Fundamenta Informaticae 46 (2001) 1–15
IOS Press

Timed Automata with non-Instantaneous Actions

Roberto Barbuti*
Dipartimento di Informatica, Università di Pisa
Corso Italia, 40 - 56125 Pisa - Italy
email: barbuti@di.unipi.it

Nicoletta De Francesco*
Dipartimento di Ingegneria dell'Informazione, Università di Pisa
Via Diotisalvi, 2 - 56126 Pisa - Italy
email: nicoletta.defrancesco@iet.unipi.it

Luca Tesei*
Dipartimento di Informatica, Università di Pisa
Corso Italia, 40 - 56125 Pisa - Italy
email: tesei@di.unipi.it

Abstract. In this paper we propose a model, *timed automata with non-instantaneous actions*, which allows representing in a suitable way real-time systems. Timed automata with non-instantaneous actions extend the timed automata model by dropping the assumption that actions are instantaneous: in our model an action can take some time to be completed. We investigate the expressiveness of the new model, comparing it with classical timed automata. In particular, we study the set of timed languages which can be accepted by timed automata with non-instantaneous actions. We prove that timed automata with non-instantaneous actions are more expressive than timed automata and less expressive than timed automata with ϵ edges. Moreover we define the parallel composition of timed automata with non-instantaneous actions. We point out how the specification by means of a parallel timed automaton with non-instantaneous actions is, in some cases, more convenient to represent reality.

Keywords: real-time systems, timed automata, timed languages.

* Address for correspondence: Dipartimento di Informatica, Università di Pisa, Corso Italia, 40 - 56125 Pisa - Italy

1. Introduction

Transition systems have been intensively used as a model for specifying and verifying "real-life systems". The representation of a system by means of a finite transition system allows reasoning easily about qualitative properties of it, such as "safety" and "liveness" properties. When real-time systems are considered, also quantitative timing properties must be taken into account, since the correctness of the whole system may depend on the magnitude of different delays. For these systems, the specification by means of classical transition systems is no longer satisfactory because time requirements cannot be described within this model. A natural way to solve this problem is to add, to the usual model, a suitable notion of time. Examples of this extension can be found in [11, 15].

Alur and Dill proposed the model of *timed automata* [6, 7]. Since their introduction, timed automata have been widely studied from different points of view [4, 5, 8, 9], in particular for their possible use in the verification of real-time systems [1, 2, 3, 10, 14, 16, 19]. Timed automata can be represented by finite graphs augmented with a finite set of (real-valued) clocks. An edge is labeled by a symbol which represents the action performed when the edge is taken. While actions are instantaneous, time can elapse in states. An edge can reset to zero the value of a set of clocks, and, at any instant, the value of a clock is equal to the time elapsed since the last reset on it. Any edge can be equipped with a constraint on the value of clocks. An edge may be taken only if the current value of clocks satisfies its constraint.

When considering real systems, in many cases the events are not instantaneous, but have a duration. In this paper we propose a model, *timed automata with non-instantaneous actions*, which extends the timed automata model by dropping the assumption that actions are instantaneous: in our model an action can take some time to be completed. The model allows specifying in a natural way systems in which the actions have a duration. To model non-instantaneous actions, every edge of the automaton is equipped with two constraints, a *initiation constraint* and a *completion constraint*. An edge can be taken when its initiation constraint is satisfied by the current value of clocks, and it can be completed only when its completion constraint is satisfied. Analogously, every edge is associated with a set of clocks which are reset to zero when it is taken (*initiation reset*) and a set of clocks which are reset to zero when the action is completed (*completion reset*). An event with a duration can be modeled, using timed automata, with two actions, corresponding to the initiation and the completion of the event. But in this way the resulting automaton has a different alphabet with respect to the original one, and the information that the event is unique is lost.

The notion of timed language accepted by a timed automaton is redefined in the context of timed automata with non-instantaneous actions. Different notions of language acceptance are defined in the paper, which differ in the choice of when an action occurrence must be considered, either on its initiation or on its completion. The different acceptance conditions correspond to different views of the system, on which different properties can be considered: for example, some property may refer to the initiation of an action and the completion of another one.

We investigate the expressiveness of the new model, comparing it with classical timed automata. A main result of the paper is that timed automata with non-instantaneous actions are more expressive than timed automata and less expressive than timed automata with ϵ edges.

Afterwards, we define the parallel composition of timed automata with non-instantaneous actions, and we show how a classical example of use of timed automata as a specification language [7] can be more suitably modeled by a parallel timed automaton with non-instantaneous actions.

