Characterization of the expressive power of silent transitions in timed automata

Béatrice Bérard[1], Volker Diekert[2] Paul Gastin[3] and Antoine Petit[1]

[1] LSV, CNRS URA 2236, ENS de Cachan, 61 av. du Prés. Wilson,
F-94235 Cachan Cedex, France, berard,petit@lsv.ens-cachan.fr
[2] Inst. für Informatik, Universität Stuttgart, Breitwiesenstr. 20-22,
D-70565 Stuttgart, Germany, diekert@informatik.uni-stuttgart.de
[3] LIAFA, Université Paris 7, 2, place Jussieu,
F-75251 Paris Cedex 05, France, gastin@liafa.jussieu.fr

Abstract. Timed automata are among the most widely studied models for real-time systems. Silent transitions, i.e., ε-transitions, have already been proposed in the original paper on timed automata by Alur and Dill [3]. We show that the class TL_ε of timed languages recognized by automata with ε-transitions, is more robust and more expressive than the corresponding class TL without ε-transitions

We then focus on ε-transitions without reset, i.e. ε-transitions which do not reset clocks. We propose an algorithm to construct, given a timed automaton, an equivalent one without such transitions. This algorithm is in two steps, it first suppresses the cycles of ε-transitions without reset and then the remaining ones.

Then, we prove that a timed automaton such that no ε-transition which resets clocks lies on any directed cycle, can be effectively transformed into a timed automaton without ε-transitions. Interestingly, this main result holds under the assumption of non-Zenoness and it is false otherwise.

To complete the picture, we exhibit a simple timed automaton with an ε-transition, which resets some clock, on a cycle and which is not equivalent to any ε-free timed automaton. To show this, we develop a promising new technique based on the notion of precise action.

This paper presents a synthesis of the two conference communications [9] and [13].

1 Introduction

Finite transition systems form one of the most intensively studied models for the specification and verification of systems, see e.g. [11,23,24,27]. By essence, these classical transition systems are suitable for reasoning about qualitative properties. Time requirements cannot be easily described by these automata which do not contain an explicit notion of time. Several attempts have been made to propose models based on transition systems together with a suitable notion of time. For instance, timed Petri nets [22], timed transition systems [21,16], timed graphs [20] or timed I/O automata [19] have been discussed, leading to the general notion of hybrid automata, which have received considerable attention recently. Let us refer to the survey by Henzinger in [14]. We focus in this paper on the basic and natural model of so-called timed automata proposed by Alur and Dill [3,4]. Since its introduction, this model has been largely studied under several aspects: determinization [5], minimization [1], power of clocks [2,15], extensions of the model [6,8] and logical characterization [26] have been considered in particular. A self-contained exposition on timed automata and a precise logical characterization of their languages has been given by Wilke in [25], see also [26]. Moreover this model has been successfully used for verification and specification of real-time systems [12,28,17].

In the original paper [3], silent transitions (ε-transitions) in timed automata have been considered but, they disappeared in most of the following papers on timed automata (even in the extended version [4]). It is well known that in the untimed case, such transitions do not increase the power of (classical) finite automata and that these transitions are just a technical, even though very useful, tool.

We show in this paper that in the case of timed automata, the situation is completely different. We first recall the basic notions concerning timed automata in Section 2. Then, we prove that ε-transitions strictly increase the power of timed automata. Moreover, we show that the class TL_ε of timed languages recognized by such automata with ε-transitions is much more robust than the class TL of timed languages recognized by timed automata without ε-transitions. For instance, any recognizable timed language over the time domain $\mathbb{N}$ remains recognizable if the time domain is $\mathbb{R}$. Note that it is surprisingly not the case for timed automata without ε-transitions. Following similar ideas, we exhibit some very natural timed languages which become recognizable with the help of ε-transitions.

Then we propose some normal forms for timed automaton (Section 4) which will simplify the constructions of the next sections. First we briefly recall the well-known notions of disjunction free and diagonal free timed automata. Then we deal with the problem of finite timed words accepted by infinite runs (eventually performing ε-transitions only). We give an effective construction transforming a timed automaton into an equivalent one, where all finite words are accepted by finite runs.

We can then start the core of our paper: the investigation of the expressive power of ε-transitions in timed automata. We first focus on the ε-transitions which do not reset clocks. We show in section 5 how to suppress cycles of such ε-transitions. The next section is devoted to the study of closures of clock constraints. We prove that these closures are still clock constraints and can thus be added to timed automata. We use these constraints in Section 7 to derive an algorithm constructing from a timed automaton, in which ε-transitions do not reset clocks, an equivalent one without ε-transitions.

After this result, it remains to study the power of ε-transitions which reset clocks, so-called ε-reset transitions. We show that a timed automaton with ε-transitions can be effectively transformed into a timed automaton without ε-

transitions, if no ε-reset transition lies on any directed cycle of the automaton. Moreover and surprisingly, this result holds only under the assumption of non-Zenoness, otherwise it becomes false. Our main construction is quite involved and leads to some huge state explosion. However, this is just a serious argument in favor of ε-transitions. We may use them in order to have a compact and concise specification for languages recognized by automata without ε-reset transitions, although no quantitative assertion about this statement can be given at this moment.

Finally, we exhibit a very simple timed automaton with some cycle containing an ε-reset transition and in which no path uses two consecutive ε-transitions and whose language cannot be accepted without ε-transition. To this purpose, we develop a new technique based on a notion of *precise action*, which appears to be very promising by itself. This new notion yields a formal tool in order to show that some timed languages are not recognizable by any timed automaton (without ε-reset transitions).

2 Timed languages with non observable actions

In this section, we briefly recall the basic definitions and notations for timed words and finite automata with timing constraints, as well as classical properties of the corresponding families of timed languages. They come mostly from [3] and [4], with a slight difference for the non Zeno property, as we explain below.

2.1 Timed words and clocks

Let Z be any set. We write Z^* (respectively Z^ω) the set of finite (respectively infinite) sequences of elements in Z, with ε for the empty sequence, and $Z^\infty = Z^* \cup Z^\omega$ the set of all sequences of elements in Z.

Throughout this paper, we consider a time domain $\mathbb{T}$ which is either the set of non-negative integers $\mathbb{N}$ or the set of non-negative reals $\mathbb{R}_+$. A time sequence over $\mathbb{T}$ is a finite or infinite non decreasing sequence $\tau = (t_i)_{i\geq 1}$ of elements in $\mathbb{T}$. The time sequence τ is strict if all its elements are distinct and τ is divergent if the sequence is unbounded.

The observable actions of the processes we consider are given by a finite set Σ. A timed word is an element of $(\Sigma \times \mathbb{T})^\infty$, $w = (a_i, t_i)_{i\geq 1}$, also written as a pair $w = (\sigma, \tau)$, where $\sigma = (a_i)_{i\geq 1}$ is a word in Σ^∞ and $\tau = (t_i)_{i\geq 1}$ is a timed sequence in $\mathbb{T}^\infty$ of same length than σ. For instance, $(a, 1.5)(b, 2)(a, \pi)(b, 3.9)(a, 3.9)$ is a finite timed word, while $((ab)^\omega, (i + 1/i)_{i\geq 1})$ is an infinite timed word. Note that, contrary to the usual definition of a time sequence, we do not include the non Zeno property, i.e. the divergence of an infinite sequence. We will see in Section 2.2 that, if needed, this condition can be ensured through the intersection with an appropriate timed language.

Let X be a set of variables with values in $\mathbb{T}$, called clocks. The clock constraints over X are the formulas defined by the grammar:

$$A ::= x \# c \mid x - y \# c \mid A \vee A \mid A \wedge A \mid \mathit{true} \mid \mathit{false},$$

where x and y are clocks, $c \in \mathbb{T}$ and $\#$ is a binary operator in $\{<, =, >\}$. The formulas of the form $x \# c$ or $x - y \# c$ are called atomic formulas. A formula with negations can be expressed by an equivalent formula in the grammar above, so that negation needs not be included in the definition. The set of all clock constraints over X is denoted by $\mathcal{C}(X)$. A clock valuation $v : X \mapsto \mathbb{T}$ is a mapping that assigns to each clock x a time value $v(x)$. The set of all clock valuations is denoted $\mathbb{T}^X$. We write $v \models A$ when the clock valuation v satisfies the clock constraint A. If t is an element of $\mathbb{T}$ and α a subset of X, the valuations $v + t$ and $v[\alpha \leftarrow 0]$ are defined respectively by $(v + t)(x) = v(x) + t$, for each clock x in X and $(v[\alpha \leftarrow 0])(x) = 0$ if $x \in \alpha$, $v(x)$ otherwise. In the same way the valuation $v - t$ is defined only if $v(x) \geq t$, for each clock $x \in X$ and in this case, $(v - t)(x) = v(x) - t$, for each clock x in X.

2.2 Timed automata with ε-transitions

A Büchi timed automaton over $\mathbb{T}$ is a tuple $\mathcal{A} = (\Sigma, Q, T, I, F, R, X)$, where Σ is a finite alphabet of observable actions, Q is a finite set of states, X is a finite set of $\mathbb{T}$-valued clocks, $T \subseteq Q \times [\mathcal{C}(X) \times (\Sigma \cup \{\varepsilon\}) \times \mathcal{P}(X)] \times Q$ is the set of transitions, $I \subseteq Q$ is the subset of initial states, $F \subseteq Q$ is the subset of final states and $R \subseteq Q$ is a subset of repeated states corresponding to a Büchi condition, as recalled below. Thus, a transition has the form (p, A, a, α, q), where A is a constraint, a is an action in $\Sigma \cup \{\varepsilon\}$ and $\alpha \subseteq X$ is the subset of clocks to be reset. A transition (p, A, a, α, q) is called an ε-transition or a silent transition when $a = \varepsilon$. If in addition, $\alpha = \emptyset$ then we say ε-transition without reset, while we use ε-reset transition when $\alpha \neq \emptyset$.

The finite set of constants occurring in the constraints of $\mathcal{A}$ is written $C(\mathcal{A})$. Note that in most papers, the set $C(\mathcal{A})$ is assumed to be contained in the set $\mathbb{Q}$ of rational numbers. There can be two motivations for this hypothesis: first, one can effectively compute sums, differences and orderings of these constants, and second, the subgroup generated by $C(\mathcal{A})$ is discrete. This latter property is useless for the results of the present paper, we only need to compute effectively sums, differences and orderings, which is possible if C is constituted of recursive real numbers, whose rational dependencies are known[1]. Therefore, throughout the paper, we will assume that the constants appearing in the constraints of $\mathcal{A}$ are rational combinations of rationally independent recursive reals.

In order to define how a timed word is recognized by a timed automaton, we recall the notions of path and of timed run through a path. A path in $\mathcal{A}$ is a

[1] A real number c is recursive if there exists a recursively enumerable sequence of rational numbers $(c_n)_{n \geq 0}$ such that $|c - c_n| < 2^{-n}$, for all $n \geq 0$. Equivalently, a real number is recursive if and only if its successive digits, say in a binary notation, can be enumerated by a Turing machine. For instance, π, e and all algebraic numbers are recursive reals. Let $a_1, \cdots, a_p$ be p rationally independent recursive real numbers. Then, in the set $\mathbb{Q}(a_1, \cdots, a_p)$ of rational combinations of $a_1, \cdots, a_p$, one can effectively compute sums, differences and orderings.

finite or infinite sequence of consecutive transitions:

$$P = q_0 \xrightarrow{A_1, a_1, \alpha_1} q_1 \xrightarrow{A_2, a_2, \alpha_2} q_2 \cdots$$

where $(q_{i-1}, A_i, a_i, \alpha_i, q_i) \in T$ for all $i > 0$. The path P is accepting, if it starts in an initial state and either it is finite ending in a final state or it is infinite and a Büchi condition holds: $inf(P) \cap R$ is nonempty, where $inf(P)$ is the set of states which occur infinitely often in P.

Equivalently, other conditions like Muller or extended Büchi conditions (see e.g. [15]) can also be defined for infinite paths. Since both conditions can be simulated with a Büchi condition (and the other way around), we will use only Büchi conditions in this work.

A run of the automaton through the path P is a sequence of the form:

$$q_0 \xrightarrow[t_1]{A_1, a_1, \alpha_1} q_1 \xrightarrow[t_2]{A_2, a_2, \alpha_2} q_2 \cdots$$

where t_i is the time at which transition $p_{i-1} \xrightarrow{A_i, a_i, \alpha_i} p_i$ has been executed. Intuitively, this transition can be performed, only if the clock constraint A_i are satisfied by the current clock valuation and the clocks in α_i are reset at that time. More precisely, the value of clock x at time t is written $x(t)$, so that the clock valuation at time t can be written as the tuple $X(t) = (x(t))_{x \in X}$. Beginning with time $t_0 = 0$ and clock values $x(t_0) = 0$, for all x, the clock valuation at time t is completely determined by the run: for each $i \geq 1$,

$$x(t_i) = \begin{cases} 0 & \text{if } x \in \alpha_i \\ x(t_{i-1}) + t_i - t_{i-1} & \text{otherwise} \end{cases}$$

and $x(t) = x(t_{i-1}) + t - t_{i-1}$ for $t_{i-1} \leq t < t_i$. Moreover, we require that $X(t_{i-1}) + t_i - t_{i-1} \models A_i$ for all $i \geq 1$.

A pair (q, v), where q is a state of $\mathcal{A}$ and $v = X(t)$ is a clock valuation is called an *extended state* of $\mathcal{A}$.

With every finite (infinite resp.) run we can associate in a natural way a finite (infinite resp.) timed ε-word

$$(a_1, t_1)(a_2, t_2) \cdots \in ((\Sigma \cup \{\varepsilon\}) \times \mathbb{R})^{\infty}.$$

Since an ε-transition is viewed as an invisible action we may cancel all pairs (a_i, t_i) where $a_i = \varepsilon$. In this way we obtain a timed word (which might be finite even if the underlying timed ε-word is infinite):

$$(a_{i_1}, t_{i_1})(a_{i_2}, t_{i_2}) \cdots \in (\Sigma \times \mathbb{R})^{\infty}.$$

The timed language accepted by the timed automaton $\mathcal{A}$ over the time domain $\mathbb{T}$ is the set denoted by $L(\mathcal{A}, \mathbb{T})$, or simply $L(\mathcal{A})$ if there is no ambiguity about the time domain, which contains the timed words in $(\Sigma \times \mathbb{T})^{\infty}$ generated by (finite or infinite) accepting runs of $\mathcal{A}$. Two automata are said to be equivalent if they accept the same timed language.

Note that an infinite timed word is always generated by an infinite run, whereas a finite timed word may be generated either by a finite run or by an infinite timed run which eventually performs ε-transitions only. In the untimed case, it is easy to obtain, from a given automaton, another one accepting the same language, with the additional property that a finite word can always be accepted by a finite run. In the case of timed automata, the problem is slightly more difficult and a solution is given in Section 4.

