Timed Automata with Urgent Transitions

Roberto Barbuti and Luca Tesei [1,2]

Dipartimento di Informatica
Università di Pisa
Corso Italia, 40
56125 Pisa - Italy

Abstract

In this paper we propose an extension to the formalism of timed automata by allowing urgent transitions. A urgent transition is a transition which must be taken within a fixed time interval from its enabling time. We give a set of rules formally describing the behaviour of urgent transitions and we show that, from a language theoretic point of view, the addition of urgency does not improve the expressive power of timed automata. However, from a specification point of view, the use of urgent transitions is crucial, especially in modular specification of systems.

Keywords: real-time systems, timed automata, modular specification, parallel composition.

1 Introduction

Timed automata are widely recognized as a standard model for describing systems in which the time plays a fundamental role [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 [1,2,3,10,19,20,22].

Usually the expressiveness of timed automata is given in terms of accepted timed languages, but, because of their use as a specification formalism, also the ease to describe real-time systems must be taken into account. For this purpose many extensions to the basic model have been proposed (see for example [11,16,17,18,21]). All these extensions have been discussed with respect to the expressiveness of the original model.

In this paper we present a further extension: *timed automata with urgent transitions*. The notion of urgency in timed automata was already introduced

[1] Email: barbuti@di.unipi.it
[2] Email: tesei@di.unipi.it

in [13,14,15], where the urgency of transitions outgoing from a state is induced by the impossibility (due to the failure of a condition) to stay in the state while the time elapses. In this paper we consider a slightly different notion of urgency. Urgent transitions are transitions which must be performed within a given time interval starting from their enabling.

From the expressiveness point of view, both the approaches are suitable for specifying timed systems. The approach in [13,14,15] allows, in some cases, more general urgency conditions, while, in other cases, such as "as soon as possible" transitions, our approach allows more general time constraints.

In this preliminary version of the paper we impose that at most one urgent transition can exit from a state. We precisely define such a behavior by means of an operational semantics.

We show that, from the language theoretic point of view, timed automata and timed automata with urgent transitions are equivalent. This is proved by defining a transformation from a timed automaton with urgent transitions to a timed automaton which accepts the same language. However, the transformation is not a congruence with respect to parallel composition, thus it cannot be applied to a component which is supposed to be used in different environments. Thus, the use of urgent transitions is fundamental for the modular specification of systems.

2 Timed automata

We recall the definition of timed automata [7]. In the following, $\mathbb{R}$ is the set of real numbers and $\mathbb{R}^+$ the set of non-negative real numbers. $\mathbb{Q}$ is the set of rational numbers and $\mathbb{Q}^+$ is the set of positive rational numbers. A *clock* takes values from $\mathbb{R}^+$. 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 $\mathbb{R}^+$. Given $\nu \in \mathcal{V}_\mathcal{X}$ and $\delta \in \mathbb{R}^+$, with $\nu + \delta$ (resp. $\nu - \delta$) we denote the valuation that maps each clock $x \in \mathcal{X}$ into $\nu(x) + \delta$ (resp. $\nu(x) - \delta$). Note that if there exists $x \in \mathcal{X}$ such that $\nu(x) - \delta < 0$, $\nu(x) - \delta$ is not a clock evaluation.

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, with $\nu\backslash\gamma$ we denote the valuation

$$\nu\backslash\gamma(x) = \begin{cases} 0 & \text{if } x \in \gamma \\ \nu(x) & \text{if } x \notin \gamma \end{cases}$$

Given a set $\mathcal{X}$ of clocks, the set $\Psi_\mathcal{X}$ of *clock constraints* over $\mathcal{X}$ are defined by the following grammar:

$$\psi ::= \mathit{true} \mid \mathit{false} \mid \psi \wedge \psi \mid x \# t$$

where $x, y \in \mathcal{X}$, $t \in \mathbb{N}$ is a natural number, and $\#$ is a binary operator in $\{<, >, \leq, \geq, =\}$. Note that the negation operator is not needed because the negation of an atomic constraint $x\#t$ ($\#$ different from $=$) can be expressed as another constraint of the same kind. The negation of a constraint $x = t$ can be expressed by $x < t \vee x > t$. The disjunction can be simulated, as usual, by duplicating the edges in the automaton. Actually, in this version of the work, we do not allow more than one urgent action outgoing from a state, thus we have a restriction on the constraints of such actions.

Clock constraints are evaluated over clock valuations. The satisfaction by a valuation $\nu \in \mathcal{V}_{\mathcal{X}}$ of the clock constraint $\psi \in \Psi_{\mathcal{X}}$, denoted $\nu \models \psi$, is defined as follows:

$\nu \models true$ and $\nu \not\models false$

$\nu \models \psi_1 \wedge \psi_2$ iff $\nu \models \psi_1 \wedge \nu \models \psi_2$

$\nu \models x\#t$ iff $\nu(x)\#t$

Definition 2.1 [Timed automaton] A timed automaton T is a tuple $(Q, \Sigma, \mathcal{E}, I, R, \mathcal{X})$, where: Q is a finite set of states, Σ is a finite alphabet of actions, $\mathcal{E}$ is a finite set of edges, $I \subseteq Q$ 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 $Q \times \Psi_{\mathcal{X}} \times \Gamma_{\mathcal{X}} \times \Sigma \times Q$.

If $e = (q, \psi, \gamma, \sigma, q')$ is an edge, q is the *source*, q' is the *target*, ψ is the *constraint*, σ is the *label*, γ is the *reset*.

The semantics of a timed automaton T is given in terms of accepted timed language. The definition of such a language is based on an infinite transition system $\mathcal{S}(T) = (S, \rightarrow)$, where S is a set of states and $\rightarrow$ is the transition relation. The states S of $\mathcal{S}(T)$ are pairs (q, ν), where $q \in Q$ is a state of T, and ν is a valuation. An initial state of $\mathcal{S}(T)$ is a state (q, ν), where $q \in I$ is an initial state of T and ν is the valuation which assigns 0 to every clock in $\mathcal{X}$. At any state q, given a valuation ν, T can stay idle or it can perform an action labeling an outgoing edge e. If T stays idle, a transition is possible to a state of $\mathcal{S}(T)$ where the state of T is the same, but the valuation has been modified according to the elapsed time. If T moves along an outgoing edge $e = (q, \psi, \gamma, \sigma, q')$, this corresponds to a transition, labeled by σ, of $\mathcal{S}(T)$ from the state (q, ν) to the state $q', \nu \backslash \gamma$. This transition is possible only if the current clock valuation respects the constraint ψ of e. The rules to derive the transitions of $\mathcal{S}(T)$ are the following:

$$1. \ \frac{\delta \in \mathbb{R}^+}{(q, \nu) \xrightarrow{\delta} (q, \nu + \delta)} \qquad 2. \ \frac{(q, \psi, \gamma, \sigma, q') \in \mathcal{E}, \nu \models \psi}{(q, \nu) \xrightarrow{\sigma} (q', \nu \backslash \gamma)}$$

Rule 1. represents the case in which T stays idle in a state and the time passes, while Rule 2. corresponds to the occurrence of an action.