2. Timed automata with non-instantaneous actions

Starting from the notion of timed automata we introduce, in this section, a new class of automata. In the following $\mathcal{R}^+$ is the set of non-negative real numbers. A *clock* takes values from $\mathcal{R}^+$. Given a set $\mathcal{X}$ of clocks, a *clock valuation* over $\mathcal{X}$ is a function assigning a non-negative real number to every clock. The set of valuations of $\mathcal{X}$, denoted $\mathcal{V}_\mathcal{X}$, is the set of total function from $\mathcal{X}$ to $\mathcal{R}^+$. Given $\nu \in \mathcal{V}_\mathcal{X}$ and $\delta \in \mathcal{R}^+$, with $\nu + \delta$ we denote the valuation that maps each clock $x \in \mathcal{X}$ into $\nu(x) + \delta$. Given a set $\mathcal{X}$ of clocks, a *reset* γ is a subset of $\mathcal{X}$. The set of all resets of $\mathcal{X}$ is denoted by $\Gamma_\mathcal{X}$. Given a valuation $\nu \in \mathcal{V}_\mathcal{X}$ and a reset γ, we define that the valuation $\nu\backslash\gamma(x) = 0$ if $x \in \gamma$ and $\nu\backslash\gamma(x) = \nu(x)$ if $x \notin \gamma$. Given a set $\mathcal{X}$ of clocks, the set $\Psi_\mathcal{X}$ of *clock constraints* over $\mathcal{X}$ are defined by the following grammar: $\psi ::= true \mid false \mid \psi \wedge \psi \mid \psi \vee \psi \mid \neg\psi \mid x\#t \mid x - y\#t$, where $x, y \in \mathcal{X}$, $t \in \mathcal{R}^+$, and $\#$ is a binary operator in $\{<, >, \leq, \geq, =\}$. Clock constraints are evaluated over clock valuations. The satisfaction by a valuation $\nu \in \mathcal{V}_\mathcal{X}$ of the clock constraint $\psi \in \Psi_\mathcal{X}$ is denoted by $\nu \models \psi$.

Definition 2.1. (Timed Automaton)
A timed automaton T is a tuple $(Q, \Sigma, \mathcal{E}, B, R, \mathcal{X})$, where: Q is a finite set of states, Σ is a finite alphabet of actions, $\mathcal{E}$ is a finite set of edges, $B \subseteq Q$ is the set of initial states, $R \subseteq Q$ is the set of repeated states, $\mathcal{X}$ is a finite set of clocks. Each edge $e \in \mathcal{E}$ is a tuple in $Q \times \Psi_\mathcal{X} \times \Gamma_\mathcal{X} \times \Sigma \times Q$. In the following, $\mathcal{T}$ denotes the class of all timed automata.

Definition 2.2. (Timed Automaton with non-Instantaneous Actions)
A timed automaton with non-instantaneous actions N is a tuple $(Q, \Sigma, \mathcal{E}, B, R, \mathcal{X})$, where: Q is a finite set of states, Σ is a finite alphabet of actions, $\mathcal{E}$ is a finite set of edges, $B \subseteq Q$ is the set of initial states, $R \subseteq Q$ is the set of repeated states, $\mathcal{X}$ is a finite set of clocks. Each edge $e \in \mathcal{E}$ is a tuple in $Q \times \Psi_\mathcal{X} \times \Gamma_\mathcal{X} \times \Sigma \times \Psi_\mathcal{X} \times \Gamma_\mathcal{X} \times Q$. If $e = (q, \psi^i, \gamma^i, \sigma, \psi^c, \gamma^c, q')$ is an edge, q is the *source*, q' is the *target*, ψ^i and ψ^c are the *initiation constraint* and the *completion constraint*, respectively, σ is the *label*, γ^i and γ^c are the *initiation reset* and the *completion reset*, respectively. The class of all timed automata with non-instantaneous actions will be denoted by $\mathcal{N}$.

The semantics of a timed automaton with non-instantaneous actions $N = (Q, \Sigma, \mathcal{E}, B, R, \mathcal{X})$ is an infinite transition system $\mathcal{S}(N)$ whose states are of two kinds:

- a pair (q, ν), where $q \in Q$ is a state of N and $\nu \in \mathcal{V}_\mathcal{X}$ is a valuation;
- a pair $(\overrightarrow{\sigma}_{(\psi,\gamma,q)}, \nu)$ where $\sigma \in \Sigma$, $\psi \in \Psi_\mathcal{X}$, $\gamma \in \Gamma_\mathcal{X}$, $q \in Q$ and $\nu \in \mathcal{V}_\mathcal{X}$. These states represent the execution of action σ after its initiation.

For an action σ, there are two particular time instants, the *initiation of the action* and the *completion of the action*, which may occur at different times. Following the notation of [17] we use the notation $\sigma\uparrow$ and $\sigma\downarrow$ for these two events. The transitions of $\mathcal{S}(N)$ are labeled either by a real number representing the elapsed time, or by an initiation or a completion of an action in Σ. The rules to derive the transitions of $\mathcal{S}(N)$ are the following:

$$N1.\ \frac{\delta \in \mathcal{R}^+}{(q,\nu) \xrightarrow{\delta} (q, \nu + \delta)} \qquad N2.\ \frac{(q, \psi^i, \gamma^i, \sigma, \psi^c, \gamma^c, q') \in \mathcal{E}, \nu \models \psi^i}{(q,\nu) \xrightarrow{\sigma\uparrow} (\overrightarrow{\sigma}_{(\psi^c,\gamma^c,q')}, \nu\backslash\gamma^i)}$$

$$N3.\ \frac{\delta \in \mathcal{R}^+}{(\overrightarrow{\sigma}_{(\psi,\gamma,q)}, \nu) \xrightarrow{\delta} (\overrightarrow{\sigma}_{(\psi,\gamma,q)}, \nu + \delta)} \qquad N4.\ \frac{\nu \models \psi}{(\overrightarrow{\sigma}_{(\psi,\gamma,q)}, \nu) \xrightarrow{\sigma\downarrow} (q, \nu \backslash \gamma)}$$

Rule $N1$. represents the case in which the automaton stays idle. If N moves along an outgoing edge $e = (q, \psi^i, \gamma^i, \sigma, \psi^c, \gamma^c, q')$, this corresponds to a transition of $\mathcal{S}(N)$ from the state (q, ν) to the state $(\overrightarrow{\sigma}_{(\psi^c,\gamma^c,q')}, \nu\backslash\gamma^i)$ in which σ is initiated (Rule $N2$.). Note that this state records the completion constraint and the completion reset of e, to be considered for the completion of σ. In this state, some time can elapse: in this case $\mathcal{S}(N)$ reaches a state where σ continues to be executed, but the valuation of clocks is modified according to the elapsed time (Rule $N3$.). When the execution of σ terminates, N reaches a new state (the target of e) with a new valuation of the clocks, given by the completion reset of e (Rule $N4$.).

