Clock Transition Systems

D. Lime[1] and O.H. Roux[1] and C. Jard[2⋆]

[1] LUNAM Université, École Centrale de Nantes, IRCCyN UMR CNRS 6597, Nantes, France
[2] ENS Cachan & INRIA, IRISA, Rennes, France
Université européenne de Bretagne

Abstract. The objective of the paper is to introduce a new model capable of modeling both Time Petri Nets (TPNs) and Networks of Timed Automata (NTA). We called 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 may be 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;

1 Introduction

Mastering the development of correct distributed real-time systems remains a priority in light of clear scientific issues they represent. One necessary lane in our opinion is the use of mathematically based models.

Low Level timed models. [10] 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. [13] defined *timed transition systems* (TTS) 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.

⋆ This work was partially funded by the ANR national research program **ImpRo** (ANR-2010-BLAN-0317).

To avoid delay actions, [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, [9] extend labeled transition systems with clocks and consider both discrete or dense time domain. Similarly, [12] propose a computational model for real-time systems called Clocked Transition Systems. This model represent time by a set of timers (clocks) which increase whenever time progress, but can be set to arbitrary values by system (program) transitions. A Clocked Transition System is also equiped of discrete variables of any type. Assertions associated with transitions allow the updates of variables and assertions over system variables specify a global restriction of the time progress.

TPNs and TA. For the class of critical systems that we aim, in which the specification of permissible behavior 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) [16, 4] and timed automata (TA for short) or networks of timed automata (NTA for short) [1], and their different extensions. These models respectively time extend Petri nets and finite automata. The paper [17] provides an overview of the theoretical known results about the relationships among these models.

Each class of model has its advantages and disadvantages. TPNs are particularly well suited for having a compact representation of concurrent behaviors with causal dependencies induced by complex synchronization between activities. The time constraints are described on transitions by intervals of firing. The mixture of concurrency and global time induces non local constraints, which are sometimes difficult to control from the designer point of view.

Timed automata better clarify how time should change. The designer's model introduces a set of temporal variables (clocks) used to form expressions guarding transitions. Transitions may reset clocks. The urgency is expressed by defining invariants on states, forcing the progress if possible. Somehow, this model is less abstract than TPNs, but is sometimes easier to build. The introduction of concurrency is achieved by connecting synchronously a set of components. The interest is the modular construction of models. The disadvantage is that it induces a kind of premature architectural decision. Another quirk of the explicit management of clocks is the ability to block the time if you are not careful.

Intermediate models for TPNs and NTA. The two main tools for TPNs and NTA are respectively TINA [5] and UPPAAL [14]. These tools use both intermediate model allowing to manipulate variables and computable functions extending the modeling concision, but in a rather restrictive way. In particular, TINA's intermediate model manipulates only interval representation for timing. It enables state class based analysis but is less general than explicit clocks used in NTA. UPPAAL manipulates bounded variables which can not represent general TPNs (which are Turing powerful). Moreover the UPPAAL synchronization mechanism, featuring only point-to-point and broadcast synchronizations, is less rich than that of TPNs.

Our contribution. A whole set of theories, methods and tools of analysis has been developed separately for TPNs and NTA. Yet we know that these models are very close, but nevertheless have subtle differences that have prevented until now to actually factorize research and development of associated technologies. The objective of the paper is to introduce an intermediate model capable of modeling both TPNs and NTA. This intermediate model is inspired from *Clocked Transition Systems* [12] but with only integer variables and with high level functions, its semantics is a *Timed Transition Systems* [13] and we called it *Clock Transition System* (CTS for short). Clock Transition System is designed to incorporate 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 may be introduced by a separate description of invariants. These are associated with a marking of the Clock Transition System, which plays the same role as the state in a timed automaton. Armed with these merits, the Clock Transition System models seems a good candidate to serve as an intermediate model to factor out the upcoming theoretical and practical 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 3 and 4 how TPNs and NTA can be easily represented by a Clock Transition System. Finally, Section 5 discusses the model and the techniques for its analysis. Proofs for the theorems can be found in [15].

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 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 $Reach(q_0) = \{q \in Q | q_0 \rightarrow^* q\}$, the set of reachable states in S.

Definition 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 3 (Equivalence up to isomorphism). *Let two models $\mathcal{A}$ and $\mathcal{A}'$ 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 $\mathcal{A} \cong \mathcal{A}'$, iff $\mathcal{S}_\mathcal{A} \cong \mathcal{S}_{\mathcal{A}'}$.*

2.2 Clock Transition Systems

Definition 4 (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 there exists $t \in T$ such that:

- $\mathsf{Pre}(t)(m)$ is true;
- $\lambda(t) = a$;
- $m' = \mathsf{Post}(t)(m)$;
- $v \models \mathsf{Guard}(t)$;
- $v' = v\big[\{x|(f,x) \in \mathsf{Resets}(t) \text{ and } f(m)\} \leftarrow 0\big]$;
- $\forall (f, J) \in \mathsf{Inv},\ f(m')$ implies $v' \models J$.

