Timed Automata with Urgent Transitions*

Roberto Barbuti
Luca Tesei
Dipartimento di Informatica - Università di Pisa,
via F. Buonarroti, 2 56127 Pisa - Italy,
email: `barbuti,tesei@di.unipi.it`

Abstract

We propose an extension to the formalism of timed automata by allowing urgent transitions. An urgent transition is a transition which must be taken within a fixed time interval from its enabling time and it has higher priority than other non-urgent transitions enabled in the same state. We give a set of rules formally describing the behavior 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. From a specification point of view, the use of urgent transitions allows shorter and clear specifications of behaviors involving urgency and priority. We use timed automata with urgent transitions for specifying a multicast protocol for mobile computing.

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 [7, 8]. Since their introduction, timed automata have been widely studied from different points of view [5, 6, 9, 10], in particular for their possible use in the verification of real-time systems [1, 2, 4, 11, 27, 28, 31].

Usually the expressiveness of timed automata is given in terms of accepted timed languages, but, because of their use as a specification formalism, also

*A first version of this paper appeared in [15] 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 [14, 23, 24, 26, 29]). 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 systems was already introduced in [18, 20, 19], where the urgency of transitions outgoing from a state is induced by a time progress condition associated to the state and derived from deadlines associated to the transitions. The semantics imposes the impossibility to stay in the state if the condition is not satisfied 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 and, in this situation, have *priority* upon non-urgent transitions.

From the expressiveness point of view, both our approach and the one of [18, 20, 19] are suitable for specifying timed systems. The latter 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. Moreover, the semantic setting is different. In op. cit. the behaviors of a timed automaton are all possible derivations of the semantic transition system in which the time progress conditions are kept satisfied. In this paper we adopt a model of behavior in which conditions in the states are not used and some behaviors of the semantic transition system are discarded according to the Büchi acceptance condition for timed automata (see Section 2). Section 8 contains a more detailed discussion on the differences of the two approaches.

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.

From a specification point of view, urgent transitions provide an easy and effective way to express some important types of behaviors of real time systems, which, otherwise, should be simulated using complex constructions. These usually yield a difficult to understand specification and increase the probability of mistakes.

The notions of priority and urgency captured by our definitions are inherently non-compositional and, hence, the transformation is not a congruence with respect to parallel composition. Thus, in the specification task, one has to design components considering that the urgency of their actions can be affected and/or enhanced by other components and by the synchronization mechanism of the parallel composition. This aspect is explained in Sections

5 and 6. The latter contains an example of specification of a multicast protocol for mobile computing, which is used to illustrate the possible use and the features of timed automata with urgent transitions. Finally, Section 7 contains some remarks and suggestions for an effective use of timed automata with urgent transitions in the verification of systems.

2 Timed automata

We recall the definition of timed automata [8]. In the following, $\mathcal{R}$ is the set of real numbers, $\mathcal{R}^{\geq 0}$ the set of non-negative real numbers and $\mathcal{R}^{+}$ is the set of positive real-numbers. $\mathcal{Q}$ is the set of rational numbers and $\mathcal{Q}^{+}$ is the set of positive rational numbers. A *clock* takes values from $\mathcal{R}^{\geq 0}$. Given a set $\mathcal{X}$ of clocks, a *clock valuation* over $\mathcal{X}$ is a function that assigns 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 functions from $\mathcal{X}$ to $\mathcal{R}^{\geq 0}$. Given $\nu \in \mathcal{V}_{\mathcal{X}}$ and $\delta \in \mathcal{R}^{\geq 0}$, we use $\nu + \delta$ (resp. $\nu - \delta$) to 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 $\nu(x) - \delta < 0$, $\nu - \delta$ is not a clock valuation.

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 $x \in \mathcal{X}$, $t \in \mathcal{N}$ 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 $x = t$ is the disjunction of $x < t$ and $x > t$. This disjunction can be simulated, as usual, by the non-deterministic choice.

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:

$$\nu \models \mathit{true}$$

$\nu \not\models \mathit{false}$

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

$\nu \models x\#t$ 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 its 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:

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

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 (Action sequence, Run) *Let $T = (Q, \Sigma, \mathcal{E}, I, R, \mathcal{X})$ be a timed automaton and let $r = s_0 \xrightarrow{l_0} s_1 \xrightarrow{l_1} \ldots$ be an* infinite *derivation of the transition system $\mathcal{S}(T)$:*

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

  $t_0 = 0$

$$t_{i+1} = t_i + \begin{cases} 0 & \textit{if } l_i \in \Sigma \\ l_i & \textit{otherwise} \end{cases}$$

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

- *The* action sequence *of r is the projection of the event sequence of r on the pairs $\{(l, t) \mid l \in \Sigma\}$*

- *The* state sequence *of r is the projection of the sequence $s_0 s_1 s_2 \cdots$ on the states $\{q_i \in Q \mid s_i = (q_i, \nu_i),\ i = 0 \ \vee\ (i \geq 1 \ \wedge\ l_{i-1} \in \Sigma)\}$.*

- *We say that r is a* run *of T if it satisfies the* Büchi acceptance condition*: there exists a state $q \in R$ such that q occurs infinitely many times along the state sequence of r.*

The Büchi acceptance condition discards all the finite runs and expresses a constraint on the infinite sequence of states taken by the automaton by edges in $\mathcal{E}$. Thus, an infinite derivation may exist which is not a run of the automaton because it violates the Büchi condition. For instance, a derivation whose state sequence ends with an infinite sequence of the same state q (obtained, say, by a self-loop transition in the state q of the timed automaton) is not considered as a run if q does not belong to R.

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 $t_i \in \mathcal{R}^{\geq 0}$ and $t_i \leq t_{i+1}$, for all $i \in I\!N$.*

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 and it is denoted by $\mathcal{L}(T)$.*

A synchronized product is defined on timed transition tables, that is to say, timed automata without the set of repeated states. A timed transition table of a certain timed automaton T can be considered an automaton that accept the action sequences of all infinite derivations of the transition system $\mathcal{S}(T)$ defining the semantics of T. A subset of such set of action sequences is $\mathcal{L}(T)$, according to the Büchi acceptance condition.

Definition 2.5 (Timed transition table) *Let $T = (Q, \Sigma, \mathcal{E}, I, R, \mathcal{X})$ be a timed automaton. The timed transition table of T is denoted $\hat{T}$ and it is the tuple $(Q, \Sigma, \mathcal{E}, I, \mathcal{X})$.*

Definition 2.6 (Product) *Let $T_1 = (Q_1, \Sigma_1, \mathcal{E}_1, I_1, R_1, \mathcal{X}_1)$ and $T_2 = (Q_2, \Sigma_2, \mathcal{E}_2, I_2, R_2, \mathcal{X}_2)$ be two timed automata with $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$. The product of the timed transition table of T_1 and of T_2, denoted by $\hat{T}_1 \parallel \hat{T}_2$, is given as follows:*

$$\hat{T}_1 \parallel \hat{T}_2 = \langle Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \mathcal{E}, I_1 \times I_2, \mathcal{X}_1 \cup \mathcal{X}_2 \rangle$$

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

1. **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* $((q_1, q_2), \psi_1 \wedge \psi_2, \gamma_1 \cup \gamma_2, \sigma, (q_1', q_2'))$
2. T_1 **actions**
   $\forall \sigma \in \Sigma_1 \setminus \Sigma_2, \forall (q, \psi, \gamma, \sigma, q') \in \mathcal{E}_1, \forall s \in Q_2$
   $\mathcal{E}$ *contains* $((q, s), \psi, \gamma, \sigma, (q', s))$
3. T_2 **actions**
   $\forall \sigma \in \Sigma_2 \setminus \Sigma_1, \forall (q, \psi, \gamma, \sigma, q') \in \mathcal{E}_2, \forall s \in Q_1$
   $\mathcal{E}$ *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.

The product of timed transition tables is a simplification of the definition of the parallel composition between timed automata. This parallel composition operation requires that in every run r of the obtained automaton, the projection of the states of r on each component j results in a sequence of states that represents a run of the j-th automaton. Thus, the parallel composition automaton has to be defined such that an infinite action sequence is a run if and only all the correspondent action sequences of the components satisfy the Büchi acceptance condition according to their sets of repeated states.

This can be done, as in [8], by adding a component at each state of the product automaton. This new component behaves as a counter that keeps track of which components entered one of their repeated states. Consider the automata T_1 and T_2 of the previous definition. The set of states of their parallel composition is $Q_1 \times Q_2 \times \{0, 1, 2\}$. Initially the counter is 0. If the first component enter a state belonging to R_1 by a transition, the new state have the third component equal to 1. Then, when the second component will enter a state belonging to R_2 the new state will have the counter at 2. The subsequent transition reset the counter to 0. The set of repeated states of the composed automaton is $R_1 \times R_2 \times \{2\}$. In this way if the composition has a run with Büchi acceptance condition, this is true, by construction, also for the two components. Conversely, again by construction, if the components have a run with Büchi acceptance condition, the resulting run of the automaton satisfies the Büchi condition. For the details see [8].

