Accepting Zeno words: a way toward timed refinements

Béatrice Bérard and Claudine Picaronny

LSV, CNRS UMR 8643, ENS de Cachan, 61 av. du Prés. Wilson,
F-94235 Cachan Cedex, France, Fax: +33 (0)1 47 40 24 64
E-mails: Beatrice.Berard@lsv.ens-cachan.fr,
Claudine.Picaronny@lsv.ens-cachan.fr

Abstract. Timed models were introduced to describe the behaviors of real-time systems and they were usually required to produce only executions with divergent sequences of times. However, when some physical phenomena are represented by convergent executions, Zeno words appear in a natural way. Moreover, time can progress if such an infinite execution can be followed by other ones.
Therefore, in a first part, we extend the definition of timed automata, allowing to generate sequences of infinite convergent executions, while keeping good properties for the verification of systems: emptiness is still decidable.
In a second part, we define a new notion of refinement for timed systems, in which actions are replaced by recognizable Zeno (timed) languages. We study the properties of these timed refinements and we prove that the class of transfinite timed languages is the closure of the usual one (languages accepted by Muller or Büchi timed automata) under refinement.

1 Introduction

The framework of timed systems. Timed models have been intensively studied for the specification and verification of real-time systems. For instance, timed Petri nets [18], timed transition systems [17,14], timed automata [1,2] or timed I/O automata [16] have been discussed. Contrary to classical (untimed) transition systems, these models provide an explicit notion of time and thus allow to describe time requirements. They have been successfully used for the verification of real-time systems [10,15].

In these systems, executions are witnessed by finite or infinite sequences of timed actions $(a_1, t_1)(a_2, t_2)\cdots$, where t_i is the date at which action a_i takes place. Surprisingly, executions for which the time sequence $t_1 t_2 \cdots$ is convergent appear in these models. The corresponding sequences, called Zeno words in reference to Zeno's paradox, are usually forbidden: the values t of time exceeding the limit cannot be reached and the state of the system at this time t is thus undefined. Of course, such a remark can also be made for finite sequences. In these cases, time is said not to progress any more.

Zeno words and transfinite automata. Nevertheless, Zeno words (as well as finite words) can be convenient to describe some physical phenomena that produce convergent time sequences: for instance when a continuous action is represented by an infinite discrete sequence or, from another point of view, when an infinite number of actions takes place within a finite interval of time. Of course, time does not stop, and other phenomena can be observed later on. This idea was already expressed in [12], where systems, in which the state can change infinitely often in a finite time, are investigated within the framework of the duration calculus. Different examples, like the fall of a satellite towards a planet or the Car-Bee problem, are proposed there. As an other illustration, consider an elastic ball which is dropped on an horizontal plane, bounces while losing amplitude, stops after some time and can then be dropped again or trigger the beginning of another observation. In this particular case, the oscillation's frequency increases, so that if action a represents the movement of the ball between two contacts, we obtain an infinite sequence $(a,t_1)(a,t_2)\cdots$ with convergent time, followed by other sequences, thus producing what is called a transfinite timed word. In fact, in the usual timed models, a Zeno word makes time stop only because an execution contains a single infinite sequence. Indeed, automata accepting transfinite untimed words have already been studied ([8,9,20,13]). Even though they do not take explicitely into account the notion of time, these automata suppose implicitely that an infinite number of actions may occur in a finite interval of time, in order to be followed by other actions. From this point of view, adding an explicit notion of time in such models is interesting in itself.

Plan of the paper. In this paper, our first purpose is to define equivalent versions of timed automata generating transfinite timed words (Sections 2, 3, 4). We then study the class, denoted by TL, of timed languages accepted by these automata (Section 5). We prove that this class has convenient closure properties, particularly under the concatenation and the derived star and ω-power operations, which become possible with Zeno words. Moreover, we show that emptiness remains decidable in the class TL, which is an important issue regarding the verification of systems. The second part of the paper is devoted to the important notion of timed refinement (Sections 6, 7). A refinement consists in the replacement of a single action (intended to represent some high level abstract action) by a language. Although many results have been obtained in the classical framework of transition systems, only a few cases have been investigated for timed models [7]. We introduce a special class of timed automata, which can be used to describe timed refinement and include the cases in [7]. Finally, the main result of the paper is the following: the class of languages accepted by timed transfinite automata is exactly the closure under refinement of the class of languages accepted by classical timed automata.

2 Preliminaries

In this section, we introduce all background, notations and definitions used along the paper.

2.1 Ordinals

All facts concerning ordinals in this paper can be found in [19]. Note that they are mainly used here to number elements of sequences.

Recall that the finite ordinals are the natural numbers and the first non-finite ordinal is denoted by ω. In general, an ordinal α represents a linearly well-ordered set (up to isomorphism) and two ordinals α, β may be summed $(\alpha + \beta)$ by considering (the isomorphism class of) the union $\alpha \cup \beta$, where all elements of α are (strictly) less than all elements of β. The ordinal $\alpha + 1$ is called the *successor* of α. This addition defines a product (as for the natural numbers): for instance, $\omega.2 = \omega + \omega$ and $\omega^2 = \omega + \omega + ...$, ω times. Note that the usual notation $\omega^0 = 1$ also applies here.

In this work, we consider only ordinals smaller than ω^ω which have a so called "polynomial" decomposition $\alpha = \sum_{k=p}^{0} \omega^k.n_k$, where $p, n_0, n_1 \cdots$ are natural numbers and the *type* of α is the least integer k such that $n_k \neq 0$. If the type of α is positive, i.e. α does not have a greatest element, then α is called a *limit* ordinal. Thus, a limit ordinal of type k can be written $\alpha = \beta + \omega^k$, where $\beta = 0$ or β is an ordinal of type greater than or equal to k.

2.2 Timed α-words and concatenation

Let α be an ordinal, $\alpha < \omega^\omega$, and let Σ be a finite alphabet of actions. We define timed words of length α over Σ and we extend the basic operations for languages of timed words.

Recall that an α-word over Σ is an α-sequence $u = (a_i)_{i<\alpha}$, where a_i is in Σ for each $i < \alpha$. A *timed α-word* (or simply *timed word*) over Σ is an α-sequence $w = ((a_i, t_i))_{i<\alpha}$, where a_i is in Σ for each $i < \alpha$ and $(t_i)_{i<\alpha}$ is a non decreasing sequence of non negative real numbers. The value t_i represents the time at which the action a_i ends, and $t_\alpha = sup(t_i, i < \alpha)$ is called the *ending time* of w and denoted by $\tau(w)$. A *Zeno (timed) word* is a timed α-word for which the ending time is finite.

As time progresses, such a word can be followed by other timed actions, so that we extend the usual operation of concatenation after Zeno words:
if $w = (a_i, t_i)_{i<\alpha}$ and $w' = (a'_i, t'_i)_{i<\alpha'}$ are timed words such that the ending time of w is finite, then

$$ww' = ((a''_i, t''_i))_{i<\alpha+\alpha'} \text{ with } \begin{cases} (a''_i, t''_i) = (a_i, t_i) & \text{for each } i < \alpha \\ (a''_{\alpha+i}, t''_{\alpha+i}) = (a'_i, \tau(w) + t'_i) & \text{for each } i < \alpha'. \end{cases}$$

If we write ε for the (timed) empty word, then of course, $\varepsilon w = w$.
A *timed language* is a set of timed words. For two timed languages L and L',

the concatenation (with the associated star and ω-product operations) becomes possible after Zeno words:

- the concatenation of L and L' is the set of timed words of the form ww', where $w \in L$ and $w' \in L'$, when the concatenation of w and w' is defined,
- the star and ω-power operations are defined as usual from the concatenation.

3 Automata accepting transfinite untimed words

Different models of automata accepting transfinite untimed words have been studied ([9,20,13,4]), since Büchi ([8]) first introduced them to prove the decidability of monadic second order logics. Restricted to ordinals less than ω^ω, these models have the same expressive power: they accept transfinite languages having regular expressions. In all these models, the basic idea is to allow an infinite execution to reach a state of the automaton in order to be able to go on. This state is a kind of limit of the sequence of states visited previously. The difference between the models lies in the different ways of defining this limit. We first give the different definitions of limit, then recall the definitions of the models.

3.1 Continuous sequences in a finite set

For a finite set Q, we denote by $\mathcal{P}(Q)$ the power set of Q without the empty set. We consider the iterations of $\mathcal{P}$, defined inductively by: $\mathcal{P}^0(Q) = Q$ and, for any natural number k, $\mathcal{P}^{k+1}(Q)$ is the powerset of $\mathcal{P}^k(Q)$ without the empty set. We write $\mathcal{P}^{\leq k}(Q) = Q \cup \mathcal{P}(Q) \cup \cdots \cup \mathcal{P}^k(Q)$.

For any limit ordinal $\alpha < \omega^\omega$, we consider the set $Succ^{<\alpha}$ of all successor ordinals less than α. Recall that if the type of α is $k > 0$, then $\alpha = \beta + \omega^k$, where $\beta = 0$ or β is an ordinal of type greater than or equal to k. Let $\mathbf{q} = (q_i)_{i \in Succ^{<\alpha}}$ be any sequence of elements in a finite set Q. We define a limit value for $\mathbf{q}$ in two different ways.

- *Choueka's limit:* In this case, $\lim^c(\mathbf{q})$ is in $\mathcal{P}^k(Q)$.
  It is defined inductively as the set of elements which appear infinitely often in the sequence $\{\lim^c((q_i)_{i \in Succ^{<\beta+\omega^{k-1}.p}})\}_{p \in \mathbb{N}}$ if $k \geq 1$ and $\{(q_i)_{i \in Succ^{<\beta+p}})\}_{p \in \mathbb{N}}$ if $k = 1$.
- *Büchi's limit:* In this case, $\lim^b(\mathbf{q})$ is in $\mathcal{P}(Q)$.
  It is the set of states q such that there exists a strictly increasing sequence of successor ordinals $(j_p)_{p \in \mathbb{N}}$ having α as limit and such that $q = q_{j_p}$, for all p in $\mathbb{N}$.

3.2 Choueka and Büchi automata

A Choueka or a Büchi automaton is a finite automaton with a global set of states split into several layers. An execution begins as in a usual automaton in the first layer of states, called states of type *zero*. After an infinite (ω) sequence of actions, the set of infinitely repeated states in the path is considered as a state of the second layer, called a *meta-state*. A new action may then be performed to get down to the first layer and the execution can go on. The difference between the two models appears after at least ω^2 executions.

In Choueka's model, there exists a natural number n such that the set of meta-states (the second layer) is itself split into n layers (meta-states of type 1, meta-states of type 2,..., meta-states of type n). If β is an ordinal of type $k \leq n$, the state reached after β transitions also is of type k. For instance, a state of the third layer will be reached after ω^2 actions.

In Büchi's model, the meta-states are used indifferently after a limit ordinal of transitions, whichever its type.

We now precisely define the two types of automata.

- A *Choueka n-automaton* (or c-n-automaton) is a tuple $\mathcal{A} = (\Sigma, Q, I, F, \Delta)$, where
  Σ is a finite alphabet of actions,
  Q is a finite set of states of the form $Q = \mathcal{P}^{\leq n}(Q[0])$, where $Q[0]$ denotes the set of type 0 states,
  $I \subseteq Q[0]$ is a subset of initial states,
  $F \subseteq Q$ is a set of final states,
  $\Delta \subseteq \mathcal{P}^{\leq n-1}(Q[0]) \times \Sigma \times Q[0]$ is the transition relation.

- A *Büchi automaton* (or b-automaton) is a tuple $\mathcal{A} = (\Sigma, Q, I, F, \Delta)$, where Σ is the alphabet and
  Q is the finite set of state of the form $\mathcal{P}^{\leq 1}(Q[0]) = Q[0] \cup \mathcal{P}(Q[0])$, where again $Q[0]$ is the set of type 0 states,
  $I \subseteq Q[0]$ and $F \subseteq Q$ are the sets of initial and final states,
  $\Delta \subseteq Q \times \Sigma \times Q[0]$ is the transition relation such that $\forall (s, a, q) \in \Delta$, with $s \in \mathcal{P}(Q)$ and $q \in Q$, then $q \notin s$.

For such an automaton $\mathcal{A}$, a run on a α-word $u = (a_i)_{i<\alpha}$ is an α-sequence $(q_i)_{i\leq\alpha}$ of states such that:

- if i is the successor of some ordinal denoted by $i-1$, there is a transition $(q_{i-1}, a_{i-1}, q_i) \in \Delta$,
- if i is a limit ordinal, let $\mathbf{q} = (q_j)_{j \in Succ^{<i}}$. Then, q_i is the meta-state defined by: $q_i = lim^c(\mathbf{q})$ in the case of a Choueka automaton, $q_i = lim^b(\mathbf{q})$ in the case of a Büchi automaton.

The run is *accepting* if $q_0 \in I$ and $q_\alpha \in F$ and in this case, the word w is *accepted* by $\mathcal{A}$. The language $L(\mathcal{A})$ is the set of words accepted by $\mathcal{A}$.

Example. The c-2-automaton in Figure 1, with initial state 1 and final state 3, accepts the single word $a^\omega b^\omega c$ with length $\omega.2+1$. After an infinite (ω) number of a's in state 1, meta-state $\{1\}$ is reached. From this point, an infinite number of b's is possible, yielding $\{2\}$ as new meta-state, and transition c allows to reach the final state 3.

