Accepting Zeno words: a way toward timed refinements

Béatrice Bérard and Claudine Picaronny

LSV, CNRS URA 2236, 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, some physical phenomena, as the movements of a damped oscillator, can be represented by convergent executions, producing Zeno words 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. Contrary to classical (untimed) transition systems, these models provide an explicit notion of time and thus allow to describe time requirements. For instance, timed Petri nets [18], timed transition systems [17,14], timed automata [1,2] or timed I/O automata [16] have been discussed. Timed models 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 are needed 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, are proposed there and the Car-Bee problem is studied in details: two cars move towards each other at uniform and equal speed and they will collide after r time units. A bee is flying repeatedly from one car to the other at twice their speed, until it is finally squashed between them. 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. 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 executions only contain one infinite sequence. Indeed, automata producing 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 we shall use along the paper.

2.1 Ordinals

We use ordinals mainly in order to number 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.

In this work, we consider only ordinals smaller than ω^ω: such an ordinal has 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 it 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 k.

All facts concerning ordinals in this paper can be found in [19].

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 tranfinite languages having regular expressions. We recall their definitions.

3.1 Choueka n-automata

If n is a natural number, a Choueka n-automaton is a finite automaton with a global set of states split into $n+1$ layers. A state in the k^{th} layer is called a state of *type* k and, for $k > 0$, such a state is in fact a set of states of type $k-1$. An execution begins as in a usual automaton in the first layer of states. 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. A new action may then be performed to get down to the first layer and the execution can go on. A state of the third layer will be reached after ω^2 actions, and so on.

Example. The $\mathcal{C}$-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, the new state becomes $\{1\}$. From this point, an infinite number of b's is possible, yielding $\{2\}$ as new state, and c allows to reach the final state 3.

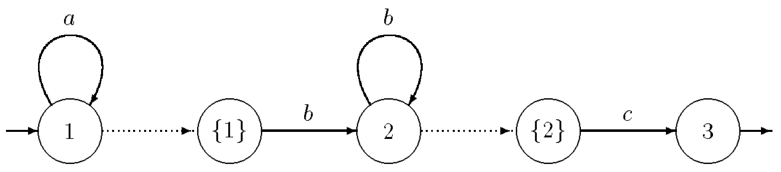


Fig. 1. A $\mathcal{C}$-2-automaton for $a^\omega b^\omega c$

More precisely, for a finite set Q, we define inductively $[Q]^0 = Q$ and, for any natural number k, $[Q]^{k+1}$ is the powerset of $[Q]^k$ without the empty set. We write $[Q]_0^k = Q \cup [Q]^1 \cup \cdots \cup [Q]^k$.
A *Choueka n-automaton* (or $\mathcal{C}$-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 type 0),
$I \subseteq Q$ is a subset of initial states,
$F \subseteq [Q]_0^n$ is a set of final states,
$\Delta \subseteq [Q]_0^{n-1} \times \Sigma \times Q$ is the transition relation.

Note that the total set of states of $\mathcal{A}$ is $S = [Q]_0^n$.
A continuous run of $\mathcal{A}$ on a α-word $u = (a_i)_{i<\alpha}$ is an α-sequence $(q_i)_{i\leq\alpha}$ of states such that:

- $q_0 \in [Q]_0^{n-1}$
- if i is the successor of some ordinal $i-1$, there is a transition $(q_{i-1}, a_{i-1}, q_i) \in \Delta$,
- if i is a limit ordinal of type $k > 0$, recall that $i = j + \omega^k$, where $j = 0$ or j is an ordinal of type greater than or equal to k. In this case, q_i is a state of type k defined by: $q_i = inf\{q_{j+\omega^{k-1}.p} / p \in \mathbb{N}\}$, where $inf\{z_1, z_2, \cdots\}$ is the set of elements z_i which appear infinitely often in the sequence.

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 $\mathcal{A}$. As $F \subseteq [Q]_0^n$, the words in $L(\mathcal{A})$ have a length smaller than ω^n.

Remark 1. The third point above explains why such a run is called continuous: the state corresponding to a limit ordinal is itself a limit state (of the same type), which is reached in an implicit way, when an execution has gone infinitely often through some set of states of the type just below. In particular, a Choueka 1-automaton is a Muller automaton, where a usual Muller acceptance condition is just a final state of type 1.

3.2 Wojciechowski automata

A Wojciechowski automaton, or $\mathcal{W}$-automaton for short, is a finite automaton with a global set of states split into only two layers. An execution begins as in a usual automaton in the first layer of states. After a sequence of actions with length a limit ordinal, a state of the second layer is reached. A new action may then be performed to get down to the first layer and the execution can go on.
A $\mathcal{W}$-*automaton* is a tuple $\mathcal{A} = (\Sigma, Q, I, F, \Delta)$, where

Σ, Q, $I \subseteq Q$ are as in a $\mathcal{C}$-automaton,
$F \subseteq [Q]_0^1$ is a set of final states,
$\Delta \subseteq [Q]_0^1 \times \Sigma \times Q$ is the transition relation.

The total set of states of $\mathcal{A}$ is $S = [Q]_0^1 = Q \cup [Q]$ and a continuous run of $\mathcal{A}$ on a α-word $u = (a_i)_{1\leq i<\alpha}$ is an α-sequence $(q_i)_{1\leq i\leq\alpha}$ of states such that:

- $q_1 \in [Q]_0^1$
- if i is the successor of some ordinal $i-1$, there is a transition $(q_{i-1}, a_{i-1}, q_i) \in \Delta$,

- if i is a limit ordinal, then q_i, in $[Q]$, is the set of states q such that there exists a strictly increasing sequence of non limit ordinals $(j_p)_{p\in\mathbb{N}}$ having i as limit and such that $q = q_{j_p}$, for all p in $\mathbb{N}$. This set is written $lim(r)$ and called the limit set of the run r.

As above, 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 $\mathcal{A}$.
To avoid, among these automata, those accepting words of any countable length, we add the following condition : For any s in $[Q]$, q in Q such that there exists a in Σ with $(s, a, q) \in \Delta$, then $q \notin s$. With this additional condition, all words accepted have length less or equal to ω^n where n is the cardinal of Q and we can call $\mathcal{A}$ a n-automaton.

Remark 2. As in the Choueka model, a continuous run is completely determined by its non limit states. In particular, the notion of continuity, for runs of length ω, is the same in a $\mathcal{C}$-automaton and in a $\mathcal{W}$-automaton.

4 Timed transfinite automata

Combining the definitions above and classical 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 $\{\vee, \wedge, \neg\}$ 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 $\mathcal{C}$-n-automata

Definition 3. A *timed* $\mathcal{C}$*-*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 $\mathcal{C}$-n-automaton,
X is a finite set of clocks and
the transition relation Δ contains elements of the form (q, A, a, ρ, q'), also written $q \xrightarrow{A,a,\rho} q'$, where $q \in [Q]_0^{n-1}$, A is a constraint, $a \in \Sigma$, $\rho \subseteq X$, and $q' \in Q$.

Let us explain the execution of a transition $q \xrightarrow{A,a,\rho} q'$, 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 $(q', X(t'))$.

As for timed automata, we wish to define a (continuous) run of the timed $\mathcal{C}$-n-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 limit state after an infinite convergent run. The following lemma (for which the proof is rather natural) gives the answer.

Remark 4. Note that, as in classical timed automata, it is not necessary to define limits of clock values for divergent sequences of times.

Lemma 5. *Let i be a limit ordinal of type $k > 0$, $i = j + \omega^k$ with $j = 0$ or j is an ordinal of type greater than or equal to k. 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 verified:*

– *Case 1: There exists some integer N such that, for each ordinal h, $\omega^{k-1}.N < h < \omega^k$, $x(t_{j+h}) \neq 0$. Then, the sequence $(x(t_{j+\omega^{k-1}.p}))_{p \in \mathbb{N}}$ is non decreasing for $p \geq N$, bounded by t_i, so that it is convergent.*
– *Case 2: For each integer N, there exists an ordinal h, $\omega^{k-1}.N < h < \omega^k$, such that $x(t_{j+h}) = 0$. In this case, the sequence $(x(t_{j+\omega^{k-1}.p}))_{p \in \mathbb{N}}$ is the term of a convergent series, so that its limit is zero.*

Finally, a run of a timed $\mathcal{C}$-n-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:

- the first state is $q_0 \in [Q]_0^{n-1}$, at time $t_0 \in \mathbb{R}_+$, with values $X(t_0)$,
- 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 $i = j + \omega^k$ is a limit ordinal of type $k > 0$, with $j = 0$ or j is of type $\geq k$, q_i is defined as in the untimed case. Moreover, thanks to Lemma 5, we have: $x(t_i) = lim_p x(t_{j+\omega^{k-1}.p})$, 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_α is finite. It is *accepting* 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 word w is then *accepted* and the timed language $L(\mathcal{A})$ is the set of accepted words.