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 real-time systems. The idea is to provide, in each state of the automaton, the possibility of labeling some outgoing edges as *urgent*. Intuitively an urgent labeled edge must be taken with higher priority with respect to the non-urgent ones and it must be taken within a certain time interval starting from its enabling.

Let q be a state of a timed automaton and let e be an outgoing transition from q. We use $\langle b_e^s, b_e^e \rangle$ to denote the time interval in which the transition e is enabled, according to its constraint, in the state q during a derivation of the semantic transition system of the automaton. Note that such an interval is represented by its bounds, b_e^s and b_e^e, which can be closed or open; that is $b_e^s \in \{(t, [t\}$ and $b_e^e \in \{t), t]\}$, where t is, in general, a non-negative rational value.

Let $\ell \in \mathcal{Q}^+$ be a fixed parameter associated to the timed automaton. Let t_q be the time instant in which the state q is entered. Suppose e is an urgent transition. The time interval in which e must be taken starts at time b_e^s, but if the constraint of e is already satisfied when q is entered then the interval starts at time t_q. The interval stops ℓ time units after its start. However, if the constraints of e becomes false before, the interval stops at instant b_e^e. Figure 1 shows three cases. The point-filled areas are the intervals in which e

is enabled. The oblique-line-filled areas are the intervals in which e is enabled but not executable. The case (a) is the basic case in which the interval has length ℓ, starts at the enabling time of e and stops after ℓ time units while e is still enabled. Case (b) shows the situation in which e becomes disabled before ℓ time units elapsed and the interval is shorter than ℓ. In case (c) the transition is already enabled when the state is entered: the interval starts at t_q and its length is ℓ.

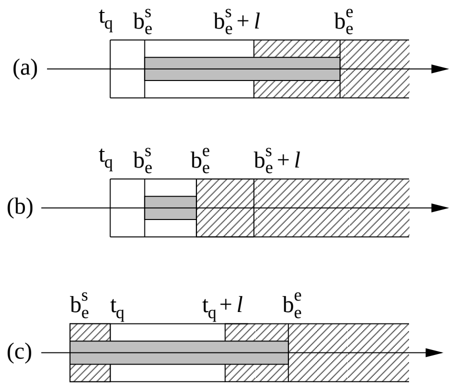

Figure 1: The semantics of urgency

This notion of urgency allows 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. To see this, 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 should the urgent transition be executed? It is not possible to answer precisely this question since the time domain is dense. To avoid this problem we introduce the constant ℓ and the interval within the action must be executed. Adopting this solution, the problem does not arise and the semantics of urgency is clear in all cases.

To complete the specification of the proposed notion of urgency we have to specify the nature of the bounds of the interval. The choice is arbitrary in principle, but the following seems to be the most natural. If the left bound of the interval is open, then we set the right bound closed, i.e. for the urgent action with constraint $x > 1$ of the example above, the interval in which such an action must be taken is that in which $x \in (1, 1 + \ell]$. Conversely, if the left bound is closed then we set the right bound open. For instance, if the transition has a constraint of the form of $x \geq 1$, instead of $x > 1$, then the interval in which it must be taken is that in which $x \in [1, 1 + \ell)$. When the state is entered and the urgent transition is already satisfied, the left bound of the interval is considered closed. In this way the interval is always well defined.

The denseness also imposes that the constant ℓ be greater than 0. The choice $\ell = 0$ could be interpreted as "immediately", but this leads, in some cases, to the problem discussed above. However, since $\ell \in \mathcal{Q}^+$, it can be chosen as small as needed. In other words, the "as soon as possible" limit behavior can be approximated with arbitrary precision.

Note that the parameter ℓ can be *local* to each urgent transition, but 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.

When a state q has a set U_q of outgoing urgent transitions, they are treated as follows. Let b^s be the lower bound of the first interval, in the state, in which some urgent transition becomes enabled. Let $S_q = \{u_1, u_2, \ldots, u_k\}$ be the set of urgent transitions which become enabled after b^s and let $I_{u_i} = \langle b^s_{u_i}, b^e_{u_i}\rangle$, $i = 1, \ldots, k$ be their enabling time intervals. The time interval in which at least one urgent transition must be taken is $\langle b^s, \mathtt{min\_bound}(b^s + \ell, b^e_{u_1}, \ldots, b^e_{u_k})\rangle$, where $\mathtt{min\_bound}$ gives the more restrictive interval upper bound. In this interval all enabled urgent transitions have the same priority and one of them is executed non-deterministically. Figure 2 shows two possible scenarios. The white area gives the interval in which at least an urgent transition must be taken. In (a) u_1 is the first enabled transition. u_2 has the minimal upper bound, thus the interval in which one urgent transition must be taken starts with $b^s_{u_1}$ and ends with $b^e_{u_2}$. In this interval one of u_1, u_2 and u_3 (in the sub-interval in which it is enabled) can be taken. In (b), the first urgent transition enabled is u_1 and it remains enabled also after ℓ time units. Thus, in this case, the interval to consider has length ℓ because the other ones are still enabled. Within this interval u_2 and u_3 can be taken, when enabled, as well.

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 no urgent transitions are enabled.

Moreover, when some urgent transition is enabled in a state, the unique way to continue the run is to execute it or another urgent transition following the rules expressed above.

Definition 3.1 (Timed Automaton with Urgent Transitions) *Let $\ell \in \mathcal{Q}^+$ be a constant. A timed automaton with urgent transitions T^ℓ_u 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,*

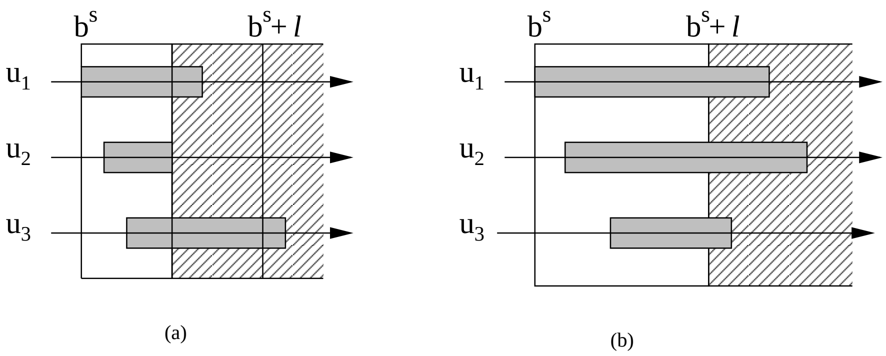

Figure 2: The case of more than one urgent transitions overlapping

$I \subseteq Q$ 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 $Q \times \Psi_{\mathcal{X}} \times \Gamma_{\mathcal{X}} \times \Sigma \times Q$.

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.

The semantics of a timed automaton with urgent transitions T_u^ℓ is defined, as for timed automata, in terms of its accepted language. This is defined in the same way of the accepted language for timed automata. The difference is that we use another infinite transition system to define action sequences and runs: the transition system $\mathcal{S}(T_u^\ell) = (S_u, \rightarrow)$. The states S_u are triples (q, ν, δ_q) such that $q \in Q$ is the current state of the automaton T_u^ℓ, ν is the current clock valuation and $\delta_q \in \mathcal{R}^{\geq 0}$ 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:

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

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

$$\textbf{(Urgent)} \qquad \frac{\begin{array}{c}(q,\psi_u,\gamma_u,\sigma_u,q') \in \mathcal{U},\ \nu \models \psi_u,\\ (\neg\exists u' = (q,\psi_{u'},\gamma_{u'},\sigma_{u'},q'_{u'}) \in \mathcal{U}.\\ (\exists\delta.0 \leq \delta \leq \delta_q \wedge \delta \geq \ell \wedge \nu - \delta \models \psi_{u'})),\\ (\neg\exists\ u' = (q,\psi_{u'},\gamma_{u'},\sigma_{u'},q'_{u'}) \in \mathcal{U}.\\ (\exists\delta.0 \leq \delta \leq \delta_q \wedge \nu \not\models \psi_{u'} \wedge \nu - \delta \models \psi_{u'})\end{array}}{(q,\nu,\delta_q) \xrightarrow{\sigma_u} (q',\nu\backslash\gamma_u,0)}$$

Rule **(Time)** lets the time elapse in a state and updates both the clock valuation and the time elapsed in the state. Recall that, due to the acceptance condition semantics, some states, in the action sequence, must be entered infinitely many times. Thus the automaton is not allowed to elapse time in a state infinitely.