The set of all timed automata will be denoted by TA_ε whereas the set of all timed automata using transitions with observable actions only (i.e. without silent transitions) will be denoted by TA. We also consider the subclass RTA_ε (for restricted TA_ε) of TA_ε, containing the timed automata in which ε-transitions do not reset clocks. The family of timed languages over the timed domain $\mathbb{T}$ generated by some timed automaton of TA_ε (TA, RTA_ε resp.) is denoted by $TL_\varepsilon(\mathbb{T})$ ($TL(\mathbb{T})$, $RTL_\varepsilon(\mathbb{T})$ resp.) or simply TL_ε (TL, RTL_εresp.) if the time domain has not to be explicitly stated.

We now recall some classical properties of these families of timed languages.

Proposition 1 ([3,4]). *1. The classes TL and TL_ε are closed under (finite) union and intersection.*
2. When only rational constants are used in the clock constraints, emptiness is PSPACE-complete in the classes TL and TL_ε.
3. The universality problem for timed automata is undecidable.

The point 3 of Proposition 1 is related to the fact that the classes TL and TL_ε are not closed under complement [4]. Also, a consequence of the first part of this proposition concerns the conditions usually required for time sequences: in some papers, time sequences are strictly increasing and/or infinite time sequences must be divergent (non Zeno). We did not include these requirements in our definition since they can be enforced respectively by constructions involving the languages *Strict* of timed words with strict timed sequences and *NZ* of timed words with divergent timed sequences.

$$\begin{aligned} Strict &= \{(a_i, t_i)_{i\geq 1} \in (\Sigma \times \mathbb{T})^\infty \mid 0 < t_1 < t_2 < \cdots\} \\ NZ &= \{(a_i, t_i)_{i\geq 1} \in (\Sigma \times \mathbb{T})^\omega \mid 0 \leq t_1 \leq t_2 \leq \cdots \text{ and } \forall t \in \mathbb{T}, \exists i : t < t_i\} \end{aligned}$$

The timed languages *Strict* and *NZ* are accepted by the automata in Figure 1. In all the figures, an initial state (respectively a final state) is represented with a short incoming (respectively outgoing) arrow, while a repeated state is represented with a double circle. Note that the transformation of a timed automaton $\mathcal{A}$ into a timed automaton $\mathcal{A}'$ such that $L(\mathcal{A}') = L(\mathcal{A}) \cap NZ$ is not as easy as it may appear because a finite timed word, which is indeed non Zeno, can be accepted by a run ending in a loop of ε-transitions where the underlying timed ε-word is Zeno.

3 Expressiveness and Robustness

We study in this section the expressiveness of timed automata with non observable actions and the robustness of the corresponding classes of languages. We

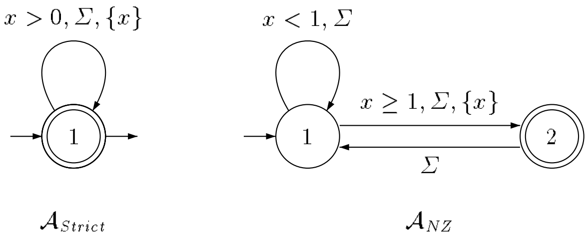


Fig. 1. Timed automata for *Strict* and *NZ*

give some examples of timed languages accepted by timed automata and we use them to compare the language classes $TL(\mathbb{T})$ and $TL_\varepsilon(\mathbb{T})$, for $\mathbb{T}$=$\mathbb{N}$ or $\mathbb{R}$.

We consider a parameterized family of languages of the class TL_ε which cannot be accepted without ε-transitions. Let $0 \leq k < p$ be two values of the time domain $\mathbb{T}$. We consider the language $Delay(k,p)$ which consists of timed words in which the delay between two consecutive actions is always in the set $k + p\mathbb{N}$:

$$Delay(k,p) = \{(a_i,t_i)_{i\geq 1} \in (\Sigma \times \mathbb{T})^\infty \mid t_{i+1} - t_i \in k + p\mathbb{N} \text{ for all } i \geq 0 \text{ (with } t_0 = 0)\}$$

The language $Delay(k,p)$ is accepted by the timed automaton $\mathcal{A}_{Delay(k,p)}$ of Figure 2. When $k = 0$ and $p = 2$, we obtain as a special case the language *Even* of finite or infinite timed words with even integer dates (see $\mathcal{A}_{Even}$ in Figure 2).

$$Even = \{(a_i,t_i)_{i\geq 1} \in (\Sigma \times \mathbb{N})^\infty \mid t_i \in 2\mathbb{N} \text{ and } t_i \leq t_{i+1} \text{ for all } i \geq 1\}$$

This simple language can be used to show that, contrary to the untimed case, silent transitions strictly increase the power of timed automata.

Proposition 2. *The class $TL(\mathbb{T})$ is strictly contained in the class $TL_\varepsilon(\mathbb{T})$.*

Proof. Let us consider the language *Even* in the simple case where the alphabet Σ reduces to a singleton $\{a\}$. We prove that *Even* does not belong to $TL(\mathbb{T})$ whatever $\mathbb{T}$. Assume on the contrary that *Even* is accepted by some timed automaton $\mathcal{A} = (\Sigma, Q, T, I, F, R, X)$ in *TA* and let $m \in \mathbb{T}$ be the largest constant appearing in the subformulas of the clock constraints. Define for any $k \in \mathbb{T}$ the clock valuations γ_k by $\gamma_k(x) = k$, for each clock x in X, and let h be a natural number such that $2h > m$. Since $(a, 2h)$ is accepted by $\mathcal{A}$, there exists a transition $(q_0, A, a, \alpha, q_f) \in T$ such that $q_0 \in I$, $q_f \in F$ and $\gamma_{2h} \models A$. Clearly, for all $p, q > m$, $\gamma_p \models A$ if and only if $\gamma_q \models A$. Therefore, for any odd number $k \in \mathbb{T}$, $k > m$, the timed word (a, k) is accepted by $\mathcal{A}$ which is a contradiction.

The proposition also holds when only infinite timed words are considered : in the proof above, we only have to consider an infinite word beginning with

$(a, 2h)$. Note that, if only rational constants were used in the timing constraints of timed automata, we could prove this proposition thanks to the notion of clock regions [3].

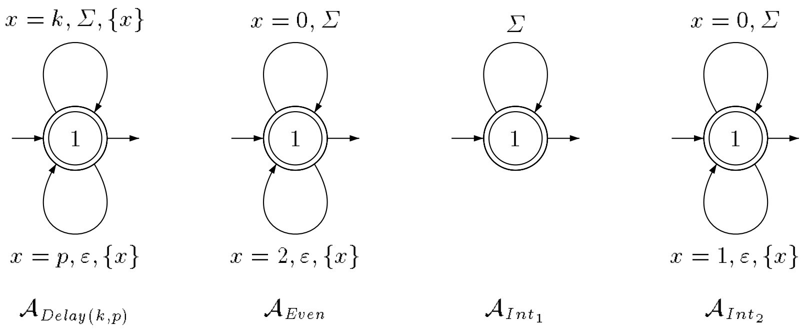


Fig. 2. Timed automata for *Delay*(k, p), *Even* and *Int*

We finally consider the set *Int* of timed words with integer dates,

$$Int = Delay(0, 1) = \{(a_i, t_i)_{i \geq 1} \in (\Sigma \times \mathbb{N})^\infty \mid t_i \leq t_{i+1} \text{ for all } i \geq 1\}$$

accepted by the timed automaton $\mathcal{A}_{Int_1}$ in TA, over $\mathbb{N}$, and also by $\mathcal{A}_{Int_2}$ in TA_ε, over $\mathbb{T} = \mathbb{N}$ or $\mathbb{R}$ (see Figure 2). With an argument similar to the one used in the proof of Proposition 2, it can be shown that *Int* does not belong to $TL(\mathbb{R})$. Therefore, we obtain the somewhat strange result that the class $TL(\mathbb{N})$ is not included in the class $TL(\mathbb{R})$. The case of the classes $TL_\varepsilon(\mathbb{N})$ and $TL_\varepsilon(\mathbb{R})$ are more satisfactory. Namely, for any timed automaton $\mathcal{A}$, the equality $L(\mathcal{A}, \mathbb{N}) = L(\mathcal{A}, \mathbb{R}) \cap Int$ holds. Since *Int* is in the class TL_ε and this class is closed under intersection, we obtain:

Proposition 3. *The class $TL(\mathbb{N})$ is not contained in the class $TL(\mathbb{R})$, but $TL_\varepsilon(\mathbb{N})$ is contained in $TL_\varepsilon(\mathbb{R})$.*

4 Normal forms for timed automata

We propose in this section several normal forms for timed automata which will simplify the constructions in the next sections. The first two are folklore and frequently used. The third one discusses the relationship between ε-transitions and finite words or repeated states respectively.

4.1 Disjunction free automata

An automaton $\mathcal{A} = (\Sigma, Q, T, I, F, R, X)$ is disjunction free if for every transition (q, A, a, α, q'), the constraint A is a conjunction of atomic formulas.

Lemma 4. *Let $\mathcal{A}$ be an automaton in TA (TA_ε resp.). There exists a disjunction free automaton $\mathcal{A}'$ in TA (TA_ε resp.) equivalent to $\mathcal{A}$.*

In order to remove the disjunctions, a clock constraint A in a transition (q, A, a, α, q') is converted into a disjunctive normal form $A = A_1 \vee A_2 \vee \ldots \vee A_k$ and the transition is replaced by the transitions (q, A_i, a, α, q'), $1 \leq i \leq k$.

4.2 Diagonal free automata

A timed automaton is diagonal free if the clock constraints do not contain any subformula of the form $x - y \# t$, where x and y are clocks, $t \in \mathbb{T}$ and $\#$ is a binary operator in $\{<, =, >\}$. The following result, mentioned in [4], says that diagonal free automata have the same expressive power than automata with diagonal constraints:

Lemma 5. *Let $\mathcal{A}$ be an automaton in TA (TA_ε resp.). There exists a diagonal free automaton $\mathcal{A}'$ in TA (TA_ε resp.) equivalent to $\mathcal{A}$.*

Proof. The transformation is done by a synchronized product of the original automaton with a finite automaton which records for all $(x, y) \in X^2$ how $x - y$ compares with the constants in C.

If $x - y \# c$ is true at time t, i.e., $x(t) - y(t) \# c$, then this is true for all times $t' \geq t$ as long as we do not perform a reset on the clock x or y. Thus, we can realize the transformation by replacing the state space Q by $Q \times \{0, 1\}$ where 1 indicates that $x - y \# c$ is true and 0 indicates that $x - y \# c$ is false. Let us call the expression $x - y \# c$ an attribute of the state. The value of the attribute $x - y \# c$ of the initial state is the truth value $0 \# c$. This value of the attribute may change with a reset on x or y.

Let (p, A, a, α, q) be a transition of the automaton. There are four cases:

1. If $\{x, y\} \cap \alpha = \emptyset$, then the attribute-value of p is passed to q.
2. If $\{x, y\} \subseteq \alpha$, then the attribute of q receives (via this transition) the value $0 \# c$.
3. If $x \notin \alpha$, $y \in \alpha$, then we double the transition. In the first one we add to A the constraint $x \# c$ and we enter the state q with the attribute value *true*. For the second, we add to A the constraint $\neg(x \# c)$ entering the state q with the attribute value *false*.
4. The last case $x \in \alpha$, $y \notin \alpha$ is dual to 3).

It should be clear to the reader how to finish the construction. Thus if k constraints have to be removed, we do this by considering the state space $Q \times \{0, 1\}^k$.

Another way to describe the technique above is to relax the syntax for transitions. If (p, A, a, α, q) is a transition with $x \notin \alpha$, $y \in \alpha$, we may add the following inscription to this transition:

$$\textbf{if } x \# c \textbf{ then } (x - y \# c) := \text{true} \textbf{ else } (x - y \# c) := \text{false} \textbf{ endif}$$

The other cases are handled analogously. Therefore, the value of an attribute changes dynamically with a run through the automaton. This syntax avoids the explicit use of several copies of the automaton.

4.3 Finite words and finite runs

We now deal with the problem mentioned in Section 2. Recall that a finite timed word may be accepted by an infinite run ending with ε-transitions only. Our aim is to modify the automaton in such a way that every accepted finite timed word can be accepted with a finite run. In the untimed case, the solution consists in changing into a final state each state from which an accepted infinite ε-path exists. With timed automata, this is not so simple, because of the clock constraints and the resets on ε-transitions. Two different situations are shown in Figure 3. For the left automaton, like in the untimed case, it is sufficient to consider state 1 as a final state.

On the contrary, state 1 should not be considered as final in the second automaton. Indeed, after each action a, it is possible to go only once through the ε-cycle. Hence, the automaton does not accept any finite timed word. The following proposition shows that a solution can be found for restricted timed automata, that is, when ε-transitions do not reset clocks.

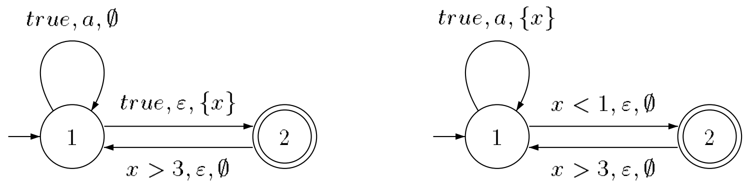


Fig. 3. Accepting finite timed words with finite or infinite runs

Proposition 6. *Let $\mathcal{A}$ be an automaton in RTA_ε. There exists an equivalent automaton $\mathcal{A}'$ such that any finite timed word in $L(\mathcal{A}')$ can be accepted by a finite run.*

Proof. In the case of a general timed automaton $\mathcal{A}$ with rational constants, a solution consists in using once more the powerful notion of clock regions. Recall that a clock region defines a set $\mathcal{H}$ of clock valuations which allow "similar" continuations of the run. There are finitely many clock regions when the automaton uses rational constants and a clock region is defined by a formula such as

$$x = 2 \wedge t > 5 \wedge 3 < y < 4 \wedge 1 < z < 2 \wedge z - y > 1.$$

Now, roughly speaking, a state q has to be considered as final with respect to a clock region $\mathcal{H}$ if starting from q with a valuation $v \in \mathcal{H}$, we can construct in $\mathcal{A}$ an accepting run using only ε-transitions. Such a test is equivalent to the non emptiness of the region automaton of $\mathcal{A}$ restricted to the ε-transitions and with $(q, \mathcal{H})$ as initial state. It is therefore decidable [3,4]. Now, to obtain a timed automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ and such that any finite accepted word is recognized by a finite path, it suffices to consider its region automaton. A state $(q, \mathcal{H})$ is final if either q is already final in $\mathcal{A}$ or q is final with respect to the clock region $\mathcal{H}$.

In the particular case where $\mathcal{A}$ is a restricted timed automaton, we sketch now a direct proof which does not use the clock regions and does not assume rationality for the constants. Note first that ε-cycles are useless in restricted timed automata. Indeed, if a timed word w is accepted by a path $P_1P_2P_3$ where P_2 is an ε-cycle, it is also accepted by the path P_1P_3 since ε-transitions do not reset clocks. Using this argument repeatedly, we will show:

Claim: If a finite timed word is accepted by an infinite run, it is accepted through a path $P_1P_2^\omega$ where P_2 is a simple ε-cycle over some repeated state s.