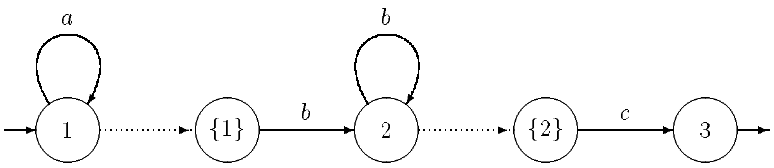

Fig. 1. A c-2-automaton for $a^\omega b^\omega c$

Remark 1. For a c-n-automaton $\mathcal{A}$, since $F \subseteq \mathcal{P}^{\leq n}(Q[0])$, the words in $L(\mathcal{A})$ have a length smaller than ω^n.
If $\mathcal{A}$ is a b-automaton, we say that $\mathcal{A}$ is a b-n-automaton, if all words accepted by $\mathcal{A}$ have length less than or equal to ω^n. Note that a b-n-automaton also is a b-m-automaton, for all $m \geq n$. Note also (cf. [13]) that a b-automaton is a b-n-automaton, for some $n \in \mathbb{N}$, thanks to the restriction introduced in the definition of the transition relation: with this condition, all accepted words have length less than or equal to ω^n where n is the cardinal of $Q[0]$, thus $\mathcal{A}$ is a n-automaton ($\mathcal{A}$ often is a m-automaton for a value $m < n$, but n may also be optimal); without the condition, a b-automaton could accept words with a length $> \omega^\omega$.
In order to simplify the presentation, we shall write o-n-automata, with $o \in \{b, c\}$, instead of c-n-automata and b-n-automata. We also omit the b or c when the context is clear.

Remark 2. The definition of a run explains why a run is sometimes called continuous: the meta-state q_i corresponding to a limit ordinal i is the set of states (of the type just below) through which the execution has gone infinitely often until reaching i. In particular, a c-1-automaton or a b-1-automaton is a Muller automaton, where a usual Muller acceptance condition is just a final meta-state of type 1.

4 Timed transfinite automata

Combining the definitions above and timed automata, as introduced in [1], we obtain different definitions of timed transfinite automata. Actually, we show that, as in the untimed case, all these models have the same expressive power. Therefore, in the rest of the paper, we choose the definition which is the more convenient with respect to the readability of the presentation or of the proof.

A timed transfinite automaton is obtained from an untimed one, using a finite set X of variables called clocks. Recall that a *constraint* (over X) is a propositional formula using the logical connectives in $\{\vee, \wedge, \neg\}$ over atomic formulae of the form $x \# c$, for $x \in X$, c some constant in $\mathbb{Q}$, and $\# \in \{<, =, >\}$. For the global time and the time values of clocks, we use non-negative real numbers. If x is a clock in X, we denote by $x(t) \in \mathbb{R}_+$ the clock value of x at time $t \in \mathbb{R}_+$, and by $X(t)$ the tuple $(x(t))_{x \in X}$ of all clock values of X. As usual, for a real number d, $X(t) + d$ is the tuple $(x(t) + d)_{x \in X}$. If the number of clocks in X is p, we may identify a constraint A with the subset of $\mathbb{R}_+^p$ of all the tuples of clock values satisfying A.

4.1 Timed o-automata

Definition 3. A *timed c-n-automaton* (over $\mathbb{R}_+$) is a tuple $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$, where Σ, Q, I and F are as in an untimed c-n-automaton, X is a finite set of clocks and the transition relation Δ contains elements of the form (s, A, a, ρ, q), where $s \in \mathcal{P}^{\leq n-1}(Q[0])$, A is a constraint, $a \in \Sigma$, $\rho \subseteq X$, and $q \in Q[0]$.

Definition 4. A *timed b-automaton* is a tuple $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$, where again Σ, Q, I and F are as in an untimed b-automaton and X is a finite set of clocks. The transition relation Δ contains elements of the form (s, A, a, ρ, q), with A, a and ρ are as above. In this case, $s \in Q$, $q \in Q[0]$ and $q \notin s$, whenever $s \in Q[1]$.

Let us recall the execution of a transition (q, A, a, ρ, q'), also written $q \xrightarrow{A,a,\rho} q'$, when q' is a state of type 0. Assume the automaton has entered state q at time t with clock values $X(t)$, the pair $(q, X(t))$ is called an *extended state* of $\mathcal{A}$. Then, the automaton may execute the transition at time $t' \geq t$, if the constraint A is satisfied by the clock values $x(t) + (t' - t)$ for all clocks $x \in X$. The automaton switches to state q' and enters this state at time t'. Moreover, the clocks in ρ are reset, so that the new clock values are $x(t') = 0$ for all $x \in \rho$ and $x(t') = x(t) + (t' - t)$ otherwise, and the automaton has reached the extended state $(q', X(t'))$. Note that, as usual, the tuple $X(t')$ of clock values associated with the date t' of a transition are those obtained after this transition.

Like for a timed automaton, we wish to define a run of the timed o-automaton $\mathcal{A}$ as an α-sequence of extended states:

$$(q_0, X(t_0)) \xrightarrow[t_0]{A_0, a_0, \rho_0} (q_1, X(t_1)) \xrightarrow[t_1]{A_1, a_1, \rho_1} (q_2, X(t_2)) \rightarrow \quad \cdots \qquad (q_\alpha, X(t_\alpha))$$

However, in order to do this, it must be possible to determine the clock values obtained when reaching a meta-state after an infinite convergent run. The following lemma (for which the proof is rather natural) gives the answer.

Remark 5. Note that, as in classical timed automata, it is not necessary to define limits of clock values for divergent sequences of times.

Lemma 6. *Let i be a limit ordinal. Assume that t_i is finite and let x be a clock in X. According to the possible positions where the clock x has been reset, one of these two cases is satisfied:*

– *Case 1: There exists a successor ordinal $j < i$ such that, for each successor ordinal h, $j < h < i$, $x(t_h) \neq 0$. Then, the sequence $(x(t_h))_{j<h<i}$ is non decreasing and bounded by t_i, so that it is convergent.*
– *Case 2: For each successor ordinal $j < i$, there exists an ordinal h, $j < h < i$, such that $x(t_h) = 0$. In this case, the sequence $(x(t_h))_{j<h<i}$ is the term of a convergent series, so that its limit is zero.*

Finally, a run of a timed o-automaton on a α-timed word $w = ((a_i, t_i))_{i<\alpha}$ is an α-sequence of extended states $(q_i, X(t_i))_{i\leq\alpha}$, such that:

- if $i \geq 1$ is the successor of some ordinal $i-1$, there is a transition $q_{i-1} \xrightarrow{A_i, a_i, \rho_i} q_i$, executed at time t_i, with $t_{i-1} \leq t_i$. The clock values $X(t_{i-1}) + t_i - t_{i-1}$ satisfy the constraint A_i and $x(t_i) = 0$ if $x \in \rho_i$, $x(t_i) = x(t_{i-1}) + t_i - t_{i-1}$ otherwise.
- if i is a limit ordinal, q_i is defined as in the untimed case. Moreover, thanks to Lemma 6, we have: $x(t_i) = lim_{h\to i} x(t_h)$, for all $x \in X$, if t_i is finite, and $X(t_i)$ is undefined otherwise.

The run is *convergent* if the ending time $t_\alpha = \tau(w)$ is finite. The word w is *accepted* if $q_0 \in I$, $t_0 = 0$, the initial valuation is $X(0)$, with $x(0) = 0$ for each clock $x \in X$ and $q_\alpha \in F$. The timed language $L(\mathcal{A})$ is the set of accepted words.

Remark 7. As in the untimed case, we say that a timed b-automaton $\mathcal{A}$ is a timed b-n-automaton, if all words in $L(\mathcal{A})$ have length less than ω^n.

Examples.
1. Let $\mathcal{A}$ be the timed c-2-automaton in Figure 2, with initial state 0 and final (type 2) state $\{\{1\}, \{2\}\}$, and let $(a_i, t_i)_{i<\omega^2}$ be a word in the language $L = L(\mathcal{A})$.

From the untimed point of view, we can compare $\mathcal{A}$ with the (untimed) automaton in Figure 1: it is then easy to see that a factor of the form $a^\omega b^\omega c$ is obtained here through the sequence of states $01^\omega\{1\}2^\omega\{2\}0$. Therefore, when the state $\{\{1\}, \{2\}\}$ is reached, the automaton accepts an infinite sequence of such factors and $(a_i)_{i<\omega^2} = (a^\omega b^\omega c)^\omega$.

Now look at the time constraints. The clock y has initial value zero in the initial state 0 and it is reset at each occurrence of c, when returning in this state. With the condition $y = 1$ in the next transition, this means that the first action a occurs at time 1 and the time difference between a c and the following a is equal to 1. Moreover, the clock x is reset each time a first a appears and must be less than 1 at the time of the next b, while the clock y must be less than 3 at the following occurrence of c. So, for each factor of the form $a^\omega b^\omega c$, the time difference between the first a and the first b is less than 1 and the time difference between this a and the c is less than 2.

2. The timed b-2-automaton in Figure 3, with initial state 0 and final state $\{0, 1, 2\}$, accepts the same language L as in Figure 2. However, in this case, any run with length ω^2 is reached by crossing infinitely often states 0, 1 and 2, at a sequence of ordinals having ω^2 as limit.

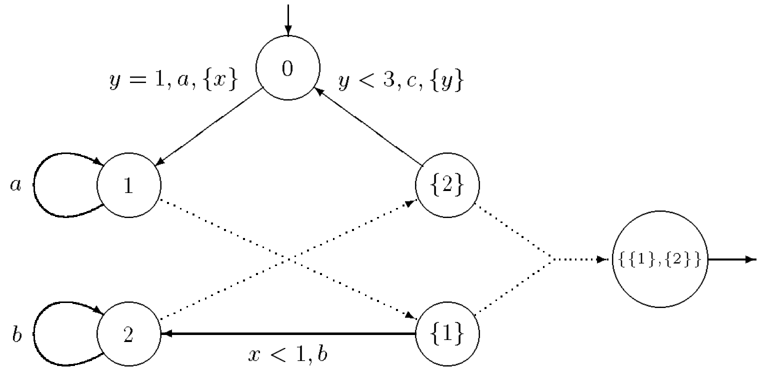

Fig. 2. A timed c-2-automaton $\mathcal{A}$ accepting L

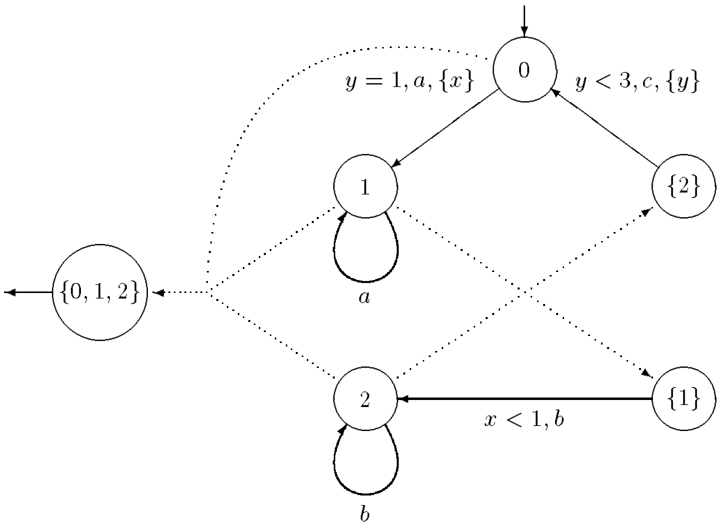

Fig. 3. A timed b-2-automaton for L

4.2 Reduced timed automata

For some further constructions, the previous models may appear too heavy to manipulate : taking some copies of zero-type states from a same automaton (in order to include in them some kind of information) will uselessly exponentially increase the number of meta-states. This phenomenon may even hide the information we want in the states. It may then be clearer if meta-states, playing a same role, are merged.

Definition 8. A *reduced timed c-n-automaton* (over $\mathbb{R}_+$) is a tuple $\mathcal{A} = (\Sigma, Q, \gamma, I, F, \Delta, X)$, where

Q is a finite set of states, disjoint union of $n+1$ subsets $Q[0], \ldots, Q[n]$, where each $Q[k]$ is the set of states of type k,
γ is a tuple $(\gamma_k)_{1 \leq k \leq n}$ where each γ_k is a mapping from $\mathcal{P}(Q[k-1])$ to $Q[k]$,
Σ, $I \subseteq Q[0]$, $F \subseteq Q$, Δ and X are as in a c-automaton.

Definition 9. A *reduced timed b-automaton* (over $\mathbb{R}_+$) is a tuple $\mathcal{A} = (\Sigma, Q, \gamma, I, F, \Delta, X)$, where

Q is a finite set of states, disjoint union of two sets $Q[0]$ and $Q[1]$,
γ is a map from $\mathcal{P}(Q[0])$ to $Q[1]$,
Σ, $I \subseteq Q[0]$, $F \subseteq Q$, Δ and X are as in a b-automaton.

As explained above, the role of the mapping γ_k in Choueka's case is to merge a set of type $k-1$ states, into a same type k state. Therefore, such a reduced automaton works exactly as a Choueka's one except for one thing. A run $(q_i, X(t_i))_{i \leq \alpha}$ satisfies the following condition:
for each limit ordinal i of type $k > 0$, if $\mathbf{q} = (q_j)_{j \in Succ^{<i}}$, then $q_i = \gamma_k(lim^c(\mathbf{q})$, instead of $q_i = lim^c(\mathbf{q})$ in the other definition. The mapping γ plays a similar role in Büchi's case, with $q_i = \gamma(lim^b(\mathbf{q}))$ instead of $q_i = lim^b(\mathbf{q})$.

Remark 10. The notion of reduced automaton actually *extends* the basic one: any timed c-n-automaton $\mathcal{A} = (\Sigma, Q, I, F, X, \Delta)$ can be viewed as a reduced one when it is denoted by $\mathcal{A} = (\Sigma, R, \gamma, I, F, X, \Delta)$, with each $R[i] = \mathcal{P}^i(Q[0])$ and each γ_i the identity map of $R[i]$.
Similarly, a timed b-automaton $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ is itself a reduced one when it is denoted by $\mathcal{A} = (\Sigma, R, \gamma, I, F, \Delta, X)$ with $R[0] = Q[0]$, $R[1] = Q[1]$ and γ the identity map of $R[1]$.

4.3 Equivalence between transfinite timed automata

A covering lemma. We first give a sufficient condition for two timed automata to recognize the same language. To make the presentation shorter, we consider the tuple $\gamma = (\gamma_k)_{1 \leq k \leq n}$, for a c-n-automaton, as a mapping partially defined on $\mathcal{P}(Q)$: $\gamma(\{q_1, ..., q_p\})$ is only defined whenever $q_1, ..., q_p$ are states of the same type $k < n$ and is then equal to $\gamma^k(\{q_1, ..., q_p\})$.