Rule **(Non-Urgent)** can be used when T_u^ℓ is in a state without outgoing urgent edges (the $\forall$ condition is trivially true). In this case the behavior is the same as timed automata. When T_u^ℓ is in a state with a set of urgent outgoing transition, the "$\neg\exists$" condition in the rule requires that every urgent transition has never been enabled since the current state was entered. If this is false the rule is not applicable. Note that when a new state is entered the time elapsed is set to 0.

Rule **(Urgent)** executes an urgent action σ_u. The first "$\neg\exists$" condition ensures that the urgent transition that is going to be executed is taken before a time ℓ has elapsed after the enabling time of any urgent transition. The second "$\neg\exists$" forbids the execution of the urgent transition if there is a urgent transition which was enabled and it is no longer so.

Example 3.2 *Figure 3 shows an example of a timed automaton with a urgent transition (graphically, we show this by attaching a "u" to the edge). 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* $(1, 2]$.

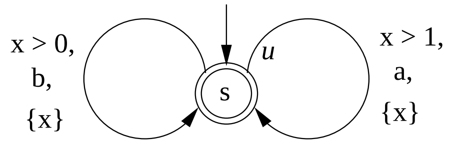

Figure 3: An automaton with urgent transitions, T_u^1

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 three-steps transformation which preserves the accepted language. Because timed automata are special cases of timed automata with urgent transitions ($\mathcal{U} = \emptyset$), the transformation is only given starting from the latter ones.

4.1 The region form of a timed automaton

Let T_u^ℓ 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 ($b \neq 0$), it is always possible to transform a $T_u^{\frac{a}{b}}$ automaton to an isomorphic one T_u^a by multiplying all the constants in the clock constraints by b. Practically this means that a different scale is used to measure time and, indeed, this does not affect specification/verification tasks. So we can assume without loss of generality that ℓ is a positive natural number.

Given a set of clocks $\mathcal{X}$, a clock region, as defined in [8], 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 both

- for every clock $x \in \mathcal{X}$, one clock constraint of the set $C_x = \{x = c \mid c = 0, 1, \ldots, c_x\} \cup \{c - 1 < x < c \mid c = 1, 2, \ldots c_x\} \cup \{x > c_x\}$ where c_x is the greatest constant, in the constraints of T, which x is compared to,

- for every pair of clocks x and y, with associated constraint $c-1 < x < c$ and $d-1 < y < d$, for some c, d, an inequality of type $\mathtt{fract}(x) \# \mathtt{fract}(y)$ where $\# \in \{<,=,>\}$ and $\mathtt{fract}(x)$ 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 in the specification of α.

In [8] 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[1].

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 that, starting from a constraint ψ, gives a *logically equivalent* constraint $\mathtt{min}(\psi)$ such that it does not contain redundancies. Essentially the transformation drops from ψ the atomic constraints which are implied by others, yielding a minimal conjunction of constraints.

Definition 4.1 *Given a constraint ψ over a set $\mathcal{X}$ of clocks. $\mathtt{min}(\psi)$ is the equivalent constraint where there is only one constraint of the form $x = c$, $x \# c$, $c \# x \#' d$ or $c \# x$, where $\#, \#' \in \{<, \leq\}$ for every clock $x \in \mathcal{X}$.*

Given $x \in \mathcal{X}$, we denote by $\mathtt{select}(\mathtt{min}(\psi), x)$ such unique constraint for x.

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. The resulting automaton is a timed automaton that is equivalent to the original one and has a structural property that will be used in the next step of the transformation for proving the correctness of the transformation itself.

Definition 4.2 *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:*

[1] This has to be imposed for those cases in which α does not belong to $\mathtt{succ}(\alpha)$ i.e. points or lines regions.

- *the states in* Q^r *(resp.* R^r*) are of the form* $\langle q, \alpha \rangle$ *where* $q \in Q$ *(resp.* R*) 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 [8]. This construction differs from the one for region automaton 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).

Also note that this step can be applied to any timed automaton yielding the following result.

Proposition 4.1 *Given a timed automaton with urgent transitions* T_u*, let* T_u^r *be* T_u *in region form. Then,* $\mathcal{L}(T_u) = \mathcal{L}(T_u^r)$*.*

Proof. By region construction correctness. □

We want to remark that the region form of a timed automaton can be, in general, a useful device for reasoning about the automaton itself and its structure. In particular, we exploit, in the next step of the transformation, the following property:

Proposition 4.2 *Let* T_u *be a timed automaton with urgent transitions and let* T_u^r *be* T_u *in region form. In every derivation of the transition system* $\mathcal{S}(T_u^r)$*, if a state* $(\langle q, \alpha \rangle, \nu)$ *is entered by performing a transition labeled by* $\sigma \in \Sigma$*, then* $[\nu] = \alpha$*.*

Proof. By definition 4.2. □

Example 4.3 *In Figure 4 it is shown the automaton of Figure 3 in region form, denoted by* T_u^{1r}*. Note that the constraints explicitly show 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*.*

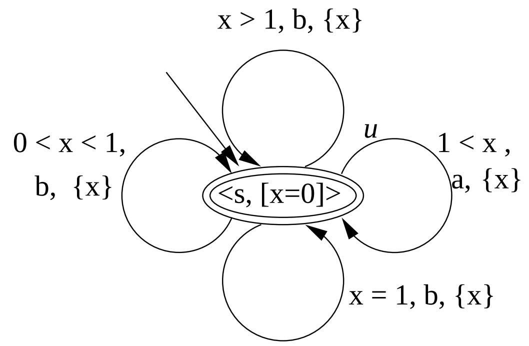

Figure 4: Automaton T_u^{1r}

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 a set of outgoing urgent transitions $U_q = \{u_1, \ldots, u_k\}$, where $u_i = (q, \psi_{u_i}, \gamma_{u_i}, \sigma_{u_i}, q'_{u_i})$, $i = 1, 2, \ldots, k$. Each of these transitions becomes, in T_u^r, a set of transitions

$$E_{u_i}^{\alpha} = \{(\langle q, \alpha \rangle, \mathtt{min}(\psi_{u_i}) \wedge \bigwedge_{x \in \mathcal{X}} R_{T_u}(\alpha'', x), \gamma_{u_i}, \sigma_{u_i}, \langle q'_{u_i}, [\gamma_{u_i} \rightarrow 0]\alpha'' \rangle) | \\ \alpha'' \in \mathtt{succ}(\alpha)\}$$

We need to determine the minimal upper bound of the enabling interval of all the urgent actions which can be enabled in a state. By the semantics of Section 3, we know that urgent transitions must be taken before such a bound.

Using the total order $\leq_\alpha$ defined in the set $\mathtt{succ}(\alpha)$ and the property of states expressed by Proposition 4.2 we can determine the set of urgent actions which will be enabled.

$$F_{\langle q, \alpha \rangle} \quad = \quad \{u_i \in U_q \mid \exists \alpha' \in \mathtt{succ}(\alpha) \cup \{\alpha\}.\ \alpha' \Rightarrow \mathtt{min}(\psi_{u_i})\}$$

If this set is empty, the state $\langle q, \alpha \rangle$ and its outgoing urgent transitions remain unchanged. Otherwise we add to each outgoing urgent transition an explicit constraint which forces it to respect both the expiry time expressed by ℓ and the minimal upper bound of the enabling interval of other urgent transitions. Such an explicit constraint does not modify the behavior of the automaton, but explicitly adds to the transition constraint the conditions expressed by the semantic rules of Section 3.

If in $F_{\langle q,\alpha\rangle}$ there are transitions that are already enabled when the state is entered we simply add to each transition in $\bigcup_{i=1}^{k} E_{u_i}^{\alpha}$ a new constraint imposing that the time elapsed in the state be less than ℓ. To do this we add in T_u^r a new clock variable. Whenever a state is entered this clock is reset, so it can be used in the constraints of outgoing edges as a measure of the time elapsed in the state.

If in $F_{\langle q,\alpha\rangle}$ there are only transitions that are enabled after some time, we determine the first clock region which satisfies a constraint of an urgent action $\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi_{u_i})) = min_{\alpha' \in \mathtt{succ}(\alpha)\cup\{\alpha\}}(\alpha' \Rightarrow \mathtt{min}(\psi_{u_i}) \wedge u_i \in F_{\langle q,\alpha\rangle})$. Note that if $\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi_{u_i})) = \alpha$ then the urgent transition is already enabled when the state is entered (remember Proposition 4.2). By convention, if the clock constraint $\mathtt{min}(\psi_{u_i})$ is equivalent to false or it is consistent, but it will never be true letting the time to elapse from α, the result of $\mathtt{fst\_succ}$ is $\top$ and it has the property of being greater than any element of $\mathtt{succ}(\alpha) \cup \{\alpha\}$. Moreover we can establish the immediate predecessor, according to the total order, of a clock region α' in the set $\mathtt{succ}(\alpha)$. Let us denote this by $\mathtt{prec}(\alpha')$. Again, if α' is the minimum in $\mathtt{succ}(\alpha)$, then its predecessor is α.