Indeed, let P be an infinite path accepting a finite timed word w. We can write $P = P_1P_2P_3\cdots$ where for all $i \geq 2$, P_i is an ε-cycle over some repeated state s. Using the remark above, we may first assume that P_i is actually a simple ε-cycle for all $i \geq 2$. Since there are only finitely many simple ε-cycles over s, one of them must occur infinitely often. Using again the remark above, we delete the other ones to obtain the form required by the claim.

We proceed now to the second step of the proof.

Claim: A finite timed word w has a run through the infinite path $P_1P_2^\omega$ if and only if it has a run through the finite path $P_1P_2P_2$.

To simplify the argument, we assume that the constraints in P_2 are disjunction free (see Lemma 4). Assume that there exists a run r through $P_1P_2P_2$ and let $v = X(t)$ be the clock valuation after P_1P_2 in this run. By stopping the time after P_1P_2, that is by keeping forever the clock valuation v, we obtain a run through the infinite path $P_1P_2^\omega$ which proves the claim. Indeed, since r is a run through $P_1P_2P_2$, each elementary constraint A of P_2 has been satisfied both at $t - \delta_1$ and $t + \delta_2$ for some $\delta_1, \delta_2 \geq 0$. Therefore, it is easy to see that A is also satisfied at t.

Note that if P_2 does not contain any elementary constraint of the form $x = c$ ($x = c$ or $x < c$ resp.) we can find an infinite run where the time is not stationary (is divergent resp.). Therefore, the second claim can be easily modified to cope with non stationary or divergent runs.

We can now construct the new automaton $\mathcal{A}' = (\Sigma, Q', T', I, F', R, X)$ by unfolding twice the simple ε-cycles over repeated states. The idea of the construction is the following: given a repeated state s and a cycle P_2 over s, we will add a path from s to a new final state f composed of the concatenation of two new unfolded copies of P_2. Formally, the set of states is $Q' = Q \cup [Q \times \mathcal{S} \times \{1, 2\}] \cup \{f\}$ where $\mathcal{S}$ is the set of all simple ε-cycles over repeated states and f is a new state. We let $F' = F \cup \{f\}$ and T' consists of all transitions from T together with new transitions that will be defined now. Let $P_2 \in \mathcal{S}$ be a simple ε-cycle over some repeated state s:

$$P_2 = s \xrightarrow{A_1, \varepsilon} s_1 \cdots s_{k-1} \xrightarrow{A_k, \varepsilon} s.$$

We add to T' the transitions which compose the following path

$$s \xrightarrow{A_1,\varepsilon} (s_1, P_2, 1) \cdots (s_{k-1}, P_2, 1) \xrightarrow{A_k,\varepsilon} (s, P_2, 1)$$
$$\xrightarrow{A_1,\varepsilon} (s_1, P_2, 2) \cdots (s_{k-1}, P_2, 2) \xrightarrow{A_k,\varepsilon} f.$$

Using the claims above, one can check that $L(\mathcal{A}) = L(\mathcal{A}')$ and that all finite timed words in $L(\mathcal{A}')$ can be accepted by a finite run of $\mathcal{A}'$.

5 Suppression of ε-cycles

We are now ready to start the core of our paper: the investigation of the expressive power of ε-transitions. We first focus on the ε-transitions which do not reset clocks. We will prove that these ε-transitions can be removed in any timed automaton.

The first step consists in the suppression of ε-cycles. In the case of (infinite) untimed words, any ε-cycle can be erased provided it does not contain a repeated state. If such a cycle contains a repeated state, the property "repeated" has to be shifted to some suitable state before erasing the cycle. In the case of timed words, the situation is not so simple. For instance in the automaton $\mathcal{A}$ of Figure 4, it is not possible to transfer the "repeated" property from state 2 to state 1. Namely, the automaton $\mathcal{B}$ of Figure 4 accepts for instance the timed word $(a, 0)(a, 1)(a, 2) \cdots$ which does not belong to $L(\mathcal{A})$. In order to keep the conditions enforced by the ε-cycle, we unfold this cycle and we obtain the automaton $\mathcal{C}$ of Figure 4 for which it is easy to prove that $L(\mathcal{A}) = L(\mathcal{C})$. A generalization of this example will allow us to suppress all ε-cycles which consist of ε-transitions without reset in any timed automaton.

Proposition 7. *Let $\mathcal{A}$ be an automaton in RTA_ε. There exists an automaton $\mathcal{A}'$ in RTA_ε equivalent to $\mathcal{A}$ such that $\mathcal{A}'$ does not contain any ε-cycle.*

The proof of Proposition 7 is based on the following two lemmas, which allow us to bound the number of consecutive ε-transitions needed in accepting runs. Let $\mathcal{A} = (\Sigma, Q, T, I, F, R, X)$ be a timed automaton in RTA_ε. Thanks to Proposition 6, we will assume that any finite word in $L(\mathcal{A})$ can be accepted by a finite run of $\mathcal{A}$.

Lemma 8. *For any finite (infinite resp.) timed word $w \in L(\mathcal{A})$, there exists an accepting timed run r of $\mathcal{A}$ over w such that r does not contain more than $|Q|-1$ ($2|Q|-2$ resp.) consecutive ε-transitions.*

Proof. We give the proof for an infinite timed word w. The finite case is similar and actually simpler. Let r be an accepting timed run generating the timed word w and let q be a repeated state which occurs infinitely often in r. Since w

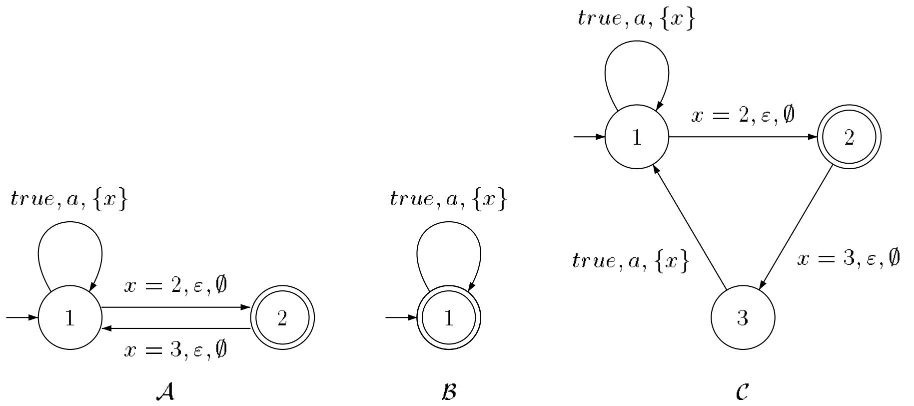


Fig. 4. Suppression of ε-cycle

is infinite, any maximal subrun of r constituted with consecutive ε-transitions is finite. For each such sub-run

$$q_i \xrightarrow[t_i]{A_i,\varepsilon,\emptyset} q_{i+1} \xrightarrow[t_{i+1}]{A_{i+1},\varepsilon,\emptyset} \dots \xrightarrow[t_{j-1}]{A_{j-1},\varepsilon,\emptyset} q_j$$

we make the following transformation. First, if all states $q_i, \dots, q_j$ in this sub-run are distinct from q then we erase the loops in order to obtain a sub-run containing at most $|Q| - 1$ transitions. Second, if $q = q_k$ for some $i \leq k \leq j$ then we erase the loops between q_i and q_k and the loops between q_k and q_j in order to obtain a sub-run which contains at most $2(|Q| - 1)$ transitions.

Since ε-transitions of $\mathcal{A}$ do not reset clocks, it is easy to verify that the sub-run r' of r obtained by applying the transformation above is still a timed run of the automaton $\mathcal{A}$. Moreover, since r' contains an infinite number of occurrences of the state q, it is an accepting run. The run r' obviously generates the same timed word than r, and the lemma is proved.

Proof. (of Proposition 7) Let $k = 2|Q| - 1$ and let $[k]$ be the set of integers $\{0, 1, \cdots, k-1\}$. We will modify the automaton $\mathcal{A}$ in order to allow at most $k - 1$ consecutive ε-transitions. To this purpose, we construct a new automaton $\mathcal{A}' = (\Sigma, Q', T', I', F', R', X)$ where $Q' = Q \times [k]$, $I' = I \times \{0\}$, $F' = F \times [k]$ and $R' = R \times [k]$. Let $q \xrightarrow{A,a,\alpha} q'$ be in T.

- If $a \neq \varepsilon$ then, for any $h \in [k]$, the transition $(q, h) \xrightarrow{A,a,\alpha} (q', 0)$ is added to T'.
- If $a = \varepsilon$ then for any $h \in [k], h \neq k-1$, the transition $(q, h) \xrightarrow{A,\varepsilon,\alpha} (q', h+1)$ is added to T' (note that in this case, $\alpha = \emptyset$).

Claim This new automaton $\mathcal{A}'$ is by construction without ε-cycle and recognizes the same timed language than $\mathcal{A}$: $L(\mathcal{A}) = L(\mathcal{A}')$.

From the construction of T', any timed run r' of $\mathcal{A}'$ can be transformed in a timed run r of $\mathcal{A}$ by erasing the second component of each state (i.e. each $(q,h) \in Q'$ is replaced by $q \in Q$). Conversely let r be a timed run of $\mathcal{A}$:

$$r = q_0 \xrightarrow[t_1]{A_1,a_1,\alpha_1} q_1 \xrightarrow[t_2]{A_2,a_2,\alpha_2} q_2 \ldots$$

Using Lemma 8, we may assume that r does not contain more than $k-1$ consecutive ε-transitions, so that r can be transformed into a sequence r' by adding to each q_i the number of consecutive ε-transitions read just before q_i:

$$r' = (q_0, h_0) \xrightarrow[t_1]{A_1,a_1,\alpha_1} (q_1, h_1) \xrightarrow[t_2]{A_2,a_2,\alpha_2} (q_2, h_2) \ldots$$

with $h_0 = 0$ and for all $i > 0$, $h_i = 0$ if $a_i \neq \varepsilon$ and $h_i = 1 + h_{i-1}$ otherwise. Since $h_i \leq 2|Q| - 2$, for all $i \geq 0$, the construction of T' shows that the sequence r' is a timed run of $\mathcal{A}'$ which accepts the same timed word as the run r of $\mathcal{A}$.

6 Closure of clock constraints

The purpose of an ε-transition without clock reset is only to check that the clock constraint carried by the ε-transition is satisfied at some point of time. More precisely, let us consider the restricted timed automaton $\mathcal{A}$ of Figure 5. The ε-transition ensures that there exists some time between the date of the execution of a and the date of the execution of c where the clock constraint B holds. The main idea to remove this ε-transition is to shift this verification either forward to the c-transition or backward to the a-transition. In order to shift this verification, we will define and study closures of clock constraints. Note that, in this section, we need to compute effectively sums, differences and orderings of constants. Hence, we will use the hypotheses on the set of constants $C(\mathcal{A})$, stated in Section 2. We start with the definition of forward closures of clock constraints.

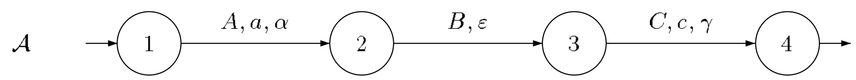


Fig. 5. A restricted timed automaton $\mathcal{A}$

Definition 9. Let B be a clock constraint. The forward closure of B, denoted by $\overrightarrow{B}$, is a formula which is satisfied by a clock valuation v if the formula B is satisfied by the clock valuation $v - \delta$ for some $\delta \geq 0$:

$$v \models \overrightarrow{B} \quad \text{if} \quad \exists \delta \geq 0, v - \delta \models B$$

The following result claims that the forward closure of any clock constraint is still a clock constraint.

Proposition 10. *Let B be a clock constraint. There exists a clock constraint C equivalent to $\overrightarrow{B}$, i.e. such that for any clock valuation v,*

$$v \models \overrightarrow{B} \quad \textit{if and only if} \quad v \models C$$

Note that, if we restrict ourselves to clock constraints with rational constants, we could use the notion of clock regions [3] to prove this proposition. However, we sketch an easy and general proof which holds whatever the clock constraints are and which does not depend on the classical region construction. This proof carries mainly two ideas which are presented in the two following lemmas. The first one addresses the case where B is a conjunction of constraints on two clocks.

Lemma 11. *Let $0 \leq a < b \leq \infty$ and $0 \leq c < d \leq \infty$. In each of the following cases, the clock constraint C is equivalent to $\overrightarrow{B}$.*

Case	B	C
1	$a < x < b \wedge c < y < d$	$a < x \wedge c < y \wedge c - b < y - x < d - a$
2	$x = a \wedge c < y < d$	$a \leq x \wedge c < y \wedge c - a < y - x < d - a$
3	$x = a \wedge y = c$	$a \leq x \wedge y - x = c - a$

Proof. The proof is a simple translation of Figure 6 which represents in each of the three cases, the constraints B and $\overrightarrow{B}$.

The second lemma is a generalization of the first one to an arbitrary number of clocks.

Lemma 12. *Let B be a conjunction of constraints of the form $a < x < b$ or $x = a$ with $0 \leq a < b \leq \infty$. Then, there exists a clock constraint C equivalent to $\overrightarrow{B}$.*

Proof. Using comparisons (which are effective) between constants, we can decide whether B is satisfiable. If it is not the case, then $\overrightarrow{B} = \textit{false}$. Otherwise, we can effectively compute for B an equivalent formula of the form $B = \bigwedge_{x \in \beta} B_x$, where $\beta \subseteq X$ is a set of clocks and for each $x \in \beta$, B_x is either the constraint $a_x < x < b_x$ or the constraint $x = a_x$ with $0 \leq a_x < b_x \leq \infty$. If $\beta = \{x\}$, then $\overrightarrow{B}$ is either $a_x < x$ or $a_x \leq x$. Otherwise, for all $x \neq y$ in β, let $C_{x,y}$ be the clock constraint associated with $B_x \wedge B_y$ by Lemma 11 and let

$$C = \bigwedge_{x \neq y} C_{x,y}.$$

We claim that C is equivalent to $\overrightarrow{B}$. Indeed, let v be a clock valuation. Assume first that $v \models \overrightarrow{B}$. Then, $v \models \overrightarrow{B_x \wedge B_y}$ for all $x \neq y$. It follows by Lemma 11 that $v \models C_{x,y}$. Whence $v \models C$.