Let $\mathcal{A} = (\Sigma, Q, \gamma, I, F, \Delta, X)$ and $\mathcal{B} = (\Sigma, S, \eta, J, G, \Omega, X)$ be reduced timed o-automata. As usual, we say that $\mathcal{A}$ covers $\mathcal{B}$ if there exists a subset U (set of useful states) of Q containing all states appearing in runs from $\mathcal{A}$ beginning in I and a map $\hat{c}$ (covering) from Q to S such that :

- $\hat{c}$ maps $\Delta(u, a)$ (the set of transitions from state u with label a) to $\Omega(\hat{c}(u), a)$ for each u in U and a in Σ
- $\hat{c}$ is *continuous*: $\hat{c}(\gamma(\{q_1, ..., q_p\})) = \eta(\{\hat{c}(q_1), ..., \hat{c}(q_p)\})$, for any subset $\{q_1, ..., q_p\}$ of $Q[0] \cap U$ if $\mathcal{A}$ is a b-automaton, for any subset $\{q_1, ..., q_p\}$ of $(Q[0] \cup Q[1] \cup ... \cup Q[n-1]) \cap U$ if $\mathcal{A}$ is a c-n-automaton.
- $\hat{c}(I) = J$
- $F = \hat{c}^{-1}(G)$

Lemma 11. *If $\mathcal{A}$ covers $\mathcal{B}$, then $L(\mathcal{A}) = L(\mathcal{B})$.*

Proof. Let $(q_0, X(t_0)) \xrightarrow{A_0, a_0, \rho_0} (q_1, X(t_1)) \dots (q_\alpha, X(t_\alpha))$ be a run in $\mathcal{A}$. Then all states q_i are in U and $(\hat{c}(q_0), X(t_0)) \xrightarrow{A_0, a_0, \rho_0} (\hat{c}(q_1), X(t_1)) \dots (\hat{c}(q_\alpha), X(t_\alpha))$ is a run in $\mathcal{B}$ with the same label. The hypothesis made on $\hat{c}$ ensures that this map sends the set of accepted runs in $\mathcal{A}$ onto the set of accepted runs in $\mathcal{B}$, which shows that $L(\mathcal{A}) \subset L(\mathcal{B})$.
Conversely, let $(s_0, X(t_0)) \xrightarrow{A_0, a_0, \rho_0} (s_1, X(t_1)) \dots (s_\alpha, X(t_\alpha))$ be a run in $\mathcal{B}$. Since $\hat{c}(I) = J$, there exists $q_0 \in I$ such that $\hat{c}(q_0) = s_0$. Let β be any ordinal strictly smaller than α. Suppose we have $(q_i)_{i<\beta}$ in U such that $\hat{c}(q_i) = s_i$ and $(q_0, X(t_0)) \xrightarrow{A_0, a_0, \rho_0} (q_1, X(t_1)) \dots (q_i, X(t_i))$ is a run in $\mathcal{A}$, $\forall i < \beta$.
Suppose we consider c-automata.

- If β is a limit ordinal of type k, written $\beta = j + \omega^k$.
  Let $q_\beta = \gamma_k(lim\{q_{j+\omega^{k-1}p} \mid p \in \mathbb{N}\})$. Then thanks to the hypothesis of continuity for $\hat{c}$, we have $\hat{c}(q_\beta) = \eta_k(lim\{s_{j+\omega^{k-1}p} \mid p \in \mathbb{N}\}) = s_\beta$.
- If β is a successor ordinal, then $\hat{c}$ maps $\Delta(q_{\beta-1}, a_{\beta-1})$ on $\Omega(s_{\beta-1}, a_{\beta-1})$. Thus there exists q_β in U such that $\hat{c}(q_\beta) = s_\beta$ and $(q_{\beta-1}, A_{\beta-1}, a_{\beta-1}, \rho_{\beta-1}, q_\beta) \in \Delta(q_{\beta-1}, a_{\beta-1})$.

Therefore, by transfinite induction, we define a run in $\mathcal{A}$: $(q_0, X(t_0)) \xrightarrow{A_0, a_0, \rho_0} (q_1, X(t_1)) \dots (q_\alpha, X(t_\alpha))$. Since q_0 has been chosen in I and $q_\alpha \in F = \hat{c}^{-1}(G)$, it is an accepted run.
The proof is similar in the Büchi case. $\square$

Simulation properties between timed automata. Using Lemma 11 above, we now proceed to prove that all the notions of transfinite timed automata presented above have the same expressive power.

Proposition 12. *A reduced timed c- (respectively b-) automaton is covered by a timed c- (respectively b-) automaton.*

Proof. If η is a mapping from a set Q to a set S, we write $[\eta]$ the mapping from $\mathcal{P}(Q)$ to $\mathcal{P}(S)$ defined for $\mathbf{q}$ in $\mathcal{P}(Q)$ by $[\eta](\mathbf{q}) = \{\eta(q); q \in \mathbf{q}\}$ and inductively $[\eta]^{i+1} = [[\eta]^i]$.

Let $\mathcal{A} = (\Sigma, S, \gamma, I, F, \Delta, X)$ be a reduced c-n-automaton (respectively a reduced b-automaton). Let m take the value n in Choueka's case and 1 in Büchi's case. Define a mapping $\hat{c}$ from $\mathcal{P}^{\leq m}(S[0])$ onto S by $\hat{c} = \gamma_i \circ [\gamma_{i-1}] \circ \cdots \circ [\gamma_1]^i$ from $\mathcal{P}^i(S[0])$ onto $S[i]$ for $i \in \{0, ..., m\}$. Let $\mathcal{B} = (\Sigma, Q, I, F', \Delta', X)$ be the timed c-n-automaton (respectively b-automaton) defined by $Q[0] = S[0]$, $\Delta' = \{(q, A, a, \rho, s) \mid q \in Q, s \in S[0], (\hat{c}(q), A, a, \rho, s) \in \Delta\}$ and $F' = \hat{c}^{-1}(F)$. Then $\hat{c}$ is a covering from $\mathcal{B}$ to $\mathcal{A}$. □

Proposition 13. *Any c-n-automaton is covered by a b-automaton.*

Proof. The proof is the same as in the untimed case (see [4]).
Let $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ be a c-n-automaton. We define a b-automaton $\mathcal{B} = (\Sigma, S, J, G, \Omega, X)$ (on the same alphabet and same set of clocks) and a covering $\hat{c}$ from $\mathcal{B}$ onto $\mathcal{A}$ by :

- $S[0] = Q[0] \cup ((\cup_{1 \leq i \leq n} \mathcal{P}^i(Q[0])) \times Q[0])$
  The *type* of a state $\mathbf{s}$ in $\mathcal{P}(S[0])$ is the smallest integer i in $\{1, \ldots, n\}$ such that $\mathbf{s} \in \mathcal{P}(Q[0] \cup (\mathcal{P}^{\leq i-1}(Q[0]) \times Q[0]))$
- The map $\hat{c}$ from S into Q is defined case by case:
  If $q \in Q[0]$, then $\hat{c}(q) = q$.
  If $(\mathbf{q}, q) \in (\cup_{1 \leq i \leq n} \mathcal{P}^i(Q[0])) \times Q[0]$, then $\hat{c}((\mathbf{q}, q)) = q$.
  If $\mathbf{s} \in \mathcal{P}(S[0])$ is of type i, then
  $\hat{c}(\mathbf{s}) = \{\mathbf{q} \in \mathcal{P}^{i-1}(Q[0]) \mid \exists q \in Q[0] \text{ such that } (\mathbf{q}, q) \in \mathbf{s}\}$.
- $J = I$
- $G = \hat{c}^{-1}(F)$
- Let $\delta = (q, A, a, \rho, q')$ be a transition in Δ.
  If $q, q' \in Q[0]$, then $\delta \in \Omega$
  If $q, q' \in Q[0]$ and $\mathbf{q} \in \cup_{1 \leq i \leq n} \mathcal{P}^i(Q[0])$, then $((\mathbf{q}, q), A, a, \alpha, q') \in \Omega$
  If $q \in \cup_{1 \leq i \leq n} \mathcal{P}^i(Q[0])$, then $\forall s \in \hat{c}^{-1}(q)$, $(s, A, a, \rho, (q, q')) \in \Omega$.

□

Proposition 14. *Any b-automaton is covered by a reduced c-n-automaton.*

Proof. Again, the proof is as in the untimed case (see [4]).
Let $\mathcal{B} = (\Sigma, S, J, G, \Omega, X)$ be a b-automaton and n the cardinal of $S[0]$. We define a reduced c-n-automaton $\mathcal{A} = (\Sigma, Q, \gamma, I, F, \Delta, X)$ (on the same alphabet and same set of clocks) and a covering $\hat{c}$ from $\mathcal{A}$ to $\mathcal{B}$ by :

- $Q[0] = S[0] \times (\mathcal{P}(S[0]))^{n-1}$
- $Q[i] = \mathcal{P}(S[0]) \times (\mathcal{P}(S[0]))^{n-i}$, for $i \in \{1, ..., n\}$
- The map $\hat{c}$ from Q into S is defined for $i \in \{0, ..., n\}$ and q in $Q[i]$ by:
  If $i = 0$, $q = (s, \mathbf{s}_1, ..., \mathbf{s}_{n-1})$ in $Q[0]$, then $\hat{c}(q) = s$
  If $i > 0$, $q = (\mathbf{s}, \mathbf{s}_i ..., \mathbf{s}_{n-1})$ in $Q[i]$, then $\hat{c}(q) = \mathbf{s}$
- If $\{q^1, ..., q^r\} \in \mathcal{P}(Q[i])$ with $q^j = (\mathbf{s}^j, \mathbf{s}^j_i, \ldots, \mathbf{s}^j_{n-1})$ then

If for $k > i$, all $\mathbf{s}_k^j$'s are equal to a same $\mathbf{s}_k$,
then $\gamma(\{q^1, \ldots, q^r\}) = (\cup_j \mathbf{s}_i^j, \mathbf{s}_{i+1}, \ldots, \mathbf{s}_{n-1})$
Otherwise, $\gamma(\{q^1, \ldots, q^r\})$ is undefined

- $I = J \times \{\emptyset\}^{n-1}$
- $G = \hat{c}^{-1}(F)$.
- If $\delta = (s, A, a, \rho, s') \in \Omega$, then
  if $s \in S[0]$, then for any $\mathbf{q}_1, ..., \mathbf{q}_n$ in $\mathcal{P}(S[0])$,
  $((s, \mathbf{q}_1, ..., \mathbf{q}_{n-1}), A, a, \rho, (s', \mathbf{q}_1 \cup \{s\}, ..., \mathbf{q}_{n-1} \cup \{s\})) \in \Delta$
  if $s \in \mathcal{P}(S[0])$, then for any $i > 0$ and $\mathbf{q}_i, ..., \mathbf{q}_{n-1}$ in $\mathcal{P}(S[0])$,
  $((s, \mathbf{q}_i, ..., \mathbf{q}_{n-1}), A, a, \rho, (s', \{s\}, \ldots, \{s\}, \mathbf{q}_i \cup \{s\}, ..., \mathbf{q}_{n-1} \cup \{s\})) \in \Delta$

Consider a run of length $\sum_{j=n-1}^{i} \omega^j.n_j$ entering the state $(\mathbf{s}, \mathbf{s}_{i+1} ..., \mathbf{s}_n)$ in $\mathcal{A}$ and the corresponding run, via $\hat{c}$, in $\mathcal{B}$. By construction, $\mathbf{s}_i$ is the set of states visited by the last ω^i executions and, for k in $\{i+1, ..., n-1\}$, $\mathbf{s}_k$ is the set of states visited by the last $\sum_{j=k}^{i} \omega^j.n_j$ executions of the run in $\mathcal{B}$. This shows the continuity of $\hat{c}$ by induction on the length of the run. □

Timed state-reset automata. A timed o-automaton is *state-reset* if all transitions entering a given state (either a type 0 or a meta-state) reset the same set of clocks.

Lemma 15. *Let $\mathcal{A}$ be a timed automaton. There exists a state-reset timed automaton $\mathcal{B}$, accepting the same language.*

Proof. For a b-automaton, as for a Muller automaton, it suffices to split each state into several copies according to the set of clocks reset by the transitions entering it, as it is in done in [3]. Note that it would not be enough for a c-automaton. Nevertheless, the proof of proposition 14 provides a state-reset timed c-automaton from a state-reset timed b-automaton. □

Remark 16. This transformation will be useful in some of the following constructions. Indeed, the information needed when reaching a meta-state is the set of infinitely repeated transitions, which is more precise than only the set of infinitely repeated states. A definition of meta-states based on transitions instead of states would allow to get rid of lemma 15. However, we chose to stay in the usual framework.

5 Properties of the class *TL*

5.1 Closure properties

In the rest of the paper, we denote by $TL(n)$ the family of timed languages accepted by timed n-automata, for each $n \geq 1$, and we write $TL = \cup_{n \geq 1} TL(n)$.

Like the class $TL(1)$ of timed automata, the class TL is closed under union and intersection, but not under complement. The interesting new fact about using Zeno words is the closure under concatenation, as well as under star and ω-power operations:

Theorem 17. *The family TL is closed under union, intersection, concatenation, star and ω-power.*

Proof. Let L and L' be two timed languages accepted respectively by a timed b-n-automaton $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ and a timed b-n'-automaton $\mathcal{A}' = (\Sigma', Q', I', F', \Delta', X')$.