Definition 4.4 (Set of Crucial Clocks) *If $\alpha \not\Rightarrow \mathtt{min}(\psi_{u_i})$, then we define the set $\mathtt{cruc}(\alpha, \mathtt{min}(\psi_{u_i}))$ as the set $\mathcal{X} - \{x \in \mathcal{X} \mid R_{T_u}(\mathtt{prec}(\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi_{u_i}))), x) \Rightarrow \mathtt{select}(\mathtt{min}(\psi_{u_i}), x)\}$. It contains the only clocks that determine the truth of the constraint $\mathtt{min}(\psi_{u_i})$ in the region $\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi_{u_i}))$.*

Example 4.5 *Let us explain the concept of "crucial". Let $\mathtt{min}(\psi_{u_i})$ be $0 < x < 2 \wedge 1 < y < 3$. If α is $[x = 0 \wedge 0 < y < 1]$ then $\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi_{u_i})) = [1 < y < 2 \wedge 0 < x < 1, \mathtt{fract}(y) < \mathtt{fract}(x)]$ and $\mathtt{prec}(\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi_{u_i}))) = [y = 1 \wedge 0 < x < 1]$. In $\mathtt{prec}(\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi_{u_i})))$, the value of clock x, $0 < x < 1$, implies the atomic constraint $\mathtt{select}(\mathtt{min}(\psi_{u_i}), x) = 0 < x < 2$, so x is not crucial for $\mathtt{min}(\psi_{u_i})$. Instead, the value of y, $y = 1$, does not imply $\mathtt{select}(\mathtt{min}(\psi_{u_i}), y) = 1 < y < 3$. Thus, we have $y \in \mathtt{cruc}(\alpha, \mathtt{min}(\psi_{u_i}))$.*

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

Proposition 4.3 *The set of crucial clocks always contains at least one element.*

Proof. By absurd, suppose it is empty. Then, the clock region $\mathtt{prec}(\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi_{u_i})))$ would imply $\mathtt{min}(\psi_{u_i})$. But, by definition, $\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi_{u_i}))$ is the minimum clock region that implies $\mathtt{min}(\psi_{u_i})$ and $\mathtt{prec}(\mathtt{fst\_succ}(\alpha, \mathtt{min}(\psi_{u_i})))$ is strictly less than it using the order defined in $\mathtt{succ}(\alpha) \cup \{\alpha\}$. A contradiction. $\square$

Let $u_i \in F_{\langle q,\alpha\rangle}$. The constraint $\mathtt{select}(\mathtt{min}(\psi_{u_i}), x)$, given any crucial clock x, can be used to determine a constraint that force the urgent action to be executed within ℓ time units from the time in which it becomes enabled (and respecting the rules discussed in Section 3). To do this we add to any transition in $\bigcup_{i=1}^{k} E^{\alpha}_{u_i}$ the additional constraint $\mathtt{add}(\alpha, \mathtt{min}(\psi_{u_i})))$ constructed as follows. Given any crucial clock x for $\mathtt{min}(\psi_{u_i})$, $x \in \mathtt{cruc}(\alpha, \mathtt{min}(\psi_{u_i}))$:

- $\mathtt{add}(\alpha, \mathtt{min}(\psi_{u_i})))$ is $(x < c + \ell)$ if $\mathtt{select}(\mathtt{min}(\psi_{u_i}), x)$ is either $(x = c)$ or $(c \leq x \# d)$ or $(c \leq x)$, where $c < d$ and $\# \in \{\leq, <\}$.
- $\mathtt{add}(\alpha, \mathtt{min}(\psi_{u_i})))$ is $(x \leq c + \ell)$ if $\mathtt{select}(\mathtt{min}(\psi_{u_i}), x)$ is either $(c < x \# d)$ or $(c < x)$ where $c < d$ and $\# \in \{\leq, <\}$.

The final step in the construction of a ℓ consistent timed automaton is to add to all urgent transitions a constraint which forces all of them to be executed (if possible) before the upper bound of the enabling interval of the first disabled one: recall Figure 2(a). If such an urgent transition does not exist, this step, for the current state, ends. To formalize this search, we exploit again the total order defined in $\mathtt{succ}(\alpha)$. Given the set $F_{\langle q,\alpha\rangle}$, we search, for each element $(q, \psi_{u_i}, \gamma_{u_i}, \sigma_{u_i}, q'_{u_i})$ of this set, the clock region, in $\mathtt{succ}(\alpha)$, in which the constraint $\mathtt{min}(\psi_{u_i})$ becomes false, if any. $\mathtt{fst\_dis}(\alpha, \mathtt{min}(\psi_{u_i}))$ denote such clock region. If the constraint will never be false, letting the time to elapse, in the current state, the result of this operation is, by convention, $\top$ and has the property of being greater than any element of $\mathtt{succ}(\alpha)$. Now we can explicitly define the set of enabled urgent transition which are first disabled.

$$\begin{aligned} L_{\langle q,\alpha\rangle} \;=\; \{u_i \in F_{\langle q,\alpha\rangle} \mid\ & \mathtt{fst\_dis}(\alpha, \mathtt{min}(\psi_{u_i})) \neq \top \;\wedge\; \forall u_j \in F_{\langle q,\alpha\rangle}. \\ & \mathtt{fst\_dis}(\alpha, \mathtt{min}(\psi_{u_i})) \leq_\alpha \mathtt{fst\_dis}(\alpha, \mathtt{min}(\psi_{u_j}))\} \end{aligned}$$

Taking any transition u_i in this set, we have to find the clock constraint to add to other urgent transitions that disables them when u_i is disabled. To this purpose we introduce the following definition.

Definition 4.6 *Let ψ be a constraint without redundancies over a set of clocks $\mathcal{X}$.*

The lower opening $\mathcal{O}^-(\psi)$ *of* ψ *is obtained by deleting from* ψ *all the constraints of the form* $c \leq x$ *and* $c < x$*, and by substituting all the constraints of the form* $x = c$ *by* $x \leq c$*, for all* $x \in \mathcal{X}$*.*

Analogously, 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* $c \leq x$*, for all* $x \in \mathcal{X}$*.*

Clearly this definition requires constraints of the form $c\#x\#d$*,* $\# \in \{<, \leq\}$*, to be considered as* $c\#x \wedge x\#d$*.*

Thus if we want that all the urgent transitions will be disabled when u_i is disabled, we need to add to all of them the constraint $\mathcal{O}^-(\mathtt{min}(\psi_{u_i}))$ which becomes false at the same time of the first disabled urgent transition.

Definition 4.7 *Let* $T_u^r = (Q^r, \Sigma, \mathcal{E}^r, \mathcal{U}^r, I^r, R^r, \mathcal{X})$ *be a timed automaton with urgent transitions in region form. The* ℓ-consistent *version of it,* ℓT_u^r*, is the timed automaton* $(Q^r, \Sigma, \mathcal{E}^r, \mathcal{U}_\ell^r, I^r, R^r, \mathcal{X}^r)$ *where* $\mathcal{X}^r = \mathcal{X} \cup \{x_{\mathtt{time\_state}}\}$ *and* $\mathcal{U}_\ell^r$ *is constructed as follows:*