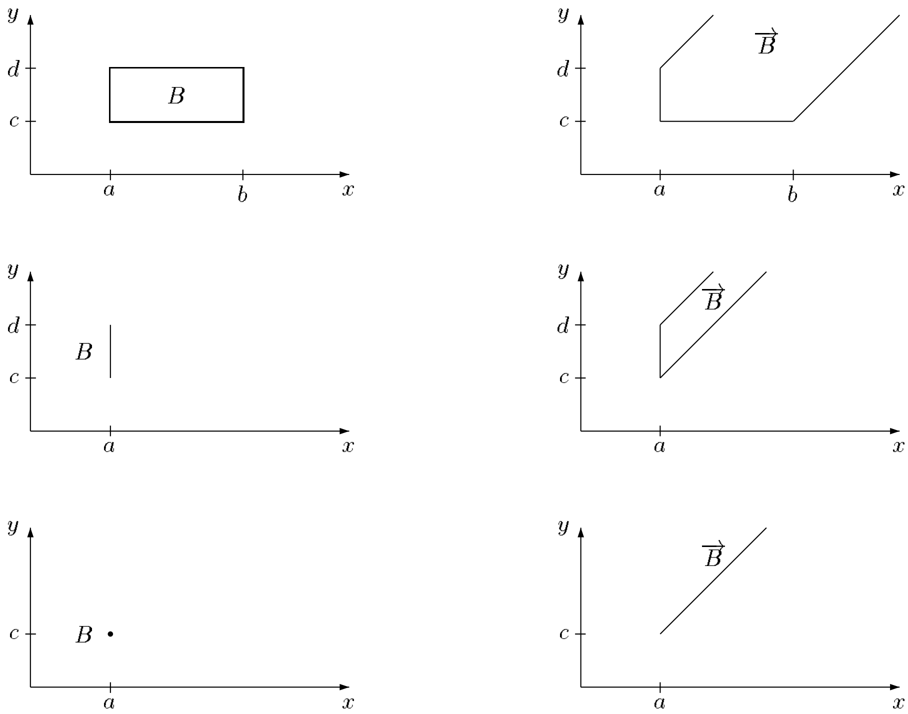


Fig. 6. Regions and their forward closures

Conversely, assume that $v \models C$. For all $x \in \beta$, define the interval I_x by :

$$I_x = \begin{cases}]v(x) - b_x, v(x) - a_x[& \text{if } B_x \text{ is } a_x < x < b_x \\ \{v(x) - a_x\} & \text{if } B_x \text{ is } x = a_x \end{cases}$$

Clearly, $v - \delta \models B_x$ if and only if $\delta \in I_x$. Now, for all $x \neq y$, $v \models C_{x,y}$. Hence, by Lemma 11 $v \models \overrightarrow{B_x \wedge B_y}$ which means precisely $\mathbb{R}_+ \cap I_x \cap I_y \neq \emptyset$. Since $(I_x)_{x \in \beta}$ is a finite family of intervals, it follows that

$$\mathbb{R}_+ \cap \bigcap_{x \in \beta} I_x \neq \emptyset$$

which is equivalent to $v \models \overrightarrow{B}$

Proof. (of Proposition 10) We begin the proof by two easy facts. First, let A, B be two clock constraints. It is easy to show that $\overrightarrow{A \vee B}$ is equivalent to $\overrightarrow{A} \vee \overrightarrow{B}$. Second, assume in addition that the atomic constraints of A are of the form $x - y \# a$ where $\# \in \{<, =, >\}$. Since $v - \delta \models A$ if and only if $v \models A$ for all clock valuation v and all $\delta \geq 0$, it follows directly that $\overrightarrow{A \wedge B}$ is equivalent to $A \wedge \overrightarrow{B}$.

Now, let B be any clock constraint. Then, the disjunctive normal form of the formula B is $\bigvee_i (A_i \wedge B_i)$ where A_i is a conjunction of formulas of the form $x - y \# a$ with $\# \in \{<, =, >\}$ and B_i satisfies the hypotheses of Lemma 12. Let C_i be the clock constraint equivalent to $\overrightarrow{B_i}$ given by Lemma 12. Using the two facts above, we deduce that $\overrightarrow{B}$ is equivalent to $\bigvee_i (A_i \wedge C_i)$ and Proposition 10 is proved.

The forward closure of a clock constraint allows to shift the verification of the clock constraint carried by an ε-transition to the next transition. This method cannot be used when the ε-transition is the last one of a run. For instance, let $\mathcal{D}$ be the RTA_ε of Figure 7. In order to remove the ε-transition of $\mathcal{D}$ we must shift the verification of the constraint B backward to the a-transition. To this purpose, we define the backward closure of a clock constraint with respect to a set of clocks.

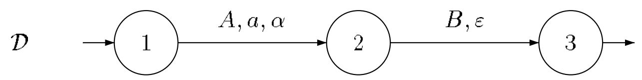


Fig. 7. A restricted timed automaton $\mathcal{D}$

Definition 13. Let B be a clock constraint and α be a set of clocks. The backward closure of B with respect to α, denoted by $\overleftarrow{B}^{\alpha}$, is a formula which is

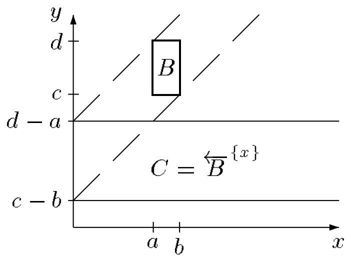


Fig. 8. A region and its backward closure

satisfied by a clock valuation v if the formula B is satisfied by the clock valuation $v[\alpha \leftarrow 0] + \delta$ for some $\delta \geq 0$:

$$v \models \overleftarrow{B}^{\alpha} \quad \text{if} \quad \exists \delta \geq 0, v[\alpha \leftarrow 0] + \delta \models B$$

The clock constraint $B = a < x < b \wedge c < y < d$ and its backward closure with respect to $\{x\}$ are presented in Figure 8. It is immediate to verify on this example that the backward closure is equivalent to a clock constraint. This result holds in general, the backward closure of any clock constraint with respect to a set of clocks is also a clock constraint.

Proposition 14. *Let B be a clock constraint and let α be a set of clocks. There exists a clock constraint C equivalent to $\overleftarrow{B}^{\alpha}$, i.e. such that for any clock valuation v,*

$$v \models \overleftarrow{B}^{\alpha} \quad \textit{if and only if} \quad v \models C$$

Proof. The proof of this proposition is similar to that of Proposition 10. It is only slightly more complicated due to the set of clocks which are reset. We start by considering the case where B is a conjunction of constraints on two clocks. In each of the following cases, the clock constraint C is equivalent to $\overleftarrow{B}^{\alpha}$. The verification of this claim is left to the reader. Figure 8 corresponds to the first subcase of case 2 of the following table.

Case	B	C
$x, y \notin \alpha$	$a < x < b \wedge c < y < d$	$x < b \wedge y < d \wedge c - b < y - x < d - a$
	$x = b \wedge c < y < d$	$x \leq b \wedge y < d \wedge c - b < y - x < d - b$
	$x = b \wedge y = d$	$x \leq b \wedge y - x = d - b$
$x \in \alpha \wedge y \notin \alpha$	$a < x < b \wedge c < y < d$	$c - b < y < d - a$
	$x = b \wedge c < y < d$	$c - b < y < d - b$
	$a < x < b \wedge y = d$	$d - b < y < d - a$
	$x = b \wedge y = d$	$y = d - b$
$x, y \in \alpha$	$a < x < b \wedge c < y < d$	*true* if $c - b < 0 < d - a$, *false* otherwise
	$x = b \wedge c < y < d$	*true* if $c < b < d$ and *false* otherwise
	$x = b \wedge y = d$	*true* if $b = d$ and *false* otherwise

Now, we deal with the case where B is a conjunction of constraints on an arbitrary number of clocks. As in the proof of Lemma 12, we may assume that $B = \bigwedge_{x\in\beta} B_x$, where β is a set of clocks and for each $x \in \beta$, B_x is either the constraint $a_x < x < b_x$ or $x = a_x$ with $0 \leq a_x < b_x \leq \infty$. If $\beta = \{x\} \subseteq \alpha$, then $\overleftarrow{B}^{\alpha} = \mathit{true}$ and if $\beta = \{x\} \not\subseteq \alpha$, then $\overleftarrow{B}^{\alpha}$ is either $x < b_x$ or $x \leq b_x$. We assume now that $|\beta| > 1$. For all $x \neq y$ in β, let $C_{x,y}$ be the clock constraint associated with $B_x \wedge B_y$ by the above table and let $C = \bigwedge_{x\neq y} C_{x,y}$. We claim that C is equivalent to $\overleftarrow{B}^{\alpha}$. The proof is similar to that of Lemma 12 using for all $x \in \beta$ the interval I_x defined by :

$$I_x = \begin{cases}]a_x - v(x), b_x - v(x)[& \text{if } B_x \text{ is } a_x < x < b_x \text{ and } x \notin \alpha \\ ]a_x, b_x[& \text{if } B_x \text{ is } a_x < x < b_x \text{ and } x \in \alpha \\ \{a_x - v(x)\} & \text{if } B_x \text{ is } x = a_x \text{ and } x \notin \alpha \\ \{a_x\} & \text{if } B_x \text{ is } x = a_x \text{ and } x \in \alpha \end{cases}$$

Finally, let A, B be two clock constraints. Clearly, the formula $\overleftarrow{A \vee B}^{\alpha}$ is equivalent to $\overleftarrow{A}^{\alpha} \vee \overleftarrow{B}^{\alpha}$. Assume in addition that the atomic constraints of A are of the form $x - y \mathrel{\#} a$ where $\# \in \{<, =, >\}$. Let A' be the clock constraint obtained from A by replacing all occurrences of $x - y$ by :

$$\begin{cases} x - y & \text{if } x, y \notin \alpha \\ x & \text{if } x \notin \alpha \text{ and } y \in \alpha \\ -y & \text{if } x \in \alpha \text{ and } y \notin \alpha \\ 0 & \text{if } x, y \in \alpha \end{cases}$$

Clearly, for all clock valuation v, it holds $v[\alpha \leftarrow 0] \models A$ if and only if $v \models A'$. Therefore, it is easy to verify that $\overleftarrow{A \wedge B}^{\alpha}$ is equivalent to $A' \wedge \overleftarrow{B}^{\alpha}$.

The conclusion of the proof is similar to that of Proposition 10.

7 Silent transitions without clock reset

We have proved in Section 3 that transitions with non observable actions add power to timed automata (Proposition 2). The aim of this section is to show that ε-transitions add power to timed automata only if they reset clocks.

Recall that RTA_ε (for restricted TA_ε) is the class of timed automata with ε-transitions without clock reset and RTL_ε the class of timed languages accepted by automata of RTA_ε. Using these notations, the main theorem of this section is

Theorem 15. *For any restricted timed automaton there exists an equivalent timed automaton without ε-transition:*

$$RTL_\varepsilon = TL$$

To prove this result, we will construct, from any timed automaton $\mathcal{A}$ in RTA_ε, a timed automaton $\mathcal{B}$ in TA which accepts the same timed language. This construction will be done in several steps.

The first step will deal with a problem about repeated states: an infinite accepting run may reach repeated states only with ε-transitions. If we remove them, all repeated states may become unreachable. Hence we have to shift repeated states to states which are entered by visible transitions only. In addition, we will split the state set into two disjoint sets $Q = Q' \dot\cup Q''$ such that the states of Q' will be only entered by visible transitions while the states of Q'' will be only entered by ε-transitions.

Proposition 16. *Let $\mathcal{A}$ be an automaton in RTA_ε. There exists a restricted timed automaton $\mathcal{A}' = (\Sigma, Q' \dot\cup Q'', T', I', F', R', X)$ equivalent to $\mathcal{A}$ such that $Q' \cap Q'' = \emptyset$, $R' \subseteq Q'$ and for all transitions $p \xrightarrow{A,a,\alpha} q$ in T', $a = \varepsilon$ if and only if $q \in Q''$. Note in particular that no ε-transition in $\mathcal{A}'$ enters a repeated state.*

Proof. The construction will add two flags to the states of a timed automaton. The first flag will be set if the state is reached by an ε-transition and reset otherwise. The second flag will be set if a repeated state was taken since the previous visible transition.

Let $\mathcal{A} = (\Sigma, Q, T, I, F, R, X)$ be a restricted timed automaton. From Proposition 6, assume that any finite word of $L(\mathcal{A})$ can be recognized through a finite path of $\mathcal{A}$. We define the new automaton $\mathcal{A}' = (\Sigma, Q' \dot\cup Q'', T', I', F', R', X)$ in RTA_ε as follows. $Q' = Q \times \{0\} \times \{0,1\}$, $Q'' = Q \times \{1\} \times \{0,1\}$, $I' = I \times \{(0,0)\}$, $F' = F \times \{0,1\}^2$, $R' = Q \times \{(0,1)\}$. If $p \xrightarrow{A,a,\alpha} q \in T$ is a visible transition of $\mathcal{A}$ then for all $e, r \in \{0,1\}$ we put in T' the transition $(p,e,r) \xrightarrow{A,a,\alpha} (q,0,s)$ where $s = 1$ if either $q \in R$ or $e = r = 1$ and $s = 0$ otherwise. Finally, if $p \xrightarrow{A,\varepsilon} q \in T$ is an ε-transition of $\mathcal{A}$ then for all $e, r \in \{0,1\}$ we put in T_1 the ε-transition $(p,e,r) \xrightarrow{A,\varepsilon} (q,1,s)$ where $s = 1$ if either $q \in R$ or $e = r = 1$ and $s = 0$ otherwise.

To each path P of $\mathcal{A}$:

$$P = p_0 \xrightarrow{A_1,a_1,\alpha_1} p_1 \xrightarrow{A_2,a_2,\alpha_2} p_2 \cdots$$

corresponds a unique path

$$P' = p'_0 = (p_0,0,0) \xrightarrow{A_1,a_1,\alpha_1} p'_1 = (p_1,e_1,r_1) \xrightarrow{A_2,a_2,\alpha_2} p'_2 = (p_2,e_2,r_2) \cdots$$

in $\mathcal{A}'$. Moreover, $p_0 \in I \iff p'_0 \in I'$ and $p_n \in F \iff p'_n \in F'$ for all $n \geq 0$. If P is an infinite path, we may assume from the hypothesis made on $\mathcal{A}$ that the word $a_1 a_2 \ldots$ is infinite. Thus, for all $n \geq 0$, if $p_n \in R$ then $p'_m \in R'$ for some $m \geq n$. Therefore, $L(\mathcal{A}) \subseteq L(\mathcal{A}')$.

Conversely, the projection of a path P' of $\mathcal{A}'$ gives a path P of $\mathcal{A}$. With the same notations, we have $p_0 \in I \iff p'_0 \in I'$ and $p_n \in F \iff p'_n \in F'$ for all $n \geq 0$. Finally, for all $m \geq 0$, if $p'_m \in R'$ then $a_m \neq \varepsilon$ and there exists some $n \leq m$ such that $p_n \in R$ and $a_k = \varepsilon$ for all $n \leq k < m$. Therefore, $L(\mathcal{A}') \subseteq L(\mathcal{A})$ which concludes the proof.

Note that if we start with a restricted timed automaton $\mathcal{A}$ without ε-cycle as given by Proposition 7 then the automaton $\mathcal{A}'$ obtained in the proof above is still ε-cycle free. Therefore, we can start the proof of Theorem 15 with a restricted timed automaton $\mathcal{A} = (\Sigma, Q, T, I, F, R, X)$ which is ε-cycle free and such that $Q = Q' \dot{\cup} Q''$ with $R \subseteq Q'$ and for all transitions $p \xrightarrow{A,a,\alpha} q$ in T, $a = \varepsilon$ if and only if $q \in Q''$.

