow 0\big] \text{ with } R = \bigcup_{i \in [1..n], e_i \neq \perp} \mathsf{Resets}_i(e_i), \\ \forall i, v' \models \mathsf{Inv}_i(m'(p_i)), \end{cases}$$

- $(m,v) \xrightarrow{d\in\mathbb{R}_{\geq 0}} (m, v+d)$ iff $\forall i, \forall 0 < d' \leq d, v+d' \models \mathsf{Inv}_i(m(p_i))$.

The very last condition exactly means that for all $\mathcal{A}_i$, $(m(p_i), v) \xrightarrow{d} (m(p_i), v+d)$ in its semantics. We can now rewrite the other condition as: $(m,v) \xrightarrow{b\in A} (m',v')$ iff there exists $t = (e_1, \ldots, e_n) \in T$ such that $b = f(\lambda(e_1), \ldots, \lambda(e_n))$ and for all $i \in [1..n]$:

- either $\lambda_i(e_i) = \bullet$ and $m(p_i) = m'(p_i)$;
- or $\lambda_i(e_i) \neq \bullet$, $\exists l_i, l_i'$ s.t. $e_i = (l_i, l_i')$ and $\begin{cases} m(p_i) = l_i \text{ and } m'(p_i) = l_i', \\ v \models \mathsf{Guard}_i(e_i), \\ v' = v\big[R \leftarrow 0\big] \text{ with } R = \bigcup_{i\in[1..n], e_i \neq \perp} \mathsf{Resets}_i(e_i), \\ v' \models \mathsf{Inv}_i(l_i') \end{cases}$

This means: $(m,v) \xrightarrow{b\in A} (m',v')$ iff there exists $(a_1, \ldots, a_n) \in (A \cup \{\bullet\})^n$ such that $b = f(a_1, \ldots, a_n)$ and for all $i \in [1..n]$:

- either $a_i = \bullet$ and $m(p_i) = m'(p_i)$;
- or $a_i \neq \bullet$ and $\exists l_i, l_i'$ such that: $\begin{cases} m(p_i) = l_i \text{ and } m'(p_i) = l_i', \\ (l_i, v) \xrightarrow{a_i} (l_i', v''), \text{ for some } v'' \text{ s.t. } v''(x) = 0 \text{ implies } v'(x) = 0. \end{cases}$

The bijection $h : L_1 \times \cdots \times L_n \times \mathbb{R}_{\geq 0} \to \mathbb{N}^P \times \mathbb{R}_{\geq 0}$ such that $h(l_1, \ldots, l_n, v)(p_i) = (m, v)$, with $\forall i, m(p_i) = l_i$, is therefore a graph isomorphism between the semantics of $\mathcal{A}$ and $\mathcal{T}(\mathcal{A})$ since it allows to find exactly the semantics of $\mathcal{T}(\mathcal{A})$ from that of $\mathcal{A}$. □

To illustrate the encoding, consider the NTA in Fig. 2. Its equivalent in CTS is given in Table 2.

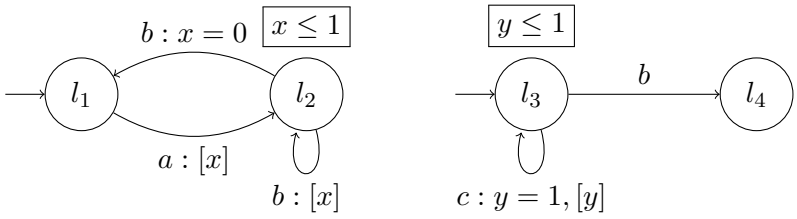


Figure 2. A network of two timed automata with two clocks x and y. The boxed constraints above the locations are invariants. Transitions are labeled by their action (a, b or c), the guard on clocks and resets (bracketed).

Theorem 3.8. Every bounded CTS $\mathcal{T}$ can be translated into a TA $\mathcal{A}(\mathcal{T})$ s.t. $\mathcal{T} \cong \mathcal{A}(\mathcal{T})$.