1. $(\langle q, \alpha \rangle, \psi, \gamma \cup \{x_{\mathtt{time\_state}}\}, \sigma, \langle q', \alpha' \rangle) \in \mathcal{U}_\ell^r$ *iff* $(\langle q, \alpha \rangle, \psi, \gamma, \sigma, \langle q', \alpha' \rangle) \in \mathcal{U}^r$ *and* $F_{\langle q, \alpha \rangle} = \emptyset$
2. $(\langle q, \alpha \rangle, \psi \wedge x_{\mathtt{time\_state}} < \ell, \gamma \cup \{x_{\mathtt{time\_state}}\}, \sigma, \langle q', \alpha' \rangle) \in \mathcal{U}_\ell^r$ *iff* $(\langle q, \alpha \rangle, \psi, \gamma, \sigma, \langle q', \alpha' \rangle) \in \mathcal{U}^r$ *and* $u \in F_{\langle q, \alpha \rangle}$ *and* $\alpha \Rightarrow \mathtt{min}(\psi_u)$ *and* $L_{\langle q, \alpha \rangle} = \emptyset$
3. $(\langle q, \alpha \rangle, \psi \wedge x_{\mathtt{time\_state}} < \ell \wedge \mathcal{O}^-(\mathtt{min}(\psi_{u'})), \gamma \cup \{x_{\mathtt{time\_state}}\}, \sigma, \langle q', \alpha' \rangle) \in \mathcal{U}_\ell^r$ *iff* $(\langle q, \alpha \rangle, \psi, \gamma, \sigma, \langle q', \alpha' \rangle) \in \mathcal{U}^r$ *and* $u \in F_{\langle q, \alpha \rangle}$ *and* $\alpha \Rightarrow \mathtt{min}(\psi_u)$ *and* $u' \in L_{\langle q, \alpha \rangle}$
4. $(\langle q, \alpha \rangle, \psi \wedge \mathtt{add}(\alpha, \mathtt{min}(\psi_u)), \gamma \cup \{x_{\mathtt{time\_state}}\}, \sigma, \langle q', \alpha' \rangle) \in \mathcal{U}_\ell^r$ *iff* $(\langle q, \alpha \rangle, \psi, \gamma, \sigma, \langle q', \alpha' \rangle) \in \mathcal{U}^r$ *and* $u \in F_{\langle q, \alpha \rangle}$ *and* $\alpha \not\Rightarrow \mathtt{min}(\psi_u)$ *and* $L_{\langle q, \alpha \rangle} = \emptyset$
5. $(\langle q, \alpha \rangle, \psi \wedge \mathtt{add}(\alpha, \mathtt{min}(\psi_u)) \wedge \mathcal{O}^-(\mathtt{min}(\psi_{u'})), \gamma \cup \{x_{\mathtt{time\_state}}\}, \sigma, \langle q', \alpha' \rangle) \in \mathcal{U}_\ell^r$ *iff* $(\langle q, \alpha \rangle, \psi, \gamma, \sigma, \langle q', \alpha' \rangle) \in \mathcal{U}^r$ *and* $u \in F_{\langle q, \alpha \rangle}$ *and* $\alpha \not\Rightarrow \mathtt{min}(\psi_u)$ *and* $u' \in L_{\langle q, \alpha \rangle}$

Proposition 4.4 *Given a timed automaton with urgent transitions in region form* T_u^r *and its* ℓ*-consistent version* ℓT_u^r*, then* $\mathcal{L}(T_u^r) = \mathcal{L}(\ell T_u^r)$*.*

Proof. Let us first add the clock $x_{\mathtt{time\_state}}$ to all edges of T_u^r. This operation results in an equivalent timed automaton with urgent transitions.

The proof proceeds for cases.

Case 1. Consider the hypothesis $F_{\langle q,\alpha\rangle} = \emptyset$.

We have

$F_{\langle q,\alpha\rangle} = \emptyset$

$\equiv$ {By definition of $F_{\langle q,\alpha\rangle}$}

$\neg\exists u \in U_q.(\exists\alpha' \in \mathtt{succ}(\alpha) \cup \{\alpha\}.\alpha' \Rightarrow \mathtt{min}(\psi_u))$

$\equiv$ {By Proposition 4.2}

$\neg\exists(\langle q,\alpha\rangle, \psi, \gamma, \sigma, \langle q',\alpha'\rangle) \in \mathcal{U}^r.[\nu] \in \alpha \cup \mathtt{succ}(\alpha) \wedge \nu \models \psi$

Thus, no urgent action can be performed in $\langle q,\alpha\rangle$ and the state is equivalent in both T^r_u and ℓT^r_u.

Case 2. Consider the hypothesis $u \in F_{\langle q,\alpha\rangle}$ and $\alpha \Rightarrow \mathtt{min}(\psi_u)$ and $L_{\langle q,\alpha\rangle} = \emptyset$.

We have:

(a)

$(u \in F_{\langle q,\alpha\rangle} \wedge \alpha \Rightarrow \mathtt{min}(\psi_u)) \;\wedge\; L_{\langle q,\alpha\rangle} = \emptyset$

$\Rightarrow$ {by definition of $L_{\langle q,\alpha\rangle}$}

$(u \in F_{\langle q,\alpha\rangle} \wedge \alpha \Rightarrow \mathtt{min}(\psi_u)) \wedge$
$(\neg\exists u \in F_{\langle q,\alpha\rangle}. \mid (\exists\alpha' \in \mathtt{succ}(\alpha).\ \mathtt{fst\_dis}(\alpha, \mathtt{min}(\psi_u)) \neq \top))$

$\Rightarrow$ {by definition of $F_{\langle q,\alpha\rangle}$, $L_{\langle q,\alpha\rangle}$ and Proposition 4.2}

$\forall\nu : [\nu] \in \alpha \cup \mathtt{succ}(\alpha).$
$(\forall\delta_{\langle q,\alpha\rangle} \geq \ell.(\exists u = \langle q,\alpha\rangle, \psi, \gamma, \sigma, \langle q',\alpha'\rangle) \in \mathcal{U}^r.(\exists\delta.0 \leq \delta \leq \delta_{\langle q,\alpha\rangle} \wedge \delta \geq \ell \wedge \nu - \delta \models \psi)) \wedge$
$(\forall\delta_{\langle q,\alpha\rangle} < \ell.(\neg\exists u = \langle q,\alpha\rangle, \psi, \gamma, \sigma, \langle q',\alpha'\rangle) \in \mathcal{U}^r.(\exists\delta.0 \leq \delta \leq \delta_{\langle q,\alpha\rangle} \wedge \delta \geq \ell \wedge \nu - \delta \models \psi)) \wedge$
$\forall\delta_{\langle q,\alpha\rangle}.(\neg\exists u = \langle q,\alpha\rangle, \psi, \gamma, \sigma, \langle q',\alpha'\rangle) \in \mathcal{U}^r.(\exists\delta.0 \leq \delta \leq \delta_{\langle q,\alpha\rangle} \wedge \nu \not\models \psi \wedge \nu - \delta \models \psi)$

Now, let us consider $(\langle q,\alpha\rangle, \psi, \gamma, \sigma, \langle q',\alpha'\rangle) \in \mathcal{U}^r$.

The condition for performing such an action, given by the **(Urgent)** rule is:

$\nu \models \psi \wedge$
$(\neg\exists u' = (\langle q,\alpha\rangle, \psi_{u'}, \gamma_{u'}, \sigma_{u'}, \langle q',\alpha'\rangle) \in \mathcal{U}^r.$
$(\exists\delta.0 \leq \delta \leq \delta_{\langle q,\alpha\rangle} \wedge \delta \geq \ell \wedge \nu - \delta \models \psi_{u'}))$

$(\neg\exists u' = (\langle q,\alpha\rangle, \psi_{u'}, \gamma_{u'}, \sigma_{u'}, \langle q',\alpha'\rangle) \in \mathcal{U}^r.$
$(\exists\delta.0 \le \delta \le \delta_{\langle q,\alpha\rangle} \wedge \nu \not\models \psi_{u'} \wedge \nu - \delta \models \psi_{u'}))$

which, under the hypothesis and proof (a), is equivalent to:

$\nu \models \psi \wedge \delta_{\langle q,\alpha\rangle} < \ell$

Now $\nu \models \psi \wedge \delta_{\langle q,\alpha\rangle} < \ell$ iff $\nu \models (\psi \wedge x_{\texttt{time\_state}} < \ell)$, thus an urgent action $(\langle q,\alpha\rangle, \psi \wedge x_{\texttt{time\_state}} < \ell, \gamma \cup \{x_{\texttt{time\_state}}\}, \sigma, \langle q',\alpha'\rangle) \in \mathcal{U}^r_\ell$ can be performed if and only if $(\langle q,\alpha\rangle, \psi, \gamma, \sigma, \langle q',\alpha'\rangle) \in \mathcal{U}^r$ can be performed.

Case 3. Consider the hypothesis $u \in F_{\langle q,\alpha\rangle}$ and $\alpha \Rightarrow \texttt{min}(\psi_u)$ and $u' \in L_{\langle q,\alpha\rangle}$.

Under the hypothesis $u \in F_{\langle q,\alpha\rangle}$ and $\alpha \Rightarrow \texttt{min}(\psi_u)$, the proof of Case 2. states that the introduction of the constraint $x_{\texttt{time\_state}} < \ell$ guarantees that no urgent action can be performed after ℓ time units from the enabling of the first urgent action.

Now we have an additional hypothesis:

$u' \in L_{\langle q,\alpha\rangle}$

$\Rightarrow$ {By the definitions of $L_{\langle q,\alpha\rangle}$, $\mathcal{O}^-$ and the hypothesis}

$\forall\nu : [\nu] \in \alpha \cup \texttt{succ}(\alpha).\nu \not\models \mathcal{O}^-(\texttt{min}(\psi_{u'})) \equiv ((\exists\delta.0 \le \delta \le \delta_{\langle q,\alpha\rangle} \wedge \nu \not\models \psi_{u'} \wedge \nu - \delta \models \psi_{u'}))$

Thus, for the **(Urgent)** rule, an urgent action could be performed only if $\nu \models \mathcal{O}^-(\texttt{min}(\psi_{u'}))$. We can conclude that an urgent action $(\langle q,\alpha\rangle, \psi \wedge x_{\texttt{time\_state}} < \ell \wedge \mathcal{O}^-(\texttt{min}(\psi_{u'}), \gamma \cup \{x_{\texttt{time\_state}}\}, \sigma, \langle q',\alpha'\rangle) \in \mathcal{U}^r_\ell$ can be performed if and only if $(\langle q,\alpha\rangle, \psi, \gamma, \sigma, \langle q',\alpha'\rangle) \in \mathcal{U}^r$ can be performed.

