Centre Fédéré en Vérification

Technical Report number 2004.30

Robustness and Implementability of Timed Automata

Martin De Wulf (ULB), Laurent Doyen (ULB), Nicolas Markey (ULB), Jean-François Raskin (ULB)

This work was partially supported by a FRFC grant: 2.4530.02

http://www.ulb.ac.be/di/ssd/cfv

Robustness and Implementability of Timed Automata*

Martin De Wulf[1], Laurent Doyen[1]**, Nicolas Markey[2], and Jean-François Raskin[1]

[1] Computer Science Department, Université Libre de Bruxelles, Belgium
[2] Lab. Spécification & Vérification, CNRS & ENS Cachan, France

Abstract. The *Almost ASAP* semantics is a parameterized semantics for Timed Automata that has been developped in [DDR05] to cope with the reaction delay of the controller. That semantics is *implementable* provided there exists strictly positive values for the parameters Δ and ε for which the strategy is correct. In this paper, we define the implementability problem to be the question of the existence of such values for the parameters. We show that this question is closely related to a notion of robustness for timed automata defined in [Pur98], and prove that the implementability problem is decidable.

1 Introduction

Timed automata are an important formal model for the specification and analysis of real-time systems. Formalisms like timed automata and hybrid automata are central in the so-called *model-based development methodology for embedded systems.* The steps underlying that methodology can be summarized as follows: *(i)* construct a (timed/hybrid automaton) model Env of the environment in which the controller will be embedded; *(ii)* make clear what is the control objective: for example, prevent the environment to enter a set of Bad states; *(iii)* design a (timed automaton) model Cont of the control strategy; *(iv)* verify that $\mathsf{Reach}([\![\mathsf{Env} \parallel \mathsf{Cont}]\!]) \cap \mathsf{Bad} = \varnothing$ (where $\mathsf{Reach}([\![\mathsf{Env} \parallel \mathsf{Cont}]\!])$ denotes the set of states reachable in the transition system associated to the synchronized product of the automaton for the environment and the automaton for the controller). When Cont has been proved correct, it would be valuable to ensure that an implementation Impl of that model can be obtained in a systematic way in order to ensure the conservation of correctness, that is to ensure that $\mathsf{Reach}([\![\mathsf{Env} \parallel \mathsf{Impl}]\!]) \cap \mathsf{Bad} = \varnothing$ is obtained by construction.

Unfortunately, this is often not possible for several *fundamental* and/or *technical* reasons. First, the notion of time used in the traditional semantics of timed automata is *continuous* and defines *perfect clocks* with *infinite precision*, while implementations can only access time through *digital* and *finitely precise* clocks.

* Supported by the FRFC project "Centre Fédéré en Vérification" funded by the Belgian National Science Foundation (FNRS) under grant nr 2.4530.02
** Research fellow supported by the Belgian National Science Foundation (FNRS)

Second, timed automata react *instantaneously* to events and time-outs while implementations can only react within a given, usually small but not zero, *reaction delay*. Third, timed automata may describe control strategies that are *unrealistic*, like *zeno-strategies* or strategies that demand the controller *to act faster and faster* [CHR02]. For those reasons, a model for a digital controller that has been proved correct may not be implementable (at all) or it may not be possible to turn it systematically into an implementation that is still correct.

To overcome those problems, [DDR05] recently proposed an alternative semantics for timed automata. This semantics is called the *Almost ASAP* semantics, AASAP for short. The AASAP-semantics of a timed automaton A, denoted by $[\![A]\!]_{\Delta}^{\varepsilon}$, is a 2-parameter semantics that leaves as a parameter the *reaction delay* Δ of the controller and the *drifts* (*i.e.* imprecision) ε on clocks. Both parameters take their values in $\mathbb{Q}^{+}$. This semantics relaxes the classical semantics of timed automata in that it does not impose on the controller to react instantaneously but imposes on the controller to react *within Δ time units*. As shown in [DDR05], a timed controller is implementable with a *sufficiently fast* and *sufficiently precise* hardware if, and only if, there exists $\Delta, \varepsilon \in \mathbb{Q}^{>0}$ such that $\mathsf{Reach}([\![\mathsf{Env} \parallel \mathsf{Cont}]\!]_{\Delta}^{\varepsilon}) \cap \mathsf{Bad} = \varnothing$. The *implementability problem* is to determine the existence of such values for Δ and ε. Details on the notion of implementability can be found in [DDR05].

The use of the AASAP-semantics in the verification phase can be understood intuitively as follows. When we verify a control strategy using the AASAP-semantics, we test if the proposed strategy is *robust* in the following sense[3]: "*Is the strategy still correct if it is perturbed a little bit when executed on a device that has a finite speed and uses finitely precise clocks?*"

In this paper, we show that this intuition relating robustness and implementability allows us to draw an interesting and important link with a paper by Puri [Pur98] and allows us to *positively* answer the open question about the decidability of the implementability problem.

Related works. In this paper, we focus on timed controllers and environments that can be modeled using timed automata. There exist related works where the interested reader will find other ideas about implementability.

In [AT05], Altisen and Tripakis tackle the problem of implementability by modeling the execution platform as a timed automaton. Their approach is much more expressive than ours, but it suffers from not verifying the "faster is better" property.

In [HKSP03], Tom Henzinger *et al.* introduce a programming model for real-time embedded controllers called GIOTTO. GIOTTO is an embedded software model that can be used to specify a solution to a given control problem independently of an execution platform but which is closer to executable code than a mathematical model. So, GIOTTO can be seen an intermediary step between mathematical models like hybrid automata and real execution platforms.

[3] Our notion of robustness is different from another interseting one introduced in [GHJ97].

In [AIK$^+$03], Rajeev Alur *et al.* consider the more general problem of generating code from hybrid automata, but they only sketch a solution and state interesting research questions. The work in this paper should be useful in that context. In [AFM$^+$02,AFP$^+$03], Wang Yi et al. present a tool called Times that generates executable code from timed automaton models. However, they make the synchrony hypothesis and so they *assume* that the hardware on which the code is executed is infinitely fast and precise.

Our paper is structured as follows. In Section 2, we recall some classical definitions related to timed automata and we introduce a general notion of enlarged semantics for those automata. In Section 3, we recall the essential notions and problems related to the AASAP-semantics, and we recall the notion of robustness as introduced in [Pur98]. In Section 4, we present a small example that illustrates the enlarged semantics and the problems that we want to solve on this semantics. In Section 5, we make formal the link between our notion of implementability and the notion of robustness introduced by Puri. In Section 6, we give a direct proof that the implementability problem is decidable, and study its complexity.

2 Preliminaries

Definition 1 A *timed transition system* (TTS for short) $\mathcal{T}$ is a tuple $\langle S, \iota, \Sigma, \rightarrow\rangle$ where S is a (possibly infinite) set of states, $\iota \in S$ is the initial state, Σ is a finite set of labels, and $\rightarrow \subseteq S \times (\Sigma \cup \mathbb{R}^{\geq 0}) \times S$ is the transition relation where $\mathbb{R}^{\geq 0}$ is the set of positive real numbers. If $(q, \sigma, q') \in \rightarrow$, we write $q \xrightarrow{\sigma} q'$. □

A *trajectory* of a TTS $\mathcal{T} = \langle S, \iota, \Sigma, \rightarrow\rangle$ is a sequence $\pi = (s_0, t_0) \ldots (s_k, t_k)$ such that for $0 \leq i \leq k$, we have $(s_i, t_i) \in S \times \mathbb{R}$, and for $0 \leq i < k$, we have $s_i \xrightarrow{\sigma_i} s_{i+1}$, and either $\sigma_i \in \Sigma$ and $t_{i+1} = t_i$, or $\sigma_i \in \mathbb{R}^{\geq 0}$ and $t_{i+1} = t_i + \sigma_i$. We sometimes refer to this trajectory as $\pi[t_0, t_k]$ and write $\pi(t_i)$ instead of s_i.

The *trace* of such a trajectory π is the sequence $(t_i)_i$ where $t_i = \sigma_i$ if σ_i belongs to Σ, and $t_i = \tau$ if $\sigma_i \in \mathbb{R}^{\geq 0}$ (assuming τ is a special symbol not in Σ).

A trajectory is *stutter-free* iff its trace contains alternately a symbol in Σ and a τ, *i.e.* does not contains two consecutive symbols in Σ or two consecutive τ's. Any trajectory can obviously be made stutter-free.

A state s of $\mathcal{T}$ is *reachable* from a state q if there exists a trajectory $\pi = (s_0, t_0) \ldots (s_k, t_k)$ such that $s_0 = q$ and $s_n = s$. Given a set of states $Q \subseteq S$, we write $\mathsf{Reach}(Q)$ for the set of states that are reachable from (at least) one state in Q. We abusively write $\mathsf{Reach}(\mathcal{T})$ for $\mathsf{Reach}(\{\iota\})$, the set of states that are reachable from the initial state of $\mathcal{T}$.

Given a set $\mathsf{Var} = \{x_1, \ldots, x_n\}$ of clocks, a *clock valuation* is a function $v \colon \mathsf{Var} \rightarrow \mathbb{R}^{\geq 0}$. In the sequel, we often identify a clock valuation with a point in $\mathbb{R}^n$. If $R \subseteq \mathsf{Var}$, then $v[R := 0]$ denotes the valuation v' such that

$$v'(x) = \begin{cases} 0 & \text{if } x \in R \\ v(x) & \text{otherwise} \end{cases}$$

A *closed rectangular guard* g over $\{x_1, \ldots, x_n\}$ is a set of inequalities of the form $a_i \leq x_i \leq b_i$, one for each x_i, where $a_i, b_i \in \mathbb{Q}^{\geq 0} \cup \{+\infty\}$ and $a_i \leq b_i$. We write $\mathsf{Rect}_c(\mathsf{Var})$ for the set of closed rectangular guards over Var. For $\varDelta \geq 0$, we define $[\![g]\!]_\varDelta = \{(x_1, \ldots, x_n) \mid a_i - \varDelta \leq x_i \leq b_i + \varDelta\} \subseteq \mathbb{R}^n$. When $\varDelta = 0$, we write $[\![g]\!]$ instead of $[\![g]\!]_0$.

We slightly modify the classical definitions related to timed automata [AD94]. In particular, all clocks are assumed to be bounded by some constant M, and guards on edges are rectangular and closed. The first two requirements are not restrictive (except w.r.t. the conciseness of the models [BC05]), the third one is discussed in a remark below.

Definition 2 A *timed automaton* is a tuple $A = \langle \mathsf{Loc}, \mathsf{Var}, q_0, \mathsf{Lab}, \mathsf{Edg} \rangle$ where

- Loc is a finite set of locations representing the discrete states of A.
- $\mathsf{Var} = \{x_1, \ldots, x_n\}$ is a finite set of real-valued variables.
- $q_0 = (l_0, v_0)$, where $l_0 \in \mathsf{Loc}$, is the initial location and v_0 is the initial clock valuation such that $\forall x \in \mathsf{Var}.\ v_0(x) \in \mathbb{N} \wedge v_0(x) \leq M$.
- Lab is a finite alphabet of labels.
- $\mathsf{Edg} \subseteq \mathsf{Loc} \times \mathsf{Loc} \times \mathsf{Rect}_c(\mathsf{Var}) \times \mathsf{Lab} \times 2^{\mathsf{Var}}$ is the set of transitions. A transition (l, l', g, σ, R) represents a jump from location l to location l' with guard g, event σ and a subset $R \subseteq \mathsf{Var}$ of variables to be reset. $\square$

We now define a family of semantics for timed automata that is parameterized by $\varepsilon \in \mathbb{Q}^{\geq 0}$ (drift on clocks) and $\varDelta \in \mathbb{Q}^{\geq 0}$ (imprecision on guards).

Definition 3 The semantics of a timed automaton $A = \langle \mathsf{Loc}, \mathsf{Var}, q_0, \mathsf{Lab}, \mathsf{Edg} \rangle$, when the two parameters ε and $\varDelta$ are fixed is given by the TTS $[\![A]\!]^\varepsilon_\varDelta = \langle S, \iota, \Sigma, \rightarrow \rangle$ where

1. $S = \{(l, v) \mid l \in \mathsf{Loc} \wedge v \colon \mathsf{Var} \rightarrow [0, M]\}$;
2. $\iota = q_0$;
3. $\Sigma = \mathsf{Lab}$;
4. The transition relation $\rightarrow$ is defined by
   (a) For the discrete transitions: $((l, v), \sigma, (l', v')) \in \rightarrow$ iff there exists an edge $(l, l', g, \sigma, R) \in \mathsf{Edg}$ such that $v \in [\![g]\!]_\varDelta$ and $v' = v[R := 0]$;
   (b) For the continuous transitions: $((l, v), t, (l', v')) \in \rightarrow$ iff $l = l'$ and $v'(x_i) - v(x_i) \in [(1 - \varepsilon)t, (1 + \varepsilon)t]$ for $i = 1 \ldots n$. $\square$

In the sequel, we omit $\varDelta$ and/or ε when they are equal to zero, and, for instance, write $[\![A]\!]$ for $[\![A]\!]^0_0$, which is the classical semantics of timed automata.

Remark. Our definition of timed automata does not use strict inequalities; This is not restrictive in the presence of guard enlargement. Indeed, consider a timed automaton A with (possibly open) rectangular guards and the closure automaton $\widehat{A}$ resulting from A by replacing strict inequalities by non-strict ones. It appears obviously that

$$\mathsf{Reach}([\![\widehat{A}]\!]^\varepsilon_{\frac{\varDelta}{2}}) \subseteq \mathsf{Reach}([\![A]\!]^\varepsilon_\varDelta) \qquad \text{and} \qquad \mathsf{Reach}([\![A]\!]^\varepsilon_\varDelta) \subseteq \mathsf{Reach}([\![\widehat{A}]\!]^\varepsilon_\varDelta),$$

and hence the implementability problem on A ("*Do there exist* $\Delta, \varepsilon \in \mathbb{Q}^{>0}$ *such that* $\mathsf{Reach}(\llbracket A \rrbracket^{\varepsilon}_{\Delta}) \cap \mathsf{Bad} = \varnothing$ *?*") is equivalent to the the implementability problem on $\widehat{A}$.

We now recall some additional classical notions related to timed automata. In the sequel, $\lfloor x \rfloor$ denotes the integer part of x (the greatest integer $k \leq x$), and $\langle x \rangle$ denotes its fractional part.

Definition 4 A *clock region* is an equivalence class of the relation $\sim$ defined over the clock valuations in $\mathsf{Var} \to [0, M]$. We have $v \sim w$ iff the following three conditions hold:

- $\forall x \in \mathsf{Var}.\ \lfloor v(x) \rfloor = \lfloor w(x) \rfloor$;
- $\forall x \in \mathsf{Var}.\ \langle v(x) \rangle = 0$ iff $\langle w(x) \rangle = 0$.
- $\forall x, y \in \mathsf{Var}.\ \langle v(x) \rangle \leq \langle v(y) \rangle$ iff $\langle w(x) \rangle \leq \langle w(y) \rangle$; $\square$

We write (v) for the clock region containing v. It is easy to see that (v) contains the valuations that agree with v on the integer part of the variables, and on the ordering of their fractional part and zero. Given a region r, its topological closure is denoted by $[r]$. Such a set is called a *closed region* hereafter. We write $[v]$ instead of $[(v)]$ for the closed region containing valuation v. The following Lemma explains why we mainly deal with closed regions in the sequel:

Lemma 5 *Let A be a timed automaton (with closed guards), and $[r]$ be a closed region of A. Then* $\mathsf{Reach}([r])$ *is a union of closed regions.*

It is easily proved by showing that the successor of a closed region by a single transition is a union of closed regions.

Definition 6 Given the TTS $\llbracket A \rrbracket = \langle S, s_0, \Sigma, \to_A \rangle$ of a timed automaton A, we define the corresponding *region graph* $G = \langle \mathcal{C}, \to_G \rangle$ of A:

- $\mathcal{C} = \{(l, (v)) \mid (l, v) \in S\}$ is the set of regions;
- $\to_G \subseteq \mathcal{C} \times \mathcal{C}$, and $((l, (v)), (l', (v'))) \in \to_G$ if, and only if, $(l, v) \to_A (l', v')$ and $(l, (v)) \neq (l', (v'))$. $\square$

This definition is meaningful since $\mathcal{C}$ is finite (the total number of regions, denoted in the sequel by $W = |\mathcal{C}|$, is exponential in the size of A) and, if $(l, (v)) \to_G (l', (v'))$, then for any $s \in (v)$, there exists an $s' \in (v')$ such that $(l, s) \to_A (l', s')$, and conversely, for any $s' \in (v')$, there exists $s \in (v)$ such that $(l, s) \to_A (l', s')$ [AD94].

Given a path $p = p_0\, p_1 \cdots p_N$ in the region graph of a timed automaton A, and a trajectory $\pi = (s_0, t_0)\,(s_1, t_1) \cdots (s_k, t_k)$ of $\llbracket A \rrbracket$, we say that π *follows* p if there exists a function $f \colon [0, k] \to [0, N]$ such that $f(0) = 0$, $f(k) = N$, for all $i \in [1, k]$ either $f(i) = f(i-1)$ or $f(i) = f(i-1) + 1$ and for all $i \in [0, k]$: $s_i \in [p_{f(i)}]$.