1. The closure under (disjoint) union is straightforward, because the automata are non deterministic, and yields a timed b-$max(n, n')$-automaton.
2. In a similar way, a timed b-$min(n, n')$-automaton accepting the intersection of L and L' is obtained as in the untimed case by a cartesian product of $\mathcal{A}$ and $\mathcal{A}'$. On a synchronized transition, the constraint is the conjunction of the two original constraints and the set of clocks to be reset is the union of the two original sets.
3. The concatenation LL' is accepted by a timed b-h-automaton, where $h = n'$ if $n < n'$ and $h = n + 1$ otherwise. The new automaton is built in the following way: any transition (q_1, A, a, ρ, q_2) from $\mathcal{A}$ is replaced by $(q_1, A, a, \rho \cup X', q_2)$ and the transitions of $\mathcal{A}'$ stay unchanged. Moreover, if q' is a state of Q' such that there is a transition (q'_0, A, a, ρ, q') in $\mathcal{A}'$, with $q'_0 \in I'$, then for each final state q of $\mathcal{A}$, the transition (q, A, a, ρ, q') is added. The set initial states is I, while the set of final states is F' if $I' \cap F' = \emptyset$ and $F \cup F'$ otherwise.
4. For the star and the ω-power operations, we use two copies of $\mathcal{A}$, and reset all clocks of one copy in the other one. Added to the transitions of $\mathcal{A}$ in each copy, each transition leaving a starting state in one copy is doubled from all final states of the other copy. Hence, a run in this automaton may be decomposed into a concatenation of runs in $\mathcal{A}$ in a unique way, just by looking at the transitions from one copy to the other. Precisely, we define a b-automaton $\mathcal{B} = (\Sigma, S, J, G, \Omega, Y)$ by:

- $S[0] = Q[0] \times \{0, 1\}$ (and $S[1] = \mathcal{P}(S[0])$)
- $J = I \times \{0, 1\}$
- $Y = X_0 \cup X_1$ consists of two disjoint copies of X, namely X_0 and X_1. Any constraint B on Y may be considered as a conjunction $B_0 \wedge B_1$, where B_0 (respectively B_1) is a constraint on X identified to X_0 (respectively X_1). In the same way, any subset ρ of Y may be considered as a disjoint union $\rho_0 \cup \rho_1$ where ρ_0 (respectively ρ_1) is a subset of X identified to X_0 (respectively X_1).
- The transitions in Ω are defined as follows. Let i be in $\{0, 1\}$ and let j be such that $\{i, j\} = \{0, 1\}$.
  - Let q be in $Q[0]$.
    - For each $(q, A, a, \lambda, q') \in \Delta$, $((q, i), B, a, \rho, (q', i)) \in \Omega$ with $B_i = A$, $B_j = \text{True}$, $\rho_i = \lambda$ and $\rho_j = X$.
    - If $q \in F$, $q_0 \in I$ and $(q_0, A, a, \lambda, q') \in \Delta$, $((q, i), B, a, \rho, (q', j)) \in \Omega$ with $B_i = \text{True}$, $B_j = A$, $\rho_i = X$ and $\rho_j = \lambda$.
  - Let $\mathbf{q} = \{q_1, ..., q_m\}$ be in $Q[1] = \mathcal{P}(Q[0])$. and $\mathbf{s} = \{(q_1, i)..., (q_m, i)\}$ be in $S[1]$.
    - For each $(\mathbf{q}, A, a, \lambda, q') \in \Delta$, $(\mathbf{s}, B, a, \rho, (q', i)) \in \Omega$ with $B_i = A$, $B_j = \text{True}$, $\rho_i = \lambda$ and $\rho_j = X$.

- $*$ If $\mathbf{q} \in F$, $q_0 \in I$ and $(q_0, A, a, \lambda, q') \in \Delta$, $(\mathbf{s}, B, a, \rho, (q', j)) \in \Omega$ with $B_i = \text{True}$, $B_j = A$, $\rho_i = X$ and $\rho_j = \lambda$.

– We finally define the set G of final states.
  - For

$$G = (F \cap Q[0]) \times \{0,1\}$$
$$\cup \{\{(q_1, i)..., (q_m, i)\} \in S[1] \mid \{q_1, ..., q_m\} \in F \text{ and } i \in \{0,1\}\},$$

    the language accepted by $\mathcal{B}$ is exactly L^+. To add the empty word, we simply add a new starting and accepting state.
  - If L does not contain the empty word and $G = \{\{(q_1, i_1)..., (q_m, i_m)\} \in S[1] \mid \{q_1, ..., q_m\} \in F \text{ and } \{i_1, ... i_m\} = \{0,1\}\}$, then the language accepted by this automaton is exactly L^ω.
  - If L contains the empty word, we have to take $G = (F \cap Q[0]) \times \{0,1\} \cup \{\{(q_1, i_1)..., (q_m, i_m)\} \in S[1] \mid \{q_1, ..., q_m\} \in F\}$ and add a new starting and accepting state.

□

5.2 Decidability of emptiness

In the evaluation of a model, the test for emptiness is an important question. Indeed, the design of verification algorithms becomes possible with a positive answer. Thus, theorem 18 shows that our extension of timed automata retains the property of the original class.

Theorem 18. *Emptiness is decidable in the class TL.*

The proof follows the standard technique introduced in [1] and extended in [5]. From a timed language L, the (untimed) language $Untime(L)$ is obtained by removing the dates of actions, with the property that $Untime(L)$ is empty if and only if L is empty. From a timed automaton $\mathcal{A}$ accepting L, we must then build an untimed automaton $\mathcal{A}'$, called a *region automaton*, accepting $Untime(L)$. The result follows from the decidability of emptiness for the corresponding class of untimed languages [9]. The proof builds a reduced untimed b-automaton $\mathcal{A}'$ from a timed b-automaton $\mathcal{A}$ and proceeds in four main steps: removing useless divergent executions (step 1), defining limit zones and the reduced b-automaton (steps 2 and 3) and proving the correctness of the construction (step 4).

Step 1: On Time Divergence. The first step consists in removing executions which arrive on a non-accepting state at infinite time, thus cannot be pursued. This is done by next lemma.

Lemma 19. *Let $\mathcal{A}$ be a timed o-automaton. There is a timed o-automaton $\mathcal{A}'$ accepting the same language, with the following property called On Time Divergence (OTD): if q is a meta-state of $\mathcal{A}'$ with some outgoing transition, then, for any run of $\mathcal{A}'$ entering q, the corresponding sequence of time is convergent.*

Note that On Time Divergence is true for a timed 1-automaton, if there is no transition going out of type 1 states. Moreover, the *OTD* property is preserved by the transformations between o-automata, so that we chose to give the construction with c-automata, where it is more explicit.

Proof. We use several copies of $\mathcal{A}$ and add enough clocks to make sure that all executions with infinite ending time arrive on a dead state.

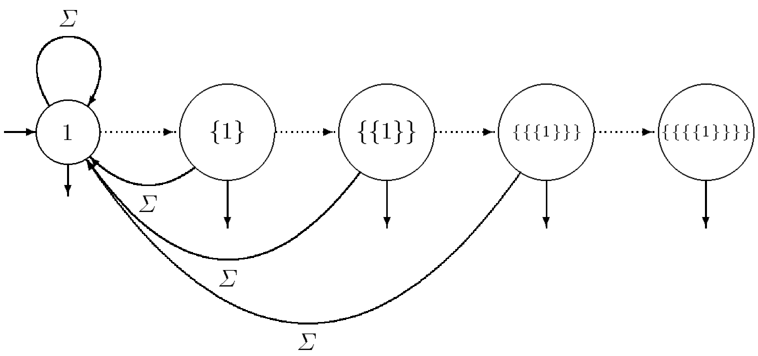

Fig. 4. The automaton $\mathcal{B}_4$

The construction is illustrated (for $n = 4$) in figures 4 and 5 for the simple case of a timed n-automaton $\mathcal{B}_n$, accepting all timed words with length less than or equal to ω^n, producing $\mathcal{B}'_n$. For the sake of readability, the labels Σ of transitions have been omitted in figure 5, and the last level is not completely represented. In order to ensure On Time Divergence, the automaton $\mathcal{B}'_n = (\Sigma, Q, I, F, \Delta, X)$ has n states of type 0, $Q[0] = \{1, \dots, n\}$, and $n - 1$ clocks $X = \{x_2, \dots, x_n\}$. Again, the initial state is 1 and all states are final. With each state k in $Q[0]$ are associated k meta-states $\{k\}$, $\{\{k\}\}$, ... (with respective types $1, \dots, k$) and the clock x_k. Meta-states of type less than k produce only convergent time sequences reaching the ordinal ω^{k-1}, while the dead final meta-state of type k accepts words with a possibly divergent sequence of length ω^k (but all the subsequences of length ω^{k-1} are convergent).

The automaton $\mathcal{A}'$ can then be obtained either in a similar way, or by the classical construction of intersection for $\mathcal{A}$ and $\mathcal{B}'_n$, thus ensuring that the OTD property also holds for $\mathcal{A}'$. □

Step 2: Definition of limit zones. We now assume $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ is a timed state-reset b-automaton, for which the *OTD* property of first step holds, and we briefly recall the notion of *regions*, introduced in [2].

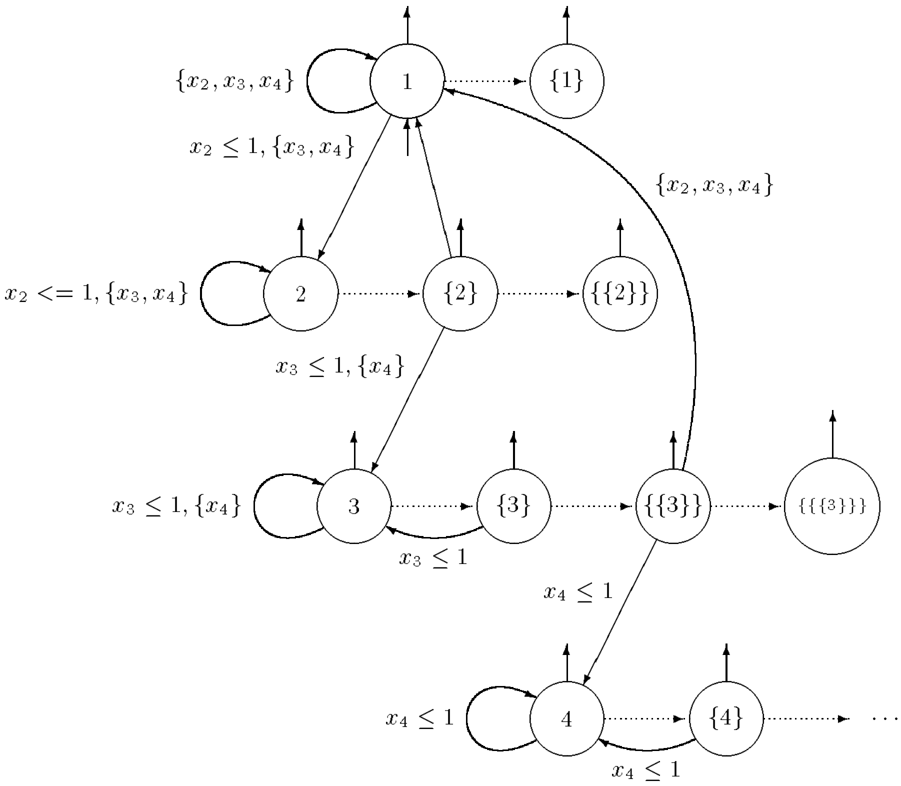

Fig. 5. The automaton $\mathcal{B}'_4$

Let p be the cardinal of X and let M be the largest constant appearing in the clock constraints of $\mathcal{A}$. A *region* is a subset of $\mathbb{R}_+^p$ (the set of all clock values) described by specifying:

1. for each clock x, one clock constraint from the set $\{x = c \mid c = 0, 1, \ldots, M\} \cup \{c - 1 < x < c \mid c = 1, \ldots, M\} \cup \{x > M\}$
2. for every pair of clocks x and y such that $c - 1 < x < c$ and $d - 1 < y < d$ appear in 1. for some c, d, an ordering of the fractional parts, i.e. wether $fract(x)$ is less than, equal to, or greater than $fract(y)$.

The set of regions, denoted by $\mathcal{V}$, form a finite partition of $\mathbb{R}_+^p$, associated with an equivalence relation (see [2]). The region containing point $z = (z_i)_{1 \leq i \leq p}$ of $\mathbb{R}_+^p$ (its equivalence class) is written $[z]$. In particular, the region $[0] = \wedge_{x \in X}(x = 0)$ is reduced to the origin point. With these notations, any clock constraint can be expressed as a (finite) disjunction of regions. To reflect the geometric point of view, we call *zone* a finite union of regions and write $\mathcal{Z}$ for the (finite) set of zones.

A *time-successor* relation between regions (corresponding to the iterated *Post* function) is defined by: $v \leq v'$ if for each point $z = (z_i)_{1 \leq i \leq p}$ in v, there is a real number $t \geq 0$ such that $z' = (z_i + t)_{1 \leq i \leq p}$ belongs to v'.

In order to build (step 3) a reduced untimed b-automaton from $\mathcal{A}$, we will define new states of type 0 (like in [2]) as pairs of the form (q, v), consisting of a state q in $\mathcal{A}$ and a region v in $\mathcal{V}$. Now, the main problem to adapt the classical construction to b-automata is to define in a consistent way the reachable states of type 1 and the mapping γ. For this, we will associate with a limit set $\mathbf{s} = \{(q_1, v_1), \cdots, (q_m, v_m)\}$ of type 0 states, a pair of the form $(\mathbf{q}, Z)$, where the untimed component is the meta-state $\mathbf{q} = \{q_1, \ldots, q_m\}$ and Z is the zone effectively reached by the runs along the cycle. This zone, which will be denoted by $lim(\{v_1, \ldots, v_m\})$, must be a subset of the intersection of the closures of the v_j's, which is itself a union of regions containing the limits of sequences of clock values.

From the automaton $\mathcal{A}$, it is possible to know the subset X_0 of clocks, which are reset infinitely often in any execution r for which the limit set is $\mathbf{s}$. For a clock in X_0, we already know that the limit value is 0. The clocks in $X \setminus X_0$ are not reset any more from a certain time on. For such a clock x, we consider the conjunction of all the constraints appearing in the v_j's. There are three cases, defining three additional subsets of clocks:

- if the resulting constraint is of the form $x = c_x$, $c_x \geq 0$, then $x \in X_s$, where the subscript s stands for "single value",
- if it is of the form $c_x < x < d_x$, with $0 \leq c_x$ and d_x is either $c_x + 1$ or ∞, then $x \in X_{oi}$, where oi means "open interval",
- if it reduces to $false$, then $x \in X_e$, where e is for "empty".