Examples.

1. Let $\mathcal{A}$ be the timed 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.

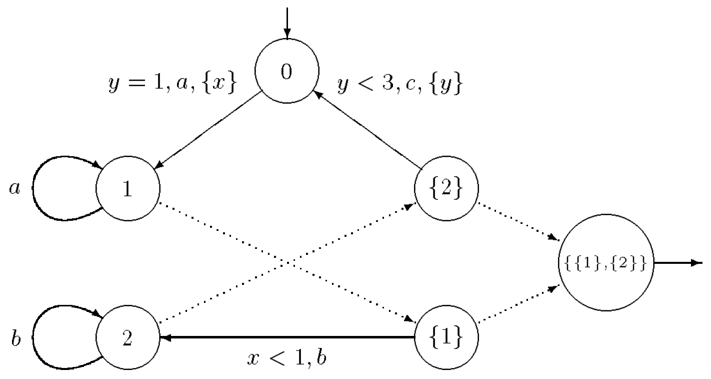


Fig. 2. A timed $\mathcal{C}$-2-automaton $\mathcal{A}$ accepting L

2. The timed 2-automaton in Figure 3 can be used to describe the successive steps in testing the resisting power of a spring: after it has been extended, the spring oscillates an infinite number of times (states 1 and 2) until it stops in the state $\{1,2\}$. If the oscillation time is too short (compared with some time unit, taken to be 1 here), the spring is faulty (state $F1$). Otherwise, the operation is repeated until the spring breaks. If it breaks too early, it is again faulty (state $F2$), else the test is successful (state S).

4.2 Timed $\mathcal{W}$-automata

Definition 6. A *timed $\mathcal{W}$-automaton* is a tuple $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$, where

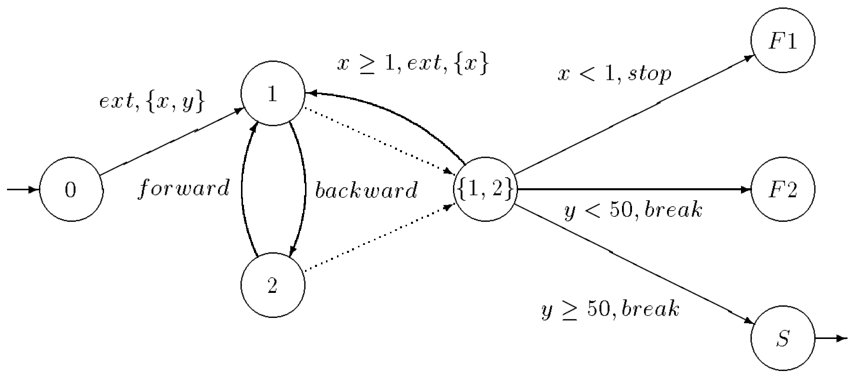


Fig. 3. A timed 2-automaton for testing a spring

Σ, Q, I and F are as in an (untimed) $\mathcal{W}$-automaton,
X is a finite set of clocks and the transition relation Δ contains elements of the form (q, A, a, ρ, q'), also written $q \xrightarrow{A,a,\rho} q'$, where $q \in [Q]_0^1$, A is a constraint, $a \in \Sigma$, $\rho \subseteq X$, and $q' \in Q$.

The execution of a transition $q \xrightarrow{A,a,\rho} q'$, when q' is a state of type 0 is the same as in a timed $\mathcal{C}$-automaton. As above, in order to define a continuous run of the timed $\mathcal{W}$-automaton $\mathcal{A}$, we need the clock values obtained when reaching a limit state after an infinite convergent run. These values are determined by the following lemma (which is exactly lemma 5 above, just not stated the same way) :

Lemma 7. *Let i be a limit ordinal, with t_i finite, and let x be a clock in X. One of the two following cases is verified:*

- *Case 1: There exists an ordinal $j < i$ such that, for each 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 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.*

A run of a timed $\mathcal{W}$-automaton on a α-timed word $w = ((a_i, t_i))_{1 \leq i < \alpha}$ is an α- sequence $(q_i, X(t_i))_{0 \leq i \leq \alpha}$, such that:

- the first state is $q_0 \in [Q]_0^1$, at time $t_0 \in \mathbb{R}_+$, with values $X(t_0)$,
- 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 7, 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 (see Remark 4).

The definitions of convergent or accepting run is the same as before, as well as the definition of the timed language $L(\mathcal{A})$ of accepted words.

For example, the timed $\mathcal{W}$-2-automaton in Figure 4, 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 state 0 (respectively states 1 and 2), at a sequence of ordinals having ω^2 as limit.

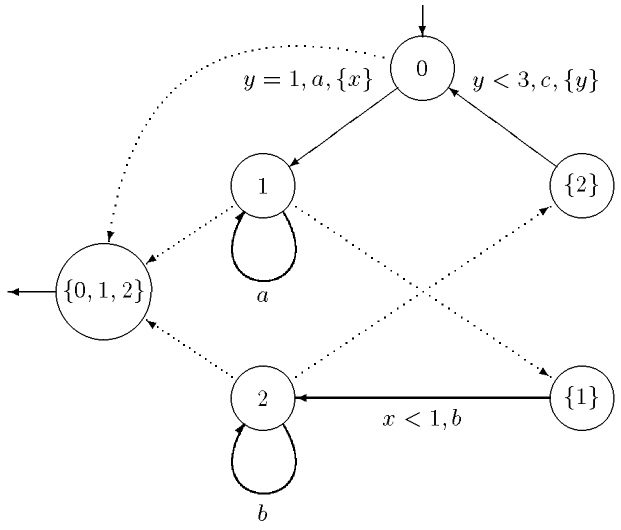


Fig. 4. A timed $\mathcal{W}$-2-automaton for L

4.3 Reduced timed automata

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

Reduced timed $\mathcal{C}$-n-automata.

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

S is a finite set of states, disjoint union of $n+1$ subsets $Q^{[0]}, \cdots, Q^{[n]}$; each $Q^{[i]}$ is the set of states of type i

$\gamma = (\gamma_i)_{1,\ldots,n}$; each γ_i is a mapping from $[Q^{[i-1]}]$ to $Q^{[i]}$
Σ, $I \subseteq Q^{[0]}$, $F \subseteq Q$, Δ and X are as in a $\mathcal{C}$-automaton.

As explained above, the role of the mapping γ_i is to merge a set of type $i-1$ states, into a same type i state. Therefore, such an automaton works exactly as a Choueka's one except for one thing: a run $(q_i, X(t_i))_{i \leq \alpha}$ is continuous if for each ordinal i of type $k > 0$, decomposed as $i = j + \omega^k$, where $j = 0$ or is an ordinal of type greater than or equal to k, then $q_i = \gamma_i(inf\{q_{j+\omega^{k-1}.p}/p \in \mathbb{N}\})$, instead of $q_i = inf\{q_{j+\omega^{k-1}.p}/p \in \mathbb{N}\}$ in the other definition.
Note that a timed $\mathcal{C}$-n-automaton $\mathcal{A} = (\Sigma, Q, I, F, X, \Delta)$ is a reduced one when it is denoted by $\mathcal{A} = (\Sigma, [Q]_0^n, \gamma, I, F, X, \Delta)$ with each γ_i the identity map of $[Q]^i$.

Reduced timed $\mathcal{W}$-automata.

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

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

Reduced $\mathcal{W}$-automata work as $\mathcal{W}$'s one except that a run $(q_i, X(t_i))_{i \leq \alpha}$ is continuous if for each limit ordinal i, then $q_i = \gamma(s)$ (instead of s itself in the other definition) where s is the set of states q such that there exists a strictly increasing sequence of non limit ordinals $(j_n)_{n \in \mathbb{N}}$ having i as limit and such that $q = q_{j_n}$, for all n in $\mathbb{N}$.
As above, a timed $\mathcal{W}$-automaton $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ is a reduced one when it is denoted by $\mathcal{A} = (\Sigma, [Q]_0^1, \gamma, I, F, \Delta, X)$ with γ the identity map of $[Q]$.