Definition 7 A *zone* $Z \subseteq \mathbb{R}^n$ is a *closed set* defined by inequalities of the form

$$x_i - x_j \leq m_{ij}, \quad \alpha_i \leq x_i \leq \beta_i$$

where $1 \leq i, j \leq n$ and $m_{ij}, \alpha_i, \beta_i \in \mathbb{Z}$. A set of states is called a *zone-set* if it is a finite union of sets of the form $\{l\} \times Z$ where l is a location and Z is a zone. □

Definition 8 A region r is *time-elapsing* if time can elapse in that region, *i.e.* if $\exists v \in r. \exists t > 0. v + t \in r$. A cycle in the region graph is *time-elapsing* if it contains a time-elapsing region. □

Definition 9 A *progress cycle* in the region graph of a timed automaton is a cycle in which each clock is reset at least once. □

Our main results heavily rely on the fact that timed automata should not have *weird* behaviors. More precisely, we rule out timed automata that contains instant cycles[4]. However, several interesting intermediate results hold in the general case. We thus formulate the following assumption, but we will always explicitely refer to it when it is needed.

Assumption 10 *We only consider timed automata satisfying the following condition:* Every cycle in the region graph is time-elapsing.

In his original work, Puri formulates a more restrictive assumption that all cycles in the region graph are progress cycles. The only possible progress cycles that are not time-elapsing are cycles along which all regions bind all clocks to zero. Apart from those very special cycles, our assumption is more general than Puri's one. The progress cycle assumption itself is not very restrictive, since it is weaker than classical non-Zeno assumptions in the literature (for example in [AMPS98], they impose that "in every cycle in the transition graph of the automaton, there is at least one transition which resets a clock variable x_i to zero, and at least one transition which can be taken only if $x_i \geq 1$").

3 AASAP semantics and enlarged semantics

The "Almost ASAP" semantics has been introduced in [DDR05]. That semantics relaxes the usual semantics of timed automata, its main characteristics are summarized as follows:

– any transition that can be taken becomes urgent only after a small delay Δ (which may be left as a parameter);
– a distinction is made between the occurrence of an event in the environment (sent) and in the controller (received), however the time difference between the two events is bounded by Δ;

[4] See Section 6.5 for comments on the general case.

– guards are enlarged by some small amount depending on Δ.

In the same paper, in Theorem 6, we show that this semantics can be encoded using a syntactical transformation of the automaton controller and by enlarging the guards by the parameter Δ which takes its value in the positive rationals. So we can study the AASAP-semantics of $\mathsf{Env} \parallel \mathsf{Cont}$ by considering the semantics $[\![\mathsf{Env} \parallel \mathsf{Cont}']\!]^0_\Delta$ where Cont' is obtained syntactically from Cont. So in the rest of this paper, we will consider the Δ-enlarged semantics instead of the AASAP-semantics.

In this previous work, we have shown that the AASAP-semantics and so the Δ-enlarged semantics allow us to reason about the implementability of a control strategy defined by a timed automaton. The problems that we want to solve algorithmically on the Δ-enlarged semantics are the following ones:

– **[Fixed]** given a zone-set of Bad states, the timed automata Env and Cont, and the value of $\Delta \in \mathbb{Q}^{>0}$, decide whether $\mathsf{Reach}([\![\mathsf{Env} \parallel \mathsf{Cont}']\!]^0_\Delta) \cap \mathsf{Bad} = \varnothing$;
– **[Existence]** given a zone-set of Bad states, Env and Cont, decide whether there exists $\Delta \in \mathbb{Q}^{>0}$ such that $\mathsf{Reach}([\![\mathsf{Env} \parallel \mathsf{Cont}']\!]^0_\Delta) \cap \mathsf{Bad} = \varnothing$. This is also called the implementability problem.
– **[Maximization]** given a zone-set of Bad states, Env and Cont, compute the least upper bound of the set of $\Delta \in \mathbb{Q}^{>0}$ such that $\mathsf{Reach}([\![\mathsf{Env} \parallel \mathsf{Cont}']\!]^0_\Delta) \cap \mathsf{Bad} = \varnothing$. Intuitively, this gives us the information about the slowest hardware that can implement correctly the strategy.

To solve the fixed version, we use the usual reachability algorithm for timed automata defined in [AD94]. To solve the maximization version (in an approximative way), we observe that for any timed automaton A, any two positive rational numbers Δ_1, Δ_2, if $\Delta_1 \leq \Delta_2$ then $\mathsf{Reach}([\![A]\!]^0_{\Delta_1}) \subseteq \mathsf{Reach}([\![A]\!]^0_{\Delta_2})$. So, given a tolerance $\eta \in \mathbb{Q}^{>0}$, the maximal value of Δ can be approached by η as follows: assuming Bad is reachable in $[\![A]\!]^0_{\Delta=1}$, it suffices to solve the **[Fixed]** problems with values $\Delta_i = i\eta$ $(0 \leq i \leq \lceil \frac{1}{\eta} \rceil)$, and take as approximation of the maximal Δ the value Δ_i such that the answer of the **[Fixed]** problem is YES for Δ_i and NO for Δ_{i+1}, which can be found more efficiently with a binary search.

The decidability of the **[Existence]** problem under assumption 10 is established in the next sections. To achieve this, we draw a strong link with the robust semantics defined by Puri in [Pur98]. In that paper, Puri shows that the traditional reachability analysis defined in [AD94] is not correct when the clocks may drift, even by a very small amount. He then reformulates the reachability problem as follows: given a timed automaton A, instead of computing $\mathsf{Reach}([\![A]\!]^0_0)$, he proposes an algorithm that computes $\cap_{\varepsilon \in \mathbb{Q}^{>0}} \mathsf{Reach}([\![A]\!]^\varepsilon_0)$. When A is clear from the context, this set is denoted by R^*_ε. This is the set of states that can be reached when the clocks drift by an infinitesimally small amount. He shows that this set has nice robustness properties with respect to modeling errors. In particular, he establishes that if the clocks are drifting, then guards can be checked with some small imprecisions (see [Pur98] for details).

In this paper, in order to make the link with the implementability problem, we study a variant of this robust semantics where small imprecisions on guards checking are allowed: the set of reachable states in this semantics is the set $\cap_{\Delta\in\mathbb{Q}^{>0}}\mathsf{Reach}([\![A]\!]^0_\Delta)$. When A is clear from the context, this set is abbreviated by R^*_Δ. We first show that for any timed automaton A, any zone-set Bad, we have that: $\cap_{\Delta\in\mathbb{Q}^{>0}}\mathsf{Reach}([\![A]\!]^0_\Delta) \cap \mathsf{Bad} = \varnothing$ iff there exists $\Delta \in \mathbb{Q}^{>0}$ such that $\mathsf{Reach}([\![A]\!]^0_\Delta) \cap \mathsf{Bad} = \varnothing$. After, we establish that the algorithm proposed by Puri to compute the set $\cap_{\varepsilon\in\mathbb{Q}^{>0}}\mathsf{Reach}([\![A]\!]^\varepsilon_0)$ is also valid to compute the set of states $\cap_{\Delta\in\mathbb{Q}^{>0}}\mathsf{Reach}([\![A]\!]^0_\Delta)$. We also reprove the results of Puri in our broader context. This yields an algorithm for deciding the implementability problem, and proves that, under assumption 10, both types of imprecisions have the same effect:

$$\bigcap_{\varepsilon>0}\mathsf{Reach}([\![A]\!]^\varepsilon_0) = \bigcap_{\Delta>0}\mathsf{Reach}([\![A]\!]^0_\Delta) = \bigcap_{\substack{\Delta>0\\ \varepsilon>0}}\mathsf{Reach}([\![A]\!]^\varepsilon_\Delta)$$

The proofs of our results follow the general ideas of the proofs of Puri and are based on the structure of limit cycles of timed automata (a fundamental notion introduced by Puri) but we needed new techniques to treat the imprecisions on guards instead of the drifts of clocks as in the original paper. Also, the proofs in the paper of Puri are not always convincing, so we reproved a large number of his lemmas that are needed to establish our proof and had to correct one of them.

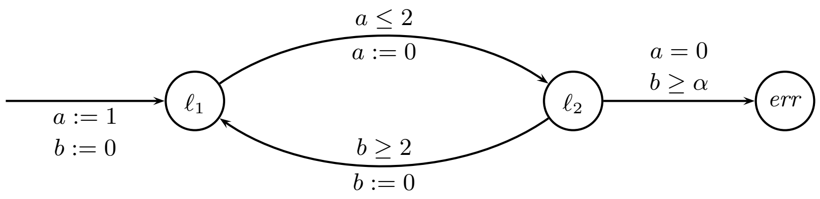
4 Example

Consider automaton A of Fig. 1(a). We examine two cases: $\alpha = 2$ and $\alpha = 3$. For both cases, the reachable states of the classical semantics $[\![A]\!]$ are the same and are depicted in Fig. 1(b) (the points $v_0 \ldots v_7$ will be used later in the paper). The safety property we want to verify is that the location *err* is not reachable. Note that in the classical semantics this is true in both cases.

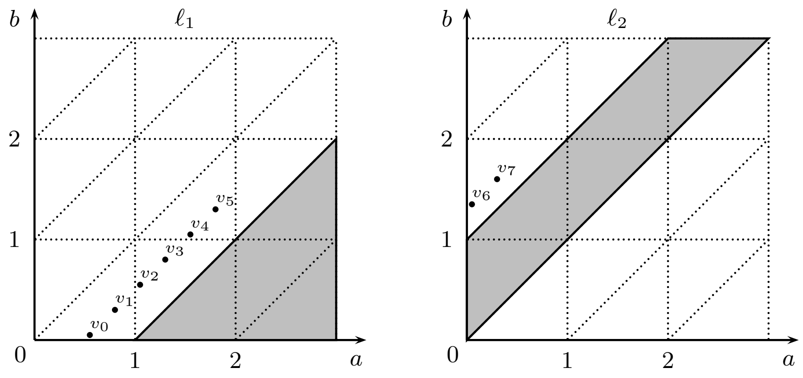
Consider now the enlarged semantics $[\![A]\!]^0_\Delta$ with $\Delta > 0$. In this semantics, guards are enlarged by the amount Δ. The edge from l_1 to l_2 has the guard $a \leq 2 + \Delta$ and the edge from l_2 to l_1 has the guard $b \geq 2 - \Delta$. Starting from the initial state $(l_1, a = 1, b = 0)$, the jump to l_2 can occur Δ time units later, so that the states $(l_2, a = 0, b \leq 1 + \Delta)$ are reached. Similarly, the transition from l_2 back to l_1 is enabled Δ time units earlier and the states $(l_1, a \geq 1 - 2\Delta, b = 0)$ can be reached. By iterating the cycle, the states $(l_1, a \geq 1 - 2k\Delta, b = 0)$ and $(l_2, a = 0, b \leq 1 + (2k-1)\Delta)$ are reachable. So, for any $\Delta > 0$ some new states are reachable in $[\![A]\!]^0_\Delta$ that were not reachable in the classical semantics. Those states are represented in Fig. 1(c).

From those new states that become reachable in location l_2, if $\alpha = 3$, the location *err* remains unreachable but if $\alpha = 2$, it becomes reachable.

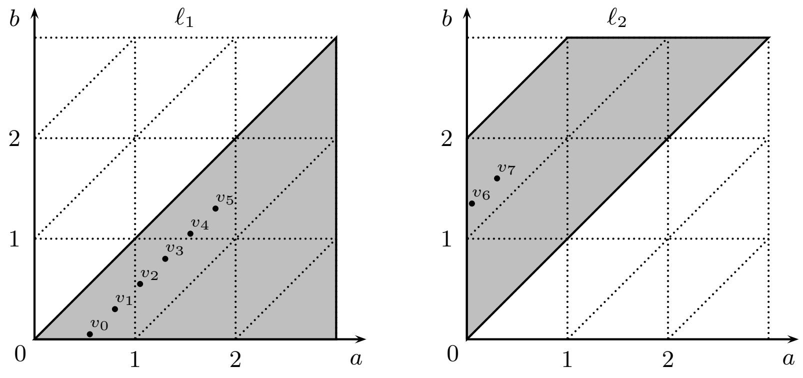
Clearly, from this example, one sees that a correct timed system (in the sense of the classical semantics) could have a completely different (and potentially bad) behavior due to an infinitesimally small inaccuracy in testing of the clocks (which is unavoidable since the clocks are discrete in embedded systems). This is the case for the automaton of figure 1(a). When $\alpha = 2$, the classical semantics is



(a) A timed automaton A.



(b) $\mathsf{Reach}(\llbracket A \rrbracket)$ for timed automaton A.



(c) The set $\bigcap_{\Delta>0} \mathsf{Reach}(\llbracket A \rrbracket^0_\Delta)$ for timed automaton A.

Fig. 1. Differences between standard and enlarged semantics

not robust since even the slightest error in guard checking allows new discrete transition to be taken. In other words, there is no strictly positive value for the parameter Δ that still ensures the safety property for $\llbracket A \rrbracket^0_\Delta$. Systems with such features cannot have a correct implementation because their correctness relies on the idealization of the mathematical model.

But this is not always the case: for the same automaton when $\alpha = 3$, no more discrete transitions are possible in the enlarged semantics than in the classical one. In this case, we can positively answer the question: "*Is there a strictly positive value for parameter Δ that allows the enlarged semantics to still satisfy the safety property?*" And indeed, we can prove that any value $\Delta < \frac{1}{3}$ is suited to that purpose.

5 Linking robustness and implementability

The classical semantics of timed automaton A is $[\![A]\!]_{\Delta=0}^{\varepsilon=0}$, which is a mathematical idealization of how we expect an implementation would behave: it makes the hypothesis that the hardware is perfectly precise and infinitely fast. Unfortunately, the execution of a timed automaton on a real hardware cannot be considered as ideal in the mathematical sense. It is thus an interesting question to know whether a small drift or imprecision of the clocks could invalidate some properties satisfied by the classical semantics. Drifts in clocks have been studied in [Pur98]. We are interested in studying imprecisions in the evaluation of the guards since it is directly connected to the question of implementability, as explained above. The main result of this paper is the following. Note that both Theorems 11 and 12 also hold with only one kind of imprecisions (guard enlargement or drift on clocks).

Theorem 11 *Under assumption 10, there exists an algorithm that decides, for any timed automaton A and any zone-set* Bad*, if there exist $\Delta, \varepsilon \in \mathbb{Q}^{>0}$ such that* $\mathsf{Reach}([\![A]\!]_{\Delta}^{\varepsilon}) \cap \mathsf{Bad} = \varnothing$.

Let $R^*_{\Delta,\varepsilon} = \bigcap_{\Delta>0} \bigcap_{\varepsilon>0} \mathsf{Reach}([\![A]\!]_{\Delta}^{\varepsilon})$. To prove Theorem 11, we show that the zone-set Bad intersects $R^*_{\Delta,\varepsilon}$ iff it intersects $\mathsf{Reach}([\![A]\!]_{\Delta}^{\varepsilon})$ for some $\Delta, \varepsilon > 0$. As shown in the next section, Algorithm 1 computes $R^*_{\Delta,\varepsilon}$. Consequently, the implementability problem is decidable.

Theorem 12 *Under assumption 10, for any timed automaton A, any zone-set* Bad*, the following equivalence holds:*

$$R^*_{\Delta,\varepsilon} \cap \mathsf{Bad} = \varnothing \qquad \textit{iff} \qquad \exists \Delta > 0,\ \varepsilon > 0.\ \mathsf{Reach}([\![A]\!]_{\Delta}^{\varepsilon}) \cap \mathsf{Bad} = \varnothing.$$

The proof of Theorem 12 relies on three intermediate lemmas, one of which corrects a wrong claim of [Pur98, Lemma 6.4]: it gives a bound on the distance between two zones with empty intersection. This bound is claimed to be $\frac{1}{2}$. We show that $\frac{1}{n}$ where n is the dimension of the space is the tightest bound. However the final results of [Pur98] are not deeply affected by this mistake. The underlying distance is defined by $d_\infty(x, y) = \|x - y\|_\infty = \max_{1\le i\le n}(|x_i - y_i|)$. Let us reformulate that lemma.

Lemma 13 *Let $Z_1 \subseteq \mathbb{R}^n$ and $Z_2 \subseteq \mathbb{R}^n$ be two zones such that $Z_1 \cap Z_2 = \varnothing$. Then, for any $x \in Z_1$ and $y \in Z_2$, $d_\infty(x, y) \ge \frac{1}{n}$. This bound is tight.*

Let us recall the definition of the 1-norm and the ∞-norm:

$$\|x\|_\infty = \max_{1\le i\le n}|x_i| \qquad\qquad \|x\|_1 = \sum_{i=1}^{n}|x_i|$$

Proof. First, we show that $\frac{1}{n}$ is a lower bound. Clearly, for any $v \in \mathbb{R}^n$, $\|v\|_1 \le n \cdot \|v\|_\infty$. We prove that $\|x-y\|_1 \ge 1$, which entails our result.

We consider two zones given under the form of Difference Bound Matrices [Dil90]: $Z_1 \equiv (m_{i,j})$ and $Z_2 \equiv \left(m'_{i,j}\right)$. Since $Z_1 \cap Z_2 = \varnothing$, there must exist a "negative cycle":

$$m^{(\prime)}_{i_1,i_2} + m^{(\prime)}_{i_2,i_3} + m^{(\prime)}_{i_3,i_4} + \cdots + m^{(\prime)}_{i_p,i_1} \le -1$$

where each term $m^{(\prime)}_{i,j}$ of the sum can be taken either in Z_1 or in Z_2. We may assume that at least one $m_{i,j}$ and one $m'_{i',j'}$ appears in this sum, since otherwise Z_1 (or Z_2) is empty and the result holds vacuously.