As announced in Section 6, we will shift forward or backward the verification of clock constraints carried by ε-transitions and then we will remove ε-transitions from the automaton. The first construction will shift the verification of clock constraints carried by ε-transitions to the next visible transition. This will be done by using the forward closures of clock constraints. To illustrate the construction, let us consider the automaton $\mathcal{A}$ of Figure 5. We first want to point out that it is not sufficient to transform $\mathcal{A}$ into the automaton $\mathcal{B}$ of Figure 9. Indeed, assume for instance that $A = (x > 3)$, $B = (x < 2 \vee x > 6)$, $C = true$ and $\alpha = \gamma = \emptyset$. The forward closure of B is $\overrightarrow{B} = true$, therefore the automaton $\mathcal{B}$ of Figure 9 accepts for instance the time word $(a, 4)(c, 5)$ whereas the automaton $\mathcal{A}$ does not accept this word. To construct a timed automaton without ε-transitions which accepts the same language, we must add a clock x_2 which is reset when entering state 2. It is then easy to verify that the automaton $\mathcal{C}$ of Figure 9 accepts the same timed language than the automaton $\mathcal{A}$.

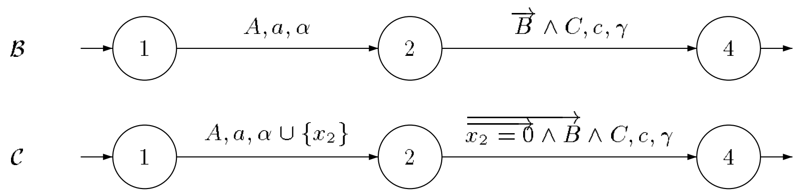


Fig. 9.

In the general case, we first construct an automaton $\mathcal{A}_1$ from $\mathcal{A}$ by adding, for each state $q \in Q'$, a new clock x_q which is reset by any transition entering the state q. Precisely, we define a set $X' = \{x_q \mid q \in Q'\}$ of clocks, disjoint from X, and we set $\mathcal{A}_1 = (\Sigma, Q, T_1, I, F, R, X_1)$ with $X_1 = X \cup X'$ and

$$T_1 = \{p \xrightarrow{A,a,\alpha \cup \{x_q\}} q \mid p \xrightarrow{A,a,\alpha} q \in T \text{ and } q \in Q'\} \cup \{p \xrightarrow{A,\varepsilon} q \mid p \xrightarrow{A,\varepsilon} q \in T\}.$$

Now, let $p, q \in Q$ and let

$$P = p \xrightarrow{B_1,\varepsilon} p_1 \cdots p_{n-1} \xrightarrow{B_n,\varepsilon} q \tag{1}$$

be an ε-path of length at least 1. In order to shift the verification of the constraints $B_1, \ldots, B_n$ to the next visible transition we define the formula

$$\varphi(P) = \overrightarrow{\overrightarrow{\overrightarrow{x_p = 0} \wedge B_1} \wedge \cdots \wedge B_n}.$$

Since there may be several ε-paths from p to q we will actually use a disjunction of such formulas. We denote by $\mathcal{P}(p,q)$ the set of ε-paths from p to q in $\mathcal{A}_1$ and we define

$$\varphi(p,q) = \bigvee_{P \in \mathcal{P}(p,q)} \varphi(P).$$

Note that since $\mathcal{A}_1$ is ε-cycle free, the set $\mathcal{P}(p,q)$ is finite and $\varphi(p,q)$ is a finite disjunction.

We are now ready for the next construction which will only add new *visible* transitions to the automaton $\mathcal{A}_1$. Let $\mathcal{A}_2 = (\Sigma, Q, T_2, I, F, R, X_1)$ where

$$T_2 = T_1 \cup \{p \xrightarrow{\varphi(p,q) \wedge A, a, \alpha} m \mid p, m \in Q' \text{ and } q \xrightarrow{A,a,\alpha} m \in T_1\}.$$

Proposition 17. *$\mathcal{A}_2$ is an ε-cycle free restricted timed automaton and*

$$L(\mathcal{A}_2) = L(\mathcal{A}_1).$$

Proof. Clearly, $\mathcal{A}_2$ is an ε-cycle free restricted timed automaton and $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. Conversely, assume that a new transition

$$p \xrightarrow{\varphi(p,q) \wedge A, a, \alpha} m \tag{2}$$

is used in a run through an accepting path P_2 of $\mathcal{A}_2$. Let v be the clock valuation when reaching this state p in the path P_2. There exists $\delta \geq 0$ such that $v + \delta \models \varphi(p,q) \wedge A$. Let

$$P = p \xrightarrow{B_1, \varepsilon} p_1 \cdots p_{n-1} \xrightarrow{B_n, \varepsilon} q$$

be an ε-path such that $v + \delta \models \varphi(P)$. There exist $\delta_n \geq 0, \ldots, \delta_0 \geq 0$ such that $v + \delta - (\delta_n + \cdots + \delta_i) \models B_i$ for all $1 \leq i \leq n$ and $v + \delta - (\delta_n + \cdots + \delta_0) \models x_p = 0$. Since $p \in Q'$ all transitions entering state p reset the clock x_p. Hence, $v(x_p) = 0$ and we obtain $\delta = \delta_0 + \cdots + \delta_n$. Therefore, $v + \delta_0 + \cdots + \delta_{i-1} \models B_i$ for all $1 \leq i \leq n$. Hence, one may use the path P followed by the transition $q \xrightarrow{A,a,\alpha} m$ instead of the transition (2). Replacing in this way all new transitions in the accepting path P_2 we obtain a path P_1 of $\mathcal{A}_1$ which accepts the same timed word as the path P_2.

We will now shift the verification of the clock constraints of an ε-path which ends in a final state backward to the previous visible transition. As above, we need first to define auxiliary formulas. Let P be the path defined in Equation (1) and let α be a set of clocks, we define

$$\psi(P,\alpha) = \overleftarrow{\overrightarrow{\overrightarrow{\overrightarrow{x_p = 0} \wedge B_1} \cdots \wedge B_{n-1}} \wedge B_n}^{\alpha}$$

and

$$\psi(p,q,\alpha) = \bigvee_{P \in \mathcal{P}(p,q)} \psi(P,\alpha).$$

Again, since $\mathcal{A}_2$ is ε-cycle free, the formula $\psi(p, q, \alpha)$ is a finite disjunction.

We consider now a new dead final state f and we will construct a new automaton from $\mathcal{A}_2$ by adding *visible* transitions to this new state. More precisely, let $\mathcal{A}_3 = (\Sigma, Q_3, T_3, I, F_3, R, X_1)$ where $Q_3 = Q \cup \{f\}$, $F_3 = F \cup \{f\}$ and

$$T_3 = T_2 \cup \{m \xrightarrow{\psi(p,q,\alpha)\wedge A,a} f \mid m, p \in Q', q \in F \text{ and } m \xrightarrow{A,a,\alpha} p \in T_2\}.$$

Proposition 18. *$\mathcal{A}_3$ is an ε-cycle free restricted timed automaton and*

$$L(\mathcal{A}_3) = L(\mathcal{A}_2).$$

Proof. Clearly, $\mathcal{A}_3$ is an ε-cycle free restricted timed automaton and $L(\mathcal{A}_2) \subseteq L(\mathcal{A}_3)$. Conversely, note first that a new transition may be taken only once in a path and as the last transition of a finite path. Assume that a new transition

$$m \xrightarrow{\psi(p,q,\alpha)\wedge A,a} f \tag{3}$$

is used in an accepting path P_3 of $\mathcal{A}_3$. By definition, $m, p \in Q'$, $q \in F$ and $m \xrightarrow{A,a,\alpha} p \in T_2$.

Let v be the clock valuation when reaching the state m of the path P_3. There exists $\delta \geq 0$ such that $v + \delta \models \psi(p, q, \alpha) \wedge A$. Let

$$P = p \xrightarrow{B_1,\varepsilon} p_1 \cdots p_{n-1} \xrightarrow{B_n,\varepsilon} q$$

be an ε-path such that $v+\delta \models \psi(P, \alpha)$. Let $v' = (v+\delta)[\alpha \leftarrow 0]$. There exist $\delta' \geq 0$ and $\delta_{n-1} \geq 0, \ldots, \delta_0 \geq 0$ such that $v' + \delta' \models B_n$, $v' + \delta' - (\delta_{n-1} + \cdots + \delta_i) \models B_i$ for all $1 \leq i \leq n-1$ and $v' + \delta' - (\delta_{n-1} + \cdots + \delta_0) \models x_p = 0$. Since $p \in Q'$, all transitions entering state p reset the clock x_p. Hence, $x_p \in \alpha$ and $v'(x_p) = 0$. We obtain $\delta' = \delta_0 + \cdots + \delta_{n-1}$. Therefore, $v' + \delta_0 + \cdots + \delta_{i-1} \models B_i$ for all $1 \leq i \leq n$. Hence, one can replace the transition (3) by the transition $m \xrightarrow{A,a,\alpha} p$ followed by the ε-path P in order to get a path P_2 of $\mathcal{A}_2$ which accepts the same timed word as the path P_3 of $\mathcal{A}_3$.

The next construction treats the case where the empty word is accepted by an ε-path in $\mathcal{A}_3$. If we simply remove ε-transitions the empty word may not be accepted anymore. To avoid this problem, we will put the dead state f into the set of initial states if the empty word is accepted by $\mathcal{A}_3$. Therefore, we define $\mathcal{A}_4 = (\Sigma, Q_3, T_3, I_4, F_3, R, X_1)$ where $I_4 = I \cup \{f\}$ if $\varepsilon \in L(\mathcal{A})$ and $I_4 = I$ otherwise. Obviously, $\mathcal{A}_3$ and $\mathcal{A}_4$ accept the same language.

Finally, we obtain an automaton $\mathcal{B}$ from $\mathcal{A}_4$ by erasing all the ε-transitions: $\mathcal{B} = (\Sigma, Q_3, X_1, T', I_4, F_3, R)$ where T' is obtained from T_3 by erasing all ε-transitions. This automaton $\mathcal{B}$ which is by construction without ε-transition accepts the same language than $\mathcal{A}$.

Proposition 19. *$\mathcal{B}$ is a timed automaton without ε-transitions and*

$$L(\mathcal{B}) = L(\mathcal{A}).$$

Proof. By definition, $\mathcal{B}$ is a timed automaton without ε-transition such that $L(\mathcal{B}) \subseteq L(\mathcal{A}_4) = L(\mathcal{A})$. Conversely, let $u \in L(\mathcal{A})$. If u is the empty word, then by construction the state f is both an initial and a final state of $\mathcal{B}$ and thus $u \in L(\mathcal{B})$. If u is nonempty, $u = (a_1, t_1)(a_2, t_2) \cdots \in L(\mathcal{A})$ and let P be a path in $\mathcal{A}$ which accepts u. If p, q are states of $\mathcal{A}$, we use the notation $p \xrightarrow{\varepsilon^*} q$ for either $p = q$ or there exists a non-empty ε-path from p to q in $\mathcal{A}$. We can thus write the path P as follows:

$$P = p_0 \xrightarrow{\varepsilon^*} q_0 \xrightarrow{A_1, a_1, \alpha_1} p_1 \xrightarrow{\varepsilon^*} q_1 \xrightarrow{A_2, a_2, \alpha_2} p_2 \cdots$$

Note that, by definition of the automaton $\mathcal{A}$, only the states $p_0, p_1, p_2, \ldots$ may be repeated in the path P.

We first remove the ε-transitions which precede visible actions. Hence, if u is infinite we obtain an ε-transition free path and if u is finite, we obtain a path which may contain ε-transitions after the last visible transition only. For convenience, we set $\varphi(p, p) = \mathit{true}$ for all $p \in Q$.

Claim: The following path P_2 is a path in $\mathcal{A}_2$ which accepts the timed word u.

$$P_2 = p_0 \xrightarrow{\varphi(p_0, q_0) \wedge A_1, a_1, \alpha_1} p_1 \xrightarrow{\varphi(p_1, q_1) \wedge A_2, a_2, \alpha_2} p_2 \cdots$$

Note first that P and P_2 start in the same state, contain the same repeated states and, if finite, end in the same state. Now, assume that in the path P_2 we reach the state p_k with the clock valuation v and assume that $p_k \xrightarrow{\varepsilon^*} q_k$ is actually the non empty ε-path

$$K = p_k \xrightarrow{B_1, \varepsilon} p'_1 \cdots p'_{n-1} \xrightarrow{B_n, \varepsilon} q_k$$

Then, there exist $\delta_0 \geq 0, \ldots, \delta_n \geq 0$ such that $\delta = \delta_0 + \cdots + \delta_n = t_{k+1} - t_k$, $v + \delta \models A_{k+1}$ and $v + \delta_0 + \cdots + \delta_{i-1} \models B_i$ for all $1 \leq i \leq n$. Hence $v + \delta \models \varphi(K)$ and then $v + \delta \models \varphi(p_k, q_k)$. The claim follows easily.

If there is no ε-transition left in the path P_2 (this is in particular the case when u is infinite) then P_2 is actually a path in $\mathcal{B}$ which accepts u.

To complete the proof, assume that u is a finite timed word of length k and that the path P_2 ends with the non empty ε-path

$$K = p_k \xrightarrow{B_1, \varepsilon} p'_1 \cdots p'_{n-1} \xrightarrow{B_n, \varepsilon} q_k$$

Since $\mathcal{A}$ is ε-cycle free, the path K is actually finite. Let $A'_{i+1} = \varphi(p_i, q_i) \wedge A_{i+1}$ for all $0 \leq i < k$.

Claim: The following path P_3 is a path in $\mathcal{B}$ which accepts the timed word u.

$$P_3 = p_0 \xrightarrow{A'_1, a_1, \alpha_1} p_1 \xrightarrow{A'_2, a_2, \alpha_2} p_2 \cdots p_{k-1} \xrightarrow{\psi(p_k, q_k, \alpha_k) \wedge A'_k, a_k} f$$

Let v be the clock valuation when reaching the state p_{k-1} in the path P_2. Let $\delta = t_k - t_{k-1}$ so that $v + \delta \models A'_k$. Let $v' = (v+\delta)[\alpha_k \leftarrow 0]$, there exists $\delta_0 \geq 0, \ldots, \delta_{n-1} \geq 0$ such that $v' + \delta_0 + \cdots + \delta_{i-1} \models B_i$ for all $1 \leq i \leq n$. Let $\delta' = \delta_0 + \cdots + \delta_{n-1}$. We have $v' + \delta' \models B_n$, $v' + \delta' - (\delta_{n-1} + \cdots + \delta_i) \models B_i$ for all $1 \leq i \leq n-1$ and $v' + \delta' - (\delta_{n-1} + \cdots + \delta_0) \models x_{p_k} = 0$. Therefore, $\psi(K, \alpha_k)$ holds and the claim follows easily.