Proof:

Let $\mathcal{T} = \langle V, T, \mathsf{Pre}, \mathsf{Post}, m_0, A, \lambda, X, \mathsf{Guard}, \mathsf{Resets}, \mathsf{Inv}\rangle$ be a k-bounded CTS. Let us define the TA $\mathcal{A}(\mathcal{T}) = \langle L, \ell_0, E, A, \lambda_{\mathcal{A}}, X, \mathsf{Guard}_{\mathcal{A}}, \mathsf{Resets}_{\mathcal{A}}, \mathsf{Inv}_{\mathcal{A}}\rangle$ such that:

$V = \{p_1, p_2\}$	$m_0 = (l_1, l_3)$
$T = \{A, B_1, B_2, C\}$	$X = \{x, y\}$
$\lambda(A) = a, \lambda(B_1) = \lambda(B_2) = b, \lambda(C) = c$	$\mathsf{Guard}(A) = \mathsf{Guard}(B_1) = \mathsf{true}$
$\mathsf{Pre}(A) = (p_1 = l_1), \mathsf{Post}(a) = (p_1 := l_2)$	$\mathsf{Guard}(B_2) = (x = 0)$
$\mathsf{Pre}(B_1) = ((p_1, p_2) = (l_2, l_3))$	$\mathsf{Guard}(C) = (y = 1)$
$\mathsf{Post}(B_1) = ((p_1, p_2) := (l_2, l_4))$	$\mathsf{Resets}(A) = \mathsf{Resets}(B_1) = \{(\mathsf{true}, x)\}$
$\mathsf{Pre}(B_2) = ((p_1, p_2) = (l_2, l_3))$	$\mathsf{Resets}(B_2) = \emptyset$
$\mathsf{Post}(B_2) = ((p_1, p_2) := (l_1, l_4))$	$\mathsf{Resets}(C) = \{(\mathsf{true}, y)\}$
$\mathsf{Pre}(C) = (p_2 = l_3), \mathsf{Post}(C) = (p_2 := l_3)$	

$$\mathsf{Inv} = \{((p_1 = l_1), \mathsf{true}), ((p_1 = l_2), (x \leq 1)), ((p_2 = l_3), (y \leq 1)), ((p_2 = l_4), \mathsf{true})\}$$

Table 2. CTS coding the NTA of Fig. 2.

- L is a set of $(k+1)^{|V|}$ locations. To each location $l \in L$, is associated a value $\mathbf{m}(l)$ of the set of variables V ($\mathbf{m} : L \to \mathbb{N}^V$);
- ℓ_0 is the location such that $\mathbf{m}(\ell_0) = m_0$
- E is a subset of $L \times L$ and is the set of directed edges;
- The invariant associated with each location $l \in L$ is defined by:

$$\mathsf{Inv}_{\mathcal{A}}(l) = \{J \mid \exists (f, J) \in \mathsf{Inv} \textit{ st } f(\mathbf{m}(l)) = \mathsf{true}\}$$

- $\forall e = (l, l') \in L \times L$, such that $\exists t \in T$, $\mathsf{Pre}(t)(\mathbf{m}(l)) = \mathsf{true}$ and $\mathbf{m}(l') = \mathsf{Post}(\mathbf{m}(l))$, we add e to E and we label e by:
  - the action name $\lambda_{\mathcal{A}}(e) = \lambda(t)$,
  - the guard: $\mathsf{Guard}_{\mathcal{A}}(e) = \mathsf{Guard}(t)$,
  - the clocks assignments: $\mathsf{Resets}_{\mathcal{A}}(e) = \{x \mid \exists (h, x) \in \mathsf{Resets}(t) \textit{ s.t. } h(\mathbf{m}(l)) = \mathsf{true}\}$