Definition 2.2 [run, action sequence] Given a timed automaton $T = (Q, \Sigma, \mathcal{E}, I, R, \mathcal{X})$, a *run* of the automaton is an infinite sequence of states and transitions of $\mathcal{S}(T)$

$s_0 \xrightarrow{l_0} s_1 \xrightarrow{l_1} \ldots$

where

- $s_0 = (q, \nu)$ where $q \in I$ and $\nu(x) = 0$ for every $x \in \mathcal{X}$
- a state $q \in R$ exists such that q occurs infinitely often in the pairs of the sequence $\{s_i\}$

Note that, given a run $s_0 \xrightarrow{l_0} s_1 \xrightarrow{l_1} \ldots$, for each i, $l_i \in (\Sigma \cup \mathbb{R}^+)$. Let r be a run.

- The *time sequence* $\overline{t_j}$ of the time elapsed from state s_0 to state s_j in r is defined as follows:

  $t_0 = 0$

  $$t_{i+1} = t_i + \begin{cases} 0 \text{ if } l_i \in \Sigma \\ \\ l_i \text{ otherwise} \end{cases}$$
- The *event sequence* of the events occurring during r, including the elapsed times, is defined as follows:

  $(l_0, t_0)(l_1, t_1) \ldots$
- The *action sequence* of r is the projection of the event sequence of r on the pairs $\{(l, t) | l \in \Sigma\}$

Definition 2.3 [timed word, timed language] Let Σ be an alphabet. A *timed word* over Σ is an infinite sequence of pairs $(\sigma_0, t_0)(\sigma_1, t_1) \ldots$ such that $\sigma_i \in \Sigma$, and $t_i \in \mathbb{R}^+$, $t_i \leq t_i + 1$, for all i.

A *timed language* over Σ is a subset of the set of all timed words over Σ.

Definition 2.4 [acceptance] Given a timed automaton $T = (Q, \Sigma, \mathcal{E}, I, R, \mathcal{X})$, a timed word w over Σ is *accepted* by T if a run r of T exists such that $w = v$, where v is the action sequence of r. The set of timed words accepted by T is called the *accepted language* of T.

Note that we use the Büchi acceptance condition for the runs.

A parallel composition is defined on timed automata. Here we recall the definition given in [7].

Definition 2.5 [Product] Let $T_1 = (Q_1, \Sigma_1, \mathcal{E}_1, I_1, \mathcal{X}_1)$ and $T_2 = (Q_2, \Sigma_2, \mathcal{E}_2, I_2, \mathcal{X}_2)$ be two timed transition tables with $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$. The product of T_1 and T_2, denoted by $T_1 \parallel T_2$, is given as follows:

$$T_1 \| T_2 = \langle Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \mathcal{E}, I_1 \times I_2, R', \mathcal{X}_1 \cup \mathcal{X}_2 \rangle$$

where $\mathcal{E}$ is defined by:

(i) **Synchronization actions**
$\forall\sigma \in \Sigma_1 \cap \Sigma_2, \forall(q_1, \psi_1, \gamma_1, \sigma, q_1') \in \mathcal{E}_1, \forall(q_2, \psi_2, \gamma_2, \sigma, q_2') \in \mathcal{E}_2$
$\mathcal{E}$ contains $((q_1, q_2), \psi_1 \wedge \psi_2, \gamma_1 \cup \gamma_2, \sigma, (q_1', q_2'))$

(ii) T_1 **actions**
$\forall\sigma \in \Sigma_1 \backslash \Sigma_2, \forall(q, \psi, \gamma, \sigma, q') \in \mathcal{E}_1, \forall s \in Q_2$
$\mathcal{E}$ contains $((q, s), \psi, \gamma, \sigma, (q', s))$

(iii) T_2 **actions**
$\forall\sigma \in \Sigma_2 \backslash \Sigma_1, \forall(q, \psi, \gamma, \sigma, q') \in \mathcal{E}_2, \forall s \in Q_1$
$\mathcal{E}$ contains $((s, q), \psi, \gamma, \sigma, (s, q'))$

This definition shows what we expect in parallel behaviors:

- Common symbols of the alphabets are synchronization actions. A synchronization action can be executed if and only if all the component automata involved can execute it. The action must be executed synchronously by all of them.
- Other symbols can be executed by each component independently according to its original specification.

Defining the set of repeated states, R', of a parallel composition according to the Büchi acceptance condition requires a slight different construction with a lot of details. We refer to [7] for this.

3 Timed automata with urgent transitions

In this section we extend the model of timed automata with a new feature which is useful in the specification of a real-time systems. The idea is to provide, in each state of the automaton, the possibility of labeling one of the outgoing edges as *urgent*. Intuitively the labeled edge must be taken with higher priority with respect to the others, provided that its constraint is satisfied by the current clock valuation.

To be more precise, we introduce a constant $\ell \in \mathbb{Q}^+$ which represents the length of a time interval in which an enabled urgent action must be executed. The time interval is $[t, t + \ell)$ where t is the instant in which the constraint associated to the urgent transition becomes satisfied by the current clock valuation.

Choosing this notion of urgency allow us to define precisely the behavior of urgent actions. The intuitive idea "urgent transitions must be taken as soon as possible" introduces some problems when applied in a model with a dense time domain. Consider a state of a timed automaton in which the current value of clock x is in $[0, 1]$ and there is an outgoing urgent transition with a clock constraint $x > 1$. Letting the time to elapse, at which time would the urgent transition be executed? It is not possible to answer precisely to this question since the time domain is dense. To avoid this problem we introduced the constant ℓ and the interval $[0, \ell)$ the action must be executed within. The choice of a right-open interval is also related to the denseness. Suppose, now, $\ell = 1$. If we chose an interval $[0, \ell]$ the upper bound of the interval within the urgent transition must be executed would not be precisely defined. This is because the lower bound is not precisely defined. The right-open interval allow us to express the upper bound as $x \leq 2$. If the transition has a constraint in the form of $x \geq 1$ then the upper bound is expressed by $x < 2$. The denseness also imposes the constant ℓ be greater than 0. The choice $\ell = 0$ could be interpreted as "immediately", but this leads to the problem discussed above. However, since $\ell \in \mathbb{Q}^+$, it can be chosen as small as we want. In other words, the "as soon as possible" limit behavior can be approximated with arbitrary precision.