We will now discuss the effectiveness of the constructions used to prove Theorem 15. Actually, one can check that all constructions are effective except maybe for the automaton $\mathcal{A}_4$. In order to carry out this before last step, one has to decide whether the empty word is accepted by the timed automaton $\mathcal{A}$. Note that in the general case of timed automata with ε-transitions, deciding whether the empty word is accepted is clearly equivalent to the emptiness problem for timed automata which is known to be PSPACE-complete if the constants used in the constraints of the automata are rational [4]. This result was obtained using the complex notion of region automata. In the case of restricted timed automata we will give a much simpler decision procedure using the tools introduced in this paper.

Let $\mathcal{A}$ be a diagonal free and ε-cycle free restricted timed automata (Propositions 7 and 5). By removing all visible transitions, we may assume that no transition of $\mathcal{A}$ resets clocks and that $\mathcal{A}$ uses only one clock. We need to decide whether there exists a finite path

$$P = i \xrightarrow{B_1,\varepsilon} p_1 \cdots p_{n-1} \xrightarrow{B_n,\varepsilon} f$$

from an initial state i to a final state f which admits a valid run. This is clearly equivalent to the satisfiability of the formula

$$\varphi(P) = \overrightarrow{\overrightarrow{\overrightarrow{x=0} \wedge B_1} \wedge \cdots \wedge B_{n-1}} \wedge B_n \,.$$

Since $\mathcal{A}$ is ε-cycle free, there are only finitely many ε-paths in $\mathcal{A}$ and our problem is equivalent to the satisfiability of the *finite* formula

$$\chi = \bigvee_{(i,f)\in I\times F} \; \bigvee_{P\in\mathcal{P}(i,f)} \varphi(P).$$

Using the results in Section 6 and the effectiveness for sums, differences and orderings of constants, one can effectively compute a finite formula $\chi' = \vee_i A_i$ equivalent to χ, where each A_i is of the form $x = a$ or $a < x < b$, with $0 \leq a < b \leq \infty$. The satisfiability of χ' is then clear.

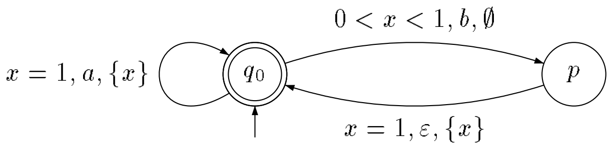
This simple construction is only possible because there is no reset which is a really strong condition. Indeed, contrary to the general case, the subclass of timed automata without reset is closed under complement, hence inclusion and equality are also decidable in this subclass.

The constructions of Section 7 are valid provided the language of constraints is closed under forward and backward closures. Hence, Theorem 15 remains true for more general constraints, including for instance comparisons modulo some value, as investigated in [10].

8 Silent transitions with clock reset

We know from Theorem 15 that ε-transitions which do not reset clocks can be removed in a timed automaton and we have proved in section 3 that ε-transitions with clock reset strictly increase the expressive power of timed automata. These ε-reset transitions are actually very useful in timed automata constructions. They often allow to give much smaller and simpler automata. For instance, they occur very naturally when we build an automaton for the concatenation: we mainly have to add ε-transitions which reset all clocks between the final states of the first automaton and the initial states of the second one. It is therefore important to study more closely these ε-reset transitions. The main result of this section is a construction which removes all ε-transitions, if there is no directed cycle of the automaton including an ε-transition with reset. This is the best one can expect as explained in the following example.

Example 20. Consider the following automaton $\mathcal{A}$ where q_0 is both initial and repeated.



The accepted language can be described as follows: In each open time interval $(i, i+1)$, $i \geq 0$, there occurs at most one b. Moreover, there is an a at time $i+1$ if and only if there is no b in $(i, i+1)$. Note that, by construction, no infinite run of $\mathcal{A}$ yields any Zeno word.

We will see (Corollary 29) that $L(\mathcal{A})$ cannot be accepted by a timed automaton without ε-transition. Therefore, an ε-transition with reset which lies inside a directed cycle cannot be removed in general.

Note that the language $L(\mathcal{A})$ can be accepted by an automaton without ε-transition which uses the periodic constraints introduced in [10]. On the other hand, it is shown in that paper, that the language

$$L = \{(a, t_1)(a, t_2) \cdots \mid \exists \alpha > O, \forall i > 0, \alpha + i - 1 < t_i < \alpha + i\}$$

cannot be accepted by an automaton without ε-transition even using modular clock constraints. Indeed, this language is easily accepted by an automaton with ε-reset transitions.

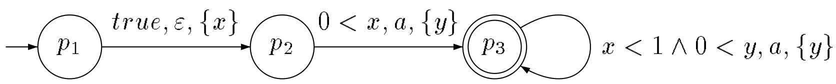
Theorem 21. *Let $\mathcal{A}$ be a timed automaton such that no ε-reset transition lies on any directed cycle. Then we can effectively construct a timed automaton $\mathcal{A}'$*

without any ε-transition such that

$$L(\mathcal{A}) \subseteq L(\mathcal{A}')$$
$$L(\mathcal{A}) \cap NZ = L(\mathcal{A}') \cap NZ$$

Before starting the construction of $\mathcal{A}'$, we would like to discuss the non-Zenoness assumption used to obtain the equality in the theorem above. One might wonder whether this hypothesis is really necessary. The following example shows that indeed one cannot simply drop this assumption.

Example 22. Consider the following automaton $\mathcal{A}$.



One can verify that the language accepted by this automaton is

$$L = \{(a, t_2)(a, t_3) \cdots \mid \exists \gamma > 0 \text{ such that } 0 \le t_2 < t_3 < \cdots < t_2 + 1 - \gamma\}.$$

Therefore, L consists of Zeno words only.

We will show that L cannot be recognized by a timed automaton without ε-transitions, even if we allow for the automaton any set of constants which is a discrete subspace of $\mathbb{R}$.

Assume by contradiction that L is recognized by some ε-free automaton $\mathcal{A}'$. Let $C \subseteq \mathbb{R}$ be its set of constants assuming that C is a discrete subspace of $\mathbb{R}$ (e.g., C is finite). Let $t_2 > 0$ be an arbitrary positive real and choose $0 < \delta < 1$ such that $\delta < \min\{c \in C \mid c > 0\}$ and $(t_2 + 1 - \delta, t_2 + 1) \cap (C \cup (t_2 + C)) = \emptyset$.

Consider a timed word $w = (a, t_2)(a, t_3)(a, t_4) \cdots$ with $t_2 + 1 - \delta < t_3 < t_4 < \cdots$ and $\lim_{i \to \infty} t_i = t_2 + 1 - \gamma$ for some $0 < \gamma < \delta$. Clearly $w \in L(\mathcal{A})$.

Then we have $w \in L(\mathcal{A}')$ and there exists some accepting path π in $\mathcal{A}'$ for the timed word w. Due to the choice of δ one can verify that the same path accepts also the timed word $w' = (a, t_2)(a, t_3 + \gamma)(a, t_4 + \gamma) \cdots$. However, since $\lim_{i \to \infty} t_i + \gamma = t_2 + 1$, we have $w' \notin L$ and therefore a contradiction.

Now that we have shown the relevance of the hypotheses of Theorem 21, we will explain the construction. The *ε-depth* of a timed automaton is defined as the maximal number of ε-reset transitions which can be found on some directed path through the automaton. An ε-reset transition is of *maximal depth*, if it is the last one on such a directed path.

The proof of Theorem 21 will be done by induction on the ε-depth of the automaton. Our strategy is as follows. First (Steps 1 to 5), we transform $\mathcal{A}$ into some normal form without increasing the ε-depth. Then we explain how to remove all ε-reset transitions of maximal depth (this is the crucial part). We end up with an automaton $\mathcal{A}'$ (being not in normal form anymore) where the ε-depth is decreased by one. The result follows by induction.

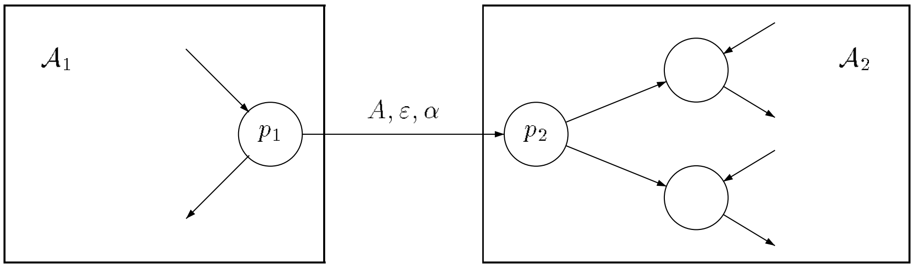
Step 1: Remove all ε-transitions without reset by the procedure of Section 7. Note that this procedure does not increase the ε-depth of $\mathcal{A}$ and that if the ε-depth of $\mathcal{A}$ is zero then we are done.

Step 2: Remove all constraints of the form $x - y \# c$ where x, y are clocks and c is some constant (note that Step 1 introduces such constraints). Note that the transformation given in Section 4.2 does not create any directed cycle. In particular, if the original automaton does not contain a directed cycle including an ε-transition, the same fact holds for the new automaton. Moreover, this construction never adds reset to transition and it adds constraints only to transitions which reset clocks. Therefore, the ε-depth of the automaton is unchanged by this construction.

By duplicating some transitions, we may also assume that all constraints are conjunction of atomic formulae of type $y \# c$ with $\# \in \{<, =, >\}$ (Section 4.1).

Note that, for the proof below, Steps 1 and 2 may be restricted to the part of the automaton which follows ε-reset transitions of maximal depth.

Step 3: Using copies of the automaton, we may now assume that every ε-reset transition of maximal depth $p_1 \xrightarrow{A,\varepsilon,\alpha} p_2$ divides the automaton into two disjoint parts and the only bridge between these parts is this ε-reset transition (See the figure below). Moreover, we may assume that p_2 has no other in-going transition. Note that $\mathcal{A}_2$ contains no ε-transition and we may assume that it contains no initial state either.



Thanks to this step, we can carry out the remaining steps simultaneously on all ε-reset transitions of maximal depth. Hence we will reduce the ε-depth by one and Theorem 21 will follow by induction.

Step 4: We replace α by a singleton $\{x\}$ where x is a new clock which is never reset in $\mathcal{A}_2$. Again, this is an easy construction on timed automata which does not change $\mathcal{A}_1$ and hence does not change the ε-depth. The idea is to keep the set $\bar{\alpha}$ of clocks in α which have not yet been reset in $\mathcal{A}_2$. Whenever a clock in $\bar{\alpha}$ is compared to some constant, one uses the new clock x instead.

Using copies of the clocks used in the constraint A which are reset with their originals inside $\mathcal{A}_1$ and are substituted to their originals in A, we may also assume that the clocks used in A are different from x and are not reset in $\mathcal{A}_2$.

Step 5: We use a new clock x_u which is reset on each transition entering state p_1. We then duplicate the ε-reset transition and the following automaton $\mathcal{A}_2$.

We add the constraint $x_u = 0$ to one ε-reset transition and the constraint $x_u > 0$ to the other one.

Similarly, we duplicate each transition issued from p_2 and we add the constraint $x = 0$ to one of them and the constraint $x > 0$ to the other one.

Step 6: This is the main step of the construction. We will replace in $\mathcal{A}$ the automaton $\mathcal{A}_2$ by a new one $\mathcal{A}_2'$ which will not use the clock x and such that there will be a correspondence between the legal paths of the new automaton $\mathcal{A}'$ and the legal ones of $\mathcal{A}$. Then we will replace α by $\emptyset$ transforming the ε-reset transition into an ε-transition without reset.

Case 6a: Assume that A contains a constraint of the form $y = c$ (note that we have added such a constraint in Step 5 to one copy of the ε-reset transition). Since neither x nor y are reset inside $\mathcal{A}_2$ (Step 4) we replace in $\mathcal{A}_2$ each constraint of the form $x \# c'$ by $y \# c + c'$ to obtain the automaton $\mathcal{A}_2'$ and we are done.

Case 6b: Assume now that A does not contain any constraint of the form $y = c$. We will first give the construction which deals with the paths not containing constraints of the form $x = c$. We will explain later how to modify the construction to cope with paths containing a constraint $x = c$. Actually, the situation is simpler in this case since we know then exactly the date of the ε-reset transition.

The states of $\mathcal{A}_2'$ will be triples (p, L, U) where p is a state of $\mathcal{A}_2$ and L, U are its lower and upper attributes. Intuitively, these attributes will keep track of the possible interval for the clock x in the corresponding run of $\mathcal{A}$. More precisely, when we reach a state p of $\mathcal{A}_2$ with a legal run of $\mathcal{A}$, then the corresponding run of $\mathcal{A}'$ will reach the state (p, L, U) with the value $x(t)$ in the open interval $(L(t), U(t))$. Conversely, for each legal run of $\mathcal{A}'$ leading to the state (p, L, U) and for each value in $(L(t), U(t))$, there exists a corresponding run in $\mathcal{A}$ leading to state p with this value for $x(t)$.

We can write $A = \bigwedge_r m_r < x_r < m_r'$ with $m_r \in C \cup \{-\infty\}$ and $m_r' \in C \cup \{+\infty\}$. Let $L_0 = \max_r(x_r - m_r')$ and $U_0 = \min_r(x_r - m_r)$ be the initial attributes given to the state p_2. One should consider the max and min used in L_0 and U_0 just as formal notations. We do not try to find out which one is the max or the min even though this can be done by duplicating the ε-reset transition and adding constraints.

Apart from the initial values L_0 and U_0, the possible values for L and U are $\{x_\ell + c \mid c \in C\}$ and $\{x_u + c \mid c \in C\}$ respectively where x_u is the clock introduced in Step 5 and x_ℓ is another new clock. Hence, there are finitely many possible attribute values and the automaton $\mathcal{A}_2'$ remains finite.

Moreover, a state (p, L, U) of $\mathcal{A}_2'$ is final (repeated resp.) if and only if the state p of $\mathcal{A}_2$ is final (repeated resp.).

For each transition $p \xrightarrow{(a<x<b)\wedge B,\sigma,\beta} q$ of $\mathcal{A}_2$ where B does not contain the clock x and $a \in C \cup \{-\infty\}$ and $b \in C \cup \{+\infty\}$, we add the following transitions

to the automaton $\mathcal{A}'_2$.

$$
\begin{aligned}
(p, L, U) &\xrightarrow{(a \le L < U \le b) \wedge B, \sigma, \beta} (q, L, U) \\
&\xrightarrow{(a \le L < b < U) \wedge B, \sigma, \beta \cup \{x_u\}} (q, L, x_u + b) \\
&\xrightarrow{(L < a < U \le b) \wedge B, \sigma, \beta \cup \{x_\ell\}} (q, x_\ell + a, U) \\
&\xrightarrow{(L < a < b < U) \wedge B, \sigma, \beta \cup \{x_\ell, x_u\}} (q, x_\ell + a, x_u + b)
\end{aligned} \tag{4}
$$

where $L_0 < a$ is an abbreviation for $\bigwedge_r (x_r - m'_r) < a$ and similarly for $a \le L_0$, $b < U_0$ and $U_0 \le b$.