A timed language is a set of *timed words*. A timed word over an alphabet Σ is an infinite sequence $\overline{(\sigma, t)} = (\sigma_0, t_0)(\sigma_1, t_1)\ldots$, where $\sigma_i \in \Sigma$ and $t_0 t_1 t_2 \ldots$ is an infinite sequence of non-decreasing time values associated to the symbols $\sigma_0\sigma_1\sigma_2 \ldots$. A timed word $\overline{(\sigma, t)}$ is accepted by a timed automaton if there exists a path in the automaton, labeled by $\sigma_0\sigma_1\sigma_2\ldots$, such that any edge labeled by σ_i is taken at time t_i, and satisfying some acceptance condition. The timed language accepted by a timed automaton T is denoted by $\mathcal{L}(T)$, and $\mathcal{L}(\mathcal{T})$ denotes the set of timed languages acceptable by automata in $\mathcal{T}$.

As for timed automata with alphabet Σ, the languages accepted by timed automata with non-instantaneous actions $N = (Q, \Sigma, \mathcal{E}, B, R, \mathcal{X})$ are timed languages over Σ. The difference is that here we have different acceptance notions: we can choose, for each action $\sigma \in \Sigma$, if we want to consider the time of its initiation or the time of its completion. If, for some action σ, we choose to consider its initiation, σ is considered as occurring when $\sigma\uparrow$ occurs, and $\sigma\downarrow$ is ignored, while, if we choose to consider its completion, σ is considered as occurring when $\sigma\downarrow$ occurs, and $\sigma\uparrow$ is ignored.

Given $A \subseteq \Sigma$, let us denote by $A\uparrow = \{\sigma\uparrow | \sigma \in A\}$ and $A\downarrow = \{\sigma\downarrow | \sigma \in A\}$ the set of initiations and completions of the actions in A, respectively.

Definition 2.3. (Run, Selected Sequence)

Given a timed automaton with non-instantaneous actions $N = (Q, \Sigma, \mathcal{E}, B, R, \mathcal{X})$, a *run* of the automaton is an infinite sequence of states and transitions of $\mathcal{S}(N)$ $s_0 \xrightarrow{l_0} s_1 \xrightarrow{l_1} \ldots$ such that $s_0 = (q, \nu)$, $q \in B$ and $\nu(x) = 0$ for every $x \in \mathcal{X}$, and a state $q \in R$ exists such that q occurs infinitely often in the pairs of the sequence $\{s_i\}$. Note that we use the Büchi acceptance condition for the runs [7].

- The *time sequence* $t_0 t_1 t_2 \ldots$ of the time elapsed from state s_0 to state s_j in r is defined as follows:
$$t_0 = 0 \text{ and } t_{i+1} = t_i + \begin{cases} 0 & \text{if } l_i \in \Sigma \\ l_i & \text{otherwise} \end{cases}$$

- The *event sequence* of the events occurring during r, including the elapsed times, is defined as follows: $(l_0, t_0)(l_1, t_1)\ldots$

- Given a partition $\Sigma = (I, C)$, $I \cup C = \Sigma$ and $I \cap C = \emptyset$, the (I, C) *selected action sequence* of r is the projection of the event sequence of r on the pairs $\{(l, t) | l \in I\uparrow \cup C\downarrow\}$.

Definition 2.4. (Selected Acceptance, Initiation Acceptance, Completion Acceptance)
Let $N = (Q, \Sigma, \mathcal{E}, B, R, \mathcal{X})$ be a timed automaton with non-instantaneous actions, and (I, C) a partition of Σ.

- A timed word w over Σ is (I, C) *selected accepted* by N if a run r of N exists such that, for all $\sigma \in I, \sigma' \in C$, $w = v[\sigma/\sigma\uparrow, \sigma'/\sigma'\downarrow]$, where v is the (I, C) selected action sequence of r and $v[\sigma/\sigma\uparrow]$ denotes the sequence v in which every symbol $\sigma\uparrow \in I\uparrow$ is substituted by σ (analogously for $v[\sigma/\sigma\downarrow]$). The set of timed words (I, C) selected accepted by N is called the (I, C) *selected accepted language* of N and is denoted by $\mathcal{L}^s_{(I,C)}(N)$.

- A timed word w over Σ is *initiation accepted* by N iff it is $(\Sigma, \emptyset)$ selected accepted by N. We shall use $\mathcal{L}^i(N)$ for $\mathcal{L}^s_{(\Sigma,\emptyset)}(N)$, to denote the *initiation accepted language* of N.