Cases 4. and 5. The proof proceeds analogously to the ones of the other cases.

□

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_u^r. 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 the state $\langle q,\alpha\rangle$ in ℓT_u^r becomes true. The disjunction of the upper opening (Definition 4.6) of all urgent edges constraints (without redundancies) outgoing from a state in ℓT_u^r, describes a right-infinite time interval to the beginning of which an urgent transition must be taken. The negation of this disjunction must be added to all the constraints of non-urgent edges of T_u^r 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 4.8 *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_u^{quiet}, 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 4.9 *Figure 5 shows the automaton T^{1quiet} which is the 1-consistent and quieted version of the automaton T_u^1 of Figure 3. Note that the constraint $1 < x$ on one of the edges 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 $x_{\mathtt{time\_state}}$ is omitted because it is useless in this case.*

Theorem 4.5 *Let T_u be a timed automaton with urgent transitions, T_u^r the corresponding timed automata in region form and T_u^{quiet} its quiet version. Then $\mathcal{L}(T_u) = \mathcal{L}(T^{quiet})$.*

Proof. Starting from the result of Proposition 4.4, the proof proceeds analogously to the one of Proposition 4.4 itself. □

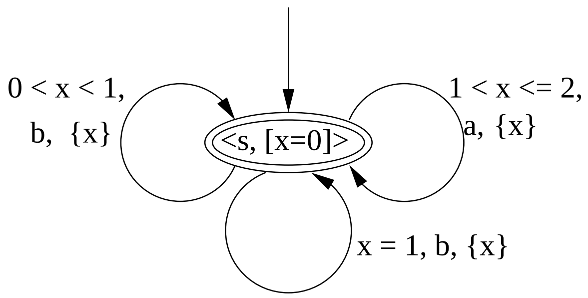

Figure 5: Automaton T^{1quiet}

As a last remark of this section we want to discuss the size of the output of the transformation. Indeed, it turns out that our notion of urgent actions is a succinct way to express urgency. To see this consider that one could just design his/her automaton as the quiet version we obtain by the transformation. That is to say, one can think about states as pairs $\langle q, \alpha \rangle$ and use suitable constraints to express urgency as we do by the algorithm.

However, it is clear that the quiet automaton has a larger size than the automaton with urgent transitions. As a first approximation the size of the quiet version of the automaton is at least the size of the region automaton of [8]. Here we consider the size of a timed automaton as the sum of the number of states and the number of clock constraints on the edges. The size of the quiet version is greater than that because we add constraints (and edges) to induce the behavior specified by the formal semantics.

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 (see Section 2). 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 2.6,

- the urgency of a transition of a component extends to the transition obtained by the rules (*ii*) and (*iii*) of Definition 2.6.

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 $T_1^{quiet} \parallel T_2^{quiet}$ (the standard parallel composition of the quiet version of them).

Figure 6: Automaton A_u^1 and its quiet version A^{1quiet}

Example 5.1 *Consider the automaton A_u^1 of Figure 6(a), the automaton T_u^1 of Figure 3 and the parallel composition $T_u^1 \parallel A_u^1$. The action c, having the constraint $y > 0$ and being urgent, surely preempts the action b at the beginning of every run. The parallel composition of the quiet version of T_u^1 (Figure 5) and of the quiet version of A_u^1 (Figure 6(b)) has a different behavior: in the automaton $T^{1quiet} \parallel A^{1quiet}$ the action b can be performed at the beginning of a run before the urgent action c. This is because the transformation of T_u^1 into T^{1quiet} does not consider the urgency of the action c belonging to the component A^{1quiet}.*

Thus, the interpretation of the non-urgent outgoing transitions in a state of an automaton respect to the urgency of a transition depends on the context in which the automaton is considered. If it is viewed as a component instead of a stand-alone system, the enabling of those transitions should change taking into account other component's urgent transitions that could interleave with them. This aspect turns out explicitly in the example of the following section.

6 An example

As an example of specification of a system with urgent actions we use a part of a multicast protocol for mobile computing presented in [12]. There the protocol is specified by the Calculus of Communicating Systems (CCS) [30], and the Concurrency Workbench tool [13] is used to state some required properties.

Let us start with an informal description of the protocol, quoted from [12]:

"...We consider a system composed of mobile hosts and stationary hosts.

Communication occurs solely via message-passing. Some stationary hosts (gateways) are connected to a wired network that provides reliable and FIFO-ordered communication and to a wireless link, that covers a spatially limited *cell* nearby the gateway. Mobile hosts may move and communicate through wireless links. A gateway may broadcast messages to all mobile hosts in its cell. A mobile host may only exchange messages with the gateway of the cell where it happens to be located.

We shall associate a different meaning with the terms "receiving" and "delivering" a message. A host C *receives* a message msg when msg arrives at its protocol layer. Upon receiving msg, the protocol may discard msg or pass msg up to one of its applications. In the latter case, C is said to *deliver* msg.

The protocol works as follows. A dedicated stationary host acts as the *coordinator*, denoted as SC. A mobile host generates a multicast by sending a message to a gateway, which forward it to SC. SC constructs a message containing an increasing sequence number, then transmits the resulting message to gateways through a FIFO-multicast. Gateways broadcast this message to mobile hosts in the respective cells.

Due to its movement across cells, any mobile host m could receive duplicates or could miss multicasts. By maintaining a history of the received sequence numbers, m discards duplicates in the former case and sends to the stationary hosts a proper *nack* message in the latter (e.g., upon receiving an out-of-order message). Upon receiving a *nack*, the stationary host will relay to m a copy of the missing multicasts."

The protocol was designed to guarantee several properties, in particular: "Each mobile host m delivers each multicast under reasonable assumptions, as follows: m stops delivering messages if: (i) m starts entering and leaving cells so quickly that its messages never arrive to any stationary host or messages from gateways are systematically lost; and (ii) this pattern of movements persists forever.

..."

The explicit assumption in the formulation of the property is due to the fact that CCS specifications cannot express constraints on the speed of mobile hosts. Thus, using this specifications, the system can have "bad" behaviors, which must be ruled out in the property formulation. This means that the verification of the property is restricted to all the possible "good" behaviors. That is, the execution paths in which the mobile host enters and leaves cells without getting any message, although possible, are disregarded.

With timed automata with urgent transitions we can easily describe the behavior of the components involved in the protocol together with important time parameters such as the speed of a mobile agents, that is the minimal amount of time it can stay in a cell or the frequency it exchanges messages with the stationary hosts. This allows to state the property on the whole system behavior which, due to time parameters, should not allow "bad" executions.

To make a simple example of this feature of timed automata with urgent transitions, we extract from the above presented protocol the part dealing with the iterated request of a mobile host to broadcast a message until it effectively receive it. For reasons of simplicity we assume only one mobile host, two stationary ones and the coordinator, as shown in Figure 7. Of course, the mobile host, being unique, cannot receive multicasts generated by other mobile hosts. We assume that multicasts are generated directly by the gateways.

The system is specified by the parallel composition of the automata specifying each entity.

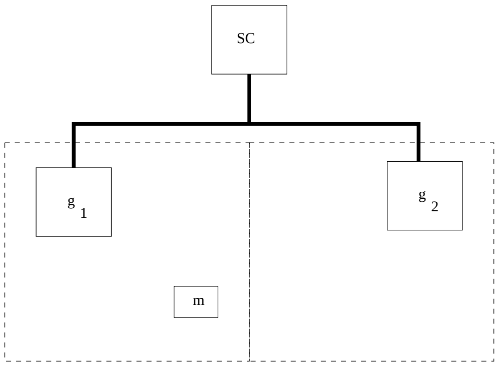

Figure 7: The scenario

The coordinator SC is defined in Figure 8. It accepts a request from a gateway g_j ($\texttt{req}_{g_j}(i)$) for broadcasting the $i-th$ message. Then it tries to contact all the gateways to signal that the $i-th$ message must be sent to their cells. Note that the coordinator tries to contact the stationary hosts sequentially, it passes to contact the next one only if either the previous accepted the request ($\texttt{signal}_{g_j}(i)$) or it does not respond for twenty time units ($\texttt{fail}_{g_j}$). Note that in the latter case the coordinator proceeds in, at most, twenty one time units. It is important to remark that the action of contacting the stationary hosts is urgent. That is, when enabled, it cannot be skipped and it must be done within a given time interval. One could observe that the automaton always have the urgent transition enabled when it enters state 2 (and 3) and so the $\texttt{fail}_{g_1}(i)$ ($\texttt{fail}_{g_2}(i)$) transition can never be taken. But, as observed in Section 5, this automaton has to be understood as a component of a whole system. This means that the semantics of urgency is defined in terms of the whole system where the action $\texttt{signal}_{g_1}(i)$ (and $\texttt{signal}_{g_1}(i)$) is a synchronization action and can be executed only if the partner (the gateway) can execute it. This depends on the state in which it currently is. Thus, in some runs of the whole system, it could happen that the action $\texttt{fail}_{g_1}(i)$ ($\texttt{fail}_{g_2}(i)$) is taken depending on the relative speed of the components and on the interleaving of non-synchronization actions. Note that without urgent transitions we cannot impose the priority and the urgency of the action $\texttt{signal}_{g_1}(i)$ ($\texttt{signal}_{g_1}(i)$) with respect to the action $\texttt{fail}_{g_1}(i)$ ($\texttt{fail}_{g_2}(i)$) in state 2 (3) because the standard semantics of timed automata would execute them non-deterministically.