Claim: All timed words (Zeno or non-Zeno) accepted by a path of $\mathcal{A}$ which does not contain a constraint of the form $x = c$ are also accepted by $\mathcal{A}'$. Conversely, all non-Zeno timed words accepted by $\mathcal{A}'$ are also accepted by $\mathcal{A}$.

Proof. Let $\pi = q_0 \xrightarrow{A_1, \sigma_1, \alpha_1} q_1 \xrightarrow{A_2, \sigma_2, \alpha_2} q_2 \cdots$ be a path of $\mathcal{A}$ and let $t_1 t_2 \cdots$ be a timed sequence such that the timed ε-word $w = (\sigma_1, t_1)(\sigma_2, t_2) \cdots$ is accepted by π. Assume that the i-th transition of π is $p_1 \xrightarrow{A, \varepsilon, \{x\}} p_2$, hence the path from q_0 to $q_{i-1} = p_1$ runs in $\mathcal{A}_1$ while the path from $q_i = p_2$ runs in $\mathcal{A}_2$. Assume finally that the path from $q_i = p_2$ does not contain any constraint of the form $x = c$.

We will construct a path π' of $\mathcal{A}'$ which accepts precisely the same timed ε-word w. The path π' starts as the path π and its i-th transition enters the "initial" state of $\mathcal{A}'_2$:

$$
q_0 \xrightarrow{A_1, \sigma_1, \alpha_1} q_1 \cdots \xrightarrow{A_{i-1}, \sigma_{i-1}, \alpha_{i-1}} q_{i-1} = p_1 \xrightarrow{A, \varepsilon, \emptyset} (p_2, L_0, U_0) = (q_i, L_i, U_i)
$$

Since the constraint A has been satisfied at time t_i, we have $L_0(t_i) < 0 < U_0(t_i)$. Hence, we have the time invariant $L_0(t) < x(t) < U_0(t)$ for all $t \ge t_i$.

Assume that the path π' has been constructed up to state $(q_{j-1}, L_{j-1}, U_{j-1})$ and that the time invariant $L_{j-1}(t) < x(t) < U_{j-1}(t)$ for all $t \ge t_{j-1}$ holds. Assume also that the constraint $A_j = (a_j < x < b_j) \wedge B_j$ where B_j does not contain the clock x. Thanks to the time invariant, we can see that among the transitions described in (4), there is exactly one transition

$$
(q_{j-1}, L_{j-1}, U_{j-1}) \xrightarrow{A'_j, \sigma_j, \alpha'_j} (q_j, L_j, U_j)
$$

whose constraint A'_j is true at time t_j. This transition is used to extend the path π'. One can easily verify that the time invariant $L_j(t) < x(t) < U_j(t)$ for all $t \ge t_j$ holds. We have thus obtained the desired path π' and we have proved the first part of the claim. Note that we do not need to assume that the time sequence $t_1, t_2, \cdots$ diverges for this part of the proof.

Conversely, let π' be a path of $\mathcal{A}'$ whose i-th transition is $p_1 \xrightarrow{A, \varepsilon, \emptyset} (p_2, L_0, U_0)$. Thus the path π' has the form

$$
q_0 \xrightarrow{A_1, \sigma_1, \alpha_1} q_1 \cdots q_{i-1} \xrightarrow{A, \varepsilon, \emptyset} (q_i, L_i, U_i) \xrightarrow{A'_{i+1}, \sigma_{i+1}, \alpha'_{i+1}} (q_{i+1}, L_{i+1}, U_{i+1}) \cdots
$$

Let $w = (\sigma_1, t_1)(\sigma_2, t_2)\cdots$ be a timed ε-word accepted by π'. For all $j > i$, let

$$q_{j-1} \xrightarrow{(a_j<x<b_j)\wedge B_j,\sigma_j,\alpha_j} q_j$$

be the transition of $\mathcal{A}_2$ from which the corresponding transition of π' was obtained. We have thus constructed a path π of $\mathcal{A}$. The timed ε-word w is not necessarily accepted by the path π but we will see that we can find a time t'_i such that the corresponding timed ε-word w' is accepted by the path π. Since w and w' differs only by the time of the i-th action which is ε, this will prove the second part of the claim.

Let $I = \bigcap_{j\geq i}(t_j - U_j(t_j), t_j - L_j(t_j))$. By construction of $\mathcal{A}'_2$, since the constraint A'_j was satisfied at time t_j, we have $L_j(t_j) < U_j(t_j)$. Hence the open intervals considered in this intersection are nonempty. Moreover, we have $L_j(t_j) = \max(L_{j-1}(t_j), a_j)$. Hence, $L_j(t_j) \geq L_{j-1}(t_j) = L_{j-1}(t_{j-1}) + t_j - t_{j-1}$. Using the same argument for the upper attribute, we deduce that these intervals are decreasing.

We will see now that $I \neq \emptyset$. This is clear if the path π' is finite. Assume now that π is infinite and that the time sequence $t_1t_2\cdots$ diverges. Note that this is the only point where we need to use non-Zenoness.

By construction, $x > 0$ is part of the constraint A_{i+1}. Hence, $a_{i+1} \geq 0$ and $L_{i+1}(t_{i+1}) = \max(L_i(t_{i+1}), a_{i+1}) \geq 0$. Moreover, for all $j > i$ we have $L_j(t_j) \geq L_{i+1}(t_{i+1}) + t_j - t_{i+1} \geq t_j - t_{i+1}$. Let j_0 be such that $t_{j_0} - t_{i+1} > \max C$ where we recall that $C \subseteq \mathbb{R}$ is the finite set of constants which are used in the constraints of the automaton $\mathcal{A}$. Since the lower attribute is changed only when it is lower than some constant in C, we deduce that $L_j = L_{j_0}$ for all $j \geq j_0$. Now, for all $j \geq j_0$, we have $\max C < L_j(t_j) < U_j(t_j) = \min(U_{j-1}(t_j), b_j)$ and then $b_j = +\infty$. It follows that $U_j = U_{j_0}$ for all $j \geq j_0$. Finally, we have proved that $I = (t_{j_0} - U_{j_0}(t_{j_0}), t_{j_0} - L_{j_0}(t_{j_0}))$ which is non empty.

To complete the proof of the claim we will show that for all $t'_i \in I$, the timed ε-word w' obtained from w by replacing time t_i by t'_i is accepted by the path π.

First, $t'_i \in (t_i - U_i(t_i), t_i - L_i(t_i))$ therefore, $L_0(t'_i) = L_0(t_i) + t'_i - t_i < 0 < U_0(t_i) + t'_i - t_i = U_0(t'_i)$ and the constraint A is satisfied at time t'_i.

Second, let $j > i$, we have $t'_i \in (t_j - U_j(t_j), t_j - L_j(t_j))$ therefore, $a_j \leq L_j(t_j) < t_j - t'_i < U_j(t_j) \leq b_j$ and the constraint $(a_j < x < b_j) \wedge B_j$ is satisfied at time t_j. This concludes the proof of the claim.

To complete the proof of Theorem 21, it remains to deal with paths which contain constraints of the form $x = c$. For this, we will enlarge the automaton $\mathcal{A}'_2$. For each transition

$$p \xrightarrow{(x=c)\wedge B,\sigma,\beta} q$$

of $\mathcal{A}_2$ where B does not contain the clock x and $c \in C$, we add the following transition to the automaton $\mathcal{A}'_2$:

$$(p, L, U) \xrightarrow{(L<c<U)\wedge B,\sigma,\beta\cup\{x_\ell\}} (q, c)$$

where the state (q, c) lies inside a new copy $\mathcal{A}_{2,c}$ of $\mathcal{A}_2$ in which each constraint of the form $x \# c'$ is replaced by $x_\ell \# c' - c$.

It is now easy to extend the claim above to cope with paths containing at least one constraint of the form $x = c$.

9 A notion of precise action

We end this paper by proposing the new notion of precise actions in an execution. This notion yields a formal tool in order to show that some timed languages are not recognizable by any timed automaton (without ε-reset transitions).

Other techniques to show the non recognizability of some timed languages have been developed in several papers [7,18,25].

In this section, we consider a timed automaton $\mathcal{A}$ which uses a set $C \subseteq \mathbb{Q}$ of *rational* constants. Moreover, we assume that $\mathcal{A}$ is disjunction free and diagonal free. Note that, from Lemmata 4 and 5, any timed automaton (with a rational set of constants) is equivalent to such a timed automaton.

Let $C_{\max}$ be the maximum value of C. Let $\delta > 0$ be such that $C \subseteq \delta\mathbb{Z}$ (the value δ can be chosen such that δ^{-1} is the common denominator of the finitely many rational constants occurring in C). Consider a finite or infinite path

$$P = p_0 \xrightarrow{A_1, a_1, \alpha_1} p_1 \xrightarrow{A_2, a_2, \alpha_2} p_2 \cdots$$

in $\mathcal{A}$. By $TS(P)$ we denote the set of all possible time sequences $t_0 t_1 t_2 \cdots$ such that $0 = t_0 \leq t_1 \leq t_2 \leq \cdots$ and

$$p_0 \xrightarrow[t_1]{A_1, a_1, \alpha_1} p_1 \xrightarrow[t_2]{A_2, a_2, \alpha_2} p_2 \cdots$$

defines a run of $\mathcal{A}$, through the path P. Let us assume that $TS(P) \neq \emptyset$. By $TS_n(P)$ we denote the set of values $r \in \mathbb{R}_+$ such that there exists a time sequence $t_0 t_1 \cdots t_n \cdots \in TS(\pi)$ with $t_n = r$. Assuming that n is not greater than the length of P, we have $TS_n(P) \neq \emptyset$.

Definition 23. The occurrence a_n of P is called a precise action in P if $TS_n(P)$ reduces to a singleton.

The aim of the present section is to show the following theorem about precise actions in a given path.

Theorem 24. *Let P, δ and $C_{\max}$ be as above. Assume that $TS(P) \neq \emptyset$ and let $t \in TS_n(P)$.*

- *If the occurrence a_n is a precise action in P, then $t \in \delta\mathbb{N}$. Moreover if $t > C_{\max}$ then there exists an occurrence a_m, with $m < n$, such that a_m is also a precise action in P and the unique element of $TS_m(P)$ belongs to the half-open interval $[t - C_{\max}, t)$.*
- *Otherwise, $TS_n(P)$ is a non-empty open interval (r, s) with $r \in \delta\mathbb{N}$ and $s \in \delta\mathbb{N} \cup \{+\infty\}$.*

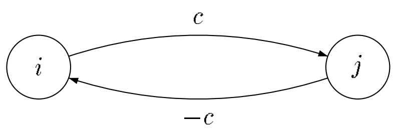
The main idea of the proof is to associate with the path P a directed graph G with edge weights such that the interval $TS_n(P)$ can be calculated from the graph. We first normalize the automaton in two simple steps as follows:

Step 1: We may assume that there is some clock x_0 such that $x_0 \geq 0$ is a constraint of A_i and $x_0 \in \alpha_i$ for all $i \geq 0$. (As we shall see later, by this assumption the graph will become connected and the condition $t_0 \leq t_1 \leq t_2 \leq \cdots$ will be a consequence of the graph construction.)

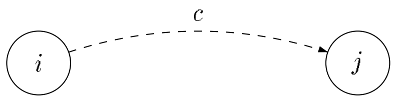
Step 2: We may assume that the path P is infinite. (If the path P is finite, enter a new state performing any action with the only constraint $x_0 > 0$. Loop in this state. This modification does not touch $TS_n(P)$, since $TS_n(P) \neq \emptyset$.)

We are now ready to define the graph G. For convenience we put $\alpha_0 = X$. The vertex set of the graph is $V = \{0, 1, 2, \ldots\} = \mathbb{N}$. There are two types of directed edges. Let $x \in X$ be a clock and $c \in C$ be a constant such that $x \in \alpha_i$ and $(x \# c) \in A_j$ for some $i < j$ and $x \notin (\alpha_{i+1} \cup \ldots \cup \alpha_{j-1})$.

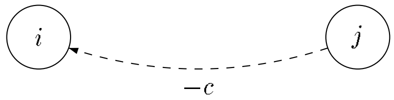
If $(x = c) \in A_j$, then we define two *strong arcs* with weights c and $-c$ respectively:



If $(x < c) \in A_j$, then we define a *soft arc* with weight c:



If, finally, $(x > c) \in A_j$ we define a soft arc with weight $-c$:



Let $t_0 t_1 t_2 \cdots \in TS(P)$ be the time sequence of some run through P. From the construction of G, it holds $t_j - t_i = w$ if there is a strong arc with weight w from i to j and $t_j - t_i < w$ if there is a soft arc with weight w from i to j. By an immediate induction on the length of a path in G, we obtain:

Lemma 25. *Let $t_0 t_1 t_2 \cdots \in TS(P)$ be the time sequence of some run through P and let $i, j \in V$ and γ_{ij} be the weight of some directed path of G from i to j.*

1. *If the path uses strong edges only, then we have $t_j - t_i = \gamma_{ij}$.*
2. *If the path uses at least one soft edge, then we have $t_j - t_i < \gamma_{ij}$.*

We define $i \sim j$, if i and j are connected by some path using strong edges, only. Clearly $\sim$ is an equivalence relation on V. Moreover, if $i, j \in V$, assuming $\inf \emptyset = +\infty$, we set

$$c_{ij} = \inf\{\gamma_{ij} \mid \gamma_{ij} \text{ is the weight of a directed path from } i \text{ to } j\}$$

In order to prove the theorem, we need a few technical results:

Lemma 26. *Let $t_0 t_1 t_2 \cdots \in TS(P)$ be the time sequence of some run through P and let $i, j, m \in V$. Then,*

1. $-c_{ji} \leq t_j - t_i \leq c_{ij}$.
2. $c_{ij} \in \delta\mathbb{Z} \cup \{+\infty\}$ *and if* $c_{ij} \neq +\infty$ *then there exists a directed path in* G *from* i *to* j *of weight* c_{ij}.
3. $i \sim j \iff t_j - t_i = c_{ij} \iff c_{ij} + c_{ji} = 0$.
4. $c_{ij} \leq c_{im} + c_{mj}$.
5. $c_{ij} = c_{im} + c_{mj}$ *if* $i \sim m$ *or* $m \sim j$.
6. $c_{ij} < c_{im} + c_{mj}$ *if* $i \sim j$ *and* $i \not\sim m$.

Proof.