If the constraint of the urgent transition is already satisfied when a state is entered then the interval starts at the instant in which the state is entered. We have chosen this approach for a uniform treatment of urgent transitions.

Note that the specification of ℓ could be *local* to each urgent transition. For the sake of simplicity we discuss the case in which ℓ is a global parameter. The case of local specification can be caught by a slight modification of the definition, the semantics and the transformation.

If a state has no urgent outgoing edges then the behavior is the usual one of timed automata. This also happens when a state is entered and the urgent transition is not enabled.

Moreover, when an urgent transition is enabled in a state, the unique way to continue the run is to execute it, both while the associated constraint is satisfied and within the interval specified above.

Definition 3.1 [Timed automaton with urgent transitions] Let $\ell \in \mathbb{Q}^+$ be a constant. A timed automaton with urgent transitions T_u^ℓ is a tuple $(Q, \Sigma, \mathcal{E}, \mathcal{U}, I, R, \mathcal{X})$, where: Q is a finite set of states, Σ is a finite alphabet of actions, $\mathcal{E}$ and $\mathcal{U}$ are finite sets of edges, the non-urgent and the urgent ones, $I \subseteq Q$ 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 $e \in \mathcal{E} \cup \mathcal{U}$ is a tuple in $Q \times \Psi_{\mathcal{X}} \times \Gamma_{\mathcal{X}} \times \Sigma \times Q$. $\mathcal{U}$ is the set of urgent transitions. To impose that at most one urgent transition can exit from a state we require that $(q_1, \psi_1, \gamma_1, \sigma_1, q_1'), (q_2, \psi_2, \gamma_2, \sigma_2, q_2') \in \mathcal{U}$ iff $q_1 \neq q_2$.

The class of all timed automata with urgent transitions will be denoted by $\mathcal{T}_u^\ell$.

In the following the superscript ℓ could be omitted and, when this happens, it should be considered implicitly defined.

For timed automata with urgent transitions T_u^ℓ we define an infinite transition system $\mathcal{S}(T_u^\ell) = (S_u, \rightarrow)$ as for timed automata. The states S_u are triples (q, ν, δ_q) such that $q \in Q$ is the current state of the automata T_u^ℓ, ν is the current clock valuation and $\delta_q \in \mathbb{R}^+ \cup \{0\}$ is a number recording the time elapsed since the state q has been entered. The rules to derive the transitions of $\mathcal{S}(T_u^\ell)$ are the following:

(Time)
$$\frac{\delta \in \mathbb{R}^+}{(q,\nu,\delta_q) \xrightarrow{\delta} (q,\nu+\delta,\delta_q+\delta)}$$

(Non-Urgent 1)
$$\frac{(q,\psi,\gamma,\sigma,q') \in \mathcal{E},\ \nu \models \psi,\ (\neg\exists(q,\psi_u,\gamma_u,\sigma_u,q'_u) \in \mathcal{U})}{(q,\nu,\delta_q) \xrightarrow{\sigma} (q',\nu\backslash\gamma,0)}$$

(Non-Urgent 2)
$$\frac{\begin{array}{c}(q,\psi,\gamma,\sigma,q') \in \mathcal{E},\ \nu \models \psi \\ (q,\psi_u,\gamma_u,\sigma_u,q'_u) \in \mathcal{U},\ (\neg\exists\delta.\ 0 \le \delta < \delta_q \wedge \nu-\delta \models \psi_u)\end{array}}{(q,\nu,\delta_q) \xrightarrow{\sigma} (q',\nu\backslash\gamma,0)}$$

(Urgent)
$$\frac{(q,\psi_u,\gamma_u,\sigma_u,q') \in \mathcal{U},\ \nu \models \psi_u, (\nu-\ell \not\models \psi_u \vee \delta_q < \ell)}{(q,\nu,\delta_q) \xrightarrow{\sigma_u} (q',\nu\backslash\gamma_u,0)}$$

Rule **(Time)** lets the time elapse in a state and updates both the clock valuation and the time elapsed in the state.

Rule **(Non-Urgent 1)** is used when T_u is in a state without outgoing urgent edges. In this case the behavior is the same as timed automata. Note that when a new state is entered the time elapsed is set to 0.

Rule **(Non-Urgent 2)** manages the case in which T_u is in a state with a (unique by definition) urgent transition. The "$\neg\exists$" condition in the rule requires that the urgent transition has never been enabled since the current state was entered. If this is false the rule is not applicable.

Rule **(Urgent)** executes an urgent action σ. The condition $(\nu - \ell \not\models \psi \vee \delta_q < \ell)$ ensures that an urgent transition is taken either before a time ℓ is elapsed after its enabling time, or the time elapsed in the state is less than ℓ. Without this guard an urgent transition could be fired after the expiry time expressed by ℓ. This would not be sound with respect to the notion of urgency.

Note that if $S(T_u)$ is in a state in which an urgent transition can be executed by the rule **(Urgent)** it cannot be postponed until its constraint becomes false. This because the "$\neg\exists$" condition of rule **(Non-Urgent 2)** will never be true and the run could not proceed.

The notion of run for a timed automata with urgent transitions is defined in the same way as for timed automata using the transition system $S(T_u)$. Similarly for the accepted timed language.

Example 3.2 Figure 1 shows an example of a timed automaton with a urgent transition, indicated by the letter u. In this example we consider $\ell = 1$. The automaton can execute the action b when the value of the clock x is in the interval $(0, 1]$. When the value of x becomes greater than 1, b cannot be performed any longer and the urgent action a must be executed. Moreover, because of the urgency, a must be performed while the value of x is in the interval $(1, 2]$.

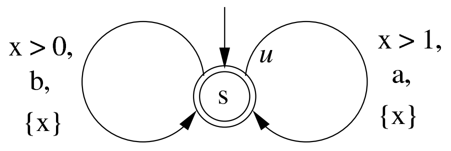

Fig. 1. An automaton with urgent transitions, T^1_u

4 The expressive power of timed automata with urgent transitions

In this section we show that, from a language theoretic point of view, the expressive power of timed automata with urgent transitions is equivalent to the one of timed automata. This is shown by providing a transformation which preserves the accepted language. Because timed automata are special cases of timed automata with urgent transitions, the transformation is only given starting from the latter ones.

In the next section we show that the transformation is not compositional, thus, for specification purposes, the use of urgent transitions is fundamental.