Since $m_{a,b} + m_{b,c} \ge m_{a,c}$ for any a, b, c, we can merge any two *consecutive* $m_{i,j}$'s into one while keeping the inequality. The same holds for $m'_{i',j'}$, and we can thus assume that $m_{i,j}$ and $m'_{i',j'}$ alternate in the sum above (starting with m_{i_1,i_2}, say).

Pick $x \in Z_1$ and $y \in Z_2$. Then

$$(x_{i_2} - x_{i_1}) + (y_{i_3} - y_{i_2}) + (x_{i_4} - x_{i_3}) + \cdots + (y_{i_1} - y_{i_p}) \le -1.$$

Terms can be rearranged in this sum, yielding

$$(y_{i_1} - x_{i_1}) - (y_{i_2} - x_{i_2}) + (y_{i_3} - x_{i_3}) - \cdots - (y_{i_p} - x_{i_p}) \le -1.$$

If $i_k = 0$ for some k, then $x_{i_k} - y_{i_k} = 0$. Thus, we assume $1 \le i_k \le n$. We take the absolute value, and apply the triangle inequality:

$$\begin{aligned}1 &\le \left|(y_{i_1} - x_{i_1}) - (y_{i_2} - x_{i_2}) + (y_{i_3} - x_{i_3}) - \cdots - (y_{i_p} - x_{i_p})\right|\\ &\le |(y_{i_1} - x_{i_1})| + |(y_{i_2} - x_{i_2})| + |(y_{i_3} - x_{i_3})| + \cdots + |(y_{i_p} - x_{i_p})|\\ &\le \|x-y\|_1.\end{aligned}$$

Now, let us show that this bound is tight. Consider the zones $Z_1, Z_2 \subseteq \mathbb{R}^n$ defined by the following equations:

– If n is odd

$$Z_1 \equiv \begin{cases} x_1 = 1 \\ x_{2i} - x_{2i+1} = 0 & 1 \le i \le \frac{n-1}{2}\end{cases}$$
$$Z_2 \equiv \begin{cases} x_{2i-1} - x_{2i} = 0 & 1 \le i \le \frac{n-1}{2} \\ x_n = 0\end{cases}$$

– If n is even

$$Z_1 \equiv \begin{cases} x_1 = 1 \\ x_{2i} - x_{2i+1} = 0 & 1 \le i \le \frac{n}{2} - 1 \\ x_n = 0\end{cases}$$
$$Z_2 \equiv \left\{ x_{2i-1} - x_{2i} = 0 \quad 1 \le i \le \frac{n}{2}\right.$$

We have $Z_1 \cap Z_2 = \varnothing$: combining equations of Z_1 and Z_2 yields $\forall i, j : x_i = x_j$, which leads to a contradiction since $x_1 = 1$ and $x_n = 0$. On the other hand, let $P(1, \frac{n-2}{n}, \frac{n-2}{n}, \frac{n-4}{n}, \frac{n-4}{n}, \dots)$ and $Q(\frac{n-1}{n}, \frac{n-1}{n}, \frac{n-3}{n}, \frac{n-3}{n}, \frac{n-5}{n}, \dots)$ (take the first n coordinates). It is easy to check that $P \in Z_1$ and $Q \in Z_2$, while $d_\infty(P, Q) = \mathsf{max}(\frac{1}{n}, \dots, \frac{1}{n}) = \frac{1}{n}$. ■

The previous lemma extends to sequences of sets in the following way:

Lemma 14 *Let A_δ be a collection of sets such that $A_{\delta_1} \subseteq A_{\delta_2}$ if $\delta_1 \leq \delta_2$. Assume that $Z = \bigcap_{\delta > 0} A_\delta$ is a zone-set. Also assume the existence of a zone-set Z' such that $\exists \delta_0 > 0.\ \forall 0 < \delta < \delta_0.\ A_\delta \cap Z' = \varnothing$. Then there exists $\delta_1 > 0$ such that for all $0 < \delta < \delta_1$, we have $d_\infty(A_\delta, Z') \geq \frac{1}{2n}$.*

Proof.
We pick $\delta_0 > 0$ such that $\forall 0 < \delta < \delta_0.\ A_\delta \cap Z' = \varnothing$, and $\delta'_0 > 0$ such that

$$\forall x \in A_{\delta'_0}.\ \exists z \in Z.\ d_\infty(x, z) < \frac{1}{2n}. \tag{1}$$

Such a δ'_0 exists by definition of Z. Assume the lemma is wrong:

$$\forall \delta_1 > 0.\ \exists 0 < \delta < \delta_1.\ \exists x \in A_\delta,\ y \in Z'.\ d_\infty(x, y) < \frac{1}{2n}.$$

Applying this result with $\delta_1 = \min(\delta_0, \delta'_0)$, we pick a $\delta'_1 > 0$, and two points $x \in A_{\delta'_1}$ and $y \in Z'$ such that $d_\infty(x, y) < \frac{1}{2n}$. From (1), and since $A_{\delta'_1} \subseteq A_{\delta'_0}$, there exists $z \in Z$ such that $d_\infty(x, z) < \frac{1}{2n}$. Thus $d_\infty(y, z) < \frac{1}{n}$, and with Lemma 13, $Z \cap Z' \neq \varnothing$. Then any A_δ intersects Z', since it contain Z. This contradicts our hypotheses.

■

In the sequel, when a distance d or a norm $\|\cdot\|$ is used, we always refer to d_∞ and $\|\cdot\|_\infty$. The following lemma relies on the theory of real numbers and the basics of topology.

Lemma 15 *Let $A_\Delta (\Delta \in \mathbb{R}^{>0})$ be a collection of sets such that $A_{\Delta_1} \subseteq A_{\Delta_2}$ if $\Delta_1 \leq \Delta_2$. Let $A = \bigcap_{\Delta > 0} A_\Delta$ be nonempty. If $d(A, B) > 0$, then there exists $\Delta > 0$ such that $A_\Delta \cap B = \varnothing$.*

Before proving this Lemma, we need the following two results:

Lemma 16 *If $d(A, B) > 0$, then $A \cap B = \varnothing$.*

Lemma 17 *If $A \subseteq B$, then $d(A, C) \geq d(B, C)$ for any C.*

We can now prove Lemma 15.

Proof of Lemma 15. Let $\delta_i = \frac{1}{i}$ $(i \geq 1)$. Define the sequence $\big(d_i\big)_{i \geq 1}$ as follows: $d_i = d(A_{\delta_i}, B)$. We use Lemma 17 to show that the sequence $\big(d_i\big)$ is increasing and bounded (we can exclude the case $d(A, B) = \infty$ since it makes the lemma trivially true). For any $i \geq 1$,

– since $A_{\delta_{i+1}} \subseteq A_{\delta_i}$, we have $d_i \leq d_{i+1}$;
– since $A \subseteq A_{\delta_i}$, we have $0 \leq d_i \leq d(A, B)$.

Then the sequence $\big(d_i\big)$ converges (to $d \overset{\text{def}}{=} \sup\{d_i\} \leq d(A, B)$). Assume $d \neq d(A, B)$ (then $d < d(A, B)$). For $\varepsilon < d(A, B) - d$, there exists N such that $\forall i \geq N : d_i < d + \varepsilon < d(A, B)$. Then, for any $i \geq N$, there exists $x \in A_{\delta_i}$ such that $d(\{x\}, B) < d(A, B)$, and consequently $x \notin A$. Since for any $0 < \delta \leq \frac{1}{N}$, there exists $i \geq N$ such that $A_{\frac{1}{i}} \subseteq A_\delta$, we have for any $\delta \leq \frac{1}{N}$ that there exists $x \in A_\delta$ such that $x \notin A$. Since A is also equal to $\bigcap_{0<\delta\leq\frac{1}{N}} A_i$, our assumption leads to a contradiction. So, it must be that $d = d(A, B)$, and thus $d > 0$. This implies the existence of N such that $\forall i \geq N : d_i > 0$. Hence, for such i we have $A_{\delta_i} \cap B = \varnothing$ (*cf.* Lemma 16). ■

We conlude this section by the proof of Theorem 12.

Proof of Theorem 12. Let $R_\varepsilon(\Delta) = \bigcap_{\varepsilon>0} \mathsf{Reach}([\![A]\!]^\varepsilon_\Delta)$, for any $\Delta > 0$. If $R^*_{\Delta,\varepsilon} \cap \mathsf{Bad} = \varnothing$, since $R^*_{\Delta,\varepsilon}$ and Bad are unions of sets of the form $\{l\} \times Z_l$ where Z_l is a zone[5], Lemma 13 applies and we have $d(R^*_{\Delta,\varepsilon}, \mathsf{Bad}) > 0$. From Lemma 15, we obtain that there exists $\Delta > 0$ such that $R_\varepsilon(\Delta) \cap \mathrm{Bad} = \varnothing$. Clearly, $R_\varepsilon(\Delta)$ satisfies the conditions of Lemma 14, hence the existence of some Δ_1 such that $\forall 0 < \Delta < \Delta_1$, we have $d(R_\varepsilon(\Delta), \mathsf{Bad}) > 0$. We pick such a $\Delta_0 \in (0, \Delta_1)$. Applying Lemma 15 to $R_\varepsilon(\Delta_0)$, we get the existence of ε_0 such that $\mathsf{Reach}([\![A]\!]^{\varepsilon_0}_{\Delta_0}) \cap \mathsf{Bad} = \varnothing$.

Conversely, if there exists $\Delta > 0$ and $\varepsilon > 0$ such that $\mathsf{Reach}([\![A]\!]^\varepsilon_\Delta) \cap \mathrm{Bad} = \varnothing$, then trivially $R^*_{\Delta,\varepsilon} \cap \mathsf{Bad} = \varnothing$. ■

It should be noted that a similar proof could be achieved in the presence of only one perturbation (for guard enlargement, such a proof can be found in [DDMR04]).

6 Algorithm for computing R^*_Δ

In this section, we prove that $R^*_\varepsilon = R^*_\Delta = R^*_{\Delta,\varepsilon}$, and that they can be computed by Algorithm 1 (originally proposed in [Pur98]).

Let us first examine how that algorithm performs on the example of section 4. In the region graph of the timed automaton of Fig. 1(a), there is a cycle that runs from valuation v_0 to itself through v_1 to v_7 (see Fig. 1(b)). Thus there is a cycle through the regions containing valuations v_0 to v_7. Furthermore, the corresponding closed regions have an intersection with the set of reachable states in the classical semantics (in gray). Since those closed regions form a strongly connected component of the region graph and their intersection with the reachable states in the classical semantics is not empty, the algorithm adds all those regions to the set J^*.

[5] Assuming Algorithm 1 is correct, the set $J^* = R^*_{\Delta,\varepsilon}$ it computes is a union of closed regions.

Algorithm 1: Algorithm for computing $R^*_\varepsilon(A)$ for a timed automaton A

```
Data: A timed automaton $A = \langle \mathsf{Loc}, \mathsf{Var}, q_0, \mathsf{Lab}, \mathsf{Edg} \rangle$
Result: The set $J^* = R^*_\varepsilon$
begin
    1. Construct the region graph $G = (R_A, \longrightarrow_A)$ of $A$ ;
    2. Compute $\mathsf{PC}(G) = \{\text{progress cycles of } G\}$;
    3. $J^* \leftarrow \mathsf{Reach}(G, [q_0])$ ;
    4. while for some $S = p_0\, p_1\, \ldots\, p_k \in \mathsf{PC}(G)$, $[p_0] \not\subseteq J^*$ and $J^* \cap [p_0] \neq \varnothing$ do
        $J^* \leftarrow J^* \cup [p_0]$ ;
        $J^* \leftarrow \mathsf{Reach}(G, J^*)$ ;
end
```

One can check that all regions of R^*_Δ for the automaton A of Figure 1(a) will be correctly added by Algorithm 1 (see figure 1(c)).

In the rest of this section, we prove that this algorithm is correct. This is achieved by showing that $J^* \subseteq R^*_\Delta$ and $J^* \subseteq R^*_\varepsilon$ on the one hand, and that $R^*_{\Delta,\varepsilon} \subseteq J^*$ on the other hand. Before tackling those problems, we first study the important notion of *limit cycles.*

6.1 Limit cycles

This subsection studies the behavior of limit cycles. A *limit cycle* of a timed automaton A is a trajectory $\pi = (q_0, t_0)(q_1, t_1) \ldots (q_k, t_k)$ of $[\![A]\!]$ containing at least one action transition and such that $q_k = q_0$. As suggested in [Pur98], given a progress cycle in the region graph and a region on this cycle, we focus on the subset of points of this region having a limit cycle. We first define this subset:

Definition 18 Consider a cyclic path $p = p_0\, p_1 \ldots p_N$ with $p_N = p_0$ in the region graph of a timed automaton A. We define the *return map* $R_p \colon 2^{[p_0]} \to 2^{[p_0]}$ by $R_p(S) = \cup_{q \in S} R_p(\{q\})$ for $S \subseteq [p_0]$, and, for singletons,

$$R_p(\{q_0\}) = \left\{ q_N \in [p_N] \;\middle|\; \begin{array}{l} \text{there exists a trajectory } \pi \text{ in } [\![A]\!] \text{ such that} \\ \pi = (q_0, t_0)(q_1, t_1) \ldots (q_N, t_N) \text{ and } \forall i.\ q_i \in [p_i] \end{array} \right\}$$

The set $L_{i,p}$ of points which can return back to themselves after i cycles through p, is defined as follows: $L_{i,p} = \{q \mid q \in R^i_p(\{q\})\}$. We write $L_p = \cup_{i \in \mathbb{N}} L_{i,p}$. □

In the sequel, we write R or L instead of R_p or L_p when the path p is clear from the context. The interesting property of L_p is that it is always (forward and backward) reachable from any valuation in p:

Theorem 19 ([Pur98, Lemma 7.10]) *Let $p = p_0\, p_1 \ldots p_N$ be a cycle in G. Then for any $z \in [p_0]$, there exists $z', z'' \in L$ and trajectories in $[\![A]\!]$ from z to z' and from z'' to z.*

The proof proposed by Puri is quite sketchy, and we propose a complete proof for which intermediate Lemmas are necessary.

Lemma 20 ([Pur98, Lemma 7.1]) *Let $p = p_0\, p_1 \dots p_N$ be a path in the region graph of a timed automaton A (we do not require that p be cyclic), let $\pi = (q_0, t_0)\,(q_1, t_1) \cdots (q_N, t_N)$ and $\pi' = (q'_0, t'_0)\,(q'_1, t'_1) \cdots (q'_N, t'_N)$ be two trajectories of $[\![A]\!]$ following p. Then for all $\lambda \in [0, 1]$, there exists a trajectory from $\lambda q_0 + (1-\lambda)q'_0$ to $\lambda q_N + (1-\lambda)q'_N$ in $[\![A]\!]$ following p.*

Proof. This is immediate by considering the path

$$\pi'' = (\lambda q_0 + (1-\lambda)q'_0, \lambda t_0 + (1-\lambda)t'_0)\,(\lambda q_1 + (1-\lambda)q'_1, \lambda t_1 + (1-\lambda)t'_1) \cdots \\ (\lambda q_N + (1-\lambda)q'_N, \lambda t_N + (1-\lambda)t'_N).$$

This is a trajectory in $[\![A]\!]$ since guards are convex. ■

Lemma 21 ([Pur98, Lemma 7.3]) *Let p be a cycle in the region graph of a timed automaton. Then L_p is convex.*

Proof. Let $x, y \in L_p$, and $\lambda \in [0, 1]$. There exists l and m such that $x \in L_{k,p}$ and $y \in L_{l,p}$. Then both x and y are in $L_{kl,p}$, and $\lambda x + (1-\lambda)y \in L_{kl,p} \subseteq L$ according to Lemma 20. ■

Definition 22 Let p be a region. The set of *vertices* of p is the smallest set of points $S(p) = \{v_0, v_1, \dots, v_k\}$ such that $[p] = \mathsf{Conv}(\{v_0, v_1, \dots, v_k\})$. □

This definition (where $\mathsf{Conv}(V)$ denotes the convex hull of the set of points V) is meaningful since regions are polytopes. Such a set is unique, and the number of vertices is bounded by $n+1$, where n is the number of clocks.

Lemma 23 *The vertices of a region p are the valuations in $[p]$ whose values are integers: $S(p) = [p] \cap \mathbb{N}^n$*

Proof. Let $v \in [p]$. There exist sets of clocks X_0, X_1, …, X_k such that all valuations w verifying the following equations is in $[p]$:

$$\begin{aligned} \forall i.\ \forall x, y \in X_i.\ \langle w(x)\rangle &= \langle w(y)\rangle, \\ \forall i < j.\ \forall x \in X_i,\ y \in X_j.\ \langle w(x)\rangle &\le \langle w(y)\rangle, \\ \forall x \in X_0.\ \langle w(x)\rangle &= 0, \end{aligned}$$

Let v_0 be the valuation such that $v_0(x) = \lfloor v(x) \rfloor$. That valuation belongs to $[p]$. Also, for all $0 < j \le k$, valuation v_j defined as

$$\begin{aligned} v_j(x) &= v_0(x) && \text{if } x \in X_i \text{ with } i < j \\ v_j(x) &= v_0(x) + 1 && \text{if } x \in X_i \text{ with } i \ge j \end{aligned}$$

are in $[p]$.