4.4 Equivalence between transfinite timed automata

A covering lemma. We first give a sufficient condition for two timed automata to recognize the same language. Let $\mathcal{A} = (\Sigma, Q, \gamma, I, F, \Delta, X)$ and $\mathcal{B} = (\Sigma, S, \eta, J, G, \Omega, X)$ be reduced timed $\mathcal{C}$- or $\mathcal{W}$-automata.

As usual, we say that $\mathcal{A}$ covers $\mathcal{B}$ if there exists a subset U (set of useful states) of the set of total states containing all states appearing in continuous pathes in A beginning in I and a map c from Q to S (covering) such that

c maps $\Delta(u, a)$ (the set of transitions from state u with label a) onto $\Omega(c(u), a)$ for each u in U and a in Σ
c is continuous
$c(I) = J$
$F = c^{-1}(G)$

Lemma 10. *If $\mathcal{A}$ covers $\mathcal{B}$, then $L(\mathcal{A}) = L(\mathcal{B})$.*

Proof. Let $(q_0, X(t_0)) \xrightarrow{(A_0,a_0,\rho_0)} (q_1, X(t_1))....(q_\alpha, X(t_\alpha))$ be a continuous run in $\mathcal{A}$. Then all states q_i are in U and

$$(c(q_0), X(t_0)) \xrightarrow{(A_0,a_0,\rho_0)} (c(q_1), X(t_1))....(c(q_\alpha), X(t_\alpha))$$

is a continuous run in $\mathcal{B}$ with the same label. The hypothesis made on c ensures that this map sends the set of accepted runs in $\mathcal{A}$ onto the set of accepted runs in $\mathcal{B}$. □

Simulation properties between timed automata. Using Lemma 10 above, we now proceed to prove that all the notions of transfinite timed automata presented above have the same expressive power.

Proposition 11. *A reduced timed $\mathcal{C}$- (respectively $\mathcal{W}$-) automaton is covered by a timed $\mathcal{C}$- (respectively $\mathcal{W}$-) automaton.*

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

Let $\mathcal{B} = (\Sigma, S, \gamma, I, F, \Delta, X)$ be a reduced $\mathcal{C}$-n-automaton (respectively a reduced $\mathcal{W}$-automaton). Let m take the value n in the $\mathcal{C}$ case and 1 in the $\mathcal{W}$ case. Define a mapping c from $[S^{[0]}]_0^m$ onto S by $c = \gamma_i \circ [\gamma_{i-1}] \circ \cdots \circ [\gamma_1]^i$ from $[S^{[0]}]^i$ onto $S^{[i]}$ for $i \in \{0, ..., m\}$. Let $\mathcal{A} = (\Sigma, Q, I, F', \Delta', X)$ be the timed $\mathcal{C}$-n-automaton (respectively $\mathcal{W}$-automaton) defined by $Q = S^{[0]}$, $\Delta' = \{(s, A, a, \rho, q)/s \in [Q]_0^m, q \in S^{[0]}, (c(s), A, a, \rho, q) \in \Delta\}$ and $F' = c^{-1}(F)$. Then c is a covering from $\mathcal{A}$ to $\mathcal{B}$. □

Proposition 12. *Any $\mathcal{C}$-n-automaton is covered by a $\mathcal{W}$-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 $\mathcal{C}$-n-automaton. We define a $\mathcal{W}$-automaton $\mathcal{B} = (\Sigma, S, J, G, \Omega, X)$ (on the same alphabet and same set of clocks) and a covering c from $\mathcal{B}$ onto $\mathcal{A}$ by :

- $S = Q \cup [Q]_1^n \times Q$
  The *type* of a state $\mathbf{s}$ in $[S]$ is the smallest integer i in $\{1, \ldots, n\}$ such that $\mathbf{s} \in [Q \cup [Q]_0^{i-1} \times Q]$
- The map c from $S \cup [S]$ into $[Q]_0^n$ is defined case by case :
  If $q \in Q$, then $c(q) = q$.
  If $(\mathbf{q}, q) \in [Q]_1^n \times Q$, $c((\mathbf{q}, q)) = q$.
  If $\mathbf{s} \in [S]$ is of type i, then $c(\mathbf{s}) = \{\mathbf{q} \in [Q]^{i-1}\, ;\, \exists q \in Q \;\; (\mathbf{q}, q) \in \mathbf{s}\}$.
- $J = I$
- $G = c^{-1}(F)$
- Let $\delta = (q, A, a, \alpha, q')$ be a transition in Δ.
  If $q, q' \in Q$, then $\delta \in \Omega$

If $q, q' \in Q$ and $\mathbf{q} \in [Q]_1^n$, then $((\mathbf{q}, q), A, a, \alpha, q') \in \Omega$
If $q \in [Q]_1^n$, then $\forall s \in c^{-1}(q)$, $(s, A, a, \alpha, (q, q')) \in \Omega$

□

Proposition 13. *Any $\mathcal{W}$-automaton is covered by a reduced $\mathcal{C}$-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 $\mathcal{W}$-automaton. Let n be the cardinal of S. We define a reduced $\mathcal{C}$-n-automaton $\mathcal{A} = (\Sigma, Q, \gamma, I, F, \Delta, X)$ (on the same alphabet and same set of clocks) and a covering c from $\mathcal{A}$ onto $\mathcal{B}$ by :

- $Q^{[0]} = S \times [S]^{n-1}$
- $Q^{[i]} = [S] \times [S]^{n-i}$, for $0 < i \leq n$
- The map c from Q into $S \cup [S]$ 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 $S \times [S]^{n-1}$, then $c(q) = s$
  If $i > 0$, $q = (\mathbf{s}, \mathbf{s}_i ..., \mathbf{s}_{n-1})$ in $[S] \times [S]^{n-i}$, then $c(q) = \mathbf{s}$

– If $\{q^1, ..., q^r\} \in [Q^{[i]}]$ with $q^j = (\mathbf{s}^j, \mathbf{s}_i^j ..., {\mathbf{s}_{n-1}}^j)$ then
  If for $k > i$, all ${\mathbf{s}_k}^j$'s are equal to a same $\mathbf{s}_k$,
  then $\gamma(\{q^1, ..., q^r\}) = (\{\cup_j \mathbf{s}_i^j, \mathbf{s}_{i+1}, ..., \mathbf{s}_{n-1})$
  Otherwise, $\gamma(\{q^1, ..., q^r\})$ is undefined
– $I = J \times \{\emptyset\}^{n-1}$
– $G = c^{-1}(F)$.
– If $\delta = (s, A, a, \alpha, s') \in \Omega$, then
  if $s \in S$, then for any $\mathbf{q}_1, ..., \mathbf{q}_n$ in $[S]$,
  $((s, \mathbf{q}_1, ..., \mathbf{q}_{n-1}), A, a, \alpha, (s', \mathbf{q}_1 \cup \{s\}, ..., \mathbf{q}_{n-1} \cup \{s\}) \in \Delta$
  if $s \in [S]$, then for any $i > 0$ and $\mathbf{q}_i, ..., \mathbf{q}_{n-1}$ in $[S]$,
  $((s, \mathbf{q}_i, ..., \mathbf{q}_{n-1}), A, a, \alpha, (s', \{s\}, ...\{s\}, \mathbf{q}_i \cup \{s\}, ..., \mathbf{q}_{n-1} \cup \{s\}) \in \Delta$

Consider a continuous 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 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 c by induction on the length of the run. □

Timed state-reset automata. A timed n-automaton is *state-reset* if all transitions entering a state of type 0 reset the same set of clocks, and the same property is true when a state of type $k \geq 1$ is reached in an implicit way.

Lemma 14. *Let $\mathcal{A}$ be a timed automaton. There exists a state-reset timed automaton $\mathcal{A}'$, accepting the same language.*

Proof. For a $\mathcal{W}$-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 $\mathcal{C}$-automaton. Nevertheless, the proof of proposition 13 provides a state-reset timed $\mathcal{C}$-automaton from a a state-reset timed $\mathcal{W}$-automaton. □

Remark 15. This transformation will be useful in some of the following constructions. Indeed, the information needed when reaching a limit state is the set of infinitely repeated transitions, which is more precise than only the set of infinitely repeated states. A definition of limit states based on transitions instead of states would allow to get rid of lemma 14. 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 $TL = \bigcup_{n\geq 1} TL(n)$. As in the usual case of timed automata, the class TL is closed under union and intersection.