4.1 The region form of a timed automata

Let T^l_u be a timed automaton with urgent transitions. We give a transformation that builds a timed automata accepting the same timed language.

Note that if $\ell = \frac{a}{b}$ with a and b natural numbers, it is always possible to transform a $T^{\frac{a}{b}}_u$ automaton to an isomorphic one T^a_u by multiplying all the constants in the clock constraints by b. So we can assume without loss of generality that ℓ is a positive natural number. For the sake of simplicity we assume in this section that $\ell = 1$, but the transformation can be easily defined for any positive natural number.

Given a set of clocks $\mathcal{X}$, a clock region, as defined in [7], is an equivalence class of clock evaluations such that, given two clock evaluations ν and ν' belonging to it, for every clock constraint ψ, $\nu \models \psi$ iff $\nu' \models \psi$. Note that, given a timed automaton T and a set of clocks $\mathcal{X}$, the clock regions are finite. Let us denote such a set by $\mathtt{Reg}(T, \mathcal{X})$. We denote the equivalence class of a clock evaluation ν as $[\nu]$. A clock region $\alpha \in \mathtt{Reg}(T, \mathcal{X})$ can be uniquely identified by specifying, for every clock $x \in \mathcal{X}$, one clock constraint of the set $C_x = \{x = c | c = 0, 1, \ldots, c_x\} \cup \{c - 1 < x < c | c = 1, 2, \ldots c_x\} \cup \{x > c_x\}$ where c_x is the greatest constant to which x is compared in the constraints of T. Moreover, for every pair of clocks x and y such that we specified $c - 1 < x < c$ and $d - 1 < y < d$, for some c, d, an inequality of type $\mathtt{fract}(x)\#\mathtt{fract}(y)$

where $\# \in \{<, =, >\}$ must be specified. Here $\mathtt{fract}(x)$ is the fractional part of the value of clock x.

Given a clock region $\alpha \in \mathtt{Reg}(T, \mathcal{X})$ and $x \in \mathcal{X}$ we denote by $R_T(\alpha, x)$ the unique clock constraint in C_x specifying α.

In [7] it is shown how to construct, given a clock region $\alpha \in \mathtt{Reg}(T, \mathcal{X})$, the ordered set of clock regions that are *time successors* of α. We denote such set by $\mathtt{succ}(\alpha)$. The order $\leq_\alpha$ of the clock regions in the set $\mathtt{succ}(\alpha)$ is total and such that $\alpha \leq_\alpha \alpha'$ iff α' is a time successor of α.

Given a clock region $\alpha \in \mathtt{Reg}(T, \mathcal{X})$ and a reset $\gamma \subseteq \mathcal{X}$, we denote by $[\gamma \to 0]\alpha$ the clock region such that, for all $x \in \gamma$, the constraint in α for x is substituted by $x = 0$.

In the following we need a transformation of clock constraints ψ which gives a *logically equivalent* constraint $\mathtt{min}(\psi)$ such that it does not contain redundancies. Essentially the transformation drop from ψ the atomic constraints which are implied by others, yielding a minimal conjunction of constraints. In other words, for each clock $x \in \mathcal{X}$, there is only one constraint in $\mathtt{min}(\psi)$ of the forms $x = c$, $x \# c$, $c \# x \#' d$ or $c \# x$, where $\#, \#' \in \{<, \leq\}$. Let us denote by $\mathtt{select}(\mathtt{min}(\psi), x)$ such unique constraint.

The following definition describes a first transformation, in region form, of a timed automaton with urgent transitions. To this purpose a state of the transformed automaton records both the state of the original one and the equivalence class (clock region) of the values of clocks when the state is entered.

Definition 4.1 Let $T_u = (Q, \Sigma, \mathcal{E}, \mathcal{U}, I, R, \mathcal{X})$ be a timed automaton with urgent transitions.
The corresponding timed automaton *in region form*,
$T_u^r = (Q^r, \Sigma, \mathcal{E}^r, \mathcal{U}^r, I^r, R^r, \mathcal{X})$
is defined as follows:

- the states in Q^r and R^r are of the form $\langle q, \alpha \rangle$ where $q \in Q$ and α is a clock region,
- the states in I^r are of the form $\langle q, [\nu_0] \rangle$ where $q \in I$ and $\nu_0(x) = 0$ for all $x \in \mathcal{X}$
- $(\langle q, \alpha \rangle, \mathtt{min}(\psi) \wedge \bigwedge_{x \in \mathcal{X}} R_{T_u}(\alpha'', x), \gamma, \sigma, \langle q', [\gamma \to 0]\alpha'' \rangle) \in \mathcal{E}^r$ (resp. $\mathcal{U}^r$) iff $(q, \psi, \gamma, \sigma, q') \in \mathcal{E}$ (resp. $\mathcal{U}$), $\alpha \in \mathtt{Reg}(T_u, \mathcal{X})$, and $\alpha'' \in \mathtt{succ}(\alpha)$.

Note that the new states are built exactly as the ones of the region automaton as defined in [7].

This construction differs from the one for region automata because constraints and resets are maintained on the edges. These constraints are modified in order to force the corresponding edge to enter only one of the time successor clock regions (in the sense that for other regions the constraint is always false).

It is important to note that a timed automaton with urgent actions in region form may not meet the requirement that at most one urgent transition can exit form the same state. This is not a problem because this automaton is intended as an intermediate state in the transformation.

Example 4.2 In Figure 2 it is shown the automaton of Figure 1 in region form, denoted by T_u^{1r}. Note that the constraints explicitly shows the time successor clock region to which they refer. Note that all the edges with a *false* constraint have been removed and, in the states, there is only the $[x = 0]$ region because both the original edges reset x.

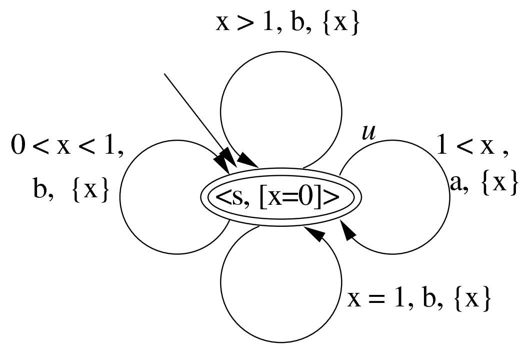

Fig. 2. Automaton T_u^{1r}

4.2 Making the urgent transitions ℓ consistent

The second step of the transformation will adapt the constraints of the urgent transitions of T_u^r making them consistent with the semantics we gave in Section 3. More precisely clock constraints are adapted according to the behavior expressed by the rule **(Urgent)**. In this step we consider only the urgent actions and neglect the other ones which remain unchanged. The third step will adapt these according to the semantics.