– $(m, v) \xrightarrow{d \in \mathbb{R}_{\geq 0}} (m, v + d)$ iff $\forall (f, J) \in \mathsf{Inv},\ f(m) \Rightarrow \forall 0 < d' \leq d, v + d' \models J$.

$\mathcal{T}$ 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.

State space and main properties. Clock Transition System allows only explicit clocks and integer variables. We can then easily extend the classical zone abstraction used in the tool Uppaal [14]. 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 1. *k-boundedness is decidable for CTS.*

Theorem 2. *Reachability is decidable for bounded CTS.*

Example. To actually illustrate how we can design a CTS, consider the following small example in Table 1.

$V = \{V_1, V_2\}$, $T = \{t_1, t_2, t_3, t_4\}$, $X = \{x_1, x_2\}$	$\mathsf{Pre}(t_1) = (V_1 > 0), \mathsf{Post}(t_1) = (V_2 := V_1 + V_2)$
$\mathsf{Pre}(t_2) = (V_2 > 0), \mathsf{Post}(t_2) = (V_2 := V_2 - 1)$	$\mathsf{Guard}(t_1) = (x_1 = 2), \mathsf{Guard}(t_2) = (x_1 = 1)$
$\mathsf{Pre}(t_3) = (V_2 < \alpha), \mathsf{Post}(t_3) = (V_1 := V_1 + 1)$	$\mathsf{Guard}(t_3) = \mathsf{Guard}(t_4) = (1 \leq x_2 \leq 3)$
$\mathsf{Pre}(t_4) = (V_1 > 1) \wedge (V_2 > \beta)$	$\mathsf{Resets}(t_1) = \mathsf{Resets}(t_2) = \{(\mathsf{true}, x_1)\}$
$\mathsf{Post}(t_4) = (V_1 := V_1 \mathsf{div}\ 2),\ m_0 = (1, 0)$	$\mathsf{Resets}(t_3) = \mathsf{Resets}(t_4) = \{(\mathsf{true}, x_2)\}$
$\mathsf{Inv} = \{((V_1 > 0), x_1 \leq 2), ((V_2 > 0), x_1 \leq 1), ((V_2 < \alpha) \vee ((V_1 > 1) \wedge (V_2 > \beta)), x_2 \leq 3)\}$	

Table 1. A small CTS (control flow).

It represents a system of flow control between a producer and a consumer. The producer produces a set of objects every two time units. The number of items produced in each burst is determined by a counter V_1 which can be controlled by the receiver. The receiver stores the objects in a container V_2 which is emptied at a rate of one object every time unit. It also knows two constants α and β ($\alpha < \beta$) defining response thresholds. Below a volume of stock α, the receiver causes the increment of V_1 to gradually accelerate the activity of the sender. Above the threshold β, V_1 is divided by two to reduce sharply the activity of the sender (risk of overflow). The acceleration and deceleration commands require between one to three units of time to be taken into account. Fig. 1 shows a possible timed execution of the system.

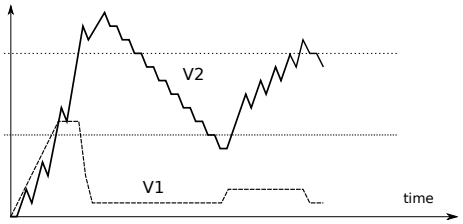

Fig. 1. A possible execution of the CTS example of Table 1.

3 Time Petri Nets and Clock Transition Systems

Definition 5 (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*;*
- $\mathsf{Pre} : P \times T \to \mathbb{N}$ *is the* backward incidence *function;*
- $\mathsf{Post} : P \times T \to \mathbb{N}$ *is the* forward incidence *function;*
- $m_0 : P \to \mathbb{N}$ *is the* initial marking *of the net;*
- A *is finite non-empty* alphabet*;*
- $\lambda : T \to A$ *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 $\mathcal{S}_\mathcal{N} = (\mathbb{N}^{|P|}, m_0, A, \to)$ 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 [16] 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 6 (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;*
- $I_s : T \to \mathcal{I}$ *assigns a* static time interval *to each transition.*

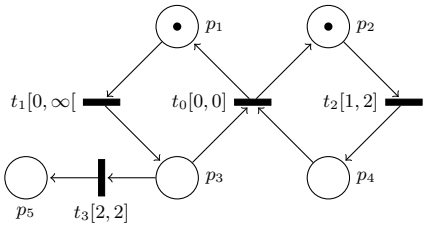

Fig. 2. A Time Petri Net.