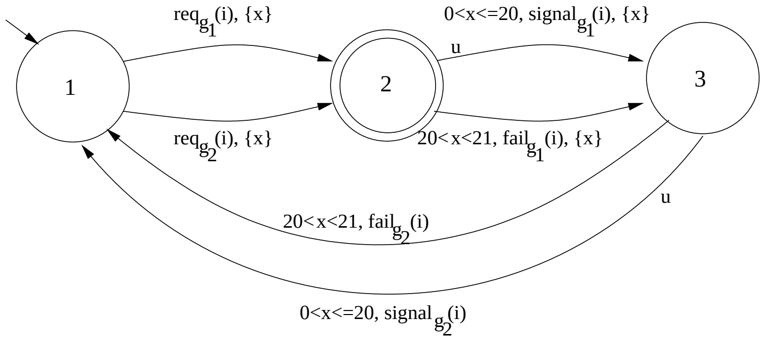

Figure 8: The coordinator SC

A gateway g_j is defined in Figure 9. It passes the requests for broadcasting the $i-th$ message from the mobile host ($\texttt{reqm}_{g_j}(i)$) to the coordinator ($\texttt{req}_{g_j}(i)$), and broadcasts the message $\texttt{msg}_{g_j}(i)$ on its cell upon request of the coordinator ($\texttt{signal}_{g_j}(i)$). Note that the action of sending the message is urgent, it can be skipped only if the mobile host do not respond within ten time units. The same remarks on urgency on parallel composition given above applies here.

Finally, the mobile host m is defined in Figure 10. It tries, every two units of time, and for a duration of two time units, to contact the gateway for requesting to broadcast the $i-th$ message. When the $i-th$ message is received and delivered a message counter is increased. Note that, due to parallel composition, both actions are urgent, so they must be done if

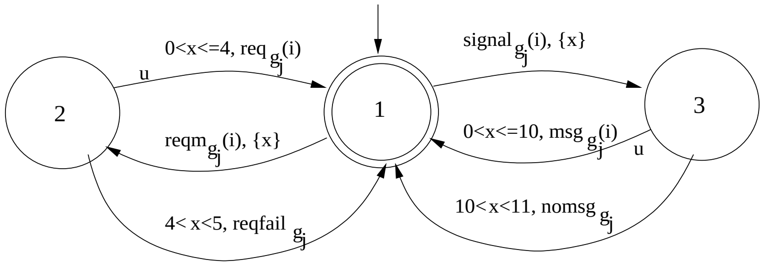

Figure 9: The gateway g_j

enabled. Moreover, the mobile host can move from a cell to another; however its speed cannot allow to stay in a cell less than fifty time units. This action is not urgent, that is a host can stay in a cell also if the action of moving is enabled.

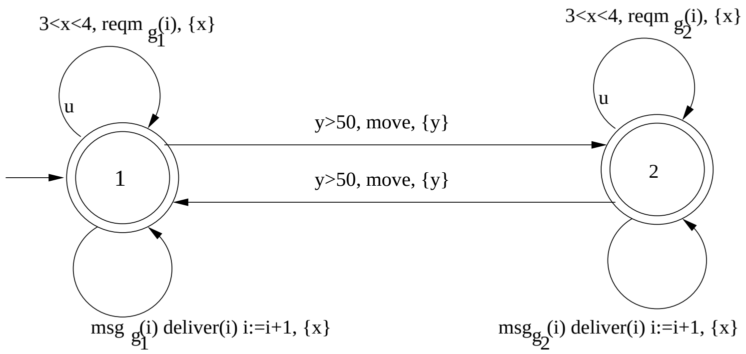

Figure 10: The mobile host m

Our notion of urgency expresses both a priority among actions and a time constraint on their execution. Both these concepts are useful in specifying real systems. In the scenario of Figure 7, the mobile host, when staying within a cell, must first of all try to communicate with the gateway and, if possible, it must do it immediately. If we remove the urgency annotation from the previous automata, the mobile host could never communicate while waiting for the time of moving.

With this specification, the correctness property of the protocol can be simply restated as: "Each mobile host m delivers each multicast".

Because we have not yet a tool implementing our notion of urgency, we have used the features of UPPAAL [16] (a verification tool for timed automata) to verify the protocol. In particular we used its notions of urgent states and urgent channels to simulate our notion of urgency. These features are not sufficient to express the notion that we have introduced in this paper and thus the system that we have tested is an approximation of the one shown in this section. We do not report here the details of this test. It results that the property expressed in the previous paragraph is verified by a system in which there are two gateways, a mobile host and a fixed number n of broadcast messages. It is interesting to report that with different relative times in the constraints of the mobile host in Figure 10 the property is not verified. This means that the speed of the mobile host is a crucial parameter for determining the truth of the property.

7 Notes on Implementation

In this section we address two problems that arise in the effective use of timed automata with urgent transitions and the defined transformation. The first is the progress model used by the automata and the second is the state explosion due to the region automaton construction.

We have used in this paper the original model of timed automata [8] in which the semantics of an automaton is its accepted timed language according to the Büchi acceptance condition. Moreover the states of an automaton have not *invariant* constraints to satisfy. This type of model always allows time to advance in a state (recall rule 1 of the transition system for the semantics of timed automata in Section 2 or rule **(Time)** of the one for timed automata with urgent transitions in Section 3). If in a derivation the time advance too much so that the automaton cannot perform transitions any more, then the derivation is discarded by the acceptance condition and is not considered as a behavior of the system. The problem with this model is the fact that we cannot decide how to progress, from each state, into a correct run using only the information of the state. In other words the progress of a correct run requires, in each local state, an information that is global to the automaton. It is difficult to implement or to construct a simulator for this model. For this reason, in verification and model checking tools, the used model is different. The information required for the progress is *local*: the states are equipped with an invariant condition which must be always true when the control is in the state (the absence of a condition corresponds to a true one). This feature can be used to assures the progress of every derivation of the transition system defining the semantics of the automaton. Acceptance conditions are not used and the implementation of the model is relatively easy. This model was introduced in [28] and is used in all simulators and verification tools developed for timed automata.

From a technical point of view, the local progress model correspond to the one defined in Sections 2 and 3 with the following differences:

- the states of the automaton can be associated with invariants (clock constraints)
- the rules of the transition system that lets the time to elapse (see above) have one more premise of the form $\forall \delta'.0 < \delta' \leq \delta \Rightarrow \nu + \delta' \models \phi$ where ϕ is the state invariant.
- the semantics of a timed automaton is the set of all infinite, time divergent derivations of the transition system.

It is easy to see that the definition of timed automata with urgent transitions and the transformation that we have developed is the same if the model of local progress is used. The only difference is in the construction of the region form of the automaton in which every state $\langle s, \alpha \rangle$ has the invariant condition of the state s in the original automaton. The proof of correctness is the same.

The real difference between the two models is in the process of specification and/or design of a real-time system by a timed automaton. It is in that process that one decides the progress model to adopt and then, accordingly, writes the automaton. Thus, we remark that our definition and transformation is correct w.r.t. both the models. We have chosen the original one because in that context we can state that the expressing power (the set of timed languages recognizable by one device) of timed automata with urgent transitions is the same than the one of original timed automata (Theorem 4.5).

The problem of state explosion in verification using timed automata is a serious aspect that has been attacked from the beginning of the presentation of the model. The main source of this explosion is the region construction that we have recalled in Section 4. The number of regions is exponential in the number of clocks and in the magnitude of the largest integer constant used in the clock constraints. The main tool to contain this complexity is the use of clock *zones* instead of clock regions. A clock zone is a convex union of clock regions and, like these ones, can be expressed by a conjunction of clock constraints. The use of zones and of other suitable data structures (e.g. difference-bound matrices) have been very useful in making practically feasible the verification of properties of timed system specified by timed automata (see, for instance, [5, 31, 2, 3]).