We now prove that those valuations generate the whole closed region: let w be a valuation in $[p]$. We define

$$w' = (1 - \langle w(x_k) \rangle)v_0 + (\langle w(x_1) \rangle - \langle w(x_0) \rangle)v_1 + \cdots + (\langle w(x_k) \rangle - \langle w(x_{k-1}) \rangle)v_k$$

where each x_i is a clock in the corresponding X_i, and claim that w' equals w. Indeed, let y be a clock, and j such that $y \in X_j$. Then

$$\begin{aligned} w'(y) &= (1 - \langle w(x_k) \rangle)v_0(y) + (\langle w(x_1) \rangle - \langle w(x_0) \rangle)v_1(y) + \cdots + \\ &\quad (\langle w(x_k) \rangle - \langle w(x_{k-1}) \rangle)v_k(y) \\ &= (1 - \langle w(x_k) \rangle)v_0(y) + (\langle w(x_1) \rangle - \langle w(x_0) \rangle)(v_0(y) + 1) + \cdots + \\ &\quad (\langle w(x_j) \rangle - \langle w(x_{j-1}) \rangle)(v_0(y) + 1) + \\ &\quad (\langle w(x_{j+1}) \rangle - \langle w(x_j) \rangle)(v_0(y)) + \cdots + \\ &\quad (\langle w(x_k) \rangle - \langle w(x_{k-1}) \rangle)(v_0(y)) \\ &= v_0(y) + w(x_j) = w(y) \end{aligned}$$

since $y \in X_j$. Thus $[p] = \mathsf{Conv}(\{v_i\})$. On the other hand, $\{v_i\}$ is the smallest set generating $[p]$, since no such valuation is a combination of the other ones. ■

This ensures that if a region r' is a sub-region of a closed region r, then its set of vertices $S(r')$ is the intersection of $[r']$ and of the set $S(r)$ of vertices of r.

Definition 24 Let $p = p_0\, p_1 \ldots p_N$ be a cycle in the region graph. The *orbit graph* is the graph $\Theta = (V_\Theta, \rightarrow_\Theta)$ such that $V_\Theta = S(p_0)$ and, for all $v, w \in V_\Theta$, $v \rightarrow_\Theta w$ iff $w \in R_p(\{v\})$. For $m \in \mathbb{N}$ and $v \in V_\Theta$, we write

$$\mathsf{Succ}^m(v) = \{w \in V_\Theta \mid v \rightarrow_\Theta^m w\} \quad \text{and} \quad \mathsf{Pred}^m(v) = \{w \in V_\Theta \mid w \rightarrow_\Theta^m v\}.$$

□

Clearly, given a vertex $v \in V_\Theta$, $R_p(\{v\})$ is a closed region thanks to Lemma 5, and $R_p(\{v\}) = \mathsf{Conv}(\{w \in V_\Theta \mid v \rightarrow_\Theta w\})$: that $R_p(\{v\})$ contains the convex hull follows from Lemma 21; conversely, any vertex in $R_p(\{v\})$ is in $\{w \in V_\Theta \mid v \rightarrow_\Theta w\}$, and, since $R_p(\{v\})$ is a closed region, it is included in the convex hull. More generally, we have $R_p^k(\{v\}) = \mathsf{Conv}(\{w \in V_\Theta \mid v \rightarrow_\Theta^k w\})$.

Lemma 25 ([Pur98, Lemma 7.4]) *Suppose there exists a run in the region graph from p to p'. Then for each vertex v of p, there exists a vertex v' of p' such that there is a trajectory from v to v', and conversely, for each vertex v' of p', there exists a vertex of p such that there is a trajectory from v to v'.*

Proof. Let v be a vertex of $[p]$. Then $\{v\}$ forms a subregion of $[p]$, and its successors in $[p']$ form a closed subregion of $[p']$. According to Lemma 23, that subregion of $[p']$ necessarily contains a vertex.

The same argument can be applied backwards, since the predecessor of a subregion of p' is a closed subregion of p. ■

Let $V = \{v \in V_\Theta \mid \exists m \in \mathbb{N}.\ v \in \mathsf{Succ}^m(v)\}$. Lemma 25 entails that any vertex has an outgoing edge in Θ. Thus for all $v \in V_\Theta$, there exists an integer m such that $\mathsf{Succ}^m(v) \cap V \neq \varnothing$, because V_Θ is finite. Let M be the largest such m. Then $\mathsf{Succ}^M(v) \cap V \neq \varnothing$ for all v. A similar argument proves the existence of M' such that $\mathsf{Pred}^{M'}(v) \cap V \neq \varnothing$ for all v.

Proof of Theorem 19. We write $z = \sum_i \lambda_i\, v_i$, where $\lambda_i \in [0,1]$, $\sum_i \lambda_i = 1$ and $v_i \in V_\Theta$. For each v_i, let w_i be an item of $\mathsf{Succ}^M(v_i) \cap V$. From Lemma 20, there is a path from z to $z' = \sum_i \lambda_i w_i \in \mathsf{Conv}(V) \subseteq L$.

Conversely, if x_i is a vertex in $\mathsf{Pred}^{M'}(v_i) \cap V$, there is a path from $z'' = \sum_i \lambda_i\, x_i \in \mathsf{Conv}(V) \subseteq L$ to z. ■

6.2 Soundness of Algorithm 1: $J^* \subseteq R^*_\Delta$ and $J^* \subseteq R^*_\varepsilon$

In this section, we prove that our algorithm can safely add progress cycles in J^*, *i.e.* that points in that cycles are indeed reachable whenever the automaton can deviate from its infinitely precise semantics. The main idea is to show that any two points in some L_p connected by a trajectory in $[\![A]\!]^0_\Delta$ (or $[\![A]\!]^\varepsilon_0$).

In the sequel, given $r \in \mathbb{R}^{\geq 0}$ and $x \in \mathbb{R}^n$, the *closed ball* of radius r centered in x is the set $B(x,r) = \{x' \mid d(x,x') \leq r\}$.

Case of imprecise guards. We first deal with the case of guard enlargement:

Theorem 26 *Let A be a timed automaton, $p = p_0\, p_1 \cdots p_N$ be a progress cycle of the region graph of A, and $\Delta \in \mathbb{Q}^{>0}$. For any two states u and v in L_p, there is a trajectory from u to v in $[\![A]\!]^0_\Delta$.*

This results immediately from the following Lemma:

Lemma 27 *Let A be a timed automaton with n clocks, $\Delta \in \mathbb{Q}^{>0}$, and $\delta = \frac{\Delta}{n}$. Let $p = p_0\, p_1\, p_2 \ldots p_N$ be a progress cycle in the region graph of A. Let u be a valuation in L_p, i.e. for which there exists a trajectory $\pi[0,T]$ in $[\![A]\!]^0_0$ following p with $\pi(0) = \pi(T) = u$. Let $v \in [p_0] \cap B(u,\delta)$ be a neighbor valuation. Then there exists a trajectory from u to v in $[\![A]\!]^0_\Delta$.*

Intuitively, the result is proved by slightly anticipating or delaying the transition dates when a clock is reset.

Proof. We first define some notations: We assume π is a cycle on valuation u, and write[6]

$$\pi = (l_0, u_0) \xrightarrow{t_0} (l_0, u'_0) \xrightarrow[R_0]{a_0} (l_1, u_1) \xrightarrow{t_1} (l_1, u'_1) \xrightarrow[R_1]{a_1} \cdots$$
$$\cdots \xrightarrow[R_{m-2}]{a_{m-2}} (l_{m-1}, u_{m-1}) \xrightarrow{t_{m-1}} (l_{m-1}, u'_{m-1}) \xrightarrow[R_{m-1}]{a_{m-1}} (l_m, u_m)$$

with $u_0 = u_m = u$. In this run, $\xrightarrow{t_i}$ is a continuous transition of duration t_i, and $\xrightarrow[R_i]{a_i}$ is an action transition a_i resetting clocks in R_i. We have in particular $T = \sum_{i=0}^{m-1} t_i$.

We define the integers z_x ($x \in \mathsf{Var}$) and the sets $X_0, \ldots, X_q$ of indices s.t., for any valuation w in region p_0:

- for all $x \in \mathsf{Var}$, $\lfloor w(x) \rfloor = z_x$;
- for all $x \in X_0$, $\langle w(x) \rangle = 0$;
- for all $x, y \in X_k$, $\langle w(x) \rangle = \langle w(y) \rangle$;
- for all $k < l$ and for all $x \in X_k,\ y \in X_l$, $\langle w(x) \rangle < \langle w(y) \rangle$;

Since p is a progress cycle, we know that each clock is reset at least once along π. For each clock x, we define $\alpha_x \in [0; m]$ to be the index of the transition in which x is reset for the last time along π. Formally, we have

$$x \in R_{\alpha_x} \qquad\qquad x \notin \bigcup_{\beta > \alpha_x} R_\beta \tag{2}$$

Then, for each clock x, we have

$$u_m(x) = u(x) = \sum_{\beta > \alpha_x} t_\beta \tag{3}$$

Now regarding valuation v, we define $\boldsymbol{\delta} = (\delta_x) = (v(x) - u(x))$. By hypothesis, $|\delta_x| \leq \delta$. Moreover, $v \in [p_0]$ implies that the order of fractional parts of the clocks must be the same in u and v, but with nonstrict inequalities. Writing $\langle\langle v(x) \rangle\rangle := v(x) - a_x$, we get:

- for all $x \in \mathsf{Var}$, $0 \leq \langle\langle v(x) \rangle\rangle \leq 1$;
- for all $x \in X_0$, $\langle\langle v(x) \rangle\rangle = 0$;
- for all $x, y \in X_k$, $\langle\langle v(x) \rangle\rangle = \langle\langle v(y) \rangle\rangle$;
- for all $k < l$ and for all $x \in X_k,\ y \in X_l$, $\langle\langle v(x) \rangle\rangle \leq \langle\langle v(y) \rangle\rangle$;

This entails that:

- for all $x, y \in X_k$, $\delta_x = \delta_y$;

[6] Other cases are possible, namely paths starting with an action transition and/or ending with a delay transition. Those cases would be treated in the same way.

– for any x, y s.t. $u(x) < u(y)$ (and hence $\alpha_x > \alpha_y$ from (3)), we have $u(x) + \delta_x \leq u(y) + \delta_y$, and

$$\delta_x - \delta_y \leq \sum_{\beta=\alpha_y+1}^{\alpha_x} t_\beta \tag{4}$$

We now build the run π' between u and v: For each $i \in [0; m]$, we define

$$t'_i = \begin{cases} t_i - \delta_x & \text{if } i = \alpha_x \\ t_i & \text{otherwise} \end{cases} \qquad t''_{i+1} = \begin{cases} t'_{i+1} + \delta_x & \text{if } i = \alpha_x \\ t'_{i+1} & \text{otherwise} \end{cases} \tag{5}$$

For the sake of simplicity, we temporarily assume that each t''_i is positive, that $\alpha_x \in [1; m-2]$ and $\alpha_x \neq \alpha_y + 1$ for any x, y, and define the run π' from π by replacing t_i with t''_i:

$$\pi' = (l_0, v_0) \xrightarrow{t''_0} (l_0, v'_0) \xrightarrow[R_0]{a_0} (l_1, v_1) \xrightarrow{t''_1} (l_1, v'_1) \xrightarrow[R_1]{a_1} \cdots$$
$$\cdots \xrightarrow[R_{m-2}]{a_{m-2}} (l_{m-1}, v_{m-1}) \xrightarrow{t''_{m-1}} (l_{m-1}, v'_{m-1}) \xrightarrow[R_{m-1}]{a_{m-1}} (l_m, v_m)$$

with $v_0 = u_0 = u$. We claim that π' corresponds to a trajectory in $[\![A]\!]^0_\Delta$. Indeed, the difference $|u(x) - v(x)|$ is always bounded by $n\delta$: The only difference between π and π' is that some clocks may sometimes be reset a little bit earlier or later, resulting in shifting their value with at most δ. This can only occur once for each clock, *i.e.* at most n times, thus the global difference between $u(x)$ and $v(x)$ is at most Δ. This ensures that each action transition can be fired along π' in $[\![A]\!]^0_\Delta$ since they were firable along π in $[\![A]\!]$.

Since clocks are reset at exactly the same transitions as along π, equation (3) applies here, and we get

$$v_m(x) \;=\; \sum_{\beta > \alpha_x} t''_\beta \;=\; t''_{\alpha_x+1} + \sum_{\beta > \alpha_x+1} t''_\beta \;=\; t_{\alpha_x+1} + \delta_x + \sum_{\beta > \alpha_x+1} t''_\beta$$

Since $t''_k + t''_{k+1} = t_k + t_{k+1}$ when k equals some α_y, and $t_k = t''_k$ when no α_y appears in $\{k, k+1\}$, we get

$$v_m(x) \;=\; \delta_x + \sum_{\beta > \alpha_x} t_\beta \;=\; u_m(x) + \delta_x \tag{6}$$

Thus $v_m = v$. It follows that π' witnesses the fact that $u \to v$ in $[\![A]\!]^0_\Delta$.

Let's consider "border cases" now:

– If $\alpha_x = \alpha_y + 1$ for some x and y, then t''_{α_x+1} equals $t_{\alpha_x+1} - \delta_y + \delta_x$. However, this does not change our result, since we still get equation (6);

- If $\alpha_x = 0$ for some x and π starts with an action transition, then we have $t''_1 = t_1 + \delta_x$ (or $t_1 + \delta_x - \delta_y$), which "enlarges"[7] the total duration of the path by δ_x. Since in that case the final value of clock x is the total duration of the path (because x is only reset at the very beginning of π), this also enlarges the final value of x by δ_x at the end of π', exactly as expected;
- Symmetrically, if $\alpha_x = m-1$ for some x and π ends with an action transition, then clock x equals zero in u. It could be the case that $x > 0$ in v; We then have to add a continuous transition at the end of π' (as if π would have ended with a continuous transition of delay zero).
- It could happen that $t''_i < 0$ for some i. However, it should be noticed that the small "shifts" that are applied to only one transition in the construction above, could be distributed over all time transitions occurring between transitions t_{α_x+1} and t_{α_y}, for "consecutive" α_x and α_y. Equation (4) ensures that we can distribute $\delta_x - \delta_y$ and get positive durations t''_i. ∎

Case of drifts on clocks. In this part, we prove a result similar to Theorem 26:

Theorem 28 *Let A be a timed automaton, $p = p_0\, p_1 \cdots p_N$ be a progress cycle of the region graph of A, and $\varepsilon \in \mathbb{Q}^{>0}$. For any two states u and v in L_p, there is a trajectory from u to v in $[\![A]\!]^{\varepsilon}_{0}$.*

The proof is more elaborate than for guard enlargement, and we will need several intermediate lemmas:

Lemma 29 *Let A be a timed automaton, r and r' be two regions s.t. $r \xrightarrow{time}_G r'$. Let $u \in r$, $v \in r'$ and $\tau \in \mathbb{R}^{>0}$ s.t. $u \xrightarrow{\tau} v$. Let $\delta \geq 0$. Then for any $y \in B(v,\delta) \cap [r']$, there exists some $x \in B(u, 2\delta) \cap [r]$ s.t. $x \xrightarrow{\tau'} y$ for some $\tau' \in \mathbb{R}^{>0}$.*

Proof. We assume that the closed regions $[r]$ and $[r']$ are characterized by the following inequalities:

$$u_i - u_j \leq m_{i,j} \qquad \alpha_i \leq u_i \leq \beta_i \qquad ([r])$$
$$v_i - v_j \leq m'_{i,j} \qquad \alpha'_i \leq v_i \leq \beta'_i \qquad ([r'])$$

W.l.o.g., we assume that $m_{i,j} \leq \beta_i - \alpha_j$ and $m'_{i,j} \leq \beta'_i - \alpha'_j$, since we have

$$u_i - u_j \leq \beta_i - \alpha_j \qquad v_i - v_j \leq \beta'_i - \alpha'_j \qquad (7)$$

Moreover, since $v = u + t \cdot \mathbf{1}$, we have

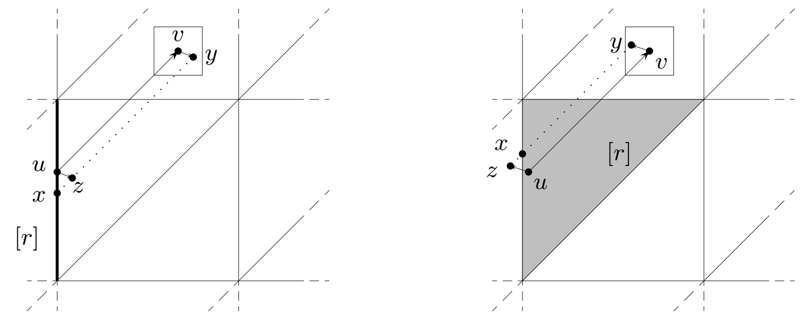
$$m_{i,j} = m'_{i,j} \qquad v_i - v_j = (u_i + t) - (u_j + t) \leq \beta_i - \alpha_j \qquad (8)$$

We define $D = y - v$. Note that $\|D\|_\infty \leq \delta$. Since $y \in [r']$, we have

$$(v_i + D_i) - (v_j + D_j) \leq m_{i,j} \qquad \alpha'_i \leq v_i + D_i \leq \beta'_i \qquad (9)$$

Now define $z = u + D$. As can be seen on Figure 2, z might not belong to $[r]$, and we have to find a neighbor x of z that belongs to $[r]$ and s.t. $x \to y$.

[7] Remember that δ_x could be negative.