We first let $\mathcal{R} \subseteq Q_{\mathcal{T}} \times Q_{\mathcal{A}}$, the relation between a state of the Timed Automaton and a state of the Clock Transition System defined by:

$$\begin{cases} \forall (m, v_{\mathcal{T}}) \in Q_{\mathcal{T}} \\ \forall (l, v_{\mathcal{A}}) \in Q_{\mathcal{A}} \end{cases}, \ (m, v_{\mathcal{T}}) \mathcal{R} (l, v_{\mathcal{A}}) \Leftrightarrow \begin{cases} m = \mathbf{m}(l) \\ v_{\mathcal{T}} = v_{\mathcal{A}} \end{cases}$$

First, by construction, given a value m, there is one and only one location $l \in L$ in $\mathcal{A}(\mathcal{T})$ such that $\mathbf{m}(l) = m$. Then, since $(m, v_{\mathcal{T}}) \mathcal{R} (l, v_{\mathcal{A}})$ implies $v_{\mathcal{T}} = v_{\mathcal{A}}$, $\mathcal{R}$ is a bijection.

Let $(m, v) \in Q_{\mathcal{T}}$ and $(l, v) \in Q_{\mathcal{A}}$ such that $(m, v) \mathcal{R} (l, v)$. In both models $\mathcal{T}$ and $\mathcal{A}(\mathcal{T})$, invariants are associated to discrete states m and l and are identical by construction: $\mathsf{Inv}_{\mathcal{A}}(l) = \{J \mid \exists (f, J) \in \mathsf{Inv} \textit{ st } f(\mathbf{m}(l)) = \mathsf{true}\}$. Moreover there is an arc between l and l' in $\mathcal{A}(\mathcal{T})$ iff $\exists t \in T$, $\mathsf{Pre}(t)(\mathbf{m}(l)) =$

true and $\mathbf{m}(l') = \mathsf{Post}(\mathbf{m}(l))$ and guards, resets and labeling are syntactically identical for arc (l, l') in $\mathcal{A}(\mathcal{T})$ and transition t leading from $m = \mathbf{m}(l)$ to $m' = \mathbf{m}(l')$ in $\mathcal{T}$. Then, for any reachable state (m, v) of $S_{\mathcal{T}}$ the conditions for the firing of an action $a \in A$ or for the elapsing of $d \in \mathbb{R}_{\geq 0}$ are identical to those from the state (l, v) of $S_{\mathcal{A}}$. Thus,

- the firing of a discrete transition gives $(m, v) \xrightarrow{a \in A} (m', v'_{\mathcal{T}}) \Leftrightarrow (l, v) \xrightarrow{a} (l', v'_{\mathcal{A}})$ and since the sets of clock resets are identical, we have $v'_{\mathcal{T}} = v'_{\mathcal{A}}$ and then $(m', v'_{\mathcal{T}})\mathcal{R}(l', v'_{\mathcal{A}})$;
- the elapsing of time increase the valuation of v and then obviously: $(m, v) \xrightarrow{d \in \mathbb{R}_{\geq 0}} (m, v + d) \Leftrightarrow \begin{cases} (l, v) \xrightarrow{d} (l, v + d) \\ (m, v + d)\mathcal{R}(l, v + d) \end{cases}$

□

Moreover, since the CTS exhibited in the proof of theorem 3.7 is clearly bounded, we have:

Corollary 3.9. The class of bounded CTSs is equivalent to the class of TA up to isomorphism of TTS.

4. Discussion

Fig. 3 summurizes how the different classes are intertwined with each other. $2CM$ stands for 2-counter machines ; CTS_0 is the CTS coding the Timed Automaton A_0 proposed in [4] such that there is no TPN weakly bisimilar to A_0. CTS_∞ is the CTS obtained from CTS_0 by adding an unbounded discrete behavior. This CTS is obviously neither a TPN nor a TA.