- A timed word w over Σ is *completion accepted* by N iff it is $(\emptyset, \Sigma)$ selected accepted by N. We shall use $\mathcal{L}^c(N)$ for $\mathcal{L}^s_{(\emptyset,\Sigma)}(N)$, to denote the *completion accepted language* of N.

The class of timed languages selected (for any (I, C)), initiation and completion accepted by automata in $\mathcal{N}$ is denoted by $\mathcal{L}^s(\mathcal{N})$, $\mathcal{L}^i(\mathcal{N})$, $\mathcal{L}^c(\mathcal{N})$, respectively.

3. Expressive power of the model

We now prove that the power of timed automata with non-instantaneous actions is greater than that of timed automata (without ϵ edges).

Example 3.1. Consider the automaton N_1 in Figure 1a. The initiation accepted language L_1 is the set of all timed words $\overline{(\sigma, t)} = (\sigma_0, t_0)(\sigma_1, t_1)(\sigma_2, t_2)\cdots$, such that each $t_i \in (i, i+1]$, for all natural numbers i, and either $\sigma_i = a$ and $t_i = i + 1$, or $\sigma_i = b$ and $t_i \in (i, i + 1)$.

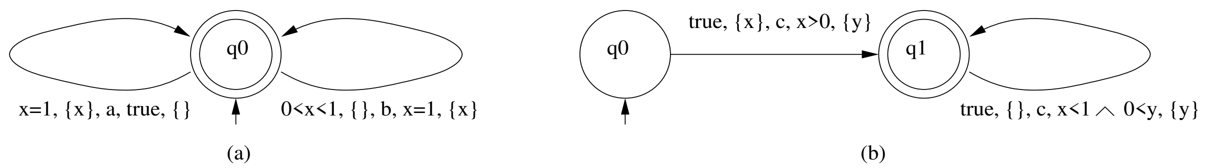

Figure 1. Automata N_1 and N_2

Example 3.2. Consider the automaton N_2 in Figure 1b. The completion accepted language L_2 is the set of all timed words $\overline{(\sigma, t)} = (c, t_0)(c, t_1)(c, t_2)\cdots$, such that $t_i < t_{i+1}$, and there exists $\gamma > 0$ such that $0 < t_i < t_0 + 1 - \gamma$, for all i.

Theorem 3.1. 1. $\mathcal{L}(\mathcal{T}) \subset \mathcal{L}^i(\mathcal{N})$, 2. $\mathcal{L}(\mathcal{T}) \subset \mathcal{L}^c(\mathcal{N})$

Proof:
To show that $\mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}^i(\mathcal{N})$ and $\mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}^c(\mathcal{N})$, we give a construction that, given a timed automaton T, builds a timed automaton with non-instantaneous actions N such that $\mathcal{L}(T) = \mathcal{L}^i(N) = \mathcal{L}^c(N)$. Given $T = (Q, \Sigma, \mathcal{E}, I, R, \mathcal{X})$, the corresponding timed automaton with non-instantaneous actions is the automaton $N = (Q, \Sigma, \mathcal{E}', I, R, \mathcal{X}')$, where $\mathcal{X}' = \mathcal{X} \cup \{x_e \mid e \in \mathcal{E}\}$ and for each $e = (q, \psi, \gamma, \sigma, q') \in \mathcal{E}$, $\mathcal{E}'$ contains $(q, \psi, \gamma \cup \{x_e\}, \sigma, x_e = 0, \emptyset, q')$. We assume that $\mathcal{X} \cap \{x_e \mid e \in \mathcal{E}\} = \emptyset$.

All actions of N are forced to be instantaneous. This is done by resetting the dedicated clock x_e at the initiation of the transition e and requiring that $x_e = 0$ at the termination of e. So N acts as T in all runs, and hence $\mathcal{L}(T) = \mathcal{L}^i(N) = \mathcal{L}^c(N)$ no matter the given selection for N.

To show that the class $\mathcal{L}(\mathcal{T})$ is a strict subset of $\mathcal{L}^i(\mathcal{N})$ ($\mathcal{L}^c(\mathcal{N})$), consider the timed automaton with non-instantaneous actions N_1 (N_2) in Figure 1a (1b). The language initiation accepted by this automaton, L_1 (L_2), is equal to the one accepted by the timed automaton with ϵ edges shown in Figure 2a (2b). This language cannot be accepted by any timed automaton without ϵ edges [12]. □

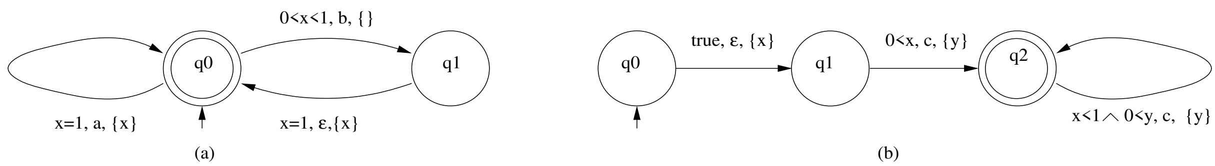

Figure 2. A timed automaton for L_1 and a timed automaton for L_2

Proposition 3.1. 1. $\mathcal{L}^i(\mathcal{N}) \not\subseteq \mathcal{L}^c(\mathcal{N})$, 2. $\mathcal{L}^c(\mathcal{N}) \not\subseteq \mathcal{L}^i(\mathcal{N})$.