First note that if the subset X_e is not empty (i.e. for some clock, one of the constraints is $false$), then no type 1 state can be reached. In this case, the limit zone will simply be the empty set. Otherwise, if X_s is not empty, time does not progress along the cycle so that the limit is the region $v_1 \cap \ldots \cap v_m$. The only difficult case occurs when X_s is empty and X_{oi} is not: if some open intervals have a finite upper bound, this bound can be reached at the limit. However, only the bounds d_x corresponding to clocks x with maximal fractional parts are reached and again from this point on, time does not progress any more, so that the remaining intervals stay open. This explains the following definition.

Definition 20. Let $\{v_1, \cdots, v_m\}$ be a set of regions with the corresponding partition of the set of clocks $X = X_0 \cup X_s \cup X_{oi} \cup X_e$ introduced above. We consider four basic zones (with the usual convention: $\wedge_{x \in Y} A_x$ is $true$ if Y is empty).

- $Z_0 = \bigwedge_{x \in X_0} (x = 0)$,
- $Z_s = \bigwedge_{x \in X_s} (x = a_x)$,
- $Z_{oi} = \bigwedge_{1 \leq j \leq m} (v_{j\,|X_{oi}})$ is the open region obtained by the intersection of the restrictions to X_{oi} of the v_j's.
  The region Z_{oi} yields an ordering on the fractional parts of the clocks in X_{oi}, and we consider the subset X_m of X_{oi}, containing the clocks x such that (i) $fract(x)$ is maximal in this ordering and (ii) d_x is finite. Then, the last zone is:
- $Z_m = \bigwedge_{x \in X_m} (x = d_x)$.

The limit zone $Z = lim(\{v_1, \cdots, v_m\})$ is then defined by:

1. If X_e is not empty, then $Z = \emptyset$.
   Otherwise, there are three cases.
2. If X_s is not empty, then $Z = Z_0 \wedge Z_s \wedge Z_{oi}$. In the particular case where X_{oi} is empty, $Z = Z_0 \wedge Z_s$ is reduced to a single point.
3. If $X_s = \emptyset$, then there are two subcases:
   If X_{oi} is empty, then $Z = [0]$ is reduced to the origin point.
   If X_{oi} is not empty, then $Z = Z_0 \wedge Z_s \wedge (Z_{oi} \vee Z_m)$.

Example. We give two examples for the last subcase ($X_s = \emptyset$ and $X_{oi} \neq \emptyset$).
First suppose $X_0 = \{x\}$ and $\mathbf{s} = \{(1, x = 0 \wedge 0 < y < 1)\}$ contains a single type 0 state. Then $X_{oi} = \{y\}$ and for some sequences, increasing values of y (which is not reset any more) may reach the value 1. In this case,
$lim(\{x = 0 \wedge 0 < y < 1\}) = (x = 0 \wedge (0 < y < 1 \vee y = 1)$
is the union of the two regions $(x = 0 \wedge 0 < y < 1)$ and $(x = 0 \wedge y = 1)$, whereas the closure of the initial region contains in addition the origin point $(x = 0 \wedge y = 0)$ which cannot be reached.

Now we look at a case where two clocks evolve in an open region. Take $X_0 = \emptyset$ and $\mathbf{s} = \{(1, v)\}$, with $v = (0 < x < 1) \wedge (0 < y < 1) \wedge (fract(x) > fract(y))$. We have $X_{oi} = \{x, y\}$, $Z_{oi} = v$, $X_m = \{x\}$ and $Z_m = (x = 1)$, because $fract(x)$ is maximal and $d_x = 1$ is reached first. The limit zone is:
$Z = lim(\{v\}) = (0 < x \leq 1) \wedge (0 < y < 1) \wedge (fract(x) > fract(y))$.

Step 3: Construction of the region automaton. We finally define from $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ the reduced (untimed) b-automaton $\mathcal{A}' = (\Sigma, S, \gamma, I', F', \Delta')$ in the following way:

- $S = S[0] \cup S[1]$, where $S[0] = Q \times \mathcal{V}$ and $S[1] = \mathcal{P}(Q) \times \mathcal{Z}$,
- $I' = I \times \{[0]\}$ and $F' = F \times \mathcal{Z}$
- The set Δ' of transitions is a subset of $S \times \Sigma \times S[0]$ defined as follows. For $a \in \Sigma$ and $s = (q, Z) \in S$, with $Z \neq \emptyset$, write $Z = V_1 \cup \ldots \cup V_k$ as a finite union of regions. A tuple $((q, Z), a, (q', V'))$ is in Δ' if there exists a transition (q, A, a, ρ, q') in Δ, a time-successor V'' of one of the V_j's, satisfying A (i.e. contained in A), such that V' is obtained from V'' by the reset of the clocks in ρ.
- The mapping γ is defined for a meta-state $\mathbf{s} = \{(q_1, v_1), \cdots, (q_m, v_m)\}$ by $\gamma(\mathbf{s}) = (\{q_1 \cdots, q_m\}, lim(\{v_1, \cdots, v_m\}))$

Note that, like $\mathcal{A}$, the automaton $\mathcal{A}'$ is state-reset.
The following lemma ensures the consistency in the definition of limit zone:

Lemma 21. *Let $\mathcal{A}$ be a timed state-reset b-automaton and $P = (q_0, A_1, a_1, \rho_1, q_1)$ $(q_1, A_2, a_2, \rho_2, q_2) \ldots$ a path (i.e. a sequence of consecutive transitions) in $\mathcal{A}$ starting from an initial state. Let $\mathbf{q}$ be a meta-state of type 1 reached through P.*

1. *Let x be a clock in X and $c > 0$ a constant, such that for any run r using P, the constraint $x = c$ is satisfied when reaching $\mathbf{q}$. Then the same constraint $x = c$ is satisfied in an infinite number of states of r having $\mathbf{q}$ as limit.*
2. *Let $r = (q_h, X(t_h))_{h \leq \alpha}$ be a run through P in $\mathcal{A}$, with $v_h = [X(t_h)]$ for each state q_h of type 0. If the meta-state $\mathbf{q} = q_i = \{q_{i_1}, \cdots, q_{i_m}\}$ is reached at finite time t_i, then $X(t_i)$ belongs to $Z_i = lim(\{v_{i_1}, \cdots, v_{i_m}\})$.*

Proof. 1. This property is just another formulation of Lemma 27, which is proved later (independently of this result) in Section 7.
2. Since $X(t_i) = lim_{h \to i} X(t_h)$ for the limit ordinal i (lemma 6), with $X(t_h) \in v_h$ for the type 0 states q_h, the proof consists in looking at all the cases in the definition of the limit zone above.

□

Example. Figure 6 shows a timed automaton accepting the language
$L = \{(a^\omega b, (t_i)_{i \leq \omega+1})$ such that $0 < t_0 < t_1 < \ldots < t_\omega \leq t_{\omega+1} < 1\}$
for which $Untime(L) = \{a^\omega b\}$ and figure 7 illustrates the construction of the region automaton accepting $Untime(L)$. As explained in the first example above, the meta-state $(\{1\}, x = 0 \wedge 0 < y \leq 1)$ is the limit state obtained when the (type 0) state $(1, x = 0 \wedge 0 < y < 1)$ is repeated infinitely often. The second type 1 (dead) state $(\{1\}, x = 0 \wedge y > 1)$ also corresponds to the third subcase of the definition, but this time, $d_y = \infty$. Thus, for the region $v' = (x = 0) \wedge (y > 1)$, the limit zone is $Z' = lim(\{v'\}) = v'$.

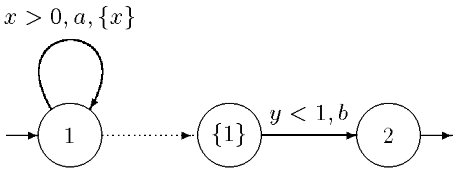

Fig. 6. A timed 2-automaton for L

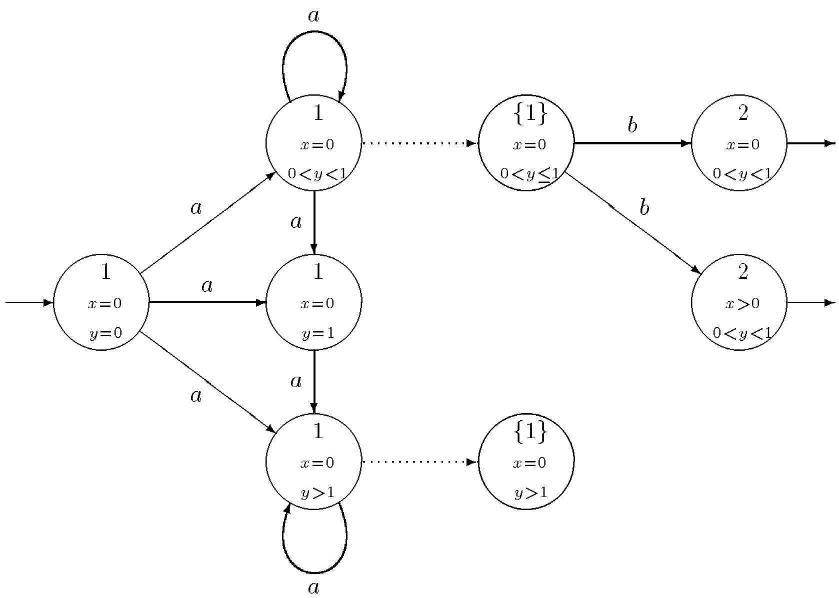

Fig. 7. The region automaton accepting $Untime(L)$

Step 4: proof of theorem 18. Finally, we verify that $Untime(L) = L(\mathcal{A}')$ and the result holds by applying the decidability result of [9].

(i) Let $u = (a_i)_{i<\alpha}$ be a word in $Untime(L)$. There exists a timed word $w = (a_i, t_i)_{1\leq i<\alpha}$ in L and a run $r = (q_i, X(t_i))_{i\leq\alpha}$ in $\mathcal{A}$, accepting w, with the transitions $(q_i, A_{i+1}, a_{i+1}, \rho_{i+1}, q_{i+1})$ in Δ, for each $i < \alpha$. With $Z_h = [X(t_h)]$ for type 0 states and lemma 21, it is easy to build inductively a run $r' = (q_i, Z_i)_{0\leq i\leq\alpha}$ of $\mathcal{A}'$ accepting u, such that $X(t_i) \in Z_i$.
• If i is of type 0, then $((q_i, Z_i), a_{i+1}, (q_{i+1}, Z_{i+1}))$ is in Δ', as usual.
• If $i < \alpha$ is a limit ordinal, then t_i is finite (because of OTD), and the limit zone Z_i (as defined above), can be written as $Z_i = V_1 \cup \ldots \cup V_k$. From the definition of $X(t_{i+1})$, there is a time-successor V of one of the V_j's, satisfying A_{i+1}, such that $[X_{i+1}]$ is obtained from V by the reset of the clocks in ρ_{i+1}. Thus, transition $((q_i, Z_i), a_{i+1}, (q_{i+1}, Z_{i+1}))$ is in Δ', with $X(t_{i+1}) \in Z_{i+1} = [X_{i+1}]$.

(ii) Conversely, let $u = (a_i)_{i<\alpha}$ be a word in $L(\mathcal{A}')$ and let $r' = (q_i, Z_i)_{i\leq\alpha}$ be a run accepting u. From the construction of $\mathcal{A}'$, we have a sequence of transitions $(q_i, A_{i+1}, a_{i+1}, \rho_{i+1}, q_{i+1})$, in $\mathcal{A}$, for all i. Starting from $t_0 = 0$, we build inductively a time sequence $(t_i)_{0\leq i\leq\alpha}$ and a sequence of clock values $X(t_i)_{0\leq i\leq\alpha}$, such that $X(t_i) \in Z_i$, for all i.
• Let $i > 0$ be of type 0 and assume that q_i has been reached at time t_i, with clock values $X(t_i)$ in the region Z_i. There is a time-successor V of Z_i, satisfying A_{i+1}, such that Z_{i+1} is obtained from V by the reset of the clocks in ρ_{i+1}. Using the definition of the time-successor relation, we can find some real number t such that $X(t_i) + t \in V$. In this case, we set $t_{i+1} = t_i + t$ and, for each clock x, $x(t_{i+1}) = x(t_i) + t$ if $x \notin \rho_{i+1}$, $x(t_{i+1}) = 0$ otherwise.
• Let $i < \alpha$ be a limit ordinal. We obtain (from OTD) a finite time $t_i = lim_{h\to i} t_h$ and (from lemma 6) limit values $X(t_i) = lim_{h\to i} X(t_h)$, with $X(t_h) \in Z_h$ by induction hypothesis. From the construction of $\mathcal{A}'$, we have $(q_i, Z_i) = \gamma(\{(q_{i_1}, Z_{i_1}), \cdots, (q_{i_m}, Z_{i_m})\})$, where the Z_{i_j}'s are regions. Again, lemma 21 ensures that $X(t_i)$ belongs to Z_i.
The only difficult point appears when the limit zone Z_i is not reduced to a single region, but contains a closed region V obtained by the limits $X(t_i)$ of increasing sequences of clock values, i.e. $Z_i = Z_0 \wedge Z_s \wedge (Z_{oi} \vee Z_m)$, and there is some clock $x \in X_{oi}$ such that $x(t_i) = d_x$. In this case, maybe no transition from $\mathcal{A}$ is possible from this particular region V, so that we must use point 1 in lemma 21: since the time sequence does not verify $x = d_x$ infinitely often, it is possible to build another partial run for which the maximal values in Z_m are not reached by the maximal clocks in X_{oi}. For this new run, $x(t_i) < d_x$ hence the limit $X(t_i)$ belongs to the (open) region $Z_i = Z_0 \wedge Z_s \wedge Z_{oi}$. Since there is now a transition from $\mathcal{A}$, we can conclude this case as for an ordinal of type 0.

□

6 Timed refinement

6.1 A tool for refinement: generalized transitions

In the framework of untimed automata, the refinement of some action a corresponds to the replacement of each transition labeled by a, by some finite automaton describing the details of the operations performed by a.