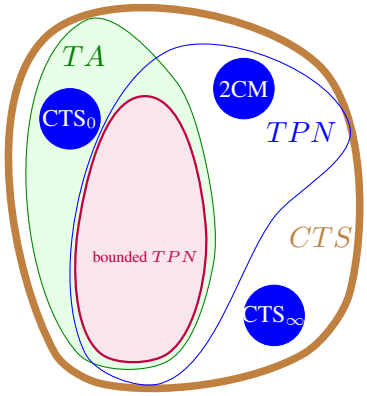


Figure 3. Expressiveness sum up

$\mathbf{CTS_0}$: $V = \{p\}$; $m_0 = l_0$; $T = \{t_a\}$; $X = \{x\}$; $\lambda(t_a) = a$; $\mathsf{Guard}(t_a) = (x < 1)$; $\mathsf{Pre}(t_a) = l_0$; $\mathsf{Post}(t_a) = l_1$; $\mathsf{Resets}(t_a) = \emptyset$;

$\mathbf{CTS_\infty}$: $V = \{p_1, p_2\}$; $m_0 = (l_0, 0)$; $T = \{t_a, t_b\}$; $X = \{x\}$; $\lambda(t_a) = a$; $\lambda(t_b) = b$; $\mathsf{Guard}(t_a) = (x < 1)$; $\mathsf{Guard}(t_b) = (x > 0)$; $\mathsf{Pre}(t_a) = (p_1 = l_0)$; $\mathsf{Post}(t_a) = (p_1 := l_1)$; $\mathsf{Pre}(t_b) = (p_1 = l_1)$; $\mathsf{Post}(t_b) = (p_1 := l_1, p_2 := p_2 := p_2 + 1)$; $\mathsf{Resets}(t_a) = \emptyset$; $\mathsf{Resets}(t_b) = \{x\}$

As we have seen in the previous section, the expressive power and conciseness of Clock Transition Systems are two of their best assets. Furtermore, since both TA and TPNs can be easily transformed in Clock Transition Systems, whose sizes are linear wrt. that of the TA or TPN, one can imagine a modeling workflow in which sequential components are modeled as TA, components featuring complex synchronization are modeled as TPNs, and complex dynamics are directly discretized in the form of Clock Transition Systems. This mixed modeling can ultimately be transformed in Clock Transition Systems for the analysis. We can lift most of the analysis techniques developed for (time) Petri nets and (timed) automata to CTS. For instance:

- For *unbounded untimed CTSs*, given adequate restrictions on the discrete guard and assignment functions (such as those in the subclass CTS-TPN), we can compute a coverability graph [13].

- For *bounded CTSs (with time)*, we can easily extend the region abstraction [1] or the zone abstraction used in the tool Uppaal [16]. We have shown in [17] how the zone graph construction can be extended to CTSs. These abstractions give a finite representation of the infinite state-space. From these basic abstractions, many analysis techniques can be constructed to decide safety, reachability, liveness, etc.

- For *potentially unbounded CTSs (with time)*, the techniques based on these abstractions become semi-algorithms as for potentially unbounded TPN [7]. A few interesting problems are still decidable though, e.g. k-boundedness and even safety control of the unbounded CTS to automatically make it bounded using the technique of [8]. It should also be possible to apply supervision techniques like in [9].

Finally, new techniques developed directly for CTSs can be immediately applied to both TPNs and TA, thus reducing the duplication of efforts.

5. Conclusion and perspectives

We have defined the new model of clock transition systems. It blends concepts from both time Petri nets and networks of timed automata. That means that CTS is a good intermediate model to develop tools, while factoring software developments. We showed that (in terms of isomorphism of TTS formal semantics):

- TPNs and TA may be encoded using CTSs;

- The syntactic subclass CTS-TPNs forms exactly the set of TPNs;

- Bounded CTSs form exactly the set of Timed Automata;

- Computation of a symbolic state-space is possible for CTSs and, in particular, allows model-checking.

The other contribution is that CTSs ultimately appear to be a powerful and concise formalism for describing timed models. One could also imagine a possible mixture of NTA, TPNs and CTSs to model complex timed behaviors, all of them being ultimately transcribed into CTSs, analyzed by a unique engine.

The outlook is therefore to start from this model for our next developments in the tool Romeo [18]. In particular, we will equip this model with a concurrent semantics to build timed unfoldings, extending techniques from [12].