Proof:
Sketch. The proof is based on the fact that the language L_1 (L_2), initiation (completion) accepted by the automaton N_1 (N_2), cannot be completion (initiation) accepted by any timed automaton with non-instantaneous actions. □

Proposition 3.2. $\mathcal{L}^i(\mathcal{N}) \cup \mathcal{L}^c(\mathcal{N}) \subset \mathcal{L}^s(\mathcal{N})$.

Proof:
Sketch. The proof is based on the fact that the language $L_1 \cup L_2$, $(\{a, b\}, \{c\})$ accepted by the automaton N_3 in Figure 3a, can be neither initiation nor completion accepted by any timed automaton with non-instantaneous actions. □

We now prove that timed automata with non-instantaneous actions are less expressive than timed automata in which ϵ (non-observables) transitions are used. The latter class will be denoted by $\mathcal{T}_\epsilon$ [12]. Let us start defining a transformation for a single automaton with non-instantaneous actions N.

Definition 3.1. (Simulator Automaton)
Let $N = (Q, \Sigma, \mathcal{E}, B, R, \mathcal{X})$ be a timed automaton with non-istantaneous actions. The simulator automaton of N is a timed automaton T^N defined as follows: $T^N = (Q', \Sigma\!\uparrow \cup \Sigma\!\downarrow, \mathcal{E}', B, R, \mathcal{X})$ where $Q' = Q \cup \{q_e \mid e \in \mathcal{E}\}$ and for all $e = (q, \psi^i, \gamma^i, \sigma, \psi^c, \gamma^c, q') \in \mathcal{E}$, $\mathcal{E}'$ contains $(q, \psi^i, \gamma^i, \sigma\!\uparrow, q_e)$ and $(q_e, \psi^c, \gamma^c, \sigma\!\downarrow, q')$.

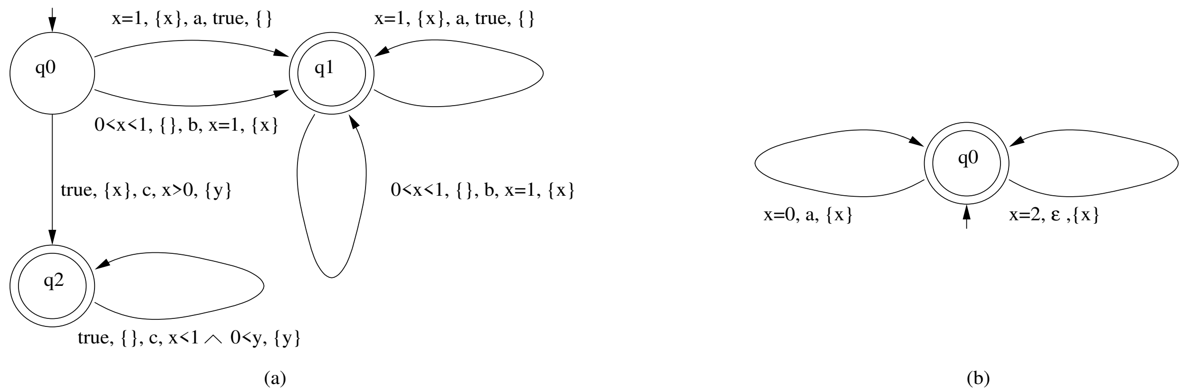

Figure 3. Automaton N_3 and an automaton for L_{even}

The simulator automaton T^N simulates all runs of N retaining both initiation symbol and termination symbol for each action. Its actions are instantaneous and represent initiating instants and terminating instants of non-instantaneous actions of N.

Definition 3.2. (Relabeled Simulator Automaton)
Let $N = (Q, \Sigma, \mathcal{E}, B, R, \mathcal{X})$ be a timed automaton with non-instantaneous actions and T^N its simulator automaton. Let (I, C) a partition of Σ.

Consider the renaming function $g_{(I,C)} : \Sigma\uparrow \cup \Sigma\downarrow \longrightarrow \Sigma \cup \{\epsilon\}$ such that

$$g_{(I,C)}(\sigma\uparrow) = \begin{cases} \sigma & \text{if } \sigma \in I \\ \epsilon & \text{if } \sigma \in C \end{cases} \qquad g_{(I,C)}(\sigma\downarrow) = \begin{cases} \epsilon & \text{if } \sigma \in I \\ \sigma & \text{if } \sigma \in C \end{cases}$$

The *relabeled simulator automaton* of N, denoted by $T^N_{(I,C)}$, is the simulator automaton T^N where for all $e \in \mathcal{E}'$ the transition label l of e is renamed by $g_{(I,C)}(l)$.

Note that the relabeled simulator automaton of N has an alphabet $\Sigma \cup \{\epsilon\}$ and belongs to $\mathcal{T}_\epsilon$.

Theorem 3.2. $\mathcal{L}^s(\mathcal{N}) \subset \mathcal{L}(\mathcal{T}_\epsilon)$.

Proof:
Sketch. First we show that $\mathcal{L}^s(\mathcal{N}) \subseteq \mathcal{L}(\mathcal{T}_\epsilon)$. Consider a timed automaton with non-instantaneous actions $N = (Q, \Sigma, \mathcal{E}, I, R, \mathcal{X})$, and a partition (I, C) of Σ. Let T^N be the simulator automaton of N and $T^N_{(I,C)}$ be the relabeled simulator automaton of N. It can be shown that $\mathcal{L}^s_{(I,C)}(N) = \mathcal{L}(T^N_{(I,C)})$ obtaining that any timed automaton with non-instantaneous actions can be simulated by an automaton in $\mathcal{T}_\epsilon$.