When dealing with timed executions, the refinement of an action may also contain time requirements, so that a transition with label a should be replaced by a timed automaton, say $\mathcal{A}_a$. Moreover, as a transition, the automaton $\mathcal{A}_a$ is accessed at some time t, with (possibly non zero) values $X(t)$ of the clocks. Starting from these values, a convergent run of $\mathcal{A}_a$ assigns a new value to each clock. Each execution of such an automaton is thus used as a function on clock values and must therefore behave like a transition. For this reason, we define a subclass, denoted by GT, of timed automata which we shall use in the refinement operations.

Definition 22. A *generalized transition* (in class GT) is a (reduced) timed o-automaton $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ such that:

1. the empty word ε does not belong to $L(\mathcal{A})$, and
2. for each convergent run of $\mathcal{A}$, starting at some time t_0 in a state $(q_0, X(t_0))$ with $q_0 \in I$ and ending in a state $(q_\alpha, X(t_\alpha))$, with $q_\alpha \in F$, for each clock x in X, if x was reset at least once along the run, then its final value is zero.

Remark 23.
- Point 1 is required because we do not allow ε-transitions in timed o-automata. Timed automata, where the empty word ε may label a transition, have been studied recently and proved to be strictly more expressive than the usual model ([6], [11]).
- Point 2 expresses the fact that a run in the timed automaton simulates a transition if the final clock value is either 0 if it was reset, or $x(t_\alpha) = x(t_0) + t_\alpha - t_0$ otherwise.
- Given a timed o-automaton $\mathcal{A}$, it can be decided if $\mathcal{A}$ is a generalized transition, by looking at the region automaton (see e.g. [2] and theorem 18) or at the state-reset automaton obtained by lemma 15.

Examples.

1. A timed o-automaton $\mathcal{A}$ is reset-free ([3]) if (i) $\varepsilon \notin L(\mathcal{A})$ and (ii) clocks are never reset. Clearly, a reset-free automaton is a generalized transition and we denote by RF the corresponding subclass of GT.
2. Another useful subclass of GT is the set of *state-reset cycles*: a state-reset cycle is a state-reset 1-automaton $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ such that $F = \{Q[0]\}$. The final condition is thus a Muller acceptance condition: an accepting run goes infinitely often through each state of type 0 of the automaton. From the definition of state-reset automata, a clock which is reset at least once is reset infinitely often, so that its final value is zero (see lemma 6).

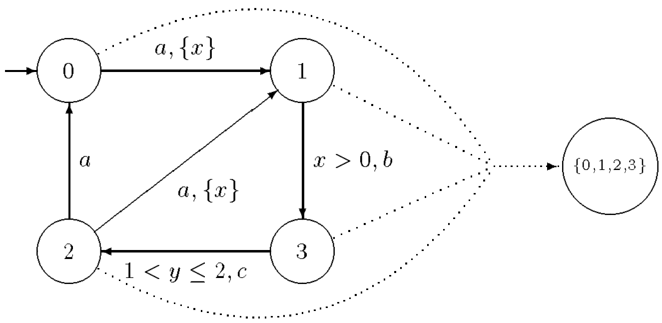

Fig. 8. A state-reset cycle

3. In [7], two particular cases of generalized transitions, based on finite timed automata, were considered.
   - The first one corresponds to instantaneous actions, which must be refined into sequences of actions, all of them occurring at the same time. The replacement is realized by a finite timed automaton, in which all initial transitions reset a single clock x and all subsequent transitions contain the constraint $x = 0$, to ensure a simultaneous execution of the refinement. Point 2 is true in a trivial way because the value of the only clock is permanently equal to zero.
   - The second case, where actions are assumed to have some duration, corresponds to the following time requirement: an action a with duration d, is refined into some sequence $a_1 a_2 \cdots a_p$ with the same global duration, i.e. $\sum_{i=1}^{p} d_i = d$, where d_i is the delay associated with a_i. This time, the replacement is realized by a finite automaton with an empty set of clocks, which makes also Point 2 true.

A generalized transition with a set X of clocks associates with each tuple of initial clock values $X(t)$ the language $L(\mathcal{A})_{X(t)}$, obtained by starting the executions of $\mathcal{A}$ at time t (with clock values $X(t)$). Note that lemma 6 ensures point 2 in definition 22): each run over a Zeno word $w = ((a_i, t_i))_{i<\alpha}$ in the language $L(\mathcal{A})_{X(t)}$ yields final clock values defined, for each $x \in X$, by $x(\tau(w)) = x(t) + \tau(w) - t$ if x has not been reset and $x(\tau(w)) = 0$ otherwise (recall that $\tau(w) = t_\alpha$ is the ending time of w). For a Zeno word $w \in L(\mathcal{A})_{X(t)}$, we write $\mathcal{X}(w, X(t))$ for the finite set of clock values obtained by all runs of $\mathcal{A}$ over w starting from $X(t)$.

6.2 Refinement of recognizable timed languages

When applied to recognizable languages, a good notion of refinement must preserve the property of being recognizable. As in [7], the question we are mainly interested in, is the closure of the class TL under refinement.

Let L be a language in TL over an alphabet Σ and $\mathcal{A}$ a timed n-automaton, with a set X of clocks, accepting L.

Definition 24. A *refinement* of L is given by a family $(\mathcal{A}_a)_{a\in\Sigma}$ of generalized transitions over an alphabet Σ', with X as set of clocks, represented by a mapping $\sigma : \Sigma \longrightarrow GT$ such that $\sigma(a) = \mathcal{A}_a$. The refinement of a timed action (a,t), denoted by $\sigma(a,t)$, is then defined for given clock values $X(t_0)$, $t \geq t_0$, by:

$$\sigma(a,t)_{X(t_0)} = \{w \in (L(\mathcal{A}_a))_{X(t_0)} \mid \tau(w) = t\}$$

Note that the initial alphabet Σ and the refinement alphabet Σ' are not necessarily disjoint sets. Also observe that, if we want to be able to refine the last timed action (a,t) of a timed word, by a non Zeno timed word, then we must allow its ending time t to be infinite.

In order to define the refinement of a timed word, we introduce an operation denoted by $\circ$, similar to a concatenation between words, and (abusively) called *composition*: if $w = (a_i,t_i)_{i<\alpha}$ and $w' = (a'_i,t'_i)_{i<\alpha'}$, with $\tau(w) \leq t'_0$, then

$$w \circ w' = ((a''_i,t''_i))_{i<\alpha+\alpha'} \text{ with } \begin{cases} (a''_i,t''_i) = (a_i,t_i) & \text{for each } i < \alpha \\ (a''_{\alpha+i},t''_{\alpha+i}) = (a'_i,t'_i) & \text{for each } i < \alpha'. \end{cases}$$

The execution of two consecutive transitions, over (a,t) followed by (b,t'), accepts $(a,t)\circ(b,t')$, if $t' \geq t$. If these transitions are replaced respectively by $\mathcal{A}_a$ and $\mathcal{A}_b$, the corresponding executions, starting from clock values $X(t_0)$, accept what we call the refinement of $(a,t)\circ(b,t')$, defined by:

$$\sigma((a,t)\circ(b,t'))_{X(t_0)} = \{w\circ w' \mid w \in (L(\mathcal{A}_a))_{X(t_0)},\ \tau(w) = t,\ w' \in (L(\mathcal{A}_b))_{X(t)} \text{ for some } X(t) \in \mathcal{X}(w,X(t_0)) \text{ and } \tau(w') = t'\}$$

The operation $\circ$ composes, in fact, runs of the automata $\mathcal{A}_a$ and $\mathcal{A}_b$.
We extend this definition to a timed word $w \in L$, inductively on its length. Finally, the refinement of L is: $\sigma(L) = \cup_{w\in L}\sigma(w)_{X(0)}$. Note that, for a language, initial time is $t_0 = 0$ and initial clock values are all equal to zero: $x(0) = 0$, for each clock $x \in X$.
If L and L' are two languages of transfinite timed words, we say that L' is a refinement of L if there exists a timed refinement σ such that $\sigma(L) = L'$. As already remarked in section 6.1, this definition of refinement generalizes those introduced in [7].

Before proving our main results in next section, we give a closer look at particular cases and their properties and we describe a problem arising in the refinement constructions.

6.3 Basic properties of timed refinement

Timed regular expressions. Besides its interest as a new tool, this notion of refinement allows us to obtain with another point of view the result of [3]: it may be used to find a timed regular expression (up to renaming) for a recognizable timed language. First note that, for the operation $\circ$, we clearly have an analogue of Kleene theorem for the family RFL of timed languages accepted by reset-free automata:

Proposition 25. *For a letter a in Σ and an interval I of time, we consider the timed language $a_I = \{(a,t) \mid t \in I\}$. The class RFL is the smallest family containing the languages a_I, and closed under union, composition ($\circ$) and the star and ω-power operations derived from $\circ$.*

Now, let L be a timed language in $TL(1)$. As in [3], we can write $L = \lambda(\bigcap_{i=1}^{p} L_i)$, where λ is a renaming mapping and each L_i is a timed language accepted by an automaton with only one clock. Moreover, $L_i = \sigma_i(M_i)$ is a refinement of a regular (untimed) language M_i and, for $a \in \Sigma$, the timed automaton $\sigma_i(a)$ resets his single clock only in the final states, which are assumed to be dead states. This result gives a regular expression for L (with renaming), based on two different operations: the usual concatenation and the composition $\circ$.

Refinement of the language a_I. Consider again the language defined above: $a_I = \{(a,t) \mid t \in I\}$, where a is a letter and I an interval of time, and let σ be the mapping sending a to some timed automaton $\mathcal{A}_a$ over an alphabet Σ, with $L_a = L(\mathcal{A}_a)$. Then, the refinement of a_I is the set of words in L_a with ending time in I:

$$\sigma(a_I) = L_a \cap \{w \mid w \text{ is a timed word over } \Sigma \text{ such that } \tau(w) \in I\}$$

It is easy to see that $\{w \mid w \text{ is finite and } \tau(w) \in I\}$ is recognizable. Hence, if L_a contains only finite words, then $\sigma(a_I)$ is also recognizable.
However, this is not the case when L_a contains infinite words because $\Sigma_I^{inf} = \{w \mid w \text{ is infinite and } \tau(w) \in I\}$ is not recognizable in general. In fact, lemma 26 and 27 below give precise results.

Lemma 26. *Let L be a timed language in TL and let I be an interval of the form $[0, c_2]$, $]c_1, \infty[$ or $]c_1, c_2]$, for real numbers c_1, c_2 such that $0 \le c_1 < c_2$. Then $L_I = \{w \in L \mid \tau(w) \in I\}$ is in TL.*

Proof.
- To obtain an automaton accepting $L_{[0,c_2]}$, we start from an automaton accepting L and add a new clock x, with the additional constraint $x \le c_2$ on each transition.
- To obtain an automaton accepting $L_{]c_1,\infty[}$, we consider two distinct copies of an automaton accepting L (but on the same set of clocks) and a new clock x. All transitions of the first copy are also sent in the second one with the addition of the clock constraint $x > c_1$. The new set of initial states is the set of initial states of the first copy and the new set of final states is the set of final states of the second copy.
- Finally, $L_{]c_1,c_2]} = L_{[0,c_2]} \cap L_{]c_1,\infty[}$.

□

Lemma 27. *The language $\Sigma_I^{inf} = \{w \mid w \text{ is infinite and } \tau(w) \in I\}$ is not accepted by a timed automaton if the interval I is of the form $[c_1, c_2[$, $\{c_2\}$ or $[c_2, \infty[$, for real numbers c_1, c_2 such that $0 \le c_1 < c_2$.*

Proof. It is proved in [11] that Σ_I^{inf} cannot be accepted by a timed automaton, when $I = [0, c_2[$. It remains now to prove that the language $\Sigma_{\{c\}}^{inf}$ is not recognizable for any positive real number c. Assume there is a Muller timed automaton $\mathcal{A}$ accepting this language. Then, there exists a path in $\mathcal{A}$ accepting a timed word with a stricly increasing time sequence $(t_i)_{i<\omega}$ convergent to c:

$$(q_0, X(t_0)) \xrightarrow[t_0]{A_0, a_0, \rho_0} (q_1, X(t_1)) \xrightarrow[t_1]{A_1, a_1, \rho_1} (q_2, X(t_2)) \rightarrow \quad \cdots \qquad (q_\omega, X(t_\omega))$$

After a finite number of steps, say k, all transitions occur infinitely often along this path.

- Let x be a clock reset after this k^{th} step. Then, it is reset infinitely often along the path. The constraints verified by x after the k^{th} step cannot be of the form $x > a > 0$ because the run would not accept a Zeno word, neither $x = 0$ because the run would not accept a strictly increasing sequence, so they must be of the form $x > 0$, or $x < a$, or $x \leq a$ or conjunctions of these. Anyway, they are verified by any strictly increasing convergent sequence of times, after a while.
- Let x be a clock which is not reset after this k^{th} step, so it is not reset anymore along the rest of the path. All constraints verified by x after this k^{th} step must accept an infinite number of the values $x(t_k)+t_i-t_k$. Therefore, they must be conjunctions of constraints of the form $x \in J$ where J is an interval containing an open interval $]x(t_k)+b-t_k\,, x(t_k)+c-t_k[$ with $b < c$. Since there are only a finite numbers of transitions and a finite number of clocks involved, we may even suppose that b is independent of x and of the transitions considered.

Thus, the automaton accepts on the same path timed words w such that $\tau(w) < c$ ($\tau(w) = (b + c)/2$ for instance), which is a contradiction. $\square$

7 The main results about timed refinement

We now give a positive answer for the closure of TL under refinement: starting from a suitable Muller timed automaton and replacing recursively some transitions by Muller timed automata in GT, we obtain a Choueka timed automaton. Therefore the refinement, when defined, preserves the recognizability of timed languages. Moreover, each class $TL(n)$ is closed under refinements using only finite timed automata in GT, so the notion of refinement proposed here is interesting in the usual class of timed languages $TL(1)$. We first give in section 7.1 the proof for closure under refinement by 1-automata. The second result (section 7.2) indicates a property of decomposition: any language in $TL(n+1)$ is obtained from a language (in $TL(1)$) accepted by a Muller timed automaton, using successive refinements. The general result of closure is established in section 7.3.