The interesting new fact about using Zeno words is the closure under concatenation, as well as under star and ω-power operations:

Theorem 16. *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 $\mathcal{W}$-n-automaton $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ and a timed $\mathcal{W}$-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 $\mathcal{W}$-$max(n,n')$-automaton.
2. In a similar way, a timed $\mathcal{W}$-$min(n,n')$-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 disjunction 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 $\mathcal{W}$-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 $q'_0 \in Q'_0$, then for each final state q of $\mathcal{A}$, the transition (q, A, a, ρ, q') is added. The set of final states is F' if $Q'_0 \cap F' = \emptyset$ 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 $\mathcal{W}$-automaton $\mathcal{B} = (\Sigma, S, J, G, \Omega, Y)$ by:
   - $S = Q \times \{0,1\}$
   - $J = I \times \{0,1\}$
   - $Y = X_0 \cup X_1$ 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).
   - Let i be in $\{0,1\}$ and let j be such that $\{i,j\} = \{0,1\}$.
     - Let q be in Q.
       * for each $(q, A, a, \lambda, q') \in \Delta$, $((q,i), B, a, \rho, (q',i)) \in \Omega$ with $B_i = A$, $B_j =$ True, $\rho_i = \alpha$ and $\rho_j = X$
       * $\forall q \in F, q_0 \in I$ and $(q_0, A, a, \alpha, q') \in \Delta$, $((q,i), B, a, \rho, (q',j)) \in \Omega$ with $B_i =$ True, $B_j = A$, $\rho_i = X$ and $\rho_j = \alpha$
     - Let $\mathbf{q} = \{q_1, ..., q_m\}$ be in $[Q]$. and $\mathbf{s} = \{(q_1,i)..., (q_m,i)\}$ be in $[S]$.
       * $\forall(\mathbf{q}, A, a, \alpha, q') \in \Delta$, $(\mathbf{s}, B, a, \rho, (q',i)) \in \Omega$ with $B_i = A$, $B_j =$ True, $\rho_i = \alpha$ and $\rho_j = X$
       * If $\mathbf{q} \in F$, $q_0 \in I$ and $(q_0, A, a, \alpha, q') \in \Delta$, $(\mathbf{s}, B, a, \rho, (q',j)) \in \Omega$ with $B_i =$ True, $B_j = A$, $\rho_i = X$ and $\rho_j = \alpha$

   We finally define the set G of final states.
   - If
   $$G = (F \cap Q) \times \{0,1\}$$
   $$\cup \{\{(q_1,i)..., (q_m,i)\} \in [S]; \{q_1, ..., q_m\} \in F \text{ and } i \in \{0,1\}\},$$
   then the language accepted by the automaton $\mathcal{B}$ 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 $G = \{\{(q_1,i_1)..., (q_m,i_m)\} \in [S]; \{q_1, ..., q_m\} \in F \text{ and } \{i_1, ...i_m\} = \{0,1\}\}$, then the language accepted by this automaton is exactly L^ω.
   - If L contains the empty word, we have to take $G = (F \cap Q) \times \{0,1\} \cup \{\{(q_1,i_1)..., (q_m,i_m)\} \in [S]; \{q_1, ..., q_m\} \in F\}$ 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, a positive answer allows the design of verification algorithms. Thus, Theorem 17 shows that our extension of timed automata retains the property of the original class.

Theorem 17. *Emptiness is decidable in the class TL.*

The proof follows the standard technique introduced in [1] and extended in [5]. For a timed language L, $Untime(L)$ is the (untimed) language obtained from L 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].

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

Lemma 18. *Let $\mathcal{A}$ be a timed automaton. There is a timed automaton $\mathcal{A}'$ accepting the same language, with the following property called On Time Divergence (OTD): if q is a state of type $k > 0$, such that there exists a transition going out from q in $\mathcal{A}'$, then, for any run of $\mathcal{A}'$ entering the state 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 the states of type 1. Moreover, since the OTD property is preserved by the transformations between $\mathcal{C}$- and $\mathcal{W}$-automata, we chose to explain the construction with $\mathcal{C}$-automata.

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. The construction is illustrated (for $n = 4$) in Figures 5 and 6 for the simplest case of a timed n-automaton $\mathcal{B}_n$, accepting all timed words with length less than or equal to ω^n, producing $\mathcal{B}'_n$.
In order to ensure On Time Divergence, the automaton $\mathcal{B}'_n = (\Sigma, Q, I, F, \Delta, X)$ has n states of type 0, $Q = \{1, \ldots, n\}$, and $n-1$ clocks $X = \{x_2, \ldots, x_n\}$. Again, the initial state is 1 and all states are final. For each $k \geq 1$, the states k, $\{k\}$, $\ldots$ (with respective type $0, \ldots, k-1$) and the clock x_k are used to produce only convergent time sequences reaching the ordinal ω^{k-1}, while the dead final state (of type k) $\{\cdots\{k\}\cdots\}$ 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 $\mathcal{A}'$. □

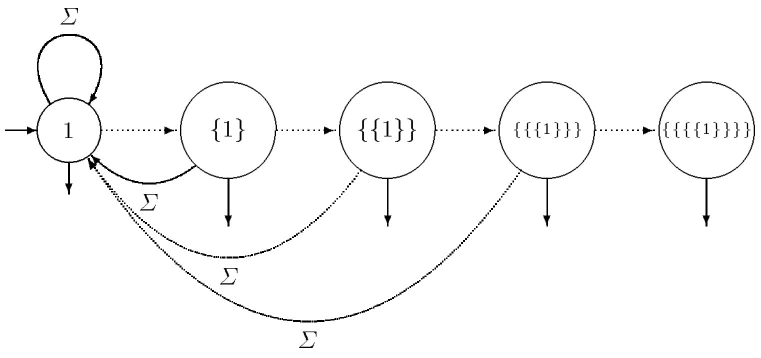


Fig. 5. The automaton $\mathcal{B}_4$

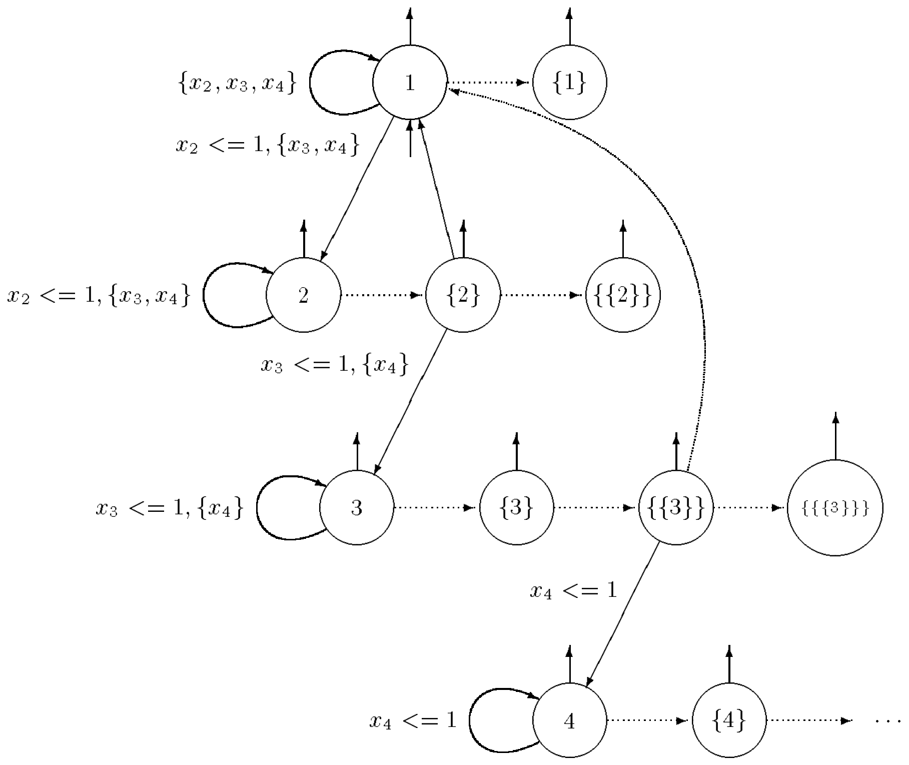


Fig. 6. The automaton $\mathcal{B}'_4$

Step 2: Definition of limit zones. We now assume $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ is a timed state-reset $\mathcal{W}$-automaton, for which the OTD property of first step holds.