Let $\langle q, \alpha \rangle$ be a state of T_u^r such that in state q of T_u there was an outgoing urgent transition $e_q = (q, \psi, \gamma, \sigma, q')$. The latter became, in T_u^r, a set of transitions $E_\alpha^q = \{(\langle q, \alpha \rangle, \mathtt{min}(\psi) \wedge \bigwedge_{x \in \mathcal{X}} R_{T_u}(\alpha'', x), \gamma, \sigma, \langle q', [\gamma \to 0]\alpha'' \rangle) \in \mathcal{U}^r \mid \alpha'' \in \mathtt{succ}(\alpha)\}$. Note that α is the clock region when $\langle q, \alpha \rangle$ is entered by T_u^r and all the outgoing urgent transitions E_α^q are labeled by the same action. We adapt the clock constraints of these to handle the expiration time expressed by ℓ, i.e. to force the action to be executed within ℓ time units from the instant it becomes enabled.

There are three possible cases.

- First, $\mathtt{min}(\psi)$ is implied by α, that is, if $\nu \in \alpha$ then $\nu \models \psi$. In other words the urgent action is already enabled when the state is entered. Here we simply add to each transition in E_α^q a new constraint imposing that the time elapsed in the state be less than $\ell = 1$. To do this we add in T_u^r a new clock variable for each state. Whenever a state is entered the correspondent clock is reset, so it can be used in the constraints of outgoing edges as a measure of the time elapsed in the state.

- Second, $\mathtt{min}(\psi)$ is equivalent to false or it is consistent, but it will never be true letting the time to elapse from α. For the latter case consider, for instance, $\alpha = [x = 1 \wedge y = 2]$ and $\psi = x < 1 \wedge y > 2$. In this case we do nothing.
- Third, $\mathtt{min}(\psi)$ is not implied by α and will be implied by a time successor of α. In this case we have to ensure that, starting from the instant in which $\mathtt{min}(\psi)$ will become true, the transition will be taken within ℓ time units. This case requires some new notation and definitions. Using the total order $\leq_\alpha$ defined in the set $\mathtt{succ}(\alpha)$ we can determine, as the time elapses, the first clock region in which $\mathtt{min}(\psi)$ will be true. Let us denote this clock region by $\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi)) = min_{\alpha' \in \mathtt{succ}(\alpha)}(\alpha' \Rightarrow \mathtt{min}(\psi))$. Moreover we can establish the immediate predecessor, according to the total order, of a clock region α' in the set $\mathtt{succ}(\alpha)$. Let use denote this by $\mathtt{prec}(\alpha')$. Note that if α' is the minimum in $\mathtt{succ}(\alpha)$, then its predecessor is α.

Definition 4.3 [Set of Crucial Clocks] The set $\mathtt{cruc}(\alpha, \mathtt{min}(\psi)) = \mathcal{X} - \{x \in \mathcal{X} \mid R_{T_u}(\mathtt{prec}(\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi))), x) \Rightarrow \mathtt{select}(\mathtt{min}(\psi), x)\}$ contains the only clocks that determine the truth of the constraint $\mathtt{min}(\psi)$ in the region $\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi))$.

Let us explain the concept of "crucial" by an example.

Example 4.4 Let $\mathtt{min}(\psi)$ be $0 < x < 2 \wedge 1 < y < 3$. If α is $[x = 0 \wedge 0 < y < 1]$ then $\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi)) = [1 < y < 2 \wedge 0 < x < 1, \mathtt{fract}(y) > \mathtt{fract}(x)]$ and $\mathtt{prec}(\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi))) = [y = 1 \wedge 0 < x < 1]$. In $\mathtt{prec}(\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi)))$, the value of clock x implies the atomic constraint $\mathtt{select}(\mathtt{min}(\psi), x) = 0 < x < 2$, so x is not crucial for $\mathtt{min}(\psi)$ becomes true. Thus, we have $\mathtt{cruc}(\alpha, \mathtt{min}(\psi)) = \{y\}$.

If α is $[y = 1 \wedge x = 0]$ then $\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi)) = [1 < y < 2 \wedge 0 < x < 1, \mathtt{fract}(y) = \mathtt{fract}(x)]$ and $\mathtt{prec}(\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi)))$ is α itself. Here both x and y are crucial clocks.

Note that the set of crucial clocks always contains at least one element. If this were not true, the clock region $\mathtt{prec}(\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi)))$ would implies $\mathtt{min}(\psi)$. But, by definition, $\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi))$ is the minimum clock region that implies $\mathtt{min}(\psi)$ and $\mathtt{prec}(\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi)))$ is strictly less than it using the order defined in $\mathtt{succ}(\alpha) \cup \{\alpha\}$. A contradiction.

The constraint $\mathtt{select}(\mathtt{min}(\psi), x)$, given any crucial clock x, can be used to determine a constraint that force the urgent action to be executed within $\ell = 1$ time units from it became enabled. To do this we add to any transition in E^q_α the additional constraint $\mathtt{add}(\alpha, \mathtt{min}(\psi)))$ constructed as follows. Given a crucial clock x (we can choose any one):

- $\mathtt{add}(\alpha, \mathtt{min}(\psi)))$ is $(x < c+1)$ if $\mathtt{select}(\mathtt{min}(\psi), x)$ is either $(x = c)$ or $(c \leq x \# d)$ or $(x \geq c)$. Here $c < d$ and $\# \in \{\leq, <\}$.
- $\mathtt{add}(\alpha, \mathtt{min}(\psi)))$ is $(x \leq c+1)$ if
  $\mathtt{select}(\mathtt{min}(\psi), x)$ is either $(c < x \# d)$ or $(c < x)$. Here $c < d$ and $\# \in \{\leq, <\}$.

Adding $\mathtt{add}(\alpha, \mathtt{min}(\psi))$ to all transitions in E^q_α we ensures that the urgent action will be executed according to its semantics.