To show that the class $\mathcal{L}^s(\mathcal{N})$ is a strict subset of $\mathcal{L}(\mathcal{T}_\epsilon)$ (and then also $\mathcal{L}^i(\mathcal{N})$ and $\mathcal{L}^c(\mathcal{N})$ are so, by Proposition 3.2) consider the timed automaton in $\mathcal{T}_\epsilon$ shown in Figure 3b. The language accepted by this automaton, L_{even}, is the set of all timed words $\overline{(\sigma, t)} = (a, t_0)(a, t_1)(a, t_2)\cdots$, such that each t_i is an even natural number and $t_i \leq t_{i+1}$, for all i.

This language cannot be selected accepted by any automaton in $\mathcal{N}$. □

As a consequence, our model can be put at an intermediate level between timed automata and timed automata with ϵ edges.

Automata with periodic clock constraints, defined in [13], also have an expressive power between timed automata and timed automata with ϵ edges. They contain constraints which are based on regularly repeated time intervals. However, their power is not comparable with the power of our model. In fact the aim of automata with periodic clock constraints is that of model periodic behaviors, while we model actions that have a duration.

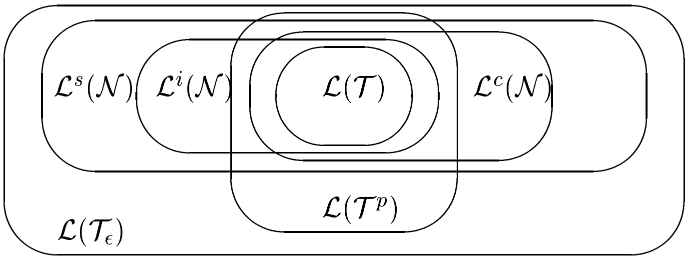

Figure 4. Inclusion among language classes

With automata with periodic clock constraints it is possible to model the timed automaton (with ϵ edges) of Figure 3b for L_{even}, which we are not able to model. On the other side, automata N_1, N_2 and N_3 cannot be modeled with periodic clock constraints.

The inclusion among language classes is given in Figure 4, where $\mathcal{L}(\mathcal{T}^p)$ is the class of languages accepted by timed automata with periodic clock constraints.

We can apply to timed automata with non-instantaneous actions the verification techniques, based on model checking, usually applied to timed automata [3, 10, 16, 19]. To perform this task, we first specify the time of occurrence of every action and then translate the timed automaton with non-instantaneous actions into a timed automaton with ϵ edges, as shown in the proof of Theorem 3.2.

4. Parallel Composition

In this section we define the semantic of a parallel composition of timed automata with non-instantaneous actions.

Let $N_1 = (Q_1, \Sigma_1, \mathcal{E}_1, B_1, R_1, \mathcal{X}_1)$ and $N_2 = (Q_2, \Sigma_2, \mathcal{E}_2, B_2, R_2, \mathcal{X}_2)$ be two timed automata with non-instantaneous actions such that $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$. The *parallel composition* of N_1 and N_2 is denoted by $(N_1 \parallel N_2)$.

The semantic is a transition system $\mathcal{S}(N_1 \parallel N_2)$ whose states are pairs $\langle \mathcal{C}_1, \mathcal{C}_2 \rangle$ in which $\mathcal{C}_1$ is a state of the transition system $\mathcal{S}(N_1)$ defining the semantic of N_1 and $\mathcal{C}_2$ is a state of the transition system $\mathcal{S}(N_2)$. An initial configuration is $\langle \mathcal{C}_1^0, \mathcal{C}_2^0 \rangle$ in which the components are initial states of $\mathcal{S}(N_1)$ and $\mathcal{S}(N_2)$ respectively.

To save notation let us define a function $\mathcal{F}_{\text{sync}} : \Sigma_1 \cup \Sigma_2 \to \{\{1,2\},\{1\},\{2\}\}$ such that $\mathcal{F}_{\text{sync}}(\sigma) = \{k \in \{1,2\} \mid \sigma \in \Sigma_k\}$. This function returns a non-singleton if applied to a synchronization action, otherwise it returns a singleton containing the index of the automaton the argument belongs to.
The system evolves starting from an initial configuration using the following rules.

$$PN1.\quad \frac{\delta \in \mathcal{R}^+}{\langle (s_1, \nu_1), (s_2, \nu_2) \rangle \xrightarrow{\delta} \langle (s_1, \nu_1 + \delta), (s_2, \nu_2 + \delta) \rangle}$$

$$PN2. \quad \frac{\mathcal{F}_{\text{sync}}(\sigma) = J, \quad \forall j \in J. \left(\mathcal{C}_j = (q_j, \nu_j),\ (q_j, \psi^i_j, \gamma^i_j, \sigma, \psi^c_j, \gamma^c_j, q'_j) \in \mathcal{E}_j,\ \nu_j \models \psi^i_j\right)}{\langle \mathcal{C}_1, \mathcal{C}_2 \rangle \xrightarrow{\sigma\uparrow} \langle \mathcal{C}'_1, \mathcal{C}'_2 \rangle}$$