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 there exists $t \in \mathsf{en}(m)$ such that:
  - $\lambda(t) = a$;
  - $\forall p \in P, m'(p) = m(p) - \mathsf{Pre}(p,t) + \mathsf{Post}(p,t)$;
  - $v(t) \in I_s(t)$;
  - $v' = v[\mathsf{new}(m,t) \leftarrow 0]$.
- $(m,v) \xrightarrow{d \in \mathbb{R}_{\geq 0}} (m, v+d)$ iff $\forall t' \in \mathsf{en}(m), \forall 0 < d' \leq d, (v+d')(t') \in I_s^{\downarrow}(t')$.

Boundedness of (time) Petri nets is defined exactly as for CTSs.

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

Theorem 3. *Every TPN $\mathcal{N}$ can be translated into a CTS $\mathcal{T}(\mathcal{N})$ s.t. $\mathcal{N} \cong \mathcal{T}(\mathcal{N})$.*

To illustrate the encoding, consider the TPN in Fig. 2. Its equivalent in CTS is given in Table 2. We now define a syntactic subclass of CTSs that is equivalent to TPNs:

Definition 7. *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.:*
  - *$k'(p,t) \geq -k(p,t)$;*
  - *$\mathsf{Pre}(t) = \bigwedge_{p \in P} p \geq k(p,t)$;*
  - *$\mathsf{Post}(t)$ is a list of assignments $\forall p, p := p + k'(p,t)$;*

$V = \{p_1, p_2, p_3, p_4, p_5\}$, $T = \{t_0, t_1, t_2, t_3\}$ $m_0 = (1, 1, 0, 0, 0)$, $X = T$

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

$\mathsf{Post}(t_0) = (p_1 := p_1 + 1, p_2 := p_2 + 1, p_3 := p_3 - 1, p_4 := p_4 - 1)$

$\mathsf{Inv} = \{(\mathsf{Pre}(t_0), (t_0 = 0)), (\mathsf{Pre}(t_1), \mathsf{true}), (\mathsf{Pre}(t_2), (t_2 \leq 2)), (\mathsf{Pre}(t_3), (t_3 \leq 2))\}$

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

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

Corollary 1. *The class CTS-TPN is equivalent to the class of TPNs up to isomorphism of TTS.*

4 Networks of Timed Automata and Clock Transition Systems

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

Definition 8 (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 9 (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* $S_{\mathcal{A}} = (Q, q_0, A, \rightarrow)$ *with* $Q = L \times (\mathbb{R}_{\leq 0})^X$, $q_0 = (l_0, \mathbf{0})$ *is the initial state and* $\rightarrow$ *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:*
  - $\lambda(e) = a$*;*
  - $v \models \mathsf{Guard}(e)$*;*
  - $v' = v[\mathsf{Resets}(e) \leftarrow 0]$*;*
  - $v' \models \mathsf{Inv}(l')$

– $(l, v) \xrightarrow{d \in \mathbb{R}_{\geq 0}} (l, v + d)$ *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 10 (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 11 (Semantics of NTA). *Let* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *be* n *timed automata with* $\mathcal{A}_i = \langle L_i, \ell_{0_i}, E_i, A, \lambda_i, X, \mathsf{Guard}_i, \mathsf{Resets}_i, \mathsf{Inv}_i\rangle$*,* $S_{\mathcal{A}_1}, \ldots, S_{\mathcal{A}_n}$ *their semantics with* $S_{\mathcal{A}_i} = (Q_i, q_{0_i}, A, \rightarrow_i)$*. Let* f *be a (partial) synchronization function* $(A \cup \{\bullet\})^n \to A$*. 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* $v' = v[\mathsf{R} \leftarrow 0]$*,*
  - *every* $\mathcal{A}_i$ *agrees on synchronization* i.e. *there 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. $(\vec{l}[i], v) \xrightarrow{d}_i (\vec{l}[i], v + d)$

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

Theorem 5. *Every NTA* $\mathcal{A}$ *can be translated into a CTS* $\mathcal{T}(\mathcal{A})$ *s.t.* $\mathcal{A} \cong \mathcal{T}(\mathcal{A})$.

To illustrate the encoding, consider the NTA in Fig. 3. Its equivalent in CTS is given in Table 3.