(a) If the interior of $[r]$ is empty (b) If the interior of $[r]$ is not empty

Fig. 2. Building a predecessor of y in the closed region $[r]$

1. If, for some i_0, $\alpha_{i_0} = \beta_{i_0}$: We define $t' = -D_{i_0}$. The value of t' does not depend on the choice of i_0. Indeed, if for some $j \neq i_0$, we have $\alpha_j = \beta_j$, then

$$u_{i_0} - u_j = \alpha_{i_0} - \alpha_j = m_{i_0,j} = -m_{j,i_0}$$

since the values of u_{i_0} and u_j are fixed in the closed region $[r]$. This entails that

$$v_{i_0} - v_j = m_{i_0,j} = -m_{j,i_0} = (v_{i_0} - D_{i_0}) - (v_j - D_j)$$

since diagonal constraints are the same in $[r]$ and $[r']$. The previous equalities entail $D_{i_0} = D_j$, as claimed above. We now define $x = z + t' \cdot \mathbf{1}$. Obviously

$$x_i - x_j = (u_i + D_i + t') - (u_j + D_j + t') = y_i - y_j \leq m_{i,j}.$$

Moreover:

- For any j s.t. $\alpha_j = \beta_j$, we have

$$x_j = u_j + D_j + t' = u_j + D_j - D_j = u_j = \alpha_j = \beta_j$$

since $t' = -D_j$;
- For any j s.t. $\alpha_j \neq \beta_j$, we have

$$\begin{aligned} x_j - x_{i_0} &= (u_j + D_j + t') - (u_{i_0} + D_{i_0} + t') \\ &= (u_j + D_j) - (u_{i_0} + D_{i_0}) \leq m_{j,i_0}. \end{aligned}$$

Since $x_{i_0} = \alpha_{i_0}$ from the previous case (since $\alpha_{i_0} = \beta_{i_0}$), we get $x_j \leq m_{j,i_0} + \alpha_{i_0} \leq \beta_j$. In the same way, we have

$$\begin{aligned} x_{i_0} - x_j &= (u_{i_0} + D_{i_0} + t') - (u_j + D_j + t') \\ &= (u_{i_0} + D_{i_0}) - (u_j + D_j) \leq m_{i_0,j} \end{aligned}$$

and we get $\alpha_j \leq x_j$.

Since $|t'| \leq \delta$, we have $\|x - u\|_\infty \leq 2\delta$, which concludes the proof for this subcase.

2. If, for all i, $\alpha_i < \beta_i$: We define two sets $I = \{i_0, \dots, i_p\}$ and $I' = \{i'_0, \dots, i'_q\}$ s.t.

$$\forall i \in I.\ \alpha_i > z_i \qquad\qquad \forall i' \in I'.\ z_{i'} > \beta_{i'}$$

If both I and I' are empty, then z belongs to $[r]$ and the proof is over (diagonal constraints are satisfied thanks to the first equation of (9)).

Otherwise, assume I is not empty, and define $t' = \max\{\alpha_i - z_i \mid i \in I\}$. We write i_0 for the index s.t. $t' = \alpha_{i_0} - z_{i_0}$. Obviously, in this case, $t' > 0$. Moreover, $t' = \alpha_{i_0} - u_{i_0} - D_{i_0}$, and since $\alpha_{i_0} \leq u_{i_0}$, we have $t' \leq -D_{i_0}$.

We now define $x = z + t' \cdot \mathbf{1}$. Diagonal constraints are still obviously verified by x. Moreover, for $i \in I$, $\alpha_i \leq x_i + t'$ by construction of t'. For $j \notin I$, $\alpha_j \leq z_j < z_j + t' = x_j$ since $0 < t'$.

Now assume that, for some index j, we have $x_j > \beta_j$. Then $x_j - x_{i_0} > \beta_j - \alpha_{i_0}$, which is impossible since $x_j - x_{i_0} = y_j - y_{i_0} \leq m_{j,i_0} \leq \beta_j - \alpha_{i_0}$. This proves that $x \in [r]$. Since $0 < t' \leq -D_{i_0} \leq \delta$, we have $\|x - u\|_\infty \leq 2\delta$.

The case where I is empty (and I' is not) is similar: Define $t' = \min\{\beta_{i'} - z_{i'} \mid i' \in I'\}$, and write i'_0 for the index s.t. $t' = \beta_{i'_0} - z_{i'_0}$. This time, $t' < 0$ since $\beta_{i'_0} < z_{i'_0}$, and $t' = \beta_{i'_0} - u_{i'_0} - D_{i'_0}$. Since $\beta_{i'_0} - u_{i'_0} \geq 0$, we have $-D_{i'_0} \leq t' < 0$.

We define $x = z + t' \cdot \mathbf{1}$. By construction of t', for $i' \in I'$, we have $x_{i'} \leq \beta_{i'}$. For $j \notin I'$, we have $x_j = z_j + t' < z_j \leq \beta_j$ since $t' < 0$.

Assume that, for some j, $\alpha_j > x_j$. Then $x_{i'_0} - x_j < \beta_{i'_0} - \alpha_j$, which leads to a contradiction since $x_{i'_0} - x_j = y_{i'_0} - y_j \leq m_{i'_0,j} \leq \beta_{i'_0} - \alpha_j$. As in the previous case, this proves that $x \in [r]$, and $\|x - u\|_\infty \leq 2\delta$ since $-\delta \leq -D_{i'_0} \leq t' < 0$. ■

Lemma 30 *Let A be a timed automaton, let $p = p_0\, p_1\, p_2 \dots p_N$ be a progress cycle in the region graph of A. Let u be a valuation in L_p, i.e. for which there exists a trajectory $\pi[0,T]$ in $[\![A]\!]_0^0$ following p with $\pi(0) = \pi(T) = u$ and $T > 0$. Then there exists a trajectory $\pi'[0,T']$ in $[\![A]\!]_0^0$, following twice p (that is following $p' = p_0\, p_1 \dots p_N\, p_1\, p_2 \dots p_N$) and satisfying $\pi'(0) = \pi'(T') = u$ with $T' \geq 1/2$.*

Proof. The result is immediate if $T \geq 1/2$. Otherwise, since all clocks are reset along π, their value is strictly less than $1/2$ in u, and all along the trajectory. Consider the trajectory π^2 obtained by concatenating two copies of π. Again, no clock will ever reach $1/2$ along π^2. Also, the hypotheses ensure that there is at least one continuous transition in π with strictly positive delay. We consider the first such transition along π^2, and increase it by an extra $1/2$ t.u. delay, yielding a new trajectory π'. Then no clock ever reach value 1 along π', which ensures that guards are still satisfied along π'. Now, since the second half of the trajectory π' is exactly π, and since all clocks are reset along π, we get that $\pi'(0) = \pi'(2T + 1/2) = u$, and clearly π' follows p'. ■

Lemma 31 *Let A be a timed automaton, x and y be two states of A, and $\varepsilon \in \mathbb{Q}^{>0}$. If $x \xrightarrow{\tau} y$ in $[\![A]\!]$, then $\forall x' \in B(x, \varepsilon\tau) : x' \xrightarrow{\tau} y$ in $[\![A]\!]_0^\varepsilon$.*

Proof. The result is immediate if $\tau = 0$. Otherwise, it suffices to set the rate of each clock c to $1 - \frac{x'(c)-x(c)}{\tau}$, which lies between $1-\varepsilon$ and $1+\varepsilon$. ■

Lemma 32 *Let A be a timed automaton, $\varepsilon \in \mathbb{Q}^{>0}$, and $K_\varepsilon = 1/(2+3\varepsilon)$. Suppose r and r' are two regions s.t. $r \rightarrow_G r'$, and pick $u \in [r]$ and $v \in [r']$.*

– *If there is a continuous transition $u \xrightarrow{\tau} v$ in $[\![A]\!]$, then for any $\eta \in \mathbb{Q}^{>0}$, we have:*

$$\forall y \in B(v, K_\varepsilon(\eta + \varepsilon\tau)) \cap [r'].\ \exists x' \in B(u, \eta) \cap [r].\ x' \xrightarrow{\tau'} y \text{ in } [\![A]\!]_0^\varepsilon.$$

– *If there is an action transition $u \xrightarrow{\sigma} v$ in $[\![A]\!]$, then for all $\eta \in \mathbb{Q}^{>0}$, we have*

$$\forall y \in B(v, \eta) \cap [r'].\ \exists x \in B(u, \eta) \cap [r].\ x \xrightarrow{\sigma} y \text{ in } [\![A]\!]_0^\varepsilon.$$

Proof. We only proof the first part, the second one being obvious. Clearly, $v = u + \tau \cdot \mathbf{1}$, where $\mathbf{1}$ is the unit vector. Let $\delta = K_\varepsilon(\eta + \varepsilon\tau)$. From Lemma 29, for all $y \in B(v, \delta) \cap [r']$, there exists some $x \in B(u, 2\delta) \cap [r]$ s.t. $x \xrightarrow{\tau'} y$ in $[\![A]\!]$ for some $\tau' \in \mathbb{R}^{>0}$. So we have $y = x + \tau' \cdot \mathbf{1}$.

Using triangle inequality,

$$\tau' = \|y - x\| = \|(v-u) - [(x-u) + (v-y)]\| \geq \tau - 3\delta \tag{10}$$

Since $[r]$ is convex and $x, u \in [r]$, the intersection between the ball $B(x, \varepsilon\tau')$ and the segment $\mathsf{Conv}(\{u, x\})$ is included in $[r]$. Since $d(u,x) \leq 2\delta$, there exists $x' \in B(x, \varepsilon\tau') \cap [r]$ s.t.

$$\begin{cases} d(u, x') = 0 & \text{if } d(u,x) \leq \varepsilon\tau' \\ d(u, x') \leq 2\delta - \varepsilon\tau' & \text{if } d(u,x) \geq \varepsilon\tau' \end{cases}$$

Since $x \xrightarrow{\tau'} y$, Lemma 31 entails that $x' \xrightarrow{\tau'} y$ in $[\![A]\!]_0^\varepsilon$. To complete the proof, we must show that $x' \in B(u, \eta)$, *i.e.* $d(u, x') \leq \eta$. This is achieved by showing that $2\delta - \varepsilon\tau' \leq \eta$: we have $\frac{\delta}{K_\varepsilon} - 2\delta = 3\varepsilon\delta \geq \varepsilon(\tau - \tau')$ according to (10). Thus $\eta = \frac{\delta}{K_\varepsilon} - \varepsilon\tau \geq 2\delta - \varepsilon\tau'$. ■

Lemma 33 *Let A be a timed automaton, $\varepsilon \in \mathbb{Q}^{>0}$, and $K_\varepsilon = 1/(2+3\varepsilon)$. Let $p = p_0\, p_1 \cdots p_N$ be a path in the region graph of A. Let $\pi[0, T]$ be a trajectory of $[\![A]\!]$ following p, and $u = \pi(0)$ and $v = \pi(T)$. Then $\forall y \in B(v, K_\varepsilon^N \varepsilon T) \cap [p_N]$, there is a trajectory π' from u to y in $[\![A]\!]_0^\varepsilon$ following p.*

Proof. We write $\pi[0,T] = (q_0,t_0)(q_1,t_1)\dots(q_N,t_N)$, with $t_0 = 0$ and $t_N = T$. Define $\varepsilon_i = K_\varepsilon^i \varepsilon t_i$. We show that for all $0 \leq i < N$, for all $y \in B(q_{i+1},\varepsilon_{i+1}) \cap [r_{i+1}]$, there exists some $x \in B(q_i,\varepsilon_i) \cap [r_i]$ s.t. there is a trajectory from x to y in $[\![A]\!]_0^\varepsilon$.

– if $q_i \xrightarrow{\sigma} q_{i+1}$ is an action transition, then the result directly follows from Lemma 32, since $\varepsilon_{i+1} \leq \varepsilon_i$.
– otherwise, we have a continuous transition $q_i \xrightarrow{\tau} q_{i+1}$. Then $t_{i+1} = t_i + \tau$. Applying Lemma 32 with $\eta = \varepsilon_i$, we get:

$$\forall y \in B(q_{i+1}, K_\varepsilon(\varepsilon_i + \varepsilon\tau)) \cap [r_{i+1}].\ \exists x' \in B(q_i,\varepsilon_i) \cap [r_i].\ x' \xrightarrow{\tau'} y \text{ in } [\![A]\!]_0^\varepsilon.$$

Since $K_\varepsilon \leq 1$, we have

$$K_\varepsilon(\varepsilon_i + \varepsilon\tau) = K_\varepsilon^{i+1}\varepsilon t_i + K_\varepsilon \varepsilon\tau \geq K_\varepsilon^{i+1}\varepsilon(t_i+\tau) = K_\varepsilon^{i+1}\varepsilon t_{i+1} = \varepsilon_{i+1}$$

Thus, we also have

$$\forall y \in B(q_{i+1},\varepsilon_{i+1}) \cap [r_{i+1}].\ \exists x' \in B(q_i,\varepsilon_i) \cap [r_i].\ x' \xrightarrow{\tau'} y \text{ in } [\![A]\!]_0^\varepsilon.$$

Applying this result N times for $0 \leq i < N$, we get that for all $y \in B(q_N,\varepsilon_N) \cap [r_N]$, there exists an $x \in B(q_0,\varepsilon_0) \cap [r_0]$ s.t. there is a trajectory from x to y in $[\![A]\!]_0^\varepsilon$, which proves our result since $\varepsilon_0 = 0$ and $B(q_0,\varepsilon_0) = \{u\}$. ∎

Lemma 34 *Let A be a timed automaton, p be a progress cycle of the region graph of A, and u, v be two states in L_p. Then there exists an integer $n(u,v)$ s.t. $\mathsf{Conv}(\{u,v\}) \subseteq L_{n(u,v),p}$.*

Proof. Let k and l s.t. $u \in L_{k,p}$ and $v \in L_{l,p}$, and $n(u,v) = kl$. Applying Lemma 20, we get the result. ∎

Lemma 35 *Let A be a timed automaton, p be a progress cycle of the region graph of A, and u, v be two states in L_p. Let $\varepsilon \in \mathbb{Q}^{>0}$. There exists $\delta > 0$ s.t. for any $x \in \mathsf{Conv}(\{u,v\})$ and any $y \in L_p \cap B(x,\delta)$, there is a trajectory from x to y in $[\![A]\!]_0^\varepsilon$.*

Proof. If p is a non-time-elapsing progress cycle, then L_p is the singleton region where each clock is equal to zero, and the result is immediate.

Thus, we consider that p contains a time elapsing region. Let $\omega = 2n(u,v)$ where $n(u,v)$ is defined by Lemma 34, and $K_\varepsilon = 1/(2+3\varepsilon)$. Let $\delta = \frac{1}{2}\varepsilon K_\varepsilon^{\omega W}$ (where W denotes the number of regions of A). Pick some $x \in \mathsf{Conv}(\{u,v\})$ and $y \in L_p \cap B(x,\delta)$. Applying Lemmas 30 and 34, there is a limit cycle on x with total duration greater than 1/2 and having at most $2\omega W$ transitions. Lemma 33 ensures that any $y \in B(x, \delta) \cap [p_0]$ is reachable from x in $[\![A]\!]_0^\varepsilon$. Since $L_p \subseteq [p_0]$, the result follows. ■

We can now complete the proof of our Theorem:

Proof of Theorem 28. Let δ given by Lemma 35 and let $k = 1 + \lfloor \frac{1}{\delta} \rfloor$. Consider the points $x_0 = u$, $x_k = v$, and $x_i = u + i\delta(v - u)$ for i between 1 and $k - 1$. It is easy to check that $d(x_i, x_{i+1}) \leq \delta \cdot d(u, v) \leq \delta$. Thus from Lemma 35, there is a trajectory from x_i to x_{i+1} in $[\![A]\!]_0^\varepsilon$ for each i, and thus a trajectory from u to v. ■

Soundness of Algorithm 1.

Theorem 36 *Let A be a timed automaton. Let $p = p_0\, p_1 \ldots p_N$ be a progress cycle in the region graph of A, and let x and y be two valuations in $[p_0]$. Then:*

- *For any $\Delta \in \mathbb{Q}^{>0}$, there exists a trajectory from x to y in $[\![A]\!]_\Delta^0$;*
- *For any $\varepsilon \in \mathbb{Q}^{>0}$, there exists a trajectory from x to y in $[\![A]\!]_0^\varepsilon$.*

Proof. From Theorem 19, there exist $u, v \in L_p$ s.t. $x \to u$ and $v \to y$ in $[\![A]\!]$. Applying Theorem 26, v is reachable from u in $[\![A]\!]_\Delta^0$, and so y is reachable from x in $[\![A]\!]_\Delta^0$. The proof is the same for the second part of the result, using Theorem 28. ■

As a consequence:

Theorem 37 *The set J^* computed by Algorithm 1 is a subset of R_Δ^* and of R_ε^*.*

Proof. Let $\Delta > 0$. If a set of regions J^* is a subset of $\mathsf{Reach}([\![A]\!]_\Delta^0)$, then so is the set $\mathsf{Reach}(J^*)$ of closed regions reachable from J^* in the region graph G. Moreover, given a progress cycle p and a region p_0 in p, if $[p_0] \cap J^* \neq \varnothing$, then any point in $[p_0]$ is reachable in $[\![A]\!]_\Delta^0$, according to Theorem 36. Thus $J^* \cup p_0 \subseteq \mathsf{Reach}([\![A]\!]_\Delta^0)$. Since J^* is built by successively applying the above two operations, this ensures that $J^* \subseteq \mathsf{Reach}([\![A]\!]_\Delta^0)$. This holds for any $\Delta > 0$, thus $J^* \subseteq R_\Delta^*$.