1. Clear by Lemma 25.
2. Follows from the fact that $\delta\mathbb{Z}$ is a discrete subset of $\mathbb{R}$ and c_{ij} is the infimum of a bounded below subset of $\delta\mathbb{Z}$.
3. Assume that $i \sim j$, there exists in G a directed path using strong edges only from i to j of weight w. From Lemma 25, $t_j - t_i = w \geq c_{ij}$. Using (1) the equality $t_j - t_i = c_{ij}$ follows.
   If $t_j - t_i = c_{ij}$, c_{ij} is finite and thus, from (2) there exists in G a directed path from i to j of weight c_{ij}. From Lemma 25, this path uses strong edges only. Therefore $i \sim j$. Moreover there is thus a path from j to i with strong edges only too, of weight $-c_{ij}$. Hence $t_i - t_j = -c_{ij} \geq c_{ji}$. Thus $c_{ji} = -c_{ij}$.
   If $c_{ji} + c_{ij} = 0$, we obtain directly from (1) that $t_j - t_i = c_{ij}$.
4. Follows from the definition of c_{ij} by considering the paths from i to j going through m.
5. From (4), we get $c_{ij} \leq c_{im} + c_{mj}$. Assume now, for instance, that $i \sim m$. Then, from (3) $c_{im} = -c_{mi}$. Thus $c_{ij} \leq -c_{mi} + c_{mj}$ i.e. $c_{mj} \geq c_{mi} + c_{ij}$. Since, from (4), the converse inequality holds, (5) follows.
6. From (3), it holds $c_{ij} = t_j - t_i = t_j - t_m + t_m - t_i$. From (1), $t_j - t_m \leq c_{mj}$ and since $i \not\sim m$, from (1) and (3), $t_m - t_i < c_{im}$. Therefore (6) holds.

We now give a characterization of the possible time sequences for the path P.

Lemma 27. *We have*

$$TS(P) = \{t_0 t_1 t_2 \cdots \mid \forall i, j \in V,\ t_j - t_i = c_{ij} \text{ if } i \sim j \text{ and } t_j - t_i < c_{ij} \text{ otherwise}\}.$$

Proof. The inclusion "$\subseteq$" follows directly from Lemma 26 (1) and (3). Conversely, from Step 1, we deduce that $c_{ji} \leq 0$ if $j > i$. Hence, $t_i \leq t_j$ for $i < j$ and the time sequence is increasing. Assume now that there is a strong arc with weight c from i to j with $i < j$. Then, $t_j - t_i = c_{ij} \leq c$ and $t_i - t_j = c_{ji} \leq -c$. It follows that $t_j - t_i = c$. Finally, assume that there is a soft arc with weight c from i to j with $i < j$. Then, either $i \sim j$ and $t_j - t_i = c_{ij} < c$ where the last inequality holds since $TS(P) \neq \emptyset$, or $i \not\sim j$ and $t_j - t_i < c_{ij} \leq c$. We deduce that the time sequence $t_0 t_1 t_2 \cdots \in TS(P)$ since it satisfies all the constraints of the path P.

We can now state the decisive lemma from which the theorem will be an easy consequence.

Lemma 28. *We have*

$$TS_n(P) = \begin{cases} \{c_{0n}\} & \textit{if } n \sim 0 \\ (-c_{n0}, c_{0n}) & \textit{otherwise.} \end{cases}$$

Proof. If $n \sim 0$ then the result is a consequence of Lemma 26 (3) since $t_0 = 0$.

Assume now that $n \not\sim 0$ and let t_n be in $TS_n(P)$. From Lemma 26 (1) and (3), we get $-c_{n0} < t_n < c_{0n}$. Conversely let t be in $(-c_{n0}, c_{0n})$, we will construct a sequence $t_0 t_1 t_2 \cdots \in TS(P)$ such that $t_n = t$. To this purpose, we will use the characterization of $TS(P)$ given by Lemma 27.

We define inductively a sequence $s(k)$ of subsets of V satisfying the following properties:

1. $s(k)$ is a finite union of equivalence classes of $\sim$.
2. For all $i \in s(k)$, a time t_i is defined.
3. For all $i, j \in s(k)$, we have

$$\begin{cases} t_j - t_i = c_{ij} \text{ if } i \sim j \\ t_j - t_i < c_{ij} \text{ otherwise.} \end{cases}$$

To begin with, let $s(0) = [0] = \{i \in V \mid 0 \sim i\}$ the equivalence class of 0. For $i \in [0]$ define $t_i = c_{0i}$. Clearly 1) and 2) are satisfied. To see 3) observe that for $i \sim j \sim 0$ we have by Lemma 26 (3) and (5), $c_{ij} = c_{i0} + c_{0j}$ and $c_{0i} = -c_{i0}$.

Since $n \not\sim 0$, $n \notin s(0)$. Thus, we define $s(1) = s(0) \cup [n] = s(0) \cup \{j \in V \mid j \sim n\}$. We set $t_n = t$ and for any $j \sim n$, $t_j = t_n + c_{nj}$. Now, 1) and 2) hold for $s(1)$. To verify 3), it suffices to note that, for any $i \sim 0$ and $j \sim n$, $t_j - t_i = (t_n + c_{nj}) - c_{0i}$. But by choice of t, $t_n < c_{0n}$. Moreover, from Lemma 26 (5), $c_{0n} + c_{nj} = c_{0j}$ and from Lemma 26 (4), $c_{0j} - c_{0i} \leq c_{ij}$. Hence $t_j - t_i < c_{ij}$. The proof that $t_i - t_j < c_{ji}$ is similar.

Assume that $s(k)$ has been defined, $k \geq 1$. If $s(k) = V$ we stop the procedure. Otherwise, for each $m \in V \setminus s(k)$, define an open interval

$$I_m^{(k)} = (r_m^{(k)}, s_m^{(k)})$$

by $r_m^{(k)} = \max\{t_i - c_{mi} \mid t_i \in s(k)\}$ and $s_m^{(k)} = \min\{t_i + c_{im} \mid t_i \in s(k)\}$.

Note that for $i \sim j \in s(k)$ and $m \in V$, by Lemma 26 (3) and (5), we have $t_i - c_{mi} = t_j - c_{mj}$ and $t_i + c_{im} = t_j + c_{jm}$. Hence the maximum and the minimum are taken over finite sets. The interval $I_m^{(k)}$ is nonempty since $t_j - c_{mj} < t_i + c_{im}$ for all $i, j \in s(k), m \notin s(k)$. Indeed, for $i \sim j$ we have $t_j - t_i = c_{ij} < c_{im} + c_{mj}$ (Lemma 26 (6)) and for $i \not\sim j$ we have $t_j - t_i < c_{ij} \le c_{im} + c_{mj}$ (Lemma 26 (4)). Hence the claim in both cases.

We now construct $s(k+1)$ as follows. Choose the smallest $m \in V \setminus s(k)$. Define $s(k+1) = s(k) \cup \{j \in V \mid j \sim m\}$ so that 1) holds for $s(k+1)$. Let t_m be any value in the open interval $I_m^{(k)}$. For all $j \sim m$ define $t_j = t_m + c_{mj}$. Now 2) holds for $s(k+1)$ as well. Finally, the proof of 3) for $s(k+1)$ is similar to the proof of 3) for $s(1)$. Let $i \in s(k)$ and let $j \sim m$. Then $t_j - t_i = t_m + c_{mj} - t_i$, But by definition, $t_m < s_m^{(k)} \le t_i + c_{im}$. Thus $t_j - t_i < c_{im} + c_{mj}$ and $c_{im} + c_{mj} = c_{ij}$ from Lemma 26 (5). The proof that $t_i - t_j < c_{ji}$ is identical.

Hence the sequence $s(k)$ is constructed by induction and $\bigcup_{k \ge 0} s(k) = V$. Thus, this defines values t_m for all $m \ge 0$ and thereby a sequence $t_0 t_1 t_2 \cdots$ with $t_n = t$. By property 3) above and Lemma 27 we see that $t_0 t_1 t_2 \cdots \in TS(P)$.

Proof of Theorem 24. Let $t \in TS_n(P)$. If the occurrence a_n is a precise action in P, it holds from Lemma 28 that $n \sim 0$ and $t = c_{0n}$. Thus, from Lemma 26 (2), $t \in \delta\mathbb{N}$. Moreover assume that $t > C_{\max}$. Since $n \sim 0$, there exists in G a path from 0 to n composed of strong arcs: $n_0 = 0 \longrightarrow n_1 \longrightarrow \cdots \longrightarrow n_k = n$. Since $c_{0n_1} \le C_{\max} < c_{0n}$, the set $\{j \mid 1 \le j \le k \wedge c_{0n_j} < c_{0n}\}$ is nonempty. Let i be its maximal element. It holds $c_{0n} \le c_{0n_{i+1}}$, thus $c_{0n} - c_{0n_i} \le c_{0n_{i+1}} - c_{0n_i} \le C_{\max}$. Whence the occurrence a_{n_i} is a precise action in P which occurs at a date in the half-open interval $[t - C_{\max}, t)$.

If the occurrence a_n is not a precise action in P, then by Lemma 28 we have $TS_n(P) = (-c_{n0}, c_{0n})$ and by Lemma 26 we obtain $-c_{n0} \in \delta\mathbb{N}$ and $c_{0n} \in \delta\mathbb{N} \cup \{+\infty\}$ which concludes the proof.

10 A simple application

The theorem above yields a tool to prove that certain languages are not recognizable by timed automata (without ε-reset transitions resp.). An easy example is the language L_{one} consisting of one infinite word:

$$\{(a, 1)(a, 1 + \frac{1}{2})(a, 2 + \frac{1}{3}) \cdots (a, (i-1) + \frac{1}{i}) \cdots\}.$$

In fact, it is quite obvious that this language cannot be recognized by any timed automaton. Our theorem yields a rigorous formal tool to see this. Indeed, assume that there exists some timed automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L_{one}$. We keep the notations of the previous section. Let P be an accepting path in $\mathcal{A}$, since $L(\mathcal{A})$ reduces to a unique word, all the occurrences in P are precise. Therefore, for any $i \ge 1$, $(i-1) + \frac{1}{i} \in \delta\mathbb{N}$ which is of course impossible.

Another application of Theorem 24 is the following.

Corollary 29. *Every timed automaton recognizing the language $L(\mathcal{A}_0)$ from Ex. 20 has an ε-reset transition lying on some directed cycle.*

Proof. By contradiction. Assume $L(\mathcal{A}_0)$ would be recognized by some automaton without any ε-reset transition lying on some directed cycle. By Theorem 21 we may assume $L(\mathcal{A}_0)$ is recognized by some automaton without any ε-transition. From Lemmata 4 and 5, we may transform this automaton into a disjunction free and diagonal free one. Let $\delta > 0$ and $C_{\max}$ be the constants for this automaton introduced at the beginning of Section 9. We consider now an accepted word of the form

$$(b, \delta_1)(b, \delta_2) \cdots (b, \delta_{d-1})(a, d)(a, d+1) \cdots$$

with $d \in \mathbb{N}$, $d \geq C_{\max}$ and $\delta_i \in (i-1, i) \setminus \delta\mathbb{N}$ for all $0 < i < d$. Let P be a path accepting this timed word. Since the actions a only occur at integer times, all the occurrences of a should be precise. In particular, the first occurrence of a is precise. Since $d > C_{\max}$, we deduce from Theorem 24 that there exists some occurrence of b which should be precise in P which is in contradiction with the hypothesis that $\delta_i \notin \delta\mathbb{N}$ for all $0 < i < d$.

References

1. R. Alur, C. Courcoubetis, D.L. Dill, N. Halbwachs, and H. Wong-Toi. Minimization of timed transition systems. In *Proceedings of CONCUR'92*, number 630 in Lecture Notes in Computer Science. Springer Verlag, 1992.
2. R. Alur, C. Courcoubetis, and T.A. Henzinger. The observational power of clocks. In *Proceedings of CONCUR'94*, number 836 in Lecture Notes in Computer Science, pages 162–177. Springer Verlag, 1994.
3. 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.
4. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
5. R. Alur, L. Fix, and T.A. Henzinger. A determinizable class of timed automata. In *Proceedings of CAV'94*, number 818 in Lecture Notes in Computer Science, pages 1–13. Springer Verlag, 1994.
6. R. Alur and T.A. Henzinger. Back to the future: towards a theory of timed regular languages. In *Proceedings of FOCS'92*, Lecture Notes in Computer Science, pages 177–186. Springer Verlag, 1992.
7. D. Beauquier. Pumping lemmas for timed automata. In *Proceedings of FoSSaCS'98*, number 1378 in Lecture Notes in Computer Science, pages 81–94. Springer Verlag, 1998.
8. B. Bérard. Untiming timed languages. *Information Processing Letters*, 55:129–135, 1995.
9. B. Bérard, P. Gastin, and A. Petit. On the power of non observable actions in timed automata. In C. Puech and R. Reischuk, editors, *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science (STACS'96)*, number 1046 in Lecture Notes in Computer Science, pages 257–268. Springer Verlag, 1996.

10. Ch. Choffrut and M. Goldwurm. Timed automata with periodic clock constraints. Tech. Rep. 225-98, Università degli Study di Milano (Italy), Dipartimento di Scienze dell'Informazione, 1998.
11. E. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal-logic specifications. *ACM Trans. Programming Languages and Systems*, 8:244–263, 1986.
12. 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.
13. 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.
14. T.A. Henzinger. The theory of hybrid automata. In *Proceedings of LICS'96*, pages 278 –292, New Brunswick, New Jersey, 1996. Invited tutorial.
15. T.A. Henzinger, P.W. Kopke, and H. Wong-Toi. The expressive power of clocks. In *Proceedings of ICALP'95*, number 944 in Lecture Notes in Computer Science, pages 335–346. Springer Verlag, 1995.
16. T.A. Henzinger, Z. Manna, and A. Pnueli. Temporal proofs methodologies for real-time systems. In *Proceedings of POPL'91*, pages 353–366, 1991.
17. 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.
18. Ph. Herrmann. Timed automata and recognizability. *Information Processing Letters*, 65:313–318, 1988.
19. N. Lynch and H. Attiya. Using mappings to prove timing properties. In *Proceedings of PODC'90*, pages 265–280, 1990.
20. X. Nicollin, J. Sifakis, and S. Yovine. From ATP to timed graphs and hybrid systems. In *Proceedings of REX workshop "Real-time: theory in practice"*, number 600 in Lecture Notes in Computer Science, pages 549–572. Springer Verlag, 1991.
21. J. Ostroff. *Temporal Logic of Real-time Systems.* Research Studies Press, 1990.
22. C. Ramchandani. Analysis of asynchronous concurrent systems by Petri nets. Technical report, Massachusetts Institute of Technology, 1974.
23. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 133–191. Elsevier Science Publishers (North-Holland), 1990.
24. M. Vardi. Verification of concurrent-programs – the automata-theoretic framework. In *Proceedings of LICS'87*, pages 167–176, 1987.
25. T. Wilke. *Automaten und Logiken zur Beschreibung zeitabhängiger Systeme.* PhD thesis, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel, 1994. Available as Tech. Rep. 9408.
26. T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *Lecture Notes in Computer Science.* Springer Verlag, 1994.
27. P. Wolper, M. Vardi, and A.P. Sistla. Reasoning about infinite computation paths. In *Proceedings of FOCS'83*, pages 185–194, 1983.
28. H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *Proceedings of the 30th IEEE Conf. on Decision and Control*, pages 1527–1528, 1991.