Let $p = card(X)$ and let M be the largest constant appearing in the clock constraints of $\mathcal{A}$. Recall ([2]) that a clock constraint, called here a zone, can be considered as a union of equivalence classes, for an equivalence relation $\equiv_M$ defined on $\mathbb{R}_+^p$ (the set of all clock values). For this relation, the quotient set $\mathcal{V} = \mathbb{R}_+^p / \equiv_M$ is finite. The equivalence class of a point $z = (z_i)_{1 \leq i \leq p}$ in $\mathbb{R}_+^p$ is written $[z]_\equiv$ and is called a *region*. 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 $d \geq 0$ such that $z' = (z_i + d)_{1 \leq i \leq p}$ belongs to v'. The (finite) set of zones is written $\mathcal{Z}$.

As usual, we shall build new states of type 0, as pairs of the form (q, v), consisting of a state q of $\mathcal{A}$ and a region v in $\mathcal{V}$. The main problem to adapt the classical construction to $\mathcal{W}$-automata is to define in a consistent way the reachable states of type 1 : given a limit set $\mathbf{s} = \{(q_1, v_1), \cdots, (q_m, v_m)\}$ of type 0 states, we already know that the limit state is $\mathbf{q} = \{q_1, \ldots, q_m\}$ and we have to describe precisely which zone, denoted by $lim(\{v_1, \cdots, v_m\})$, is effectively reached. First notice that $Z = lim(\{v_1, \cdots, v_m\})$ is 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 R. 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 intersection of all the constraints appearing in the v_j's. There are three cases, defining three additional subsets of clocks :

- the resulting constraint is of the form $x = a_x$, $a_x \geq 0$ and $x \in X_s$, where the subscript s stands for "single value",
- it is of the form $x \in]a_x, b_x[$, $0 \leq a_x < b_x$ (where b_x is possibly infinite) and $x \in X_{oi}$, where oi means "open interval",
- it reduces to $false$ and $x \in X_e$, where e is for "empty".

We are now ready to define the limit zone of a set of regions.

Definition 19. Let $\{v_1, \cdots, v_m\})$ be a set of regions and let $X = X_0 \cup X_s \cup X_{oi} \cup X_e$ be the corresponding partition of the set of clocks, as introduced above. We consider four basic zones (with the usual convention: $\bigwedge_{x \in Y} A_x$ is $true$ if Y is empty)

- $Z_0 = \bigwedge_{x \in X_0}(x = 0)$,
- $Z_s = \bigwedge_{x \in X_s}(x = a_x)$,
- $v[X_{oi}] = \bigwedge_{j=1}^{m} v_{j|X_{oi}}$ is the open region obtained by the intersection of the restrictions to X_{oi} of the v_j's.
  The region $v[X_{oi}]$ yields an ordering on the fractional parts $fract(x)$ of the clocks x 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) b_x is finite. Then, the last zone is:

- $Z_m = \bigwedge_{x \in X_m} (x = b_x)$.

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

1. if the subset X_e is not empty, i.e. for some clock, one of the constraints is $false$, then $Z = \emptyset$.
2. if X_s is not empty, then $Z = Z_0 \wedge Z_s \wedge v[X_{oi}]$, and time does not progress along the cycle. 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 $Z = [0]_\equiv$ is reduced the origin point. If X_{oi} is not empty, then $Z = Z_0 \wedge Z_s \wedge (v[X_{oi}] \vee Z_m)$.

Step 3: Construction of the region automaton. We finally define a reduced $\mathcal{W}$-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 $S^{[1]} = [Q] \times \mathcal{Z}$,
- $I' = I \times \{[0]_\equiv\}$
- $F' = F \times \mathcal{Z}$
- if $s = (q, Z) \in S$, with $Z \neq \emptyset$, and $a \in \Sigma$, then $Z = V_1 \cup \ldots \cup V_k$ is a finite union of regions. A transition $((q, Z), a, (q', V'))$ is in Δ' if there exist 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 limit state $\mathbf{s} = \{(q_1, v_1), \cdots, (q_m, v_m)\}$ by:

$$\gamma(\mathbf{s}) = (\{q_1 \cdots, q_m\}, lim(\{v_1, \cdots, v_m\}))$$

The correction of this construction is ensured by the following lemma:

Lemma 20. *Let $\mathcal{A}$ be a timed $\mathcal{W}$-automaton, P a path in $\mathcal{A}$ starting from an initial state, and $\mathbf{s}$ a state of type 1 reached by this path. Then*

1. *Let x be a clock in X and $c > 0$ a constant, such that for any run r along the path P, the constraint $x = c$ is satisfied when reaching $\mathbf{s}$. Then the same constraint $x = c$ is satisfied in an infinite number of states having $lim(r) = \mathbf{s}$ as limit.*
2. *Let $r = (q_h, X(t_h))_{h \leq \alpha}$ be a run through P such that the state $\mathbf{s} = q_i = \{q_{h_1}, \cdots, q_{h_m}\}$ is reached at finite time t_i. If $v_h = [X(t_h)]_\equiv$, for each state q_h of type 0, then $Z_i = lim(\{v_{h_1}, \cdots, v_{h_m}\})$.*

Proof. 1. This property is just another formulation of Lemma 26, which is proved later (independently of this result) in Section 7.

2. The proof consists in looking at all the cases in the definition of the limit zone above.

□

Example. Figure 7 shows a timed automaton accepting the language

$$L = \{(a^\omega b, (t_i)_{i \leq \omega+1}) \text{ such that } 0 < t_0 < t_1 < \ldots < t_\omega \leq t_{\omega+1} < 1\},$$

for which $Untime(L) = \{a^\omega b\}$.

In Figure 8, which illustrates the construction of the region automaton accepting $Untime(L)$, the (type 1) 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. This corresponds to the third subcase in Definition 19 of the limit zone: for the region $v = (x = 0) \wedge (0 < y < 1)$, we have $Z_0 = (x = 0)$, $X_{oi} = \{y\}$, so that $v[X_{oi}] = 0 < y < 1$ and $Z_m = (y = 1)$. Therefore, $Z = lim(\{v\}) = Z_0 \wedge (v[X_{oi}] \vee Z_m) = (x = 0 \wedge 0 < y \leq 1)$.

The second type 1 state $(\{1\}, x = 0 \wedge y > 1)$ also corresponds to this same subcase, but this time, with $b_y = \infty$. Thus, for the region $v' = (x = 0) \wedge (y > 1)$, the limit zone is $Z' = lim(\{v'\}) = v'$.

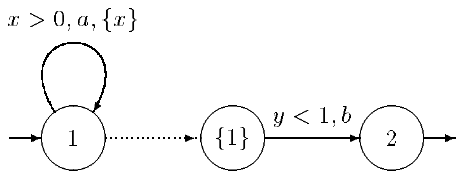


Fig. 7. A timed 2-automaton for L

Step 4: proof of theorem 17. Finally, we verify that $Untime(L) = L(\mathcal{A}')$ and the result holds by applying the decidability result of [9].
Note that, like $\mathcal{A}$, the automaton $\mathcal{A}'$ is state-reset and On Time Divergence holds.

(i) Let $u = (a_i)_{1 \leq i < \alpha}$ 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 $r = (q_i, X(t_i))_{0 \leq i \leq \alpha}$ 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 $i < \alpha$. We 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, Z_i is simply the equivalence class $[X(t_i)]_\equiv$ of $X(t_i)$, as usual, and the transition $((q_i, Z_i), a_{i+1}, (q_{i+1}, Z_{i+1}))$ is in Δ'.
- If $i < \alpha$ is a limit ordinal, then t_i finite (because of OTD) and Lemma 7 gives $X(t_i) = lim_{h \to i} X(t_h)$.
  Consider now $s_i = (q_i, Z_i) = \gamma(\{(q_{h_1}, Z_{h_1}), \cdots, (q_{h_k}, Z_{h_k})\})$. By induction hypothesis, from some point on, each $X(t_h)$ belongs to one of the regions Z_{h_j}'s. From Lemma 20 above, $X(t_i)$ belongs to Z_i. Write $Z_i = v_1 \cup \ldots \cup v_m$. 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 $[X_{i+1}]_\equiv$ is obtained from v by the reset of the clocks in ρ_{i+1}. Thus, setting $Z_{i+1} = [X_{i+1}]_\equiv$, the transition $((q_i, Z_i), a_{i+1}, (q_{i+1}, Z_{i+1}))$ is in Δ', with $X(t_{i+1}) \in Z_{i+1}$.