7.1 Closure of the class *TL* under refinement by 1-automata

Theorem 28. *Let L be a language in $TL(n)$ over an alphabet Σ, accepted by a timed b-automaton $\mathcal{A}$ with a set X of clocks. Let σ be a timed refinement, defined by a family $(\mathcal{A}_a)_{a\in\Sigma}$ of generalized transitions, such that for each $a \in \Sigma$, $\mathcal{A}_a$ is a o-1-automaton. For $a \in \Sigma$, we consider the language $L_a = L(\mathcal{A}_a)$ and the subset ρ_a of clocks reset in $\mathcal{A}_a$. If, for each $a \in \Sigma$,*

1. *for each transition $\delta = (q, A, a, \rho, r)$ in $\mathcal{A}$ with label a, we have $\rho_a \subset \rho$ and,*
2. *(a) either the language L_a contains only finite words,*
   (b) or for each transition δ in $\mathcal{A}$ with label a, for each clock $x \in X$, the clock constraint associated with δ is a disjunction of subformulas of the form $x \in I$, where $I =]c_1, c_2]$ or $I = [0, c_2]$ or $I =]c_1, \infty[$,

then $\sigma(L)$ belongs to TL.

Remark 29. 1. We may intuitively explain the first condition for a transition δ labelled by a. In the subset ρ of clocks to be reset, there is a part reset by the action and another part reset by the system. Of course, the part reset by the action a must be the same for all transitions labelled by a.
2. Lemma 27 explains why condition $2(b)$ in Theorem 28 is necessary to build a refinement, when condition $2(a)$ does not hold.

Proof. We present the construction in the particular case of the refinement $\sigma = \sigma_a$ defined by $L_a = L(\mathcal{A}_a)$ and $L_b = \{b\}$ for $b \neq a$. As in the classical framework, the general case is obtained by renaming and successive refinements of this form.
To refine action a by L_a, we have to replace in $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ each transition (q, A, a, ρ, r) by a timed automaton accepting $L_a \cap \{w \mid X(\tau(w)) \in A\}$. Write Δ_a for the subset of transitions labeled by a in Δ. We may suppose that no transitions in Δ_a are self-loops (by adding a copy of the state and looping between both states). We consider $\mathcal{A}_a = (\Sigma, R, J, G, \omega, X)$ as a b-1-automaton accepting L_a. By making on $\mathcal{A}_a$ some classical transformations over finite automata, we may assume that J is reduced to a single state j with no loops (simply add a new state) and that all final states are dead states. In order to simplify the notations, we shall make the construction in the particular case where G is reduced to a single state g. The general case may be deduced from this one by taking an alphabet $\{a^g \mid g \in G\}$ in bijection with G, replacing each transition (q, A, a, ρ, r) in Δ_a by the bunch of transitions (q, A, a^g, ρ, r), $\forall g \in G$, then refining each action labelled a^g by the generalized transition $\mathcal{A}_{a^g} = (\Sigma, R, J, \{g\}, \omega, X)$. We now give the construction of the timed automaton accepting $\sigma(L)$ in the two cases of theorem 28.

Case (a). In this case, $g \in R[0]$ and the construction is straigthforward.
For all δ in Δ_a, let $\mathcal{A}_{a,\delta} = (\Sigma, R_\delta, \{j_\delta\}, \{g_\delta\}, \omega_\delta, X)$ be distinct copies of $\mathcal{A}_a$. The b-automaton $\mathcal{B} = (\Sigma, S, H, K, \Omega, X)$, defined below, accepts $\sigma(L)$.

- $S[0] = (\cup_{\delta\in\Delta_a} R_\delta) \cup Q[0]$

- $\Omega = (\cup_{\delta \in \Delta_a}(\omega_\delta \cup \bar{\omega}_\delta)) \cup (\Delta - \Delta_a)$ where $\bar{\omega}_\delta$, for $\delta = (q, A, a, \rho, r)$ is defined as the set of the following transitions :
  - if $(j_\delta, B, b, \beta, h_\delta) \in \omega_\delta$, then $(q, B, b, \beta, h_\delta) \in \bar{\omega}_\delta$
  - if $(h_\delta, B, b, \beta, g_\delta) \in \omega_\delta$, then $(h_\delta, B \wedge A, b, \rho, r) \in \bar{\omega}_\delta$
  - if $T \in \mathcal{P}(Q[0])$ contains q and r, if $(T, B, b, \beta, s) \in \Delta$, then for each subset U of $S[0]$ such that $U \cap Q[0] = T$, we have $(U, B, b, \beta, s) \in \bar{\omega}_\delta$
- $H = I$
- $K = F \cup \{U \subseteq S[0] \mid U \cap Q[0] \in F\}$.

Case (b). In this case, $g \in \mathcal{P}(R[0])$ and the construction is more involved: in the replacement of a transition (q, A, a, ρ, r) by $\mathcal{A}_a$, a transition leaving $\mathcal{A}_a$ to return in r is obtained by limit. A clock x in ρ, which is not reset in $\mathcal{A}_a$, cannot be reset, except if we reset it all along the path in $\mathcal{A}_a$. We then have to use inside $\mathcal{A}_a$ a copy of x. Thus, reaching r, the clock x and its copy have no more the same value and this would make impossible for a run to go through $\mathcal{A}_a$ again. For this reason, we shall use two copies of each clock and split the states according to their use of one or the other. This is done in the first step.
(i) FIRST STEP.
For each clock x in X, we consider two distincts copies of x denoted by x_0 and x_1, and for any subset ρ of X, we write $\rho_0 = \{x_0 \mid x \in \rho\}$ and $\rho_1 = \{x_1 \mid x \in \rho\}$. We consider the reduced b-automaton
$\mathcal{A}' = (\Sigma, Q', I', F', \gamma', \Delta', X' = X \cup X_0 \cup X_1)$
defined by:

- $Q'[0] = Q[0] \times \{0,1\}^X$ and $Q'[1] = \mathcal{P}(Q[0]) \times \{0,1\}^X$
- $I' = I \times \{(0)_{x \in X}\}$ and $F' = F \times \{0,1\}^X$
- In Δ', clocks of $X_0 \cup X_1$ are not submitted to any constraint, so that a constraint on X' may be identified to a constraint on X without ambiguity.
  - For all (q, B, b, β, r) in $\Delta \setminus \Delta_a$ (i.e. $b \neq a$) and for all ε in $\{0,1\}^X$, $((q, \varepsilon), B, b, \beta \cup \{x_{\varepsilon_x} \mid x \in \beta\}, (r, \varepsilon)) \in \Delta'$
  - For all (q, A, a, ρ, r) in Δ_a and for all $\varepsilon = (\varepsilon_x)_{x \in X}$ in $\{0,1\}^X$, define $\varepsilon' = (\varepsilon'_x)_{x \in X}$ by
    - $\varepsilon'_x = \varepsilon_x$ if $x \notin \rho$
    - $\{\varepsilon'_x, \varepsilon_x\} = \{0,1\}$ if $x \in \rho$

    Then $((q, \varepsilon), A, a, \rho \cup \{x_{\varepsilon'_x} \mid x \in \rho\}, (r, \varepsilon')) \in \Delta'$
- The mapping γ' is defined as follows. For a subset $\{(q^j, \varepsilon^j) \mid j \in \{1, ..., k\}\}$ of $\mathcal{P}(Q'[0])$, with $\varepsilon^j = (\varepsilon^j_x)_{x \in X}$, define $\varepsilon = (\varepsilon_x)_{x \in X}$ by: For x in X,
  - if all ε^j_x are equal, ε_x is their common value,
  - otherwise $\varepsilon_x = 0$ by default.

  Then $\gamma'(\{(q^j, \varepsilon^j) \mid j \in \{1, ..., k\}\}) = (\{q^j \mid j \in \{1, ..., k\}\}, \varepsilon)$

Then the first projection from Q' onto Q defines a covering from $\mathcal{A}'$ onto $\mathcal{A}$ and $L(\mathcal{A}') = L(\mathcal{A})$. From the construction, we have:

Lemma 30. *Let (q, ε) be a state of $\mathcal{A}'$ with $\varepsilon = (\varepsilon_x)_{x \in X}$. For any run in $\mathcal{A}'$ and any clock x in X, x and x_{ε_x} have the same value when the run enters (q, ε).*

(ii) SECOND STEP.
Let $\delta' = ((q,\varepsilon), A, a, \rho \cup \{x_{\varepsilon_x} \mid x \in \rho\}, (r,\varepsilon'))$ in Δ'_a defined as above from a transition $\delta = (q, A, a, \rho, r)$ of Δ_a. The transition δ' is also written below simply as $(\mathbf{q}, A, a, \rho', \mathbf{r})$. We consider the generalized transition
$\mathcal{A}_{\delta'} = (\Sigma, R_{\delta'}, J_{\delta'} = \{j_{\delta'}\}, G_{\delta'} = \{g_{\delta'}\}, \omega_{\delta'}, \gamma_{\delta'}, X' = X \cup X_0 \cup X_1)$
obtained from A_a by the following different operations:

- we take a copy of $\mathcal{A}_a$ in which we replace each clock x of ρ by its copy x_{ε_x}. Then each clock x in ρ and its other copy $x_{\varepsilon'_x}$ are reset on each transition.
- the automaton is modified according to Lemma 26 to accept instead of L_a the language $L_a \cap \{w \mid X_\delta(\tau(w)) \in A\}$ where $X_\delta = \{x \in X \mid x \notin \rho\} \cup \{x_{\varepsilon_x} \mid x \in \rho\}$.

The reduced timed automaton $\mathcal{B} = (\Sigma, S, H, K, \Gamma, \Omega, X' = X \cup X_0 \cup X_1)$, defined below, accepts $\sigma(L)$.

- $S[0] = \cup_{\delta' \in \Delta'_a} R_{\delta'}[0] \cup Q'[0]$
- $S[1] = \cup_{\delta' \in \Delta'_a} R_{\delta'}[1] \cup Q'[1]$
- $H = I' \cup \{j_{\delta'} \mid \delta' = (\mathbf{q}, A, a, \rho', \mathbf{r}) \in \Delta'_a$ such that $\mathbf{q} \in I'\}$
- $K = F' \cup \{g_{\delta'} \mid \delta' = (\mathbf{q}, A, a, \rho', \mathbf{r}) \in \Delta'_a$ such that $\mathbf{r} \in F'\}$
- $\Omega = \cup_{\delta' \in \Delta'_a} (\omega_{\delta'} \cup \bar{\omega}_{\delta'}) \cup (\Delta' - \Delta'_a)$ where $\bar{\omega}_{\delta'}$ is the set of the following transitions:
  - If $\delta'' = (\mathbf{s}, B, b, \beta, \mathbf{q}) \in \Delta'$ with $b \neq a$, then $(\mathbf{s}, B, b, \beta, j_{\delta'}) \in \bar{\omega}_{\delta'}$.
  - If $\delta'' = (\mathbf{s}, B, b, \beta, \mathbf{q}) \in \Delta'$ with $b \neq a$, as above, but if also $\delta'_1 = (\mathbf{q}_1, A_1, a, \rho'_1, \mathbf{s}) \in \Delta'_a$ then $(g_{\delta''}, B, b, \beta, j_{\delta'}) \in \bar{\omega}_{\delta'}$.
  - If $\delta'' = (\mathbf{r}, B, b, \beta, \mathbf{s}) \in \Delta'$ with $b \neq a$, then $(g_{\delta'}, B, b, \beta, \mathbf{s},) \in \bar{\omega}_{\delta'}$.
- The mapping Γ is defined on $\mathcal{P}(S[0])$ by:
  - $\Gamma(T) = \gamma'(T)$ if $T \subseteq Q'[0]$
  - $\Gamma(T) = \gamma_{\delta'}(T)$ if there exists δ' in Δ'_a such that $T \subseteq R_{\delta'}[0]$
  - $\Gamma(T) = \gamma'((T \cap Q'[0]) \cup \{\mathbf{q}, \mathbf{r} \mid \exists \delta' = (\mathbf{q}, A, a, \alpha, \mathbf{r}) \in \Delta'_a$ and $\exists \sigma \subseteq R_{\delta'}[0]$ such that $\gamma_{\delta'}(\sigma) = g_{\delta'}$ and $\{j_{\delta'}\} \cup \sigma \subseteq T\})$.

□

The construction is illustrated on a simple example. We consider the refinement σ which leaves b unchanged and associates with a the timed language $L_a = \{(c^\omega, (t_i)_{i<\omega}) \mid$ the time sequence satisfies $t_0 > 0$ and $t_i > 1\ \forall i > 0\}$ accepted by the generalized transition $\mathcal{A}_a$ of figure 9. To apply this refinement on the timed language L accepted by the automaton of figure 10, we first construct another automaton for L which corresponds to the first step of the proof (figure 11), then we build an automaton accepting $\sigma(L)$ (figure 12).