The proof for drifts on clocks is similar. ■

6.3 Completeness of Algorithm 1: $R_{\Delta,\varepsilon}^* \subseteq J^*$

To prove the completeness of Algorithm 1, we need to better understand the relationship between trajectories of $[\![A]\!]_\Delta^\varepsilon$ and those of $[\![A]\!]$. In particular, Theorem 38 states that any trajectory π' of $[\![A]\!]_\Delta^\varepsilon$ can be approached by a trajectory π of $[\![A]\!]$, performing the same discrete transitions.

Theorem 38 *Let A be a timed automaton with n clocks and maximal constant M. For any distance $\delta \in (0,1)$, for any number of steps $k \in \mathbb{N}$, there exist two rationals $D, E \in \mathbb{Q}^{>0}$ such that for any $\Delta \in [0, D]$, for any $\varepsilon \in [0, E]$ and for any k-step stutter-free trajectory $\pi' = (q'_0, t'_0)(q'_1, t'_1) \cdots (q'_k, t'_k)$ in $[\![A]\!]^{\varepsilon}_{\Delta}$, there exists a k-step stutter-free trajectory $\pi = (q_0, t_0)(q_1, t_1) \cdots (q_k, t_k)$ in $[\![A]\!]$ such that for any position i in $[0, k]$:*

- *$q_0 \in [q'_0]$;*
- *Both π and π' have the same trace;*
- *Trajectories π and π' are "close" to each other,* i.e. *if $q_i = (l_i, v_i)$ and $q'_i = (l_i, v'_i)$, then $l_i = l'_i$ and $\|v_i - v'_i\| \leq \delta$;*

More precisely, the following values satisfy the conditions above:

$$D = \frac{\delta^2}{18(n+1)^{k+2}} \qquad\qquad E = \frac{D}{2(M+1)}$$

The proof of this theorem uses the following preliminaries and lemmas.

Preliminaries. The Difference Bound Matrices (DBM) are a classical data-structure used to represent zones. Classical operations (intersection, variable resets, time passing) are easily computed on DBM [Dil90,ACD+92,Yov96,CGP99, Bou01]. Let us briefly introduce them. If $x_1, \ldots, x_n$ are the clocks of a timed automaton and x_0 is a special clock whose value is always 0, zones can be represented by a set of constraints of the form $x_i - x_j \leq a$ where $a \in \mathbb{Z} \cup \{+\infty\}$[8].

A DBM is a $(n+1) \times (n+1)$ matrix $\mathbf{M} = \big(m_{i,j}\big)_{0 \leq i,j \leq n}$ where each $m_{i,j} = (a_{i,j}, \prec_{i,j})$ where $\prec_{i,j} \in \{<, \leq\}$ and $a_{i,j} \in \mathbb{Z}$ (that integer is called a *bound*). In the sequel, we only consider *closed* DBM, *i.e.* DBM where $\prec_{i,j}$ is always $\leq$, and we thus omit to mention it. Such a DBM represents the set

$$[\![\mathbf{M}]\!] = \{(x_1, \ldots, x_n) \in \mathbb{R}^n \mid \forall\, 0 \leq i, j \leq n : x_i - x_j \leq a_{ij} \wedge\ x_0 = 0\}.$$

When $i = j$, we can choose $m_{ij} = 0$. A DBM can be seen as a complete directed graph with nodes $0, 1, \ldots, n$ and edges (i, j) labeled with m_{ij}. A normal form for DBM is defined by the shortest path closure of the corresponding graph.

For our purpose, we introduce PDBM (*Parametric* DBM), a parametric version of the DBM [AAB00,HRSV01]. In a PDBM, each m_{ij} is a *parametric* bound, *i.e.*, a couple $m = (a, b)$ with $a \in \mathbb{Z}$ and $b \in \mathbb{N}$. Given a rational $\Lambda \in \mathbb{Q}^{\geq 0}$, the value of m is $[\![m]\!]_\Lambda = a + b\Lambda$, and the set represented by $\mathbf{M}$ is

$$[\![\mathbf{M}]\!]_\Lambda = \{(x_1, \ldots, x_n) \in \mathbb{R}^n \mid \forall\, 0 \leq i, j \leq n.\ x_i - x_j \leq [\![m_{ij}]\!]_\Lambda \wedge\ x_0 = 0\}.$$

As usual, we write $[\![\mathbf{M}]\!]$ for $[\![\mathbf{M}]\!]_0$. For a PDBM $\mathbf{M} = \big(m_{ij}\big)_{0 \leq i,j \leq n}$ with $m_{ij} = (a_{ij}, b_{ij})$, define the *width* of $\mathbf{M}$ by $w(\mathbf{M}) = \max\{b_{ij} \mid 0 \leq i, j \leq n\}$. Thus, a DBM is a zero-width PDBM.

[8] Since we consider a bounded state space, we can replace $+\infty$ by the greatest constant appearing in guards of the timed automaton.

Example. A closed rectangular guard g can be represented by a PDBM $\mathbf{M}_g$ such that for any $\varLambda \in \mathbb{Q}^{\geq 0}$: $[\![g]\!]_\varLambda = [\![\mathbf{M}_g]\!]_\varLambda$. In particular, $[\![g]\!] = [\![\mathbf{M}_g]\!]$. Also, $w(\mathbf{M}_g) = 2$. For example,

$$g = \{x = 4, 1 \leq y \leq 3\} \qquad \mathbf{M}_g = \begin{pmatrix} (0,0) & (-4,1) & (-1,1) \\ (4,1) & (0,0) & (3,2) \\ (3,1) & (-1,2) & (0,0) \end{pmatrix}$$

Lemma 39 ([Pur98]) *Let $\mathbf{M}$ be a $(n+1) \times (n+1)$ PDBM and $\varLambda \in \mathbb{Q}^{\geq 0}$ such that $\varLambda \cdot (2n+1) \cdot w(\mathbf{M}) < 1$. Let $Z' = [\![\mathbf{M}]\!]_\varLambda$ and $Z = [\![\mathbf{M}]\!]$. For any $x' \in Z'$, there exists $x \in Z$ such that $\|x' - x\| \leq n \cdot w(\mathbf{M}) \cdot \varLambda$.*

Proof. First, assume x' is a vertex of Z'. Then x' is obtained by solving a system of n equations of the form $x'_i - x'_j = m_{ij}$ or $x'_i = m_{i0}$. It is then clear that each component of x' is the sum or difference of at most n coefficients m_{ij}. Since those coefficients are entries of $\mathbf{M}$, we have for each $1 \leq i \leq n$, $x'_i = l_i + k_i\varLambda$ where $l_i, k_i \in \mathbb{Z}$ and $|k_i| \leq n \cdot w(\mathbf{M})$. Take $x_i = l_i$. Then $\|x' - x\| \leq n \cdot w(\mathbf{M}) \cdot \varLambda$ and we claim that $x \in Z$. Indeed, for any i, j, $x'_i - x'_j = l_i - l_j + (k_i - k_j)\varLambda = a_{ij} + b_{ij}\varLambda$. Hence $l_i - l_j = a_{ij} + (b_{ij} - k_i + k_j)\varLambda$. Since l_i, l_j and a_{ij} are integers, and $|(b_{ij} - k_i + k_j) \cdot \varLambda| \leq (2n+1) \cdot w(\mathbf{M}) \cdot \varLambda < 1$, it must be that $x_i - x_j = l_i - l_j \leq a_{ij}$. Therefore $x \in Z$.

If x' is not a vertex, then it can be written as $x' = \sum_i \lambda_i v'_i$ with $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$ and each v'_i is a vertex of Z'. From the proof above, there exist $v_i \in Z$ such that $\|v'_i - v_i\| \leq n \cdot w(\mathbf{M}) \cdot \varLambda$. Take $x = \sum_i \lambda_i v_i$. Clearly $x \in Z$, and we have successively $\|x' - x\| = \|\sum_i \lambda_i(v'_i - v_i)\| \leq \sum_i \lambda_i \|v'_i - v_i\| \leq \sum_i \lambda_i(n \cdot w(\mathbf{M}) \cdot \varLambda) \leq n \cdot w(\mathbf{M}) \cdot \varLambda$. ■

It is easy to extend classical operations (intersection, variable resets, time passing and emptiness test) to PDBM. It just needs to define an *order on parametric bounds*: we write $(a, b) \leq (a', b')$ iff either $a < a'$ or $a = a'$ and $b \leq b'$. This definition is consistent with any $\varLambda \in \mathbb{Q}^{\geq 0}$ such that $b\varLambda \leq 1$, that is, $(a, b) \leq (a', b')$ implies $[\![(a, b)]\!]_\varLambda \leq [\![(a', b')]\!]_\varLambda$ whenever $b\varLambda \leq 1$. If we assume $|b| \leq w$, it suffices that $w \cdot \varLambda \leq 1$ Thus provided this condition on $\varLambda$ is satisfied, we can compare parametric bounds *independently* of the valuation for $\varLambda$; in particular, the result of the comparison holds when $\varLambda = 0$.

Notice that the width is not increased by variable reseting nor by time passing. Also the normal form is preserved by those two operations. Intersection of two PDBM $\mathbf{M}_1$ and $\mathbf{M}_2$ gives a PDBM $\mathbf{M}$ whose entries are the minimum of the corresponding entries of $\mathbf{M}_1$ and $\mathbf{M}_2$. Hence $w(\mathbf{M}) \leq \max\{w(\mathbf{M}_1), w(\mathbf{M}_2)\}$. However, $\mathbf{M}$ is not necessarily in normal form. For a PDBM $\mathbf{M}$ of width $w(\mathbf{M})$, the width of the normal form of $\mathbf{M}$ is less than $n \cdot w(\mathbf{M})$. To see this, recall that each entry m_{ij} is replaced by the value of the shortest path from node i to node j, which has at most n edges. Finally, note that the consistency of order on parametric bounds is ensured if $\varLambda \leq 1/\max\{w(\mathbf{M}_1), w(\mathbf{M}_2)\}$ for intersection, and $\varLambda \leq 1/(n \cdot w(\mathbf{M}))$ for normalization. The result of an emptiness test on a PDBM $\mathbf{M}$ that has been first put in normal form is the same for $\varLambda_1 = 0$ and

	Intersection of $\mathbf{M_1}$ and $\mathbf{M_2}$	Normalization of $\mathbf{M}$	Reset	Time
Condition	$\varLambda \leq 1/\max\{w(\mathbf{M}_1), w(\mathbf{M}_2)\}$	$\varLambda \leq 1/(n \cdot w(\mathbf{M}))$	true[9]	true[9]
Result width	$\leq \max\{w(\mathbf{M_1}), w(\mathbf{M_2})\}$	$\leq n \cdot w(\mathbf{M})$	$\leq w(\mathbf{M})$	$\leq w(\mathbf{M})$

Table 1. Operations on PDBM.

$\varLambda_2 < 1/(n \cdot w(\mathbf{M}))$, that is, $[\![\mathbf{M}]\!]_{\varLambda_1} = \varnothing$ iff $[\![\mathbf{M}]\!]_{\varLambda_2} = \varnothing$. Indeed if $[\![\mathbf{M}]\!]_{\varLambda_2}$ is empty, then since $[\![\mathbf{M}]\!]_{\varLambda_1} \subseteq [\![\mathbf{M}]\!]_{\varLambda_2}$, it is also the case for $[\![\mathbf{M}]\!]_{\varLambda_1}$. On the other hand, if $\mathbf{M}$ is empty for $\varLambda_1$, then after normalization a negative integer bound must appear on the main diagonal of $\mathbf{M}$, *i.e.* for some $0 \leq i \leq n$, $m_{ii} = (a, b)$ with $a \leq -1$. Since $b \leq n \cdot w(\mathbf{M})$, we have $[\![m_{ii}]\!]_{\varLambda_2} < 0$ and $[\![\mathbf{M}]\!]_{\varLambda_2}$ is empty.

For a TTS $\mathcal{T} = \langle S, \iota, \Sigma, \rightarrow\rangle$, if $E \subseteq S$ is a set of states, and $\sigma \in \Sigma$ is a label, define $\mathsf{Post}^{\sigma}_{\mathcal{T}}(E) = \{y \in S \mid \exists x \in E.\ x \xrightarrow{\sigma} y\}$. For the special label τ (which denotes a continuous transition), we define $\mathsf{Post}^{\tau}_{\mathcal{T}}(E) = \{y \in S \mid \exists x \in E, t \in \mathbb{R}^{\geq 0}.\ x \xrightarrow{t} y\}$. For a sequence $\bar{\sigma} \in (\Sigma \cup \{\tau\})^+$ and a label $\nu \in \Sigma \cup \{\tau\}$, define recursively $\mathsf{Post}^{\nu\bar{\sigma}}_{\mathcal{T}}(E) = \mathsf{Post}^{\bar{\sigma}}_{\mathcal{T}}(\mathsf{Post}^{\nu}_{\mathcal{T}}(E))$ (for the empty word ϵ, we let $\mathsf{Post}^{\epsilon}_{\mathcal{T}}(E) = E$).

In the lemmas below, we show how to use PDBM to deal with the computation of Post. We omit to mention locations in those computations, *i.e.* we write $Z = [\![\mathbf{M}]\!]$ instead of $Z = \{l\} \times [\![\mathbf{M}]\!]$ for $l \in \mathsf{Loc}$.

Lemma 40 *Given a timed automaton A with n clocks, let M be the greatest constant appearing in guards of A. Let $\mathbf{M}$ be a PDBM of width $w(\mathbf{M})$. Let $\varLambda < 1$, $\varDelta \in \mathbb{R}^{\geq 0}$, and $\varepsilon \leq \varLambda/(2(M+1))$. Let $Z = [\![\mathbf{M}]\!]$ and $Z' = [\![\mathbf{M}]\!]_{\varLambda}$. The set $\mathsf{Post}^{\tau}_{[\![A]\!]^{\varepsilon}_{\varDelta}}(Z')$ can be over-approximated by a PDBM $\mathbf{M}'$ of width $w(\mathbf{M}') = w(\mathbf{M}) + 1$, that is $\mathsf{Post}^{\tau}_{[\![A]\!]^{\varepsilon}_{\varDelta}}(Z') \subseteq [\![\mathbf{M}']\!]_{\varLambda}$, and when $\varLambda = 0$, $\mathbf{M}'$ represents the set $\mathsf{Post}^{\tau}_{[\![A]\!]}(Z)$, that is $[\![\mathbf{M}']\!] = \mathsf{Post}^{\tau}_{[\![A]\!]}(Z)$.*

Proof. In the classical semantics $[\![A]\!]$, the time passing can be represented exactly using DBM: since the state space is a subset of $[0, M]^n$, it suffices to set each entry m_{i0} to M for $1 \leq i \leq n$. In the enlarged semantics, for a continuous transition of length t, the value of a clock may differ by $t\varepsilon$. We can bound t by $M + 1$: it can get larger than M because clocks can progress slower in the enlarged semantics (namely at a rate of $1 - \varepsilon$). Hence, the duration of a time transition is bounded by $M/(1 - \varepsilon)$, thus by $M + 1$ since $\varepsilon \leq 1/(M + 1)$. We construct $\mathbf{M}'$ by computing the time successors of $\mathbf{M}$ in the exact semantics, and then by adding $\varLambda$ to each entry of the PDBM, except the main diagonal. This gives an over-approximation of $\mathsf{Post}^{\sigma}_{[\![A]\!]^{\varepsilon}_{\varDelta}}(Z')$ since if $x_i - x_j \leq m_{ij}$ by the exact computation, then $x_i - x_j \leq m_{ij} + 2(M+1)\varepsilon \leq m_{ij} + \varLambda$ when considering drifts on clocks. Therefore $\mathsf{Post}^{\sigma}_{[\![A]\!]^{\varepsilon}_{\varDelta}}(Z') \subseteq [\![\mathbf{M}']\!]_{\varLambda}$. It appears clearly that in the construction of $\mathbf{M}'$, if we set $\varLambda = 0$, we get the time successors of Z in the exact semantics. ∎

[9] $\mathbf{M}$ must be in normal form.