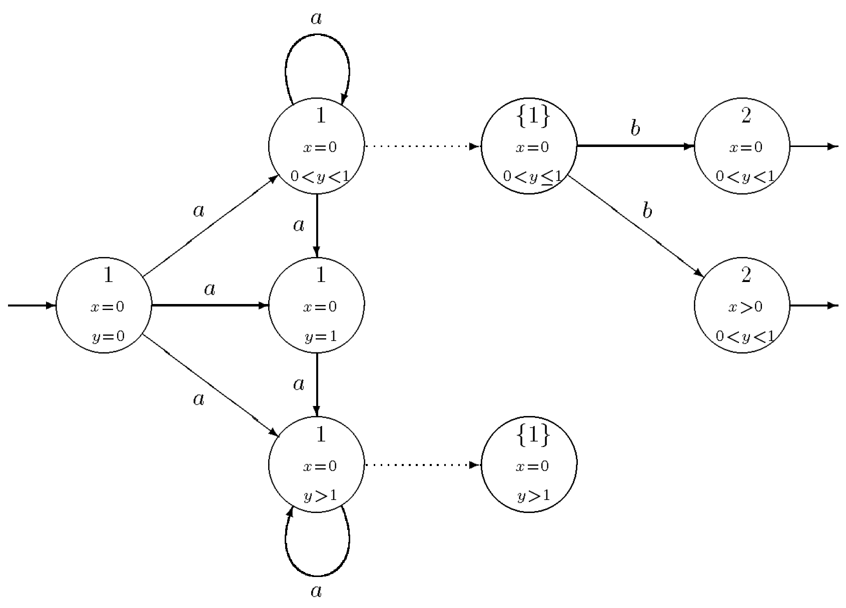


Fig. 8. The region automaton accepting $Untime(L)$

(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 d such that $X(t_i) + d \in v$. In this case, we set $t_{i+1} = t_i + d$ and, for each clock x, $x(t_{i+1}) = x(t_i) + d$ if $x \notin \rho_{i+1}$, $x(t_{i+1}) = 0$ otherwise.
- Let i be a limit ordinal, with $i < \alpha$. We obtain (Lemma 18) a finite time $t_i = lim_{h\to i} t_h$ and (Lemma 7) 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 $\mathcal{A}'$, $(q_i, Z_i) = \gamma(\{(q_{h_1}, Z_{h_1}), \cdots, (q_{h_k}, Z_{h_k})\})$, where the Z_{h_j}'s are regions. Again, Lemma 20 ensures that $X(t_i)$ belongs to Z_i. The only difficult case 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.
  - $*$ $Z_i = Z_0 \wedge Z_s \wedge (v[X_{oi}] \vee Z_m)$, and
  - $*$ there is some clock $x \in X_{oi}$ such that $x(t_i) = b_x$ belongs to Z_m.

  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 20 to build another partial run for which the maximal values in Z_m are not reached by the maximal clocks in X_{oi}. This is possible because the time sequence does not verify $x = b_x$ infinitely often. For this new run, the limit $X(t_i)$ belongs to the region $Z_i = Z_0 \wedge Z_s \wedge v[X_{oi}]$ and there is now a transition from $\mathcal{A}$. We can then 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 21. A *generalized transition* is a timed n-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 $q_0 \in Q_0$ 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 22. • Point 1 is required because we do not allow ε-transitions in timed n-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 values are either 0 if there was a reset, or $x(t_\alpha) = x(t_0) + t_\alpha - t_0$ otherwise.
- Given a timed n-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 17) or at the state-reset automaton obtained by the construction of lemma 14.

Examples.

1. An automaton is reset-free ([3]) if the transitions do not reset any clock. Clearly, a reset-free automaton is a generalized transition and we denote by RF the corresponding subclass of GT. Figure 9 shows two basic examples of reset-free automata. The first one is only a Muller timed automaton and the second one accepts a language of finite timed words.

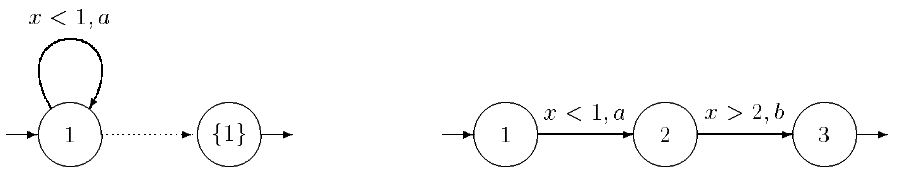


Fig. 9. Two reset-free timed automata

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 $F = \{Q\} \in [Q]^1$. The final condition is a Muller condition of acceptance: an accepting run goes infinitely often through each state 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 5).

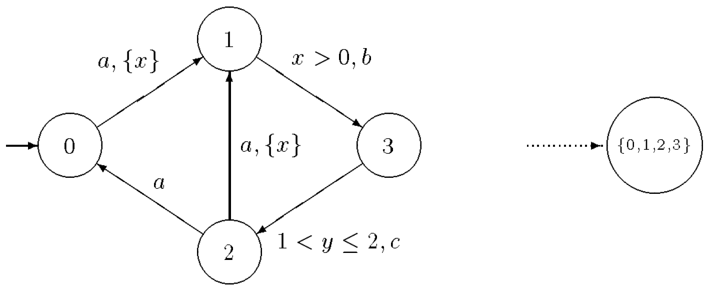


Fig. 10. A state-reset cycle

3. In [7], two particular cases of generalized transitions 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. Figure 11 shows a generalized transition, corresponding to instantaneous executions.
   - 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 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.

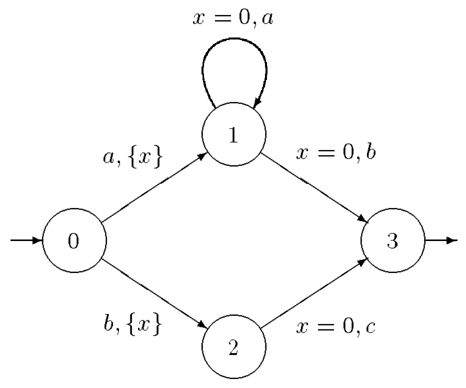


Fig. 11. An "instantaneous" finite automaton

A generalized transition 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 $(x(t))_{x \in X}$. Note that, again from lemma 5, 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 this 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 23. A *refinement* of L is given by a family $(\mathcal{A}_a)_{a\in\Sigma}$ of generalized transitions over some other 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:

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

It should be noticed 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 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* by: if $w = (a_i, t_i)_{i<\alpha}$ and $w' = (a'_i, t'_i)_{i<\alpha'}$, with $\tau(w) \leq t'_0$, then

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

Note that the execution of two consecutive transitions over (a,t) and (b,t') respectively, 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:

$\sigma((a,t) \circ (b,t'))_{X(t_0)} = \{w \circ w' / w \in (L(\mathcal{A}_a))_{X(t_0)},\ \tau(w) = t, w' \in (L(\mathcal{A}_b))_{X(t)}$ for some $X(t) \in \mathcal{X}(w, X(t_0))$ 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, the initial time is chosen as $t_0 = 0$ and the initial values of the clocks are all equal to zero: $x(0) = 0$, for each clock $x \in X$.

If L and L' are two languages of transfinite words, we say that L' is a refinement of L if there exists a timed refinement σ such that $L' = \sigma(L)$. As already remarked in Subsection 6.1, this definition of refinement generalizes those introduced in [7].

Remark 24. 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 equivalent 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 $a_I = \{(a,t)/t \in I\}$. 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 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 $\sigma_i(a)$ reset 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 $\circ$.

7 The main results about timed refinement

Before stating precisely the results, we describe a problem arising in the constructions of refinements.

7.1 Basic cases of timed refinement

Consider again the language (introduced in Section 6) $a_I = \{(a,t)/t \in I\}$, 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 the language a_I is the set of words in L_a with ending time in I:

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

It is easy to see that $\{w/w \text{ is finite and } \tau(w) \in I\}$ 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 $L_I^{inf} = \{w/w \text{ is infinite and } \tau(w) \in I\}$ is not recognizable in general. In fact, Lemma 26 and 27 below give precise results.