7.2 A decomposition theorem

Theorem 31. *Let n be an integer, $n \geq 1$. Any timed language in $TL(n+1)$ is the refinement of some language in $TL(n)$.*

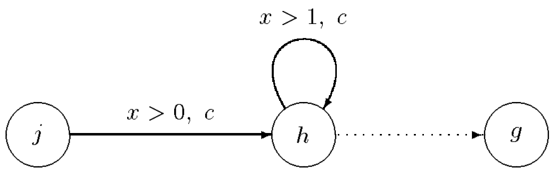

Fig. 9. The generalized transition $\mathcal{A}_a$ which defines σ

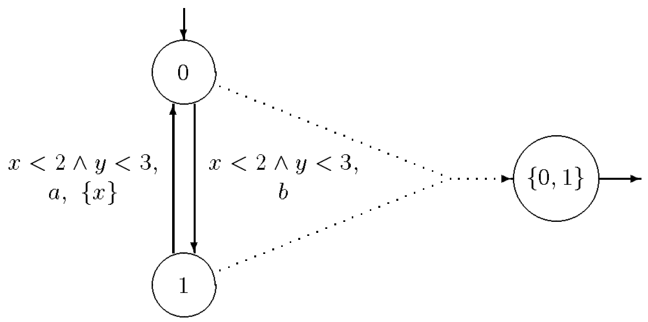

Fig. 10. A timed automaton $\mathcal{A}$ accepting L

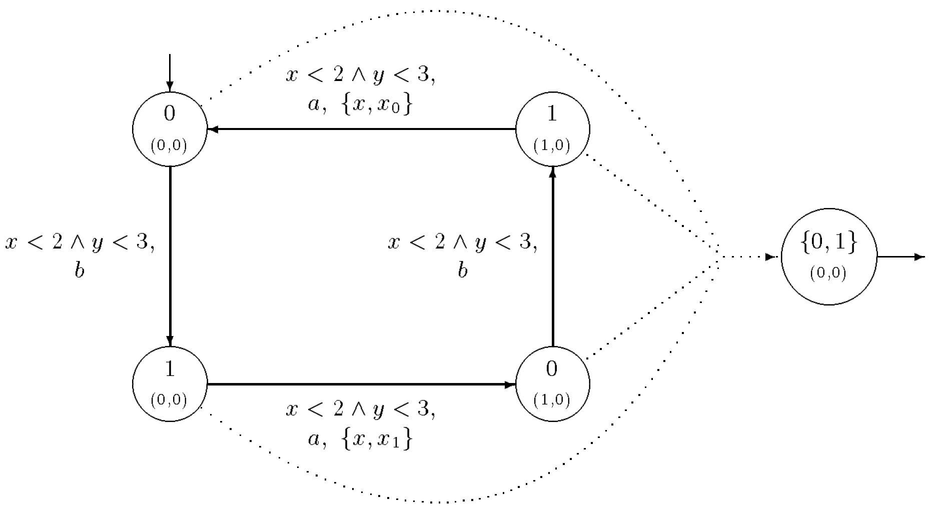

Fig. 11. The modified timed automaton $\mathcal{A}'$ accepting L (step 1 in the proof)

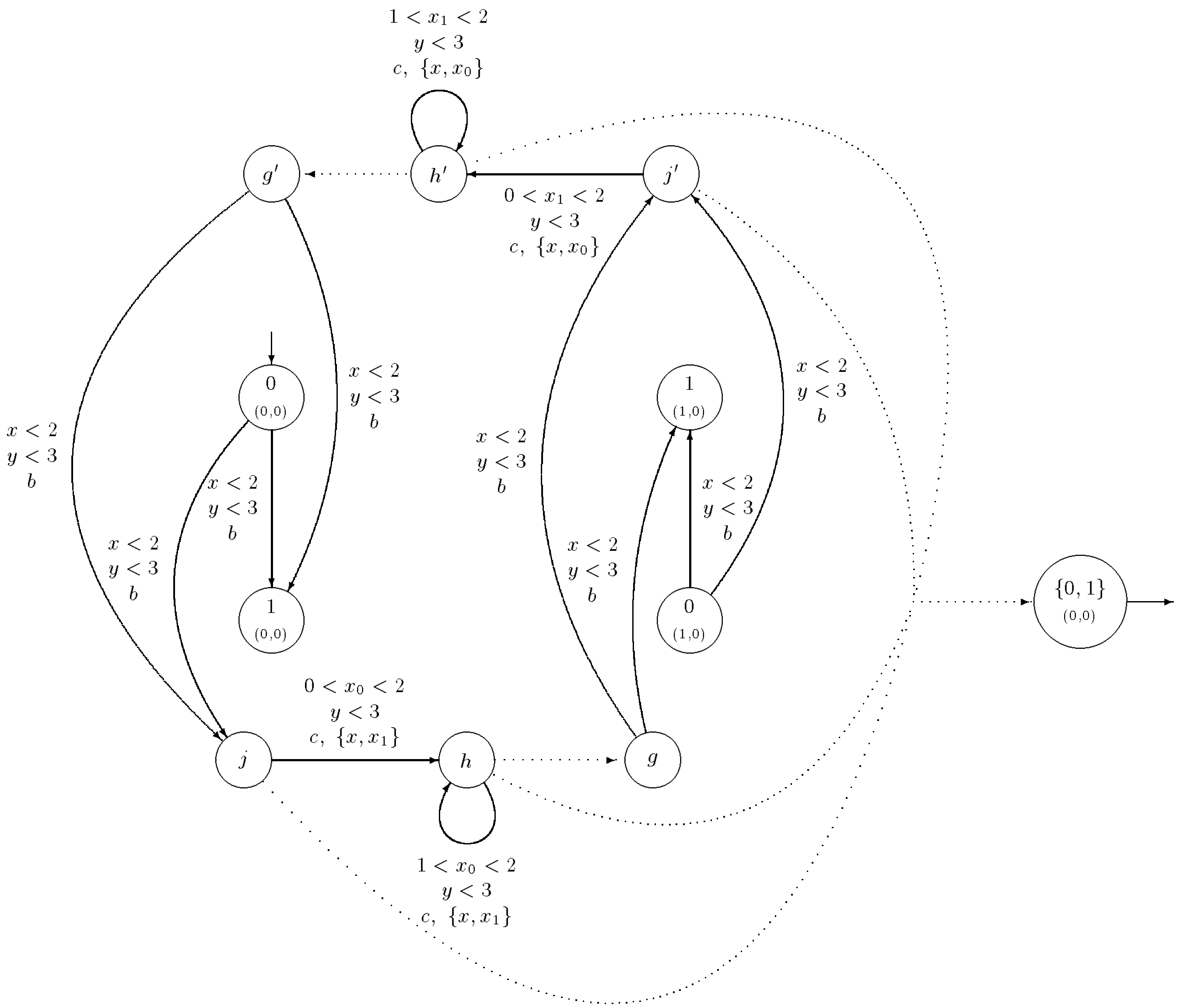

Fig. 12. A timed automaton (from step 2) accepting $\sigma(L)$

Proof. We use here c-automata to identify more easily the levels of meta-states. Let L' be a language in $TL(n+1)$, $L' = L(\mathcal{A}')$ for some $(n+1)$-automaton $\mathcal{A}' = (\Sigma', Q', I', F', \Delta', X)$. From lemma 15, we assume that $\mathcal{A}'$ is state-reset. The idea underlying the construction is to introduce new transitions, that can be taken instead of runs leading to meta-states of type 1.

For a transition $\delta = (q, A, a, \rho, q')$, we define $src(\delta) = q$ and $rge(\delta) = q'$. With a non empty subset P of $Q'[0]$, we associate the state-reset cycle $\mathcal{A}_P = (\Sigma', Q_P, I_P, F_P, \Delta_P, X)$, where

- $Q_P[0] = I_P = P$, $F_P = \{P\} \subset \mathcal{P}(Q_P[0])$ and
- $\Delta_P = \{\delta \in \Delta' \mid src(\delta) \in P \text{ and } rge(\delta) \in P\}$, i.e. the restriction of Δ' to states in P.

We now build a new automaton $\mathcal{A}$ in the following way: if the language $L(\mathcal{A}_P)$ is not empty, we add a new letter a_P (not in Σ') and transitions leading to P, which thus becomes a state of type 0, instead of a meta-state. More precisely, we consider the subset $\mathcal{R} = \{P \in \mathcal{P}(Q'[0]) \mid L(\mathcal{A}_P) \neq \emptyset\}$ of $\mathcal{P}(Q'[0])$ and the new alphabets $\Sigma^1 = \{a_P \mid P \in \mathcal{R}\}$ and $\Sigma = \Sigma' \cup \Sigma^1$.
The n-automaton $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ over the alphabet Σ and the set of clocks X is defined by:

- $Q[0] = Q'[0] \cup \mathcal{R}$ and $I = I'$.
- The set Δ consists of the following transitions:
  - if (q, A, a, ρ, q') is in Δ', with $q \in Q[0]$ and $q' \in Q'[0]$, then it is also a transition in Δ,
  - if $P \in \mathcal{R}$ and $q \in P$, then $(q, \text{ True}, a_P, \rho_P, P)$ is in Δ, where $\rho_P = \cup_{\delta \in \Delta_P}$ is the set of clocks reset along the cycle $\mathcal{A}_P$.
  - in order to obtain the transitions from meta-states in Δ, we define inductively a sequence of mappings γ_i from $\mathcal{P}^i(Q[0])$ into $\mathcal{P}^{i+1}(Q'[0])$, $1 \leq i \leq n$ by: $\gamma_1(\mathbf{q}) = \mathbf{q} \cap \mathcal{P}(Q'[0])$ for $\mathbf{q} \in \mathcal{P}(Q[0])$ and for $i \geq 2$, if $\mathbf{q} = \{q_1, \cdots, q_k\}$ is in $\mathcal{P}^i(Q[0])$, then $\gamma_i(\mathbf{q}) = \{\gamma_{i-1}(q_1), \cdots, \gamma_{i-1}(q_k)\}$. A transition $(\mathbf{q}, A, a, \rho, q')$, $\mathbf{q} \in \mathcal{P}^i(Q[0])$, $q' \in Q[0]$ belongs to Δ if and only if $(\gamma_i(\mathbf{q}), A, a, \rho, q')$ belongs to Δ'.
- the set F of final states of $\mathcal{A}$ is defined in the following way: with the convention $\gamma_0 = id$ on $Q[0]$, a state $\mathbf{q} \in \mathcal{P}^i(Q[0])$ belongs to F if and only if $\gamma_i(\mathbf{q})$ belongs to F'.

Finally, we define a refinement σ by the family $(\mathcal{A}_a)_{a \in \Sigma}$ of generalized transitions associated with the following languages: $L_{a_P} = L(\mathcal{A}_P)$ for $a_P \in \Sigma^1$, and $L_a = \{a\}$ for $a \in \Sigma'$. Note that the conditions of theorem 28 hold for this refinement. Now if L is the language accepted by $\mathcal{A}$, it is easy to prove from the construction above that $\sigma(L) = L'$. □

7.3 Final remarks

Note that theorem 31 gives in fact an algorithm to find a regular expression (with renaming) for a timed language in TL. We can also use the method proposed after proposition 25.

The general property of closure under refinement for the class TL is obtained by combining theorems 28 and 31 above. Let L be a language in TL, over an alphabet Σ, and let σ be a refinement, given by a family $(L_a)_{a \in \Sigma}$ of languages accepted by generalized transitions in $TL(n_a)$, such that the conditions of theorem 28 hold. Then, applying n_a times theorem 31 from each language L_a, we can obtain a timed 1-automaton $\mathcal{B}_a$ such that L_a is the image of $L(\mathcal{B}_a) = M_a$ by n_a successive refinements. Now, $\sigma(L)$ is also the image of L by those successive refinements, following a first refinement by $(M_a)_{a \in \Sigma}$.

8 Conclusion

In this work, we developed two original studies.

The first one is devoted to timed automata accepting sequences of Zeno words. We proved that the corresponding class of languages TL retains nice properties of the usual class ($TL(1)$): closure properties and decidability of emptiness. Furthermore, concatenation of Zeno words becomes a meaningful operation inside the class TL.

The second one concerns a new and general definition of timed refinement. Apart from a few particular cases of untimed refinement for timed languages, there was, up to now, no such systematic study in the framework of recognizable timed languages. The main result is the following: the class TL is the closure under timed refinement of the class $TL(1)$. Moreover, as an application, we obtain an exhaustive description of timed refinement (by finite timed automata in a special set GT) inside the lowest class $TL(1)$, as well as algorithms to find a timed regular expression (with renaming).

References

1. R. Alur and D.L. Dill. Automata for modeling real-time systems. In *Proceedings of ICALP'90*, number 443 in Lecture Notes in Computer Science, pages 322–335. Springer Verlag, 1990.
2. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. E. Asarin, P. Caspi, and O. Maler. A Kleene theorem for timed automata. In *Proceedings of LICS'97*, 1997.
4. N. Bedon. Finite automata and ordinals. *Theoretical Computer Science*, 156:119–144, 1996.
5. B. Bérard. Untiming timed languages. *Information Processing Letters*, 55:129–135, 1995.
6. B. Bérard, P. Gastin, and A. Petit. On the power of non observable actions in timed automata. In *Proceedings of STACS'96*, number 1046 in Lecture Notes in Computer Science, pages 257–268. Springer Verlag, 1996.
7. B. Bérard, P. Gastin, and A. Petit. Refinement and abstraction for timed languages. Technical report, LSV, CNRS URA 2236, ENS de Cachan, 1997.
8. R. Büchi. The monadic second order theory of ω_1. In Siefkes Büchi, editor, *The Monadic Second Order Theory of All Countable Ordinals*, number 328 in Lecture Notes in Mathematics, pages 1–126, 1973.

9. Y. Choueka. Finite automata, definable sets and regular expressions over ω^n-tapes. *Journal of Computer and System Sciences*, 17:81–97, 1978.
10. C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. In *Proceedings of CAV'91*, number 575 in Lecture Notes in Computer Science, pages 399–409. Springer Verlag, 1991.
11. V. Diekert, P. Gastin, and A. Petit. Removing ϵ-transitions in timed automata. In *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science (STACS'97)*, number 1200 in Lecture Notes in Computer Science, pages 583–594. Springer Verlag, 1997.
12. M.R. Hansen, P.K. Pandya, and C. Zhou. Finite divergence. *Theoretical Computer Science*, 138:113–139, 1995.
13. J. C. Hemmer and P. Wolper. Ordinal finite automata and languages. Technical report, University of Liege, 1992.
14. T.A. Henzinger, Z. Manna, and A. Pnueli. Temporal proofs methodologies for real-time systems. In *Proceedings of POPL'91*, pages 353–366, 1991.
15. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.
16. N. Lynch and H. Attiya. Using mappings to prove timing properties. In *Proceedings of PODC'90*, pages 265–280, 1990.
17. J. Ostroff. *Temporal Logic of Real-time Systems.* Research Studies Press, 1990.
18. C. Ramchandani. Analysis of asynchronous concurrent systems by Petri nets. Technical report, Massachusetts Institute of Technology, 1974.
19. J.G. Rosenstein. *Linear orderings.* Academic Press, New York, 1982.
20. J. Wojciechowski. Finite automata on transfinite sequences and regular expressions. *Fundamenta Informaticae*, 8.3-4:379–396, 1985.