Definition 4.5 Let $T^r_u = (Q^r, \Sigma, \mathcal{E}^r, \mathcal{U}^r, I^r, R^r, \mathcal{X})$ be a timed automaton with urgent transitions in region form. The *ℓ-consistent* version of it, ℓT^r_u, is the timed automaton $(Q^r, \Sigma, \mathcal{E}^r, \mathcal{U}^r_\ell, I^r, R^r, \mathcal{X}^r)$ where $\mathcal{X}^r = \mathcal{X} \cup \{x_q | q \in Q^r\}$ and $\mathcal{U}^r$ is constructed as follows:

- $(\langle q, \alpha\rangle, \psi \wedge x_{\langle q,\alpha\rangle} < \ell, \gamma \cup \{x_{\langle q',\alpha'\rangle}\}, \sigma, \langle q', \alpha'\rangle) \in \mathcal{E}'$ iff
  $(\langle q, \alpha\rangle, \psi, \gamma, \sigma, \langle q', \alpha'\rangle) \in \mathcal{U}^r$ and $(\alpha \Rightarrow \psi)$,
- $(\langle q, \alpha\rangle, \mathtt{min}(\mathtt{min}(\psi) \wedge \bigwedge_{x\in\mathcal{X}} R_{T_u}(\alpha'', x) \wedge \mathtt{add}(\alpha, \mathtt{min}(\psi))), \gamma \cup \{x_{\langle q',\alpha'\rangle}\}, \sigma, \langle q', \alpha'\rangle)$ $\in \mathcal{E}'$ iff
  $(\langle q, \alpha\rangle, \mathtt{min}(\psi) \wedge \bigwedge_{x\in\mathcal{X}} R_{T_u}(\alpha'', x), \gamma, \sigma, \langle q', [\gamma \to 0]\alpha''\rangle) \in \mathcal{U}^r$, $\alpha \in \mathtt{Reg}(T_u, \mathcal{X})$, $\alpha'' \in \mathtt{succ}(\alpha)$ and $(\alpha \not\Rightarrow \psi)$

4.3 The quiet version of a timed automaton with urgent transitions

Now, in order to achieve the desired behavior, in each state of ℓT^r_u we have to turn off all the originally non-urgent outgoing transitions when at least one of the edges obtained by the originally urgent transition is enabled. This is the third transformation step.

Let $e = (\langle q, \alpha\rangle, \psi, \gamma, \sigma, \langle q', \alpha'\rangle)$ be a non-urgent outgoing transition from a state in ℓT^r_u. We map e in some transitions $e' = (\langle q, \alpha\rangle, \psi \wedge \theta, \gamma, \sigma, \langle q', \alpha'\rangle)$ of the new automaton where θ is the constraint that will become false when at least one of the outgoing urgent transitions of a state in ℓT^r_u becomes true.

Definition 4.6 Let ψ be a constraint without redundancies over a set of clocks $\mathcal{X}$. The *upper opening* $\mathcal{O}(\psi)$ of ψ is obtained by deleting from ψ all the constraints of the form $x \leq c$ and $x < c$, and by substituting all the constraints of the form $x = c$ by $x \geq c$, for all $x \in \mathcal{X}$. Clearly this definition requires constraints of the form $c \# x \# d$, $\# \in \{<, \leq\}$, to be considered as $c \# x \wedge x \# d$.

The upper opening of the disjunction of all urgent edges constraints (without redundancies), outgoing from a state in ℓT^r_u, describes a right-infinite time interval to the beginning of which a urgent transition must be taken. The negation of this disjunction must be added to all the constraints of non-urgent edges of T^r_u outgoing from the same state.

The negation of a complex formula can introduce disjunction of constraints. We denote by $\mathtt{DNF}^+$ an operation that, given a constraint which contains negations, push the negation operator inside, using the logical axioms for $\neg, \wedge, \vee$, until it is applied to atomic constraints. Then it transforms the negations of these constraints to the correspondent positive ones ($x = c$ will be translated into $x < c \vee c < x$). Finally, it transforms the formula in disjunctive normal form. It returns the set containing all the conjunctive components of the formula.

Definition 4.7 Let $\ell T_u^r = (Q^r, \Sigma, \mathcal{E}^r, \mathcal{U}_\ell^r, I^r, R^r, \mathcal{X}^r)$ be the *ℓ-consistent* version of a timed automaton with urgent transitions in region form T_u^r. The *quiet* version of it, T^r, is the timed automaton $(Q^r, \Sigma, \mathcal{E} = \mathcal{U}_\ell^r \cup \mathcal{E}', I^r, R^r, \mathcal{X}^r)$ where $\mathcal{E}'$ is constructed as follows:
$(\langle q, \alpha\rangle, \psi \wedge \phi, \gamma \cup \{x_{\langle q', \alpha'\rangle}\}, \sigma, \langle q', \alpha'\rangle) \in \mathcal{E}'$ iff
$(\langle q, \alpha\rangle, \psi, \gamma, \sigma, \langle q', \alpha'\rangle) \in \mathcal{E}^r$, and
$\phi \in \mathtt{DNF}^+(\neg(\bigvee_u \mathcal{O}(\mathtt{min}(\psi_u)))$ for all $(\langle q, \alpha\rangle, \psi_u, \gamma_u, \sigma_u, \langle q'', \alpha''\rangle) \in \mathcal{U}_\ell^r$.

Example 4.8 Figure 3 shows the automaton T^{1r} which is the quieted version of the automaton T_u^1 of Figure 1. Note that the constraint $x > 1$ on the edge for b has been modified to $1 < x \wedge x \leq 1$ by the last transformation. Thus, being always false has been removed. In figure, the clock $x_{s,[x=0]}$ is omitted because it is useless in this case.

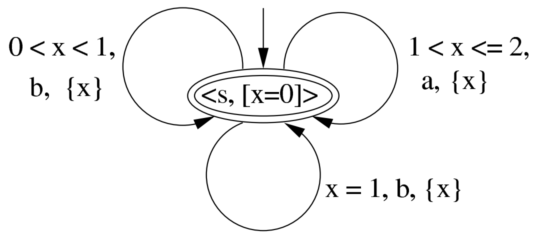

Fig. 3. Automaton T^{1r}

The transformation allows to state the following result.

Proposition 4.9 *Let T_u be a timed automaton with urgent transitions, T_u^r the corresponding timed automata in region form and T^r its quiet version. T_u and T^r accept the same timed language.*

5 Using timed automata with urgent transitions as a specification formalism

In the previous section we defined a new feature for the timed automata specification formalism. After that we showed how to compile a timed automaton with urgent transition into a standard timed automaton.

Indeed, a specification formalism needs a way to define systems as a composition of components. For timed automata this mechanism is the parallel composition of Definition 2.5. The parallel composition of timed automata with urgent transitions is defined in the same way, but the following remarks:

- the urgency of a transition of a component with a synchronization action σ extends to the transition obtained using the rule (i) of Definition 2.5 even if the transition of the partner was not urgent
- the urgency of a transition of a component extends to the transition obtained by the rules (ii) and (iii) of Definition 2.5
- if the interleaving of actions leads to a reachable state with more than one outgoing urgent transition, then the composition is not possible.