For the sake of simplicity and clarity of exposition we have used, in our transformation of Section 4, the region form of a timed automaton (Definition 4.2). Such device is very useful to define and, for its properties, also to prove the correctness of the transformation. However, it suffers of the problem of state explosion. With regard to this, we state that it is possible to define a transformation that uses clock zones instead of clock regions. Such a transformation starts with the definition of the *zone form* of a timed automaton. This automaton is the analogous of the region form of the timed automaton if the zone automaton construction [5, 31, 3] is used instead of the region automaton construction. The second and the third step of the transformation remain unchanged.

8 Related works

The notion of urgency and/or priority for timed formalisms has been studied in the past. In [17] the urgency of actions has been investigated in the process algebra field with the concept of discrete time. Different notions of priority have been introduced for timed automata in [25] and for timed process algebras in [22, 21].

A closer approach to ours can be found in [18, 20, 19]. 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 [20] the following remark.

"The relative position of d with respect to ψ determines the urgency of the action. For a given ψ, the corresponding d may take two extreme values: first, $d = \psi$, 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 ψ (cannot be disabled without enforcing its execution).*

The condition $d \Rightarrow \psi$ 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* $d = \psi = x > 2$*, implying the TPC* $x \leq 2$*, which is not right-open. Then, if* x *is initially* 2*, time cannot progress by any delay* ψ*, 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 in which x is in $[2, 2 + \ell)$. On the other hand, if a transition with a left-closed guard, say $x \geq 2$, need to be fired "as soon as possible" we can approximate this behavior using a constant ℓ small as needed.

With regard to the problem of right-closed TPCs it is also interesting to note that the introduction of urgent transitions in the tools of verification of timed automata is a difficult task. By now, the latest version of UPPAAL[2] has introduced the concept of "urgent channels"[3] which model transitions that must be fired as soon as possible. But their use is very restricting because a guard *cannot* be associated with them.

References

[1] L. Aceto, P. Bouyer, A. Burgueño, and K. G. Larsen. The power of reachability testing for timed automata. In *Proceedings of the 18th Conference on Foundations Software Technology and Theoretical Computer Science (FSTTCS'98)*, number 1530 in Lecture Notes in Computer Science, pages 245–256. Springer, Berlin, 1998.

[2] L. Aceto, A. Burgueño, and K. Guldstrand Larsen. Model checking via reachability testing for timed automata. In *Proceedings of the 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98)*, number 1384 in Lecture Notes in Computer Science, pages 263–280. Springer, Berlin, 1998.

[3] R. Alur. Timed automata. In *Proceedings of the 11th International Conference on Computer-Aided Verification (CAV'99)*, number 1633 in Lecture Notes in Computer Science, pages 8–22. Springer, Berlin, 1999.

[4] R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking in dense real-time. *Information and Computation*, 104:2–34, 1993.

[2]UPPAAL Version 3.4.2 downloadable at `http://www.uppaal.com`

[3]See the documentation at `http://www.uppaal.com`

[5] R. Alur, C. Courcoubetis, N. Halbwachs, D. L. Dill, and H. Wong-Toi. Minimization of timed transition systems. In *In Proceedings of the 3rd International Conference on Concurrency Theory (CONCUR'92)*, number 630 in Lecture Notes in Computer Science, pages 340–354. Springer, Berlin, 1992.

[6] R. Alur, C. Courcoubetis, and T. A. Henzinger. The observational power of clocks. In *Proceedings of the 5th International Conference on Concurrency Theory (CONCUR'94)*, number 836 in Lecture Notes in Computer Science, pages 162–177. Springer, Berlin, 1994.

[7] R. Alur and D. L. Dill. Automata for modeling real-time systems. In *Proceedings of the 17th Colloquium on Automata, Languages and Programming (ICALP'90)*, number 443 in Lecture Notes in Computer Science, pages 322–335. Springer, Berlin, 1990.

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

[9] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 211:253–273, 1999.

[10] R. Alur and T. A. Henzinger. Back to the future: Towards a theory of timed regular languages. In *Proceedings of the 33th Annual IEEE Symposium on Foundations of Computer Science (FOCS'92)*, pages 177–186. IEEE Computer Society Press, 1992.

[11] R. Alur and T. A. Henzinger. A really temporal logic. *Journal of the ACM*, 41:181–204, 1994.

[12] G. Anastasi, A. Bartoli, N. De Francesco, and A. Santone. Efficient verification of a multicast protocol for mobile computing. *The Computer Journal*, 44:21–30, 2000.

[13] R. Cleaveland anf S. Sims. The ncsu concurrency workbench. In *Proceedings of the 8th conference on Computer Aided Verification (CAV'96)*, number 1102 in Lecture Notes in Computer Science, pages 394–397. Springer, Berlin, 1996.

[14] R. Barbuti, N. De Francesco, and L. Tesei. Timed automata with non-instantaneous actions. *Fundamenta Informaticae*, 47(3-4):189–200, 2001.

[15] R. Barbuti and L. Tesei. Timed automata with urgent transitions. In F. Corradini and W. Vogler, editors, *Proceedings of the 2nd International Workshop on Models for Time-Critical systems (MTCS'01)*, number NS-01-5 in BRICS Notes, pages 3–21. Department of Computer Science, Aarhus University Press, 2001.

[16] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi. UPPAAL – a tool suite for automatic verification of real-time systems. In *Proceedings of the 4th DIMACS Workshop on Verification and Control of Hybrid Systems*, number 1066 in Lecture Notes in Computer Science, pages 232–243. Springer, Berlin, 1996.

[17] T. Bolognesi and F. Lucidi. Timed process algebras with urgent interactions and a unique powerful binary operator. In *Proceedings of Real-Time: Theory in Practice Workshop (REX Workshop 1991)*, number 600 in Lecture Notes in Computer Science, pages 124–148. Springer, Berlin, 1992.

[18] S. Bornot and J. Sifakis. Relating time progress and deadlines in hybrid systems. In *Proceedings of International Workshop on Hybrid and Real-Time Systems (HART'97)*, number 1201 in Lecture Notes in Computer Science, pages 286–300. Springer, Berlin, 1997.

[19] S. Bornot and J. Sifakis. An algebraic framework for urgency. *Information and Computation*, 163(1):172–202, 2000.

[20] S. Bornot, J. Sifakis, and S. Tripakis. Modeling urgency in timed systems. In *Proceedings of Compositionality Meeting (COMPOS'97)*, number 1536 in Lecture Notes in Computer Science, pages 103–129. Springer, Berlin, 1998.

[21] P. Brémond-Grégoire and I. Lee. A process algebra of communicating shared resources with dense time and priorities. *Theoretical Computer Science*, 189(1-2):179–219, 1997.

[22] M. Buchholtz, J. Andersen, and H. H. Lovengreen. Towards a process algebra for shared processors. In F. Corradini and W. Vogler, editors, *Electronic Notes in Theoretical Computer Science*, volume 52. Elsevier, 2002.

[23] C. Choffrut and M. Goldwurm. Timed automata with periodic clock constraints. *Journal of Automata, Languages and Combinatorics*, 5(4):371–403, 2000.

[24] F. Demichelis and W. Zielonka. Controlled timed automata. In *Proceedings of 9th International Conference on Concurrency Theory (CONCUR'98)*, number 1466 in Lecture Notes in Computer Science, pages 455–469. Springer, Berlin, 1998.

[25] E. Fersman, P. Petterson, and W. Yi. Timed automata with asynchronous processes: Schedulability and decidability. In *Proceedings of TACAS 2002*, number 2280 in Lecture Notes in Computer Science, pages 67–82. Springer, Berlin, 2002.

[26] V. Gupta, T. A. Henzinger, and R. Jagadeesan. Robust timed automata. In *Proceedings of International Workshop on Hybrid and Real-Time Systems (HART'97)*, number 1201 in Lecture Notes in Computer Science, pages 331–345. Springer, Berlin, 1997.

[27] T. A. Henzinger and P. W. Kopke. Verification methods for the divergent runs of clock systems. In *Proceedings of the 2nd International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'94)*, number 863 in Lecture Notes in Computer Science, pages 351–372. Springer, Berlin, 1994.

[28] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.

[29] R. Lanotte, A. Maggiolo-Schettini, and A. Peron. Timed cooperating automata. *Fundamenta Informaticae*, 43:153–173, 2000.

[30] R. Milner. *A Calculus of Communicating Systems*. Springer, Berlin, 1980.

[31] S. Yovine. Model checking timed automata. In *Lectures on Embedded Systems*, number 1494 in Lecture Notes in Computer Science, pages 114–152. Springer, Berlin, 1996.