where for all $j \in \{1,2\}$, $\mathcal{C}'_j = \begin{cases} \mathcal{C}_j & \text{if } j \notin J \\ (\overrightarrow{\sigma}_{(\psi^c_j, \gamma^c_j, q'_j)}, \nu_j \backslash \gamma^i_j) & \text{if } j \in J \end{cases}$

$$PN3. \quad \frac{\mathcal{F}_{\text{sync}}(\sigma) = J, \quad \forall j \in J. \left(\mathcal{C}_j = (\overrightarrow{\sigma}_{(\psi^c_j, \gamma^c_j, q_j)}, \nu_j),\ \nu_j \models \psi^c_j\right)}{\langle \mathcal{C}_1, \mathcal{C}_2 \rangle \xrightarrow{\sigma\downarrow} \langle \mathcal{C}'_1, \mathcal{C}'_2 \rangle}$$

where for all $j \in \{1,2\}$, $\mathcal{C}'_j = \begin{cases} \mathcal{C}_j & \text{if } j \notin J \\ (q_j, \nu_j \backslash \gamma^c_j) & \text{if } j \in J \end{cases}$

Rule $PN1$ represents the case in which both the two automata stay idle while the time passes. Rule $PN2$ describes the situation in which a set of automata initiate, at the same time, an action σ. This set is $\{1,2\}$ if σ is a synchronization action. If $\mathcal{F}_{\text{sync}}(\sigma)$ is a singleton, only one automaton proceeds according to its own behavior. Rule $PN3$ describes the case in which a set of automata in parallel, or a single, complete an action σ. Note that synchronization non-instantaneous actions must be initiated and terminated in the same instants by the two automata.

Let us remark that the notions of run and acceptance are analogous to the ones for timed automata with non-instantaneous transitions, the only difference being that at least a repeated state for each automaton must be infinitely repeated in a run according to the Büchi acceptance condition.

The definition above can be extended to the case of n, $n \geq 2$ automata simply adding components to the states of the transition system and extending the function $\mathcal{F}_{\text{sync}}$ to handle common symbols of n automata. That is the function applied to a symbol returns the set of all indexes of automata that must synchronize on the symbol.

5. An example of specification

We consider the example of an automatic controller which opens and closes a gate at a railroad crossing. This example was originally presented in [18], and it was used in [7] to show the specification and verification capabilities of timed automata. For simplicity we assume that one unit of time corresponds to a minute.

The automaton modeling the train is shown in Figure 5a. The automaton starts in state s_0. When approaching the railroad crossing, the train sends a (instantaneous) signal, *approach*, to the controller. Note that the signal *approach* belongs to both the alphabet of the train and of the controller, thus the two automata must synchronize on it. The train sends the signal *approach* at least 4 minutes before it enters the crossing. The train takes at least 1 minute to pass through the railroad crossing (action *crossing*). Note that, because this action is non-instantaneous, in [7] it was modeled by two actions, *in* and *out*, simulating the initiation and completion of it. After the signal *approach* is sent to the controller, the signal *exit* is sent within 8 minutes.

The gate is modeled by the automaton in Figure 5b. The synchronization with the controller is ensured by the instantaneous signals *lower* and *raise*. When the gate receives the signal *lower*, it starts to close the gate within 1 minute, and the closing action takes between 1 and 2 minutes. The gate responds to the signal *raise* by opening the gate with the same delays.

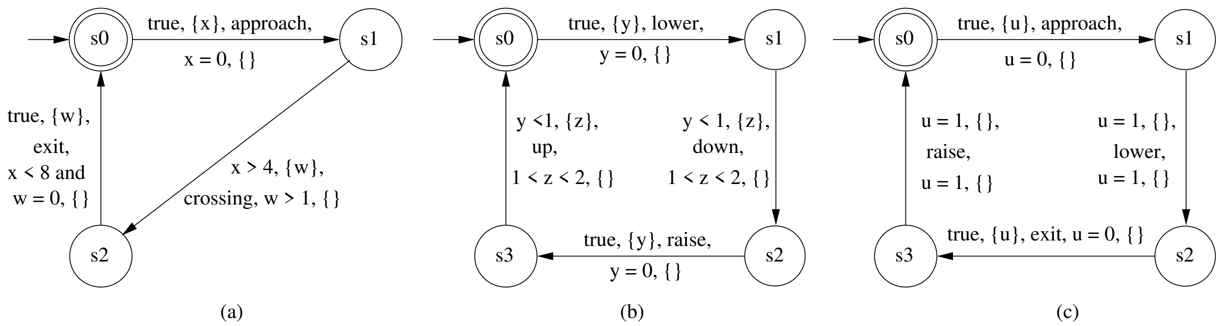

Figure 5. Train, Gate and Controller

Finally, Figure 5c shows the controller. When the controller receives the signal *approach* from the train, it sends, exactly after 1 minute, the signal for closing the gate. With the same delay it sends the signal *raise* after receiving the signal *exit*. Note that all the signals are forced to be instantaneous by asking for the value of the clocks to be the same at their initiation and completion.

The whole system is obtained by the parallel composition of the three automata.

We would like to remark that our specification allows us to state properties which better describe real constraints than timed automata. As an example, let us quote a property which was checked on the train/gate system in [7]. Such a property is a safety one: "Whenever the train is inside the gate, the gate should be closed". Using parallel timed automata with non-instantaneous actions we can state finer properties. For example we could be interested in verifying the following two ones:

1. "Whenever the train approaches the gate, the gate closes, and when the train initiates to cross the gate, the action of closing it should be completed".
2. "Every time, after the train crosses the gate, the gate must open, and the action of opening the gate should be initiated only when the train has completed the action of crossing it".