Lemma 26. *The language $L_I^{inf} = \{w/w$ 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 $0 \leq c_1 < c_2$.*

Proof. It is proved in [11] that L_I^{inf} cannot be accepted by a timed automaton, when $I = [0, c_2[$. It remains now to prove that the language $L_{\{c\}}^{inf}$ is not recognizable for any positive real number c. Assume there is a Muller timed automaton $\mathcal{A}$ accepting $L_{\{c\}}^{inf}$. 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:

$$\begin{tikzcd}
(q_0, X(t_0)) \arrow[r, "A_0, a_0, \rho_0", "t_0"'] & (q_1, X(t_1)) \arrow[r, "A_1, a_1, \rho_1", "t_1"'] & (q_2, X(t_2)) \arrow[r] & \cdots & (q_\omega, X(t_\omega))
\end{tikzcd}$$

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 $x \in I$ where I 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.

So 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.

Lemma 27. *Let L be any timed language in TL. Then $L_I = \{w\,;\, w \in L \,/\, \tau(w) \in I\}$ is in TL for any interval I of the form $]c_1, c_2]$, $[0, c_2]$, $]c_1, \infty[$, for real numbers c_1, c_2 such that $0 \leq c_1 < c_2$.*

Proof. • To obtain an automaton accepting $L_{[0,c_2]}$, we start from an automaton accepting L and we add a new clock x, with the additional constraint $x \leq c_2$ 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.

□

7.2 Closure of the class *TL* under 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 $TL(1)$.

Theorem 28. *Let L be a language in $TL(n)$ over an alphabet Σ, accepted by a timed $\mathcal{W}$-automaton $\mathcal{A}$ with a set X of clocks, and let σ be a timed refinement. For each $a \in \Sigma$, we consider the generalized transition $\sigma(a) = \mathcal{A}_a$, the language $L_a = L(\mathcal{A}_a)$ and α_a, the subset of clocks of X reset in $\mathcal{A}_a$. If, for each $a \in \Sigma$,*

1. *for each transition $\delta = (A, a, \alpha)$ in $\mathcal{A}$ labelled by a, we have $\alpha_a \subset \alpha$ and,*
2. *(a) either the language L_a contains only finite words,*
   (b) or for each transition δ in $\mathcal{A}$ labelled by 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 α 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 26 explains why condition 2(b) in Theorem 28 is necessary to build a refinement, when condition 2(a) does not hold.

Proof. Refining an action a by a language L_a, we have to replace in $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$ each transition with label (A, a, α) by a timed automaton accepting $L_a \cap \{w/X(\tau(w)) \in A\}$. Write Δ_a the subset of transitions labeled by a in Δ. We may suppose that no transitions in Δ_a are loops and we consider a

$\mathcal{W}$-automaton $\mathcal{A}_a = (\Sigma, R, J, G, \omega, X)$ accepting L_a. We may also suppose that J is reduced to a single state j with no loops and that G is reduced to a single state g which is a dead state. We give the construction of the timed automaton accepting $\sigma(L)$ in the two cases of Theorem 28.

First case: $g \in R$, i.e. L_a only contains finite words.
In this case, the construction is straigthforward.
For all δ in Δ, let $\mathcal{A}_{a,\delta} = (\Sigma, R_\delta, \{j_\delta\}, \{g_\delta\}, \omega_\delta, X)$ be distinct copies of $\mathcal{A}_a$. The $\mathcal{W}$-automaton $\mathcal{B} = (\Sigma, S, K, H, \Omega, X)$, defined below, accepts $\sigma(L)$.

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

Second case: $g \in [R]$.

In this case, the construction is more involved : we replace a transition (q, A, a, α, r) by $\mathcal{A}_a$. But a transition leaving $\mathcal{A}_a$ to return in r is an implicit one. A clock x in α, 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, these two copies 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 $\rho_0 = \{x_0; x \in \rho\}$ and $\rho_1 = \{x_1; x \in \rho\}$.
We consider the reduced $\mathcal{W}$-automaton

$$\mathcal{A}' = (\Sigma, Q', I', F', \gamma', \Delta', X' = X \cup X_0 \cup X_1)$$

defined by :

- $Q'^{[0]} = Q \times \{0,1\}^X$
- $Q^{[1]} = [Q] \times \{0,1\}^X$
- $I' = I \times \{(0)_{x \in X}\}$
- $F' = F \times \{0,1\}^X$
- 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, ρ, r) in $\Delta \setminus \Delta_a$ (i.e. $b \neq a$) and for all ε in $\{0,1\}^X$, $((q, \varepsilon), B, b, \rho \cup \{x_{\varepsilon_x}; x \in \rho\}, (r, \varepsilon)) \in \Delta'$

- 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 $((q,\varepsilon), A, a, \rho \cup \{x_{\varepsilon'_x} ; x \in \rho\}, (r, \varepsilon')) \in \Delta'$
- If $\{(q^j, \varepsilon^j) ; j \in \{1, ..., k\}$ is a subset of $[Q']$, with $\varepsilon^j = (\varepsilon^j_x)_{x \in X}$, define $\varepsilon = (\varepsilon_x)_{x \in X}$ by : Let x be in X
  - $*$ If all ε^j_x are equal, ε_x is their common value.
  - $*$ Otherwise $\varepsilon_\delta = 0$ by default.
- $\gamma'(\{(q^j, \varepsilon^j) ; j \in \{1, ..., k\}\}) = (\{q^j ; j \in \{1, ..., k\}\}, \varepsilon)$

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 $\delta' = ((q,\varepsilon), A, a, \alpha \cup \{x_{\varepsilon_x} ; x \in \alpha\}, (r, \varepsilon'))$ in Δ'_a defined as above from a transition $\delta = (q, A, a, \rho, r)$ of Δ_a. We consider the generalized transition $\mathcal{A}_{\delta'} = (\Sigma, R_{\delta'}, J_{\delta'} = \{j_{\delta'}\}, G_{\delta'} = \{g_{\delta'}\}, \omega_{\delta'}, \gamma_{\delta'}, X' = X \cup X_0 \cup X_1)$ which is 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 α by its copy x_{ε_x}. Then each clock x in α and its other copy $x_{\varepsilon'_x}$ are reset on each transition.
- the automaton is modified according to Lemma 27 to accept instead of L_a the language $L_a \cap \{w / X_\delta(\tau(w)) \in A\}$ where $X_\delta = \{x \in X ; x \notin \alpha\} \cup \{x_{\varepsilon_x} ; x \in \alpha\}$.

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

- $S^{[0]} = Q'^{[0]} \cup_{\delta' \in \Delta'_a} R^{[0]}_{\delta'}$
- $S^{[1]} = Q'^{[1]} \cup_{\delta' \in \Delta'_a} R^{[1]}_{\delta'}$
- $K = I' \cup \{j_{\delta'} ; \delta' = (\mathbf{q}, A, a, \alpha, \mathbf{r}) \in \Delta'_a$ such that $\mathbf{q} \in I'\}$
- $H = F' \cup \cup\{g_{\delta'} ; \delta' = (\mathbf{q}, A, a, \alpha, \mathbf{r}) \in \Delta'_a$ such that $\mathbf{r} \in F'\}$
- $\Omega = (\Delta' - \Delta'_a) \cup_{\delta' \in \Delta'_a} \omega_{\delta'} \cup \bar{\omega}_{\delta'}$ where, for $\delta' = (\mathbf{q}, A, a, \alpha, \mathbf{r})$ in Δ', $\bar{\omega}_{\delta'}$ is the set of the following transitions :
  - If $\delta'' = (\mathbf{s}, B, b, \beta, \mathbf{q})$ with $b \neq a$, then $\mathbf{s}, B, b, \beta, j_\delta) \in \bar{\omega}_{\delta'}$.
  - If $\delta'' = (\mathbf{s}, B, a, \beta, \mathbf{q}) \in \Delta'$, then $(g_{\delta''}, B, a, \beta, j_{\delta'}) \in \bar{\omega}_{\delta'}$.
  - If $\delta'' = (\mathbf{r}, B, b, \beta, \mathbf{s}) \in \Delta'$ with $b \neq a$, then $(g_{\delta'}, B, b, \beta, \mathbf{s},) \in \bar{\omega}_{\delta'}$.
- $\Gamma(T)$ for $T \subset [S^{[0]}]$ is defined by :
  - $\Gamma(T) = \gamma(T)$ if $T \subset Q'^{[0]}$
  - $\Gamma(T) = \gamma_{\delta'}(T)$ if there exists δ' in Δ'_a such that $T \subset R^{[0]}_{\delta'}$
  - $\Gamma(T) = T \cap Q'^{[0]} \cup \{\mathbf{q}, \mathbf{r} ; \exists \delta' = (\mathbf{q}, A, a, \alpha, \mathbf{r}) \in \Delta'_a$ such that $\{j_{\delta'}, g_{\delta'}\} \subset T\}$

□

As a simple example, consider again the automaton in Figure 2, accepting the language L and suppose we want to build the automaton accepting $\sigma(L)$, where σ is the mapping which leaves a and b unchanged and associates with c the timed language

$$L_c = \{(a_i, t_i)_{i<\omega} \;/\; \text{the time sequence } (t_i)_{i<\omega} \text{ is strictly increasing and } a_i = d \text{ for each i}\}.$$

Two generalized transitions $\mathcal{A}_c$ and $\mathcal{B}_c$ corresponding to L_c are represented in Figure 13, where $\mathcal{B}_c$ is the modification required from $\mathcal{A}_c$ to remove the self-loop. The timed automaton accepting $\sigma(L)$ is represented in Figure 14.