Lemma 41 *Given a timed automaton A with n clocks, and a PDBM $\mathbf{M}$ of width $w(\mathbf{M})$, let $\Lambda \leq 1/(n^2 \cdot (2 + w(\mathbf{M})))$, $\varepsilon \in \mathbb{R}^{\geq 0}$, and $\Delta \leq \Lambda$. Let $Z = [\![\mathbf{M}]\!]$ and $Z' = [\![\mathbf{M}]\!]_\Delta$. Let $\sigma \in \mathsf{Lab}_A$. We assume that $\mathsf{Post}^\sigma_{[\![A]\!]^\varepsilon_\Delta}(Z')$ is not empty. It can then be over-approximated by a PDBM $\mathbf{M}'$. More precisely, $\mathbf{M}'$ satisfies $[\![\mathbf{M}']\!]_\Lambda \supseteq [\![\mathbf{M}']\!]_\Delta = \mathsf{Post}^\sigma_{[\![A]\!]^\varepsilon_\Delta}(Z')$. Moreover, when $\Lambda = 0$, $\mathbf{M}'$ represents exactly the set $\mathsf{Post}^\sigma_{[\![A]\!]}(Z)$, that is, $[\![\mathbf{M}']\!] = \mathsf{Post}^\sigma_{[\![A]\!]}(Z)$. In particular, $\mathsf{Post}^\sigma_{[\![A]\!]}(Z)$ is not empty. Last, $w(\mathbf{M}') \leq n \cdot \max\{2, w(\mathbf{M})\}$.*

Proof. Computing Post^σ is achieved by first intersecting Z' with a PDBM representing the guard of the transition. That PDBM has width 2. The computation is correct since $\Delta \leq 1/\max(2, w(\mathbf{M}))$, and the width of the resulting PDBM is at most $\max(2, w(\mathbf{M}))$ (see Table 1). Since $\Delta \leq 1/(n \cdot \max(2, w(\mathbf{M})))$, that PDBM can be normalized, yielding a PDBM of width $n \cdot \max(2, w(\mathbf{M}))$. Last, clock reset can be achieved on that PDBM. The width is not enlarged. In the end, we have

$$\mathsf{Post}^\sigma_{[\![A]\!]^\varepsilon_\Delta}(Z') = [\![\mathbf{M}']\!]_\Delta \subseteq [\![\mathbf{M}']\!]_\Lambda.$$

The above computations are exact when applied to DBM, *i.e.* when $\Lambda = 0$. Thus $[\![\mathbf{M}']\!] = \mathsf{Post}^\sigma_{[\![A]\!]}(Z)$, which is non-empty since $\Delta \leq 1/(n \cdot w(\mathbf{M}'))$. ■

Lemma 42 *Given a timed automaton A with $n \geq 1$ clocks, let M be the greatest constant appearing in guards of A. Let $\mathbf{M}_0$ be a PDBM of width $w(\mathbf{M}_0) = 0$, and $k \in \mathbb{N}$. Let $\Lambda < 1/(2(1+n)^{k+2})$, $\Delta \leq \Lambda$ and $\varepsilon < \Lambda/(2(M+1))$. Let $Z'_0 = [\![\mathbf{M}_0]\!]_\Delta$. Consider a sequence $Z'_1, Z'_2, \ldots, Z'_k$ of k zones and $\sigma_1, \sigma_2, \ldots, \sigma_k$ of k labels in $\Sigma \cup \{\tau\}$, s.t. for each $1 \leq i \leq k$, $Z'_i = \mathsf{Post}^{\sigma_i}_{[\![A]\!]^\varepsilon_\Delta}(Z'_{i-1})$ is not empty. Then there exist PDBM $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_k$ over-approximating $Z'_1, Z'_2, \ldots, Z'_k$ respectively (that is $Z'_j \subseteq [\![\mathbf{M}_j]\!]_\Lambda$). When $\Lambda = 0$, the DBM $[\![\mathbf{M}_i]\!]_0$ represent the sets Z_i defined by $Z_0 = Z'_0$ and, for $1 \leq i \leq k$, $Z_i = \mathsf{Post}^{\sigma_i}_{[\![A]\!]}(Z_{i-1})$. In particular, the sets Z_i are not empty. Last, we have $w(\mathbf{M}_j) \leq 2(1+n)^j$.*

Proof. For each $j \leq k$, the PDBM $\mathbf{M}_j$ is obtained by applying Lemma 40 or 41, depending on the nature of the j-th transition. It is easy to show by induction that, for all $j \leq k$, $w(\mathbf{M}_j) \leq 2(1+n)^j$:

- This is true for $j = 1$ since $w(\mathbf{M}_0) = 0$.
- Assume this is true for some $j - 1 \geq 1$. We then have

$$\begin{aligned} w(\mathbf{M}_j) &\leq \max(w(\mathbf{M}_{j-1}) + 1, n \cdot \max(2, w(\mathbf{M}_{j-1}))) \\ &\leq \max(1 + 2(1+n)^{j-1}, n \cdot (2 + 2(1+n)^{j-1})) \\ &\leq n \cdot (2 + 2(1+n)^{j-1}) \\ &\leq 2(1+n)^j \end{aligned}$$

The hypothesis concerning Λ in Lemma 40 is clearly fulfilled. For Lemma 41, the hypothesis is $\Lambda \leq 1/(n^2 \cdot (2 + w(\mathbf{M}_{j-1})))$ for computing $\mathbf{M}_j$. It is fulfilled since

$$\Lambda \leq 1/(2(1+n)^{k+2}) \leq 1/(2(1+n)^{j+2}) \leq \\ 1/(2(n+1)^3 \cdot (1+n)^{j-1}) \leq 1/(n^2(2 + w(\mathbf{M}_{j-1}))).$$

■

Automaton refinement. For a timed automaton A, and an integer $\gamma \in \mathbb{N}$, define A_γ the γ-refinement of A by first replacing in A each constant c appearing in guards by $c\gamma$, and then replacing each edge $e = (l, l', g, \sigma, R) \in \mathsf{Edg}_A$ with g of the form $\bigcap_{x \in \mathsf{Var}} \{a_x \leq x \leq b_x\}$ by the set of edges $e'_{g'} = (l, l', g', \sigma, R)$ where g' ranges over the set of constraints of the form $\bigcap_{x \in \mathsf{Var}} \{a^i_x \leq x \leq b^i_x \mid i \in I_x\}$ with

$$\begin{cases} I_x = \{0\}, & a^0_x = a_x, b^0_x = b_x \quad \text{if } a_x = b_x \\ I_x = \{0, \ldots, b_x - a_x - 1\}, & a^i_x = a_x + i, b^i_x = a^i_x + 1 \quad \text{if } a_x < b_x \end{cases}$$

Roughly, this amounts to splitting each guard into "atomic" ones, in which all constraints are of the form $a \leq x \leq a+1$. Then, for any $\Delta, \varepsilon \in \mathbb{Q}^{\geq 0}$, the TTS $[\![A]\!]^\varepsilon_\Delta$ and $[\![A_\gamma]\!]^\varepsilon_{\gamma\Delta}$ are bisimilar, using the bijective function $\mu_\gamma \colon S_A \to S_{A_\gamma}$ such that $\mu_\gamma(l, v) = (l, \gamma v)$. Also notice that, for any $x, x' \in S_{A_\gamma}$, if $\|x - x'\| = \theta$, then $\|\mu^{-1}_\gamma(x) - \mu^{-1}_\gamma(x')\| = \theta/\gamma$.

Example. For $\gamma = 2$, an edge (l, l', g, σ, R) in A with $g = \{x = 4, 1 \leq y \leq 3\}$ is replaced in A_γ by the four edges

$$\begin{array}{ll} (l, l', \{x = 8, 2 \leq y \leq 3\}, \sigma, R) & (l, l', \{x = 8, 4 \leq y \leq 5\}, \sigma, R) \\ (l, l', \{x = 8, 3 \leq y \leq 4\}, \sigma, R) & (l, l', \{x = 8, 5 \leq y \leq 6\}, \sigma, R) \end{array}$$

In this example, we have $I_x = \{0\}$ and $I_y = \{0, 1, 2, 3\}$.

Proof of Theorem 38. Let $\gamma = \lceil 2/\delta \rceil$, $\Lambda = \delta^2/(18(1+n)^{k+2})$, $\Delta \leq \Lambda$ and $\varepsilon \leq \Lambda/(2M+2)$. Those values of Λ, Δ and ε clearly satisfy the hypotheses of Lemmas 39 and 42.

Let $\bar{\sigma} = \sigma_1 \sigma_2 \ldots \sigma_k$ be the trace corresponding to π'. Let $\rho' = \mu_\gamma(\pi')$. Then ρ' is a trajectory of $[\![A_\gamma]\!]^\varepsilon_{\gamma\Delta}$. Let $Z_0 = [q'_0]$ and $\mathbf{M}_0$ be a normalized DBM such that $Z_0 = [\![\mathbf{M}_0]\!]$. Note that $w(\mathbf{M}_0) = 0$. Using Lemma 42, there exist PDBM $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_k$ such that for $1 \leq j \leq k$:

– $[\![\mathbf{M}_j]\!]_{\gamma\Lambda} \supseteq \mathsf{Post}^{\sigma_1 \ldots \sigma_j}_{[\![A_\gamma]\!]^\varepsilon_{\gamma\Delta}}(Z_0)$;
– $[\![\mathbf{M}_j]\!] = \mathsf{Post}^{\sigma_1 \ldots \sigma_j}_{[\![A_\gamma]\!]}(Z_0)$.

Clearly, $\mu_\gamma(q'_k) \in [\![\mathbf{M}_k]\!]_{\gamma\Delta}$. Then, from Lemma 39, there exists $q_k \in [\![\mathbf{M}_k]\!]$ such that $\|\mu_\gamma(q'_k) - q_k\| \le n \cdot w(\mathbf{M}_k) \cdot \gamma \cdot \Delta \le 2(1+n)^{k+1} \cdot \gamma \cdot \Delta \le \delta$ (the latter inequality holds because $\gamma \le 2/\delta{+}1$). Since $q_k \in [\![\mathbf{M}_k]\!] = \mathsf{Post}^{\sigma_1 \cdots \sigma_k}_{[\![A_\gamma]\!]}(Z_0)$, there is a trajectory $\rho = (q_0, t_0)(q_1, t_1) \cdots (q_k, t_k)$ in $[\![A_\gamma]\!]$ such that $q_0 \in Z_0$, $q_i \in [\![g'_i]\!]$ for the set of indices $\{i_1, i_2, \ldots, i_l\}$ corresponding to action transitions. For those indices, since g'_i are "atomic" guards, we have $\|\mu_\gamma(q'_i) - q_i\| \le 1 + \gamma\Delta$. The effect of action transitions (resetting clocks) does not increase that distance, so that this inequality remains true after each reset. Since we consider stutter-free trajectories, the inequality holds for all indices i. Now, define $\pi = \mu_\gamma^{-1}(\rho)$; it is a trajectory of $[\![A]\!]$ since ρ is a trajectory of $[\![A_\gamma]\!]$. Thus, we have $\|q'_i - \mu_\gamma^{-1}(q_i)\| \le (1+\gamma\Delta)/\gamma \le 2/\gamma \le \delta$ for each $0 \le i \le k$. ■

The following theorem is the central theorem for the completeness proof: It states that points in J^* cannot reach points that are more than distance α away from J^* in $[\![A]\!]^\varepsilon_\Delta$ for sufficiently small Δ and ε.

Theorem 43 *Let A be a timed automaton with n clocks satisfying Assumption 10, and whose region graph has W regions. Let $\alpha \in \mathbb{R}^{>0}$, and*

$$\alpha' = \min\left(\alpha, 1/(12(n+2)n^2)\right) \qquad \Lambda = \frac{\alpha'^2}{18(n+1)^{12(n+2)n^2W+3}}$$

Pick some Δ in $(0, \Lambda)$ and ε in $(0, \frac{\Lambda}{2(M+1)})$. Suppose $x \in J^*$, and that there exists a trajectory in $[\![A]\!]^\varepsilon_\Delta$ from x to some position y. Then $d(y, J^*) \le \alpha'$.

Proof. We prove the result by contradiction, by assuming that there exists a stutter-free trajectory in $[\![A]\!]^\varepsilon_\Delta$ starting from some position in J^*, and ending in a position y such that $d(y, J^*) > \alpha'$.

We pick one of the shortest (w.r.t. the number of transitions) such stutter-free trajectory $\pi' = (q'_0, t'_0)(q'_1, t'_1) \ldots (q'_m, t'_m)$ in $[\![A]\!]^0_\Delta$. We have $q'_0 \in J^*$, $d(q'_i, J^*) \le \alpha'$ for $i < m$, and $d(q'_m, J^*) > \alpha'$.

We let $L = 12(n+2)n^2W$. Assume $m \le L+1$. From Theorem 38, our trajectory π' can be approached by a trajectory $\pi = (q_0, t_0) \ldots (q_m, t_m)$, starting in the closed-region $[q'_0]$, having the same trace as π', and such that for any integer $i \le m$, $\|q_i - q'_i\| \le \alpha'$. Since $q_0 \in [q'_0] \subseteq J^*$, the whole trajectory π lies in J^*. Thus $q_m \in J^*$, and $d(q'_m, J^*) \le \alpha'$, which contradicts our assumption.

Thus $m \ge L+2$. We apply Theorem 38 to the subtrajectory between q'_{m-L-2} and q'_m, and consider a trajectory $\pi = (q_{m-L-2}, t_{m-L-2}) \ldots (q_m, t_m)$ witnessing the requirements of Theorem 38. We know that $d(q'_i, J^*) \le \alpha'$ for all integers $i < m$, and that $\|q'_i - q_i\| \le \alpha'$ for i between $m-L-2$ and m. Thus, $d(q_i, J^*) \le 2\alpha' < \frac{1}{n}$ for i between $m-L-2$ and $m-1$, and, from Lemma 13, $[q_i] \cap J^* \neq \varnothing$ for those i.

Since it contains $L+1$ states, that is, $12(n+2)n^2 \cdot W + 1$ states, the prefix of π between (q_{m-L-2}, t_{m-L-2}) and (q_{m-1}, t_{m-1}) contains $6(n+2)n^2+1$ states s_0, s_1, ..., $s_{6(n+2)n^2}$ s.t. $(s_0) = (s_1) = \cdots = (s_{6(n+2)n^2})$, and with at least one action transition between each pair (s_i, s_{i+1}) (w.l.o.g., we assume that, for any i, state s_i occurs before state s_{i+1} along π). This entails that π runs through $6(n+2)n^2$ cycles in the region graph.

- Assume one of these cycles is a progress cycle; then we know that some (s_i) has a progress cycle through it in the region graph, and it intersects with J^*. Such a region has been added in J^*, and thus (s_i) lies in J^*. Thus the whole suffix of π starting in s_i is in J^*; in particular $q_m \in J^*$, and $d(q'_m, J^*) \leq \alpha'$, which is in contradiction with the choice of trajectory π'.
- Now assume that, along one of the cycles (between s_i and s_{i+1}, say), no clock is reset. Since π and the corresponding portion of π' have the same trace, no clock is reset in the corresponding portion of π', and that portion can be replaced by a single continuous transition, which leads to a shorter trajectory than π' that sufficiently moves away for J^*, and contradicts our initial assumption.
- Thus, our $6(n+2)n^2$ cycles are non-progress cycles along each of which at least one clock is reset. We now explain how to build a shorter trajectory between the same initial and final states, thus contradicting the assumption that π' was the shortest.

  We first define sets X_0, …, X_p of indices such that, for any valuation w in region (s_0) (and thus in every region (s_i)),
  - for all $x \in X_0$, $\langle w(x) \rangle = 0$ (note that X_0 can be empty);
  - for all $x, y \in X_k$, $\langle w(x) \rangle = \langle w(y) \rangle$;
  - for all $k < l$ and for all $x \in X_k$ and $y \in X_l$, $\langle w(x) \rangle < \langle w(y) \rangle$.

  We also define S_i to be the set of clocks that are reset along the i-th cycle, between s_{i-1} and s_i. We write c_i and c'_i for the i-th cycle along π and π', respectively, and t'_i for the total duration of c'_i. We denote by T' the sum of all the durations t'_i. We first notice that $T' > 0$, and that X_0 does not contain all the clocks, otherwise it would contradict Assumption 10. In other words, $p > 0$ and $X_1 \neq \varnothing$. Clearly enough, along each cycles, if a clock in some X_j is reset, then all clocks in the same X_j are also reset, as well as all the clocks in X_k for all $k \leq j$. This entails that clocks in X_p are never reset along any of the $6(n+2)n^2$ cycles (otherwise, one of the cycles would be a progress cycle). As a consequence, the total duration T' of the $6(n+2)n^2$ cycles is strictly less than $1 + 2\alpha'$. Another consequence is that the sets S_i are characterized by the number of clocks they contain.

  The new trajectory is built by removing one of the cycles. We will have to report its delay on another cycle, in order to get the correct value for clocks that are never reset. This must be done carefully in order to be sure that guards will still be satisfied along the new trajectory.
  First, if $T' \geq 1$, then we know that one of the $6(n+2)n^2$ cycles c'_i has total duration greater than $1/(6(n+2)n^2)$, thus greater than $2\alpha'$. Thus, either the first half, or the second half, of the cycles has total duration strictly less than 1. With this remark, we can assume that $T' < 1$, and that we have $3(n+2)n^2$ cycles. In that case, we are sure that a clock that has been reset will never reach value 1 along that sequence of cycles.

We now explain how we choose the cycle to be removed. As mentionned earlier, the sets S_i are characterized by the number of clocks they contain. We assume in the sequel that the cycles are numbered from 1 to $3(n+2)n^2$. We claim that we can find an index i_0 such that

- $i_0 > 1$,
- $S_{i_0} \subseteq S_{i_0+1} \subseteq S_j$ for some $j < i_0$,
- along cycles c'_{i_0-1}, c'_{i_0} and c'_{i_0+1} of π', no clock reaches a strictly positive integer value.

We now explain why such an i_0 exists: consider the sequence of couples $f_i = (|S_i|, |S_{i+1}|)$, where
$sizeS$ is the number of clocks in S. Such a couple is said to be negative if $|S_{i+1}| < |S_i|$, and nonnegative otherwise. Let N be the number of negative couples, and P be the number of nonnegative ones. For negative couples, we have $|S_{i+1}| - |S_i| \leq -1$, while for nonnegative ones, we have $|S_{i+1}| - |S_i| \leq n-1$. Thus $|S_{3(n+2)n^2}| - |S_1| \leq -N + (n-1)P$, and $(n-1)P - N > -n$. Since $P + N \geq 3(n+2)n^2$ (there are at least $3(n+2)n^2$ cycles), we get that $P \geq 3n(n+2)$. This entails that we can find $3(n+2)$ nonnegative couples f_{e_1}, f_{e_2}, ..., $f_{e_{3(n+2)}}$ having the same second element $|S_{e_j+1}|$. We assume that $e_1 < e_2 < \cdots < e_{3(n+2)}$. Now, along our $3(n+2)n^2$ cycles, we know that each clock can reach a positive integer value at most once, since $T' < 1$. This means that we can find an index $i_0 \in \{e_2, \ldots, e_{3(n+2)}\}$ such that no clock reaches a positive integer value in any of the cycles c_{i_0-1}, c_{i_0} and c_{i_0+1}. That index fulfills our three requirements (with $j = e_1 + 1$).