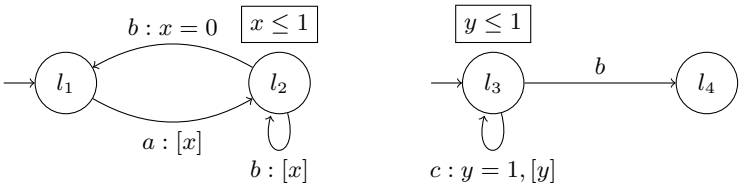

Fig. 3. A network of two timed automata with two clocks x and y. Locations are denoted in circles, and invariants boxed above them. Transitions are labeled by their action (a, b or c), the guard on clocks and resets (bracketed).

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

Table 3. CTS coding the NTA of Fig. 3.

Theorem 6. *Every bounded CTS* $\mathcal{T}$ *can be translated into a TA* $\mathcal{A}(\mathcal{T})$ *s.t.* $\mathcal{T} \cong \mathcal{A}(\mathcal{T})$.

Corollary 2. *The class of bounded CTSs is equivalent to the class of TA up to isomorphism of TTS.*

5 Discussion

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 easily be transformed in CTS, one can imagine a modelling workflow in which sequential components are modelled as TA, components featuring complex synchronization are modelled as TPNs, and complex dynamics are directly discretized in the form of CTS. This mixed modelling can ultimately be transformed in CTS 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 [11].
- For *bounded CTSs (with time)*, we can easily extend the region abstraction [1] or the zone abstraction used in the tool Uppaal [14]. 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. 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 [7]. It should also be possible to apply supervision techniques like in [8].

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

6 Conclusion and perspectives

We 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 [6]. In particular, we will equip this model with a concurrent semantics to build timed unfoldings.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. Rajeev Alur and Thomas A. Henzinger. Real-time system = discrete system + clock variables. In *Theories and Experiences for Real-Time System Development, AMAST Series in Computing*, volume 2, 1994.
3. Rajeev Alur and Thomas A. Henzinger. Real-time system = discrete system + clock variables. *Software Tools for Technology Transfer*, 1:86–109, 1997.
4. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE trans. on Soft. Eng.*, 17(3):259–273, 1991.
5. B. Berthomieu, P.-O. Ribet, and F. Vernadat. The tool TINA – construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, 42(4), July 2004.
6. Guillaume Gardey, Didier Lime, Morgan Magnin, and Olivier (H.) Roux. Roméo: A tool for analyzing time Petri nets. In *Proceedings of CAV'05*, volume 3576 of *LNCS*, Edinburgh, Scotland, UK, July 2005. Springer.
7. Guillaume Gardey, Olivier (F.) Roux, and Olivier (H.) Roux. Safety control synthesis for time Petri nets. In *8th International Workshop on Discrete Event Systems (WODES'06)*, pages 222–228, Ann Arbor, USA, July 2006. IEEE Computer Society Press.
8. Bartosz Grabiec, Louis-Marie Traonouez, Claude Jard, Didier Lime, and Olivier H. Roux. Diagnosis using unfoldings of parametric time Petri nets. In *Proceedings of FORMATS'10*, volume 6246 of *LNCS*, pages 137–151, Austria, September 2010. Springer.
9. Thomas A. Henzinger, Peter W. Kopke, and Howard Wong-Toi. The expressive power of clocks. In *Proceedings of the 22nd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 944 of *LNCS*, pages 417–428, 1995.
10. Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. Temporal proof methodologies for timed transitions systems. *Information and Computation*, 112(2):273–337, 1994.
11. Richard M. Karp and Raymond E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147 – 195, 1969.
12. Yonit Kesten, Zohar Manna, and Amir Pnueli. Verifying clocked transition systems. In *Hybrid Systems*, volume 1066 of *LNCS*, pages 13–40, 1996.
13. K. G. Larsen, P. Pettersson, and W. Yi. Model-checking for real-time systems. In Horst Reichel (Ed.), editor, *Proceedings of the 10th International Conference on Fundamentals of Computation Theory*, pages 62–88, Dresden, Germany, August 1995. LNCS 965.
14. Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, Oct 1997. http://www.uppaal.com/.
15. Didier Lime, Olivier H. Roux, and Claude Jard. Clock transition systems. Technical report, Institut de Recherche en Communications et Cyberntique de Nantes (IRCCyN), 2012. Available on HAL as hal-00725792.
16. P. M. Merlin. *A study of the recoverability of computing systems*. PhD thesis, Dep. of Information and Computer Science, University of California, Irvine, CA, 1974.
17. J. Srba. Comparing the expressiveness of timed automata and timed extensions of Petri nets. In *Proceedings of FORMATS'08*, volume 5215 of *LNCS*, pages 15–32. Springer, 2008.