Both the properties could be expressed as conditions on the language accepted by the parallel composition $(Train \parallel Gate \parallel Controller)$ by specifying that, for property 1., the selected accepted words should refer to the initiation of *crossing* and to the completion of *down*, while for property 2. they should refer to the completion of *crossing* and to the initiation of *up*.

6. Simulation for Effectiveness

In this section we define a simple construction that, given a parallel composition $P = (N_1 \parallel N_2)$ of timed automata with non-instantaneous actions and a partition (I, C) of its alphabet Σ, builds a timed automaton with ϵ edges recognizing exactly $\mathcal{L}^s_{(I,C)}(P)$. This is done in order to make the parallel composition effective: properties can be checked on the timed automaton resulting from the construction and, for this, verification techniques and developed tools for timed automata can be used [3, 10, 16, 19].

Let $P = (N_1 \parallel N_2)$ is a parallel composition of timed automata with non-instantaneous actions.

1. Construct the simulator automata T^{N_1}, T^{N_2} of N_1, N_2 respectively (see Section 3).

2. Construct the parallel composition of timed automata $T^P = (T^{N_1} \parallel T^{N_2})$ [7].
3. Apply the relabeling function $g_{(I,C)}$ defined in Definition 3.2 to the labels of the transitions of T^P obtaining the automaton $T^P_{(I,C)}$ with ϵ edges.

Theorem 6.1. $\mathcal{L}^s_{(I,C)}(P) = \mathcal{L}(T^P_{(I,C)})$. $\square$

Clearly, if we have more than 2 automata the construction extends naturally to the general case. First, do step 1 and step 2 for the automata N_1 and N_2. This yields an automaton P'. Do step 1 for the successive automaton N_3 and construct the parallel composition of P' and T^{N_3}. Iterate until all automata have been considered and, finally, do step 3.

References

[1] Aceto, L., Bouyer, P., Burgueño, A. and Guldstrand Larsen, K. The Power of Reachability Testing for Timed Automata. Proc. Foundations Software Technology and Theoretical Computer Science, Springer LNCS 1530, 245–256, 1998.

[2] Aceto, L., Burgueño, A. and Guldstrand Larsen, K. Model Checking via Reachability Testing for Timed Automata. Proc. TACAS, Springer LNCS 1384, 263–280, 1998.

[3] Alur, R., Courcoubetis, C. and Dill, D.L. Model-Checking in Dense Real-time. *Information and Computation*, 104, 2–34, 1993.

[4] Alur, R., Courcoubetis, C., Halbwachs, N., Dill, D.L. and Wong-Toi, H. Minimization of Timed Transition Systems. Proc. CONCUR 1992, Springer LNCS 630, 340–354, 1992

[5] Alur, R., Courcoubetis, C. and Henzinger, T.A. The Observational Power of Clocks. Proc. CONCUR 1994, Springer LNCS 836, 162–177, 1994.

[6] Alur, R. and Dill, D.L. Automata for Modelin Real-time Systems. Proc. ICALP'90, Springer LNCS 443, 322–335, 1990.

[7] Alur, R. and Dill, D.L. A Theory of Timed Automata. *Theoretical Computer Science*, 126, 183–235, 1994.

[8] Alur, R., Fix, L. and Henzinger, T.A. Event-Clock Automata: A Determinizable Class of Timed Automata. *Theoretical Computer Science*, 211, 253-273 (1999).

[9] Alur, R. and Henzinger, T.A. Back to the Future: Towards a Theory of Timed Regular Languages. Proc. FOCS 1992, 177–186, 1992.

[10] Alur, R. and Henzinger, T.A. A Really Temporal Logic. *Journal of ACM*, 41, 181–204, 1994.

[11] Alur, R., Itai, A., Kurshan, R.P. and Yannakakis. M. Timing Verification by Successive Approximation. *Information and Computation*, 118, 142–157,1995.

[12] Bérard, B., Petit, A., Diekert, V. and Gastin P. Characterization of the Expressive Power of Silent Transitions in Timed Automata. *Fundamenta Informaticae*, 36, 145–182, 1998.

[13] Choffrut ,C. and Goldwurm, M. Timed Automata with periodic Clock Constraint. Int. Report 225-98, 1998.

[14] Henzinger, T.A. and Kopke, P.W. Verification Methods for the Divergent Runs of Clock Systems. Proc. Formal Techniques in Real-Time and Fault-Tolerant Systems, Springer LNCS 863, 351–372, 1994.

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

[16] Henzinger, T.A., Nicollin, X., Sifakis, J. and Yovine, S. Symbolic Model Checking for Real-Time Systems. *Information and Computation*, 111, 193–244, 1994.

[17] Jahanian, F. and Mok, A.K. A Graph-Theoretic Approach for Timing Analysis and its Implementation. *IEEE Transactions on Computers*, 36, 961-975, 1987.

[18] Leveson, N. and Stolzy, J. Analyzing Safety and fault Tolerance using Timed Petri Nets. Proc. Theory and Practice of Software Development, Springer LNCS 186, 339–355, 1985.

[19] Yovine, S. Model Checking Timed Automata. Lectures on Embedded Systems, Springer LNCS 1494, 114–152, 1996.