References

[1] Alur, R., Dill, D. L.: A theory of timed automata, *Theoretical Computer Science*, **126**(2), 1994, 183–235.

[2] Alur, R., Henzinger, T. A.: Real-time system = discrete system + clock variables, *Theories and Experiences for Real-Time System Development, AMAST Series in Computing*, 2, 1994.

[3] Alur, R., Henzinger, T. A.: Real-time system = discrete system + clock variables, *Software Tools for Technology Transfer*, **1**, 1997, 86–109.

[4] Bérard, B., Cassez, F., Haddad, S., Lime, D., Roux, O. H.: The Expressive Power of Time Petri Nets, *Theoretical Computer Science*, **474**, 2013, 1–20.

[5] Berthomieu, B., Diaz, M.: Modeling and Verification of Time Dependent Systems Using Time Petri Nets, *IEEE trans. on Soft. Eng.*, **17**(3), 1991, 259–273.

[6] Berthomieu, B., Ribet, P.-O., Vernadat, F.: The tool TINA – Construction of Abstract State Spaces for Petri Nets and Time Petri Nets, *International Journal of Production Research*, **42**(4), July 2004.

[7] Boucheneb, H., Gardey, G., Roux, O. H.: TCTL model checking of Time Petri Nets, *Journal of Logic and Computation*, **19**(6), December 2009, 1509–1540.

[8] Gardey, G., Roux, O. F., Roux, O. H.: Safety Control Synthesis for Time Petri Nets., *8th International Workshop on Discrete Event Systems (WODES'06)*, IEEE Comp. Soc. Press, Ann Arbor, USA, July 2006.

[9] Grabiec, B., Traonouez, L.-M., Jard, C., Lime, D., Roux, O. H.: Diagnosis using Unfoldings of Parametric Time Petri Nets, *Proceedings of FORMATS'10*, 6246, Springer, Austria, September 2010.

[10] Henzinger, T. A., Kopke, P. W., Wong-Toi, H.: The expressive power of clocks, *Proceedings of the 22nd International Colloquium on Automata, Languages, and Programming (ICALP)*, 944, 1995.

[11] Henzinger, T. A., Manna, Z., Pnueli, A.: Temporal Proof Methodologies for Timed Transitions Systems, *Information and Computation*, **112**(2), 1994, 273–337.

[12] Jard, C., Lime, D., Roux, O. H., Traonouez, L.-M.: Symbolic Unfolding of Parametric Stopwatch Petri Nets, *Formal Methods in System Design*, 2013, To appear.

[13] Karp, R. M., Miller, R. E.: Parallel program schemata, *Journal of Computer and System Sciences*, **3**(2), 1969, 147 – 195, ISSN 0022-0000.

[14] Kesten, Y., Manna, Z., Pnueli, A.: Verifying Clocked Transition Systems, *Hybrid Systems*, 1066, 1996.

[15] Larsen, K. G., Pettersson, P., Yi, W.: Model-Checking for Real-Time Systems, *Proceedings of the 10th International Conference on Fundamentals of Computation Theory*, LNCS 965, Dresden, Germany, 1995.

[16] Larsen, K. G., Pettersson, P., Yi, W.: UPPAAL in a Nutshell, *International Journal on Software Tools for Technology Transfer*, **1**(1–2), Oct 1997, 134–152, Http://www.uppaal.com/.

[17] Lime, D., Roux, O. H., Jard, C.: *Clock Transition Systems*, Technical report, Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN), 2012, Available on HAL as hal-00725792.

[18] Lime, D., Roux, O. H., Seidner, C., Traonouez, L.-M.: Romeo: A Parametric Model-Checker for Petri Nets with Stopwatches, *15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2009)*, 5505, Springer, York, United Kingdom, March 2009.

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

[20] Srba, J.: Comparing the Expressiveness of Timed Automata and Timed Extensions of Petri Nets, *Proceedings of FORMATS'08*, 5215, Springer, 2008.