The latter restriction follows from the limitation we gave in Definition 3.1. We plan to extend our definition to manage multiple outgoing urgent transitions. Most of the future work will concern the definition of a precise behavior in that case.

It is easy to see that the transformation defined in the previous section is not a congruence with respect to parallel composition. In other words if we have two timed automata, T_u^1 and T_u^2, with urgent transition the automaton $T_u^1 \parallel T_u^2$, when defined, is not equivalent, in general, to $T_1^r \parallel T_2^r$ (the standard parallel composition of the quiet version of them).

This fact makes the transformation not feasible for modular specifications, i.e. it must be applied to the whole system (the parallel composition of all components).

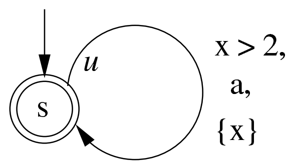

Fig. 4. Automaton $T_u'^1$

Consider the automaton $T_u'^1$ of Figure 4, the automaton T_u^1 of Figure 1 and the parallel composition $T_u^1 \parallel T_u'^1$. The action a, being a synchronization action, can be performed only when the value of the clock x is greater than 2. Thus b can be performed when the value of x is in $(0, 2]$. Using the quiet version of T_u^1 (Figure 3) in the parallel composition leads to a wrong behavior: $T^{1r} \parallel T_u'^1$ cannot perform the action a.

5.1 An example

In this last section we show a simple example. In Figure 5 is given the specification of a scheduler. The scheduler assigns resources to two processes, $P1$ and $P2$, alternatively. To each process the resources are assigned for 2 time units. If, during the elaboration of a process, an interruption occurs, it must be handled immediately. This is specified by considering the interruption handling a urgent action. It is important to note that, because of urgent transitions, the scheduler behaves in the correct way independently from the environment in which will be introduced. The interruption handling will preempt all the non-urgent transitions of the environment thus preserving the intended behavior of the scheduler.

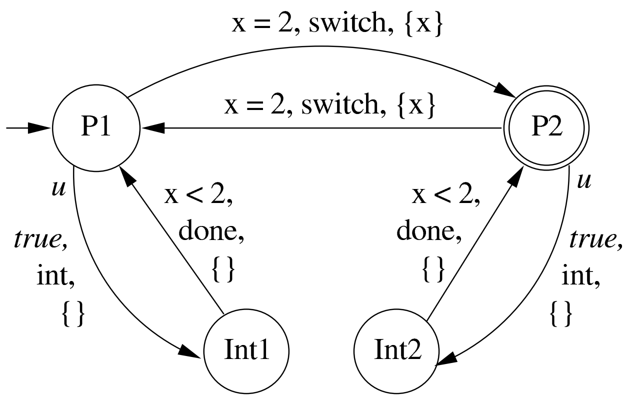

Fig. 5. A simple scheduler

6 Related works

The notion of urgency for timed formalisms has been studied in the past.

In [12] the urgency of actions has been investigated in the process algebra field with the concept of discrete time.

A closer approach to ours can be found in [13,14,15]. There the states of a timed automaton are associated with time progress conditions (TPC). TPC are state conditions which specify that the time can progress at a state by δ only if all the intermediate times δ', $0 \leq \delta' < \delta$, satisfy it.

TPC are computed from *deadlines*. Deadlines are clock constraints associated to transitions in addition to the usual constraints (which, in this setting, are called *guards*). The defined class of timed automata is called *Timed Automata with Deadlines* (TAD).

Given a state q, its TPC is intuitively computed as follows. Consider the set $I = \{i \mid t_i \text{ is a transition outgoing from } q\}$ of indexes of transitions from q. The TPC of q, c_q, is obtained as the negation of the disjunction of the deadlines, d_i, of all the transitions from q, $c_q = \neg \bigvee_{i \in I} d_i$. In a state of a run, (q, ν), the time can progress by δ, $(q, \nu) \xrightarrow{\delta} (q, \nu + \delta)$, if $\forall \delta' < \delta . \nu + \delta' \models c_q$.

Given a transition in a TAD, with guard ψ and deadline d, we can found in [14] the following remark.

"The relative position of d with respect to δ determines the urgency of the action. For a given δ, the corresponding d may take two extreme values: first, $d = \delta$, meaning that the action is eager and, second, $d = false$, meaning that

the action is lazy. A particularly interesting case is the one of a delayable action *where d is the falling edge of a right-closed guard δ (cannot be disabled without enforcing its execution).*

The condition $d \Rightarrow \delta$ guarantees that if time cannot progress at some state, then at least one action is enabled from this state. Restriction to right-open TPC guarantees that deadlines can be reached by continuous time trajectories and permits to avoid deadlock situations in the case of eager transitions. For instance, consider the case where $d = \delta = x > 2$, implying the TPC $x \leq 2$, which is not right-open. Then, if x is initially 2, time cannot progress by any delay δ, according to above definition. The guard ψ is not satisfied either, thus, the system is deadlocked."

This limitation is very intuitive: if the eager transition has a left-open guard, the time at which it can be fired is undefined. Using our concept of urgent transition we avoid this problem because the transition can be fired in the interval $[0, \ell)$. On the other hand to fire "as soon as possible" a transition with a left-closed guard, say $2 \leq x$, we have only to change it in $2 = x$.

Example 6.1 Let us use our notion of urgency to model a producer-consumer system that is considered in [15]. The partners are supposed to communicate with a zero-length buffer. Figure 6(a) and 6(b) show the producer and the consumer respectively.

The producer in state 1 is producing a new item. This process requires a time that is between l_p and u_p time units. When in state 3, the producer can communicate with a *handshake* in the time interval $[l'_p, u'_p]$ to send an item. Conversely the consumer in state 2 can communicate with a *handshake* in the time interval $[l'_c, u'_c]$ to receive an item. When in state 4 the consumer is consuming the item needing a time between l_c and u_c time units. Note that actions *produce* and *consume* represent the end of the correspondent processes. Also note that *handshake* is a urgent action for both the components. The parallel composition is shown in Figure 7. In [15] authors use both different notions of compositions of guards and deadlines to obtain different behaviors of the whole system. Here we study how these behaviors can be simulated in our model varying the constraint ψ for the *handshake* action in Figure 7 and the urgency parameter ℓ.

The first behavior to consider is the usual one coming from standard parallel composition. The constraint ψ is the conjunction of the components *handshake*-transition constraints. The parameter ℓ in this case is $\mathtt{max}((u'_p - l'_p), (u'_c - l'_c))$, i.e. the length of the largest interval expressed in the components behavior. The system can execute the *handshake* only if both clock x and y are in the intervals expressed in the specification of components. This behavior, as observed in [15], could be too restrictive since it may cause deadlock.