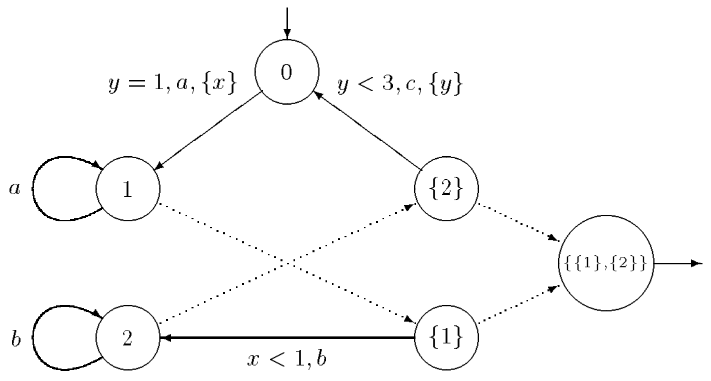


Fig. 12. A timed automaton $\mathcal{A}$ accepting L

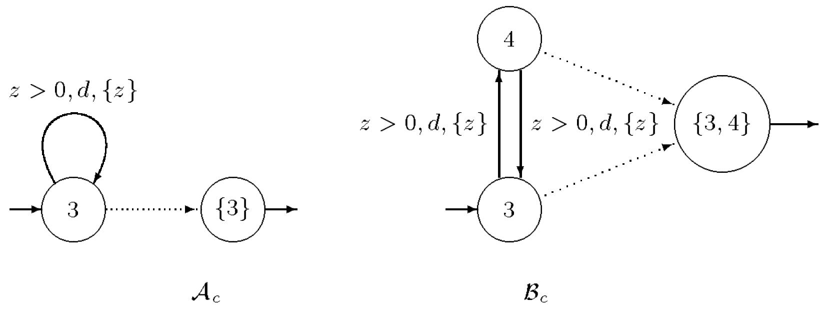


Fig. 13. Generalized transitions for σ

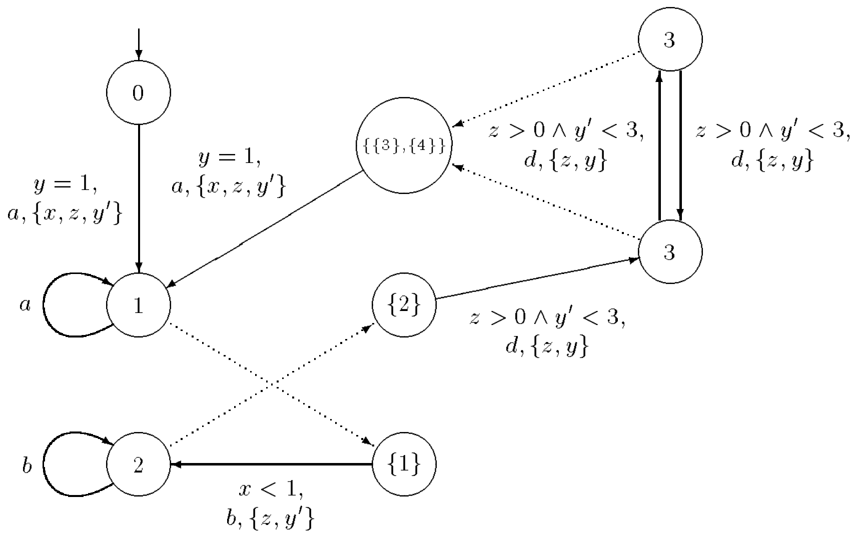


Fig. 14. A timed automaton accepting $\sigma(L)$

7.3 A decomposition theorem

The last result 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.

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

Note that conditions of Theorem 28 holds for the refinement constructed in Theorem 31.

Proof. We use here $\mathcal{C}$-automata for practical reasons. 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 14, 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 states of type 1. We first associate with each non empty subset P of Q':

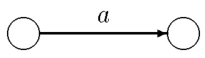
- the state-reset cycle $\mathcal{A}_P = (\Sigma', P, P, \{P\}, \Delta', X)$,
- the set $\Delta'_P = \{D \subseteq \Delta'/st(D) = P\}$, where st is defined for a transition $\delta = (q, A, a, \rho, q')$ by $st(\delta) = q'$ and $st(D) = \{st(\delta)/\delta \in D\}$,
- the clock constraint $A_P = \bigvee_{D \in \Delta'_P} A_D$, with $A_D = \bigwedge_{\delta \in D} \overline{A_\delta}$, for $D \in \Delta'_P$, where A_δ is the constraint associated with the transition δ (recall that $\overline{A}$ is the upper closure of the constraint A, i.e. the constraint obtained from A by adding the limits of non-decreasing sequences of elements in A),

- the subset of clocks $\rho_P = \cup_{D \in \Delta'_P} \cup_{\delta \in D} \rho_\delta$, where ρ_δ is the constraint associated with the transition δ,
- a new letter a_P, not in Σ'.

We consider the new alphabet $\Sigma^1 = \{a_P, P \in [Q']^1\}$ and $\Sigma = \Sigma' \cup \Sigma^1$ and we build a new automaton that will contain transitions of the form (q, A_P, a_P, ρ_P, P), so that P becomes a state of type 0. More precisely, we define a n-automaton over the alphabet Σ, $\mathcal{A} = (\Sigma, Q, I, F, \Delta, X)$, accepting L, with:

- $Q = Q' \cup [Q']^1$ and $I = I'$
- if (q, A, a, ρ, q') is a transition of Δ', with $q \in Q$ and $q' \in Q'$, then it is also a transition in Δ
- (q, A_P, a_P, ρ_P, P) is in Δ for all $q \in P$, $P \in [Q']^1$
- in order to define the transitions from states of type i, $1 \leq i \leq n$ in Δ, we define inductively a sequence of mappings γ_i from $[Q]^i$ into $[Q']^{i+1}$, $1 \leq i \leq n$ by: $\gamma_1(q) = q \cap [Q']^1$ for $q \in [Q]^1$ and for $i \geq 2$, if $q = \{q_1, \cdots, q_k\}$ is in $[Q]^i$, then $\gamma_i(q) = \{\gamma_{i-1}(q_1), \cdots, \gamma_{i-1}(q_k)\}$. A transition (q, A, a, ρ, q'), $q \in [Q]^i$, $q' \in Q$ belongs to Δ if and only if $(\gamma_i(q), A, a, \rho, q')$ 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, a state $q \in [Q]^i$ belongs to F if and only if $\gamma_i(q)$ belongs to F'.

Finally, we define a refinement σ by the mapping from Σ into GT: for $a_P \in \Sigma^1$, $\sigma(a_P) = \mathcal{A}_P$, and for $a \in \Sigma'$, $\sigma(a)$ is the elementary untimed transition (in GT) labelled by a:



Finally, it is easy from the construction above to prove that $\sigma(L) = L'$.

□

Note that this theorem 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 in Remark 24.

8 Conclusion

In this work, we developed two original studies.

The first one is devoted to timed automata accepting sequences of Zeno words. We believe that some physical phenomena deserve to be described by such sequences: those where an infinite number of actions occur in a finite lapse of time. 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 a new algorithm 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. On a decision method in restricted second order arithmetic. In *Proceedings of the International Congress on Logic, Methodology and Philosophy*, pages 1–11. Stanford University Press, 1962.
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.