To relax this condition we can set the constraint ψ of Figure 7 to $x \geq l'_p \wedge y \geq l'_c$. Now the upper bounds are not considered and, starting from the instant in which the constraint becomes satisfied, the action has to be executed

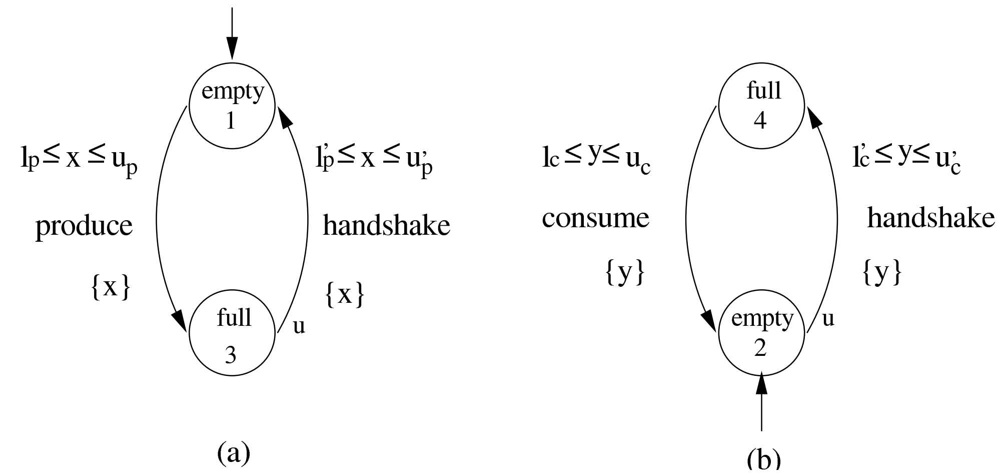

Fig. 6. A Producer-Consumer system: the components

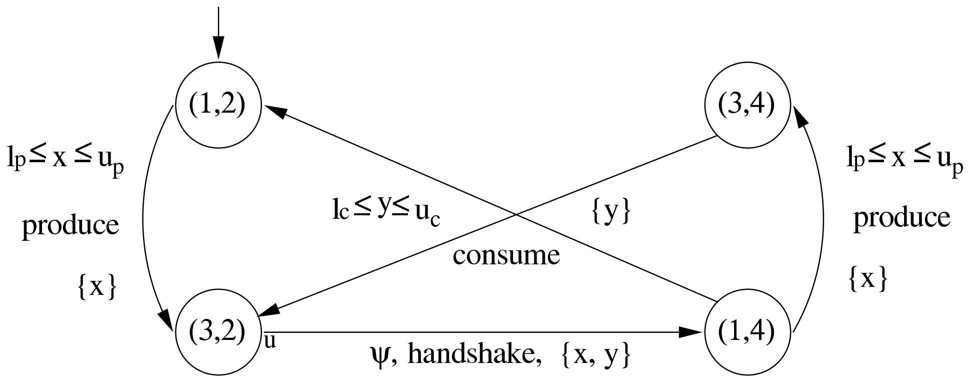

Fig. 7. A Producer-Consumer system: parallel composition

within ℓ time units. We set ℓ to $\mathtt{min}((u'_p - l'_p), (u'_c - l'_c))$. This ensures that at most one of the upper bounds can be violated. This behavior is similar to the one specified in [15] by a deadline $x = u'_p \wedge y \geq u'_c \;\vee\; y = u'_c \wedge x \geq u'_p$. There the *handshake* can be postponed as long as one of the intervals is not violated, but when the end of this interval is reached the action becomes urgent and must be executed. Clearly, since in the current approach we can specify only one outgoing action for each state, our model is a slight more restrictive. As a matter of fact, if the shortest interval is the one associated to clock y and the clock values are such that the first constraint satisfied is $y \geq l'_c$, then, when the constraint ψ becomes true (i.e. $x \geq l'_p$ becomes true), the action must be executed before the upper bound associated to y (actually it should wait until $x = u'_p$) because we definitely chose $\ell = (u'_c - l'_c)$. A similar remark can be done considering the choice of $\ell = (u'_p - l'_p)$.

There is another synchronization scheme for this example that authors in [15] call "best-effort", that is "either no upper bound is violated if possible, or the transition is executed as soon as possible". To express a similar scheme we use the constraint $\psi = x \geq l'_p \wedge y \geq l'_c$ as above and a constant ℓ as small as we want to precisely approximate the "as soon as possible" behavior. In this example this is precisely definite since the constraint $\psi = x \geq l'_p \wedge y \geq l'_c$ uses the operator $\geq$. Indeed, the action should be executed when $x = l'_p$ or $y = l'_c$ (depending on which one becomes true after). In this version of the work we always have to consider the approximation, even for precisely definite behaviors (i.e. those with operators "$\geq$" and "$=$"). However, note that by using the approximation we can manage also the constraints in which is used the operator $>$ in place of $\geq$.

7 Conclusion

In this paper we introduce a notion of urgency for timed automata. We compare it with other approaches to urgency, in particular the one of [13,14,15].

In this version of the paper we introduce the limitation that no more than one urgent transition can exit from a state. We plan to remove this limitation in future works.

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] Barbuti, R., De Francesco, N. and Tesei,L. Timed Automata with non-Instantaneous Actions To appear in *Fundamenta Informaticae.*

[12] Bolognesi, T. and Lucidi, F. Timed Process Algebras with Urgent Interactions and a Unique Powerful Binary Operator. REX Workshop 1991, Springer LNCS 600, 124-148, 1992.

[13] Bornot, S. and Sifakis, J. Relating Time Progress and Deadlines in Hybrid Systems. HART 1997, Springer LNCS 1201, 286–300, 1997.

[14] Bornot, S., Sifakis, J. and Tripakis, S. Modeling Urgency in Timed Systems. COMPOS 1997, Springer LNCS 1536, 103–129, 1998.

[15] Bornot, S. and Sifakis, J. Modeling Urgency in Timed Systems. To appear on *Information and Computation.*

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

[17] Demichelis,F. and Zielonka, W. Controlled Timed Automata Proc. CONCUR 98, Springer LNCS 1566, 455–469, 1998.

[18] Gupta, V., Henzinger, T.A. and Jagadeesan, R. Robust Timed Automata. Proc. HART 97, Springer LNCS 1201, 331-345, 1997.

[19] 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.

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

[21] Lanotte, R., Maggiolo-Schettini, A. and Peron, A. Timed Cooperating Automata *Fundamenta Informaticae*, 43, 96-107, (2000).

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