Now, we build the new trajectory π'': it is similar to π', except that we remove the i_0-th cycle, and report its delay t'_{i_0} as late as possible in the $i_0 - 1$-st cycle (thus either at the very end of that cycle, if time can elapse after the last reset (*i.e.* if $X_0 = \varnothing$), or between the resets of clocks in X_1 and clocks in X_0 otherwise). We assume that all clocks keep the same (global) rate during that extra delay as in the i_0-th cycle.
We also have to be careful at one extra problem: If X_0 is not empty, then at the end of cycle c'_{i_0-1}, clocks in X_0 have been reset at "almost the same time", since Δ t.u. may elapse between any two successive resets of clocks in X_0. For instance, the following path in π:

$$\cdots q_j \xrightarrow[x_1:=0]{} q_{j+1} \xrightarrow{0} q_{j+2} \xrightarrow[x_2:=0]{x_1=0} q_{j+3} \xrightarrow{0} q_{j+4} \xrightarrow[x3:=0]{x_2=0} \cdots$$

could result in the following path in π':

$$\cdots q_j \xrightarrow[x_1:=0]{} q_{j+1} \xrightarrow{\Delta} q_{j+2} \xrightarrow[x_2:=0]{x_1\leq\Delta} q_{j+3} \xrightarrow{\Delta} q_{j+4} \xrightarrow[x3:=0]{x_2\leq\Delta} \cdots$$

It could then be the case that cycle c''_{i_0+1} (the cycle of π'' corresponding to c'_{i_0+1}) requires that all clocks in X_0 are less than Δ. Such a guard would fail with the above example. To overcome that problem, we remove those small delays and report them just before the resets of clocks in X_0 (*i.e.* precisely where we reported the delay t'_{i_0}).

The other transitions of π'' are the same as those of π'. This way, at the end of the $i_0 + 1$-st cycle, all clocks have the same value in π' and in π''.
Last, we have to ensure that all guards along the new trajectory are still satisfied. It is clear for guards occuring before the first modified transition, as well as for guards occuring after c''_{i_0+1}, since clocks have the same values as in π'. For other guards:

- guards "$x = 0$": we have ensured that those guards are satisfied for clocks in X_0 at the beginning of cycle c''_{i_0+1} (by removing the possible superfluous delays at the end of c''_{i_0-1}). At other positions along π'', those guards are still satisfied if they were along π': this can only occur just after a reset of the corresponding clock, and that reset would also occur along π'.
- other guards are satisfied thanks to the requirement that no clock reaches an integer value along the three cycles. The only problem could come from clocks that are reset along c'_{i_0}: those clocks are not reset anymore between c''_{i_0-1} and c''_{i_0+1}. However, we know that those clocks have been reset previously (since $S_{i_0} \subseteq S_{i_0+1} \subseteq S_j$ for some $j < i_0$,) and can thus not reach value 1 at that position.

Thus, we have built a shorter path than π' that still sufficiently moves away from J^*, which contradicts our initial assumption that π' was the shortest one.

We conclude that our initial hypothesis is wrong: there does not exist a stutter-free trajcetory starting in J^* and ending in a position y with $d(y, J^*) \geq \alpha$. ■

Finally, we can establish the completeness of Algorithm 1.

Theorem 44 *Under Assumption 10, the set J^* computed by Algorithm 1 contains $R^*_{\Delta,\varepsilon}$.*

Proof. Let $\alpha \in \mathbb{R}^{>0}$. Let $y \in R^*_{\Delta,\varepsilon}$. Then for any $\Delta > 0$ and $\varepsilon > 0$, $y \in \mathsf{Reach}(\llbracket A \rrbracket^\varepsilon_\Delta)$,*i.e.* there exists a trajectory from the initial state to y in $\llbracket A \rrbracket^\varepsilon_\Delta$. From Theorem 43, $d(y, J^*) < \alpha$ since J^* contains the initial state. Hence, for any $\alpha > 0$, $d(y, J^*) < \alpha$. Therefore, $d(y, J^*) = 0$. Now, J^* is a closed set, since it is a union of closed regions. Thus $y \in J^*$. ■

In summary, we have proved the following inclusions:

$$\begin{tikzcd} & & R^*_\varepsilon \arrow[dr, phantom, "\subsetneq"] & \\ R^*_{\Delta,\varepsilon} \arrow[r, phantom, "\subseteq"] & J^* \arrow[ur, phantom, "\subsetneq"] \arrow[dr, phantom, "\subsetneq"] & & R^*_{\Delta,\varepsilon} \\ & & R^*_\Delta \arrow[ur, phantom, "\subsetneq"] & \end{tikzcd}$$

All those sets are thus equal:

Theorem 45 *Under Assumption 10, we have $R^*_{\Delta,\varepsilon} = R^*_{\Delta,\varepsilon}$, and those sets are computed by Algorithm 1.*

6.4 Complexity

Complexity issues have been studied in [Pur98], so we recall the main result.

Theorem 46 ([Pur98]) *Given a timed automaton $A = \langle \mathsf{Loc}, \mathsf{Var}, q_0, \mathsf{Lab}, \mathsf{Edg} \rangle$ and a location $l_f \in \mathsf{Loc}$, determining whether a state $(l_f, v) \in R^*_\Delta$ (or equivalently $(l_f, v) \in R^*_\varepsilon$) for some valuation v is PSPACE-Complete.*

Proof. For proving PSPACE-membership, it is not possible to use Algorithm 1 because it first constructs the region graph G of the timed automaton A which has a number of states exponential in the number of clocks of A. However, we can check accessibility of q by guessing a path from q_0 to q in polynomial space. This is a classical trick used for showing PSPACE-membership of the reachability problem for standard timed automata with an on-the-fly algorithm [AD94]. In our case, the difficulty to overcome is that the successor of a given region r can be a neighbor region r' of r (*i.e.* such that $r \cap r' \neq \varnothing$) provided r' is in a strongly connected component S of G. As we have shown, the entire region r' can be reached from r no matter how small is Δ by iterating the cycle S. Hence we can add S in one step in the set of reachable states. Such an *acceleration* has been proven correct. So, when guessing the successor of a region r, we must take into account the neighbor regions of r, and decide whether they are in a cycle or not. This can be checked in PSPACE using the procedure for standard timed automata. A polynomially bounded part of the memory space is reserved for executions of this procedure. Since the content of this part of the memory is not necessary for further computations, it can be reused by subsequent calls. With this adaptation, the proof of PSPACE-membership can be completed, and we omit the details.

We establish PSPACE-hardness by reducing the acceptance problem for Linear Bounded Turing Machines (LBTM) to our decision problem. A LBTM is a nondeterministic Turing machine that can only use a number of tape cells equal to the length of its input. A LBTM $M = (Q, \Sigma, q_0, q_f, E)$ consists of a finite set of control states Q, a finite alphabet Σ, an initial state $q_0 \in Q$, a final state $q_f \in Q$, and a set of transitions $E \subseteq Q \times \Sigma \times \Sigma \times \{\mathsf{left}, \mathsf{right}\} \times Q$. A *configuration* of M is a triple $(q, w, i) \in Q \times \Sigma^* \times \mathbb{N}$ where q is a control location, w is the content of the tape, and i is the position of the tape head. An execution of M on the input $x \in \Sigma^*$ is a sequence $s_0 s_1 \dots s_n$ of configurations starting with $s_0 = (q_0, x, 1)$ and finishing in s_n such that s_{i+1} is a successor of s_i for every $0 \leq i < n$. The configuration (q', w', i') is a *successor* of a configuration (q, w, i) iff there exists a transition $(q, \sigma, \sigma', d, q') \in E$ such that:

(1) $w_i = \sigma$;
(2) $w'_i = \sigma'$ and $w'_j = w_j$ for $j \neq i$;
(3) $i' = i - 1$ if $d = \mathsf{left}$ and $i' = i + 1$ if $d = \mathsf{right}$ with $1 \leq i' \leq |x|$.

We assume the condition $1 \leq i' \leq |x|$ is realized using input delimiters. We say that M accepts x iff M has an execution on x finishing in $s_n = (q_f, w, i)$

for some $w \in \Sigma^*$ and $i \in \mathbb{N}$. The acceptance problem for LBTM asks, given a LBTM M and an input word $x \in \Sigma^*$ whether M accepts x.

Our reduction is inspired by [CY91], where a configuration (q, w, i) of M is encoded by a location (q, i) (recording the control state q and the tape position i) and the clocks $y_1, \ldots, y_{|x|}$, one for each tape cell. We assume without loss of generality that $\Sigma = \{a, b\}$. A clock y_i has the value $y_i = n_a$ if $w_i = a$ and $y_i = n_b > n_a$ if $w_i = b$. This encoding is not preserved by time passing. Thus we need to periodically refresh the values of the clocks. This is done in two phases: (I) resetting the clock coding a 'b' (by checking $y_i = n_b$), then letting $n_b - n_a$ time unit pass, and (II) resetting the clock coding an 'a' (by checking $y_i = n_b$ again) and finally letting n_a time unit pass. This schema is modified in order to execute the transitions of the LBTM. It will result in slight changes in the reset policy.

We show how to adapt this framework to our enlarged semantics of timed automata. Due to guards enlargements, equality can not be tested precisely and the clocks can not store precise values n_a and n_b. However, if Δ is sufficiently small, we can still distinguish clocks coding an $'a'$ and clocks coding a $'b'$ if n_a and n_b are not too close. The main details of the proof follow.

Let $M = (Q, \Sigma, q_0, q_f, \delta)$ be a LBTM and $x \in \Sigma^*$ an input word. Let $n = |x|$, $n_a = 3$ and $n_b = 6$. We construct a timed automaton $\mathcal{A}(M, x)$ and a location l_f such that M accepts x iff $(l_f, v) \in R^*_\Delta$ for some valuation v. Let $\mathcal{A}(M, x) = \langle \mathsf{Loc}, \mathsf{Var}, q_0^{\mathcal{A}}, \mathsf{Lab}, \mathsf{Edg}\rangle$ with

- $\mathsf{Loc} = \{s_0, s_1, l_f\} \cup \{(q, i, j, \varphi, d) \mid q \in Q \wedge 1 \le i \le n \wedge 1 \le j \le n+1 \wedge \varphi \in \{\mathrm{I}, \mathrm{II}\} \wedge d \in \{\mathsf{left}, \mathsf{right}\}\}$; a location (q, i, j, φ, d) encodes the control state q, the tape position i, the number of the next clock to be treated j, the phase of the simulation φ, and the direction of the next head movement d.
- $\mathsf{Var} = \{y_i \mid 1 \le i \le n\} \cup \{z\}$
- $q_0^{\mathcal{A}} = (s_0, v_0)$ with $v_0(t) = 0$ for all $t \in \mathsf{Var}$
- $\mathsf{Lab} = \{\tau\}$

The set Edg contains the edges (we write $l \xrightarrow{g,R} l'$ when $(l, l', g, \tau, R) \in \mathsf{Edg}$):

- *Initialization*:
  - $s_0 \xrightarrow{z=3,\{y_i | x_i = a\}\cup\{z\}} s_1$
  - $s_1 \xrightarrow{z=3,\{z\}} (q_0, 1, 1, I, \mathsf{left})$
- *Refresh*: for every $(q, i, j, \varphi, d) \in \mathsf{Loc}$ with $j \neq i$ and $j \le n$,
  - $(q, i, j, \varphi, d) \xrightarrow{z\le 0 \wedge y_j \le 4, \varnothing} (q, i, j+1, \varphi, d)$
  - $(q, i, j, \varphi, d) \xrightarrow{z\le 0 \wedge y_j \ge 5, \{y_j\}} (q, i, j+1, \varphi, d)$
- *Execution*: for every transition $(q, \sigma, \sigma', d', q') \in E$, for every $q \in Q$, $1 \le i \le n$, $d \in \{\mathsf{left}, \mathsf{right}\}$,
  - If $(\sigma, \sigma') = (a, a)$ then $(q, i, i, \mathrm{I}, d) \xrightarrow{z\le 0\wedge y_i \le 4, \varnothing} (q', i, i+1, \mathrm{I}, d')$
  - If $(\sigma, \sigma') = (a, b)$ then $(q, i, i, \mathrm{I}, d) \xrightarrow{z\le 0\wedge y_i \le 4, \{y_i\}} (q', i, i+1, \mathrm{I}, d')$
  - If $(\sigma, \sigma') = (b, a)$ then $(q, i, i, \mathrm{I}, d) \xrightarrow{z\le 0\wedge y_i \ge 5, \varnothing} (q', i, i+1, \mathrm{I}, d')$

  - If $(\sigma, \sigma') = (b, b)$ then $(q, i, i, \mathrm{I}, d) \xrightarrow{z \leq 0 \wedge y_i \geq 5, \{y_i\}} (q', i, i+1, \mathrm{I}, d')$
  - $(q, i, i, \mathrm{II}, d) \xrightarrow{z \leq 0 \wedge y_i \leq 4, \varnothing} (q, i, i+1, \mathrm{II}, d)$
  - $(q, i, i, \mathrm{II}, d) \xrightarrow{z \leq 0 \wedge y_i \geq 5, \{y_i\}} (q, i, i+1, \mathrm{II}, d)$
- *Phase change*: for every $q \in Q$, $1 \leq i \leq n$, $j = n+1$ and $d \in \{\mathsf{left}, \mathsf{right}\}$,
  - $(q, i, n+1, \mathrm{I}, d) \xrightarrow{z=3, \{z\}} (q, i, 1, \mathrm{II}, d)$
  - $(q, i, n+1, \mathrm{II}, \mathsf{left}) \xrightarrow{z=3, \{z\}} (q, i-1, 1, \mathrm{I}, \mathsf{left})$
  - $(q, i, n+1, \mathrm{II}, \mathsf{right}) \xrightarrow{z=3, \{z\}} (q, i+1, 1, \mathrm{I}, \mathsf{right})$
- *Termination*: for every $1 \leq i \leq n$, $d \in \{\mathsf{left}, \mathsf{right}\}$,
  - $(q_f, i, 1, \mathrm{I}, d) \xrightarrow{\mathsf{true}, \varnothing} l_f$

After the *initialization step*, we have in the location $(q_0, 1, 1, I, \mathsf{left})$ when $z = 0$: $y_i = \alpha$ if $x_i = a$ and $y_i = \beta$ if $x_i = b$ with $\alpha \in [3 - \Delta, 3 + \Delta]$ and $\beta \in [6 - 2\Delta, 6 + 2\Delta]$.

After one transition $(q, \sigma, \sigma', d', q')$ of M, let x' be the new tape content (x' differs from x by at most one symbol). If we simulate that transition by executing the *refresh* steps, the *execution* step, and the *phase changes*, it is easy to check that in location $(q, i, 1, I, d)$, when $z = 0$:

- $(A1)$ if $x'_i = a$ then $3 - 2\Delta \leq y_i \leq 3 + \Delta$;
- $(A2)$ if $x'_i = b$ then $6 - 3\Delta \leq y_i \leq 6 + 2\Delta$;

Note that two clocks coding the same symbol are not necessarily equal (however, their difference is bounded by Δ). The reader can check that after having executed two transitions, there is no accumulation of the imprecisions and conditions $(A1)$ and $(A2)$ hold. Hence, provided Δ is sufficiently small (in fact $\Delta < 1/2$), the automaton $\mathcal{A}(M, x)$ will correctly distinguish clocks coding 'a' from clocks coding 'b' for any number of transitions, and thus simulate the execution of M on x. It is easy to check that the location l_f is reachable in R^*_Δ iff l_f is reachable in $[\![\mathcal{A}]\!]^0_0$ iff M accepts x.

Since our construction is polynomial in the size of M and x, and the acceptance problem for LBTM is PSPACE-hard, the proof is complete. ■

6.5 What if we relax the time-elapsing hypothesis?

We briefly show here that Assumption 10 is necessary for the completeness: consider the automaton displayed on Figure 3. That automaton does not have any progress cycle, and Algorithm 1 would end up with $J^* = \big\{(\ell_1, \{t, t\}) \mid t \in \mathbb{R}^{\geq 0}\big\}$. However, if guards are enlarged, it is possible to perform the loop on ℓ_1 each time $x = \Delta$, and then reach state *err*.

We remark that, if only drifts on clocks are allowed, Algorithm 1 behaves correctly on this example. We conjecture that it is always the case, *i.e.* that Assumption 10 is not necessary if only drifts on clocks are assumed.

On the other hand, we conjecture that our algorithm can be adapted in order to relax Assumption 10 in presence of both types of imprecisions. The algorithm

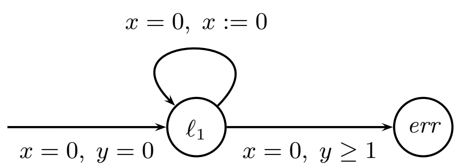


Fig. 3. A timed automaton not satisfying Assumption 10

we proposed is Algorithm 2, in which X_S is the set of clocks that are reset along cycle S, and $r_X^{\nearrow}$ is the zone obtained from r by "freezing" clocks in X and letting time elapse for the other clocks.

7 Conclusion

In this paper, we have shown that a notion of robustness defined by Puri [Pur98] is closely related to the notion of implementability introduced in [DDR05]. Making this link formal allowed us to show that our notion of implementability is decidable for the class of timed automata. To establish this link, we have proved that the algorithm proposed by Puri computes the set of reachable states of timed automata where guards are enlarged by an infinitesimally small rational value. The existence of such a value implies the implementability as shown in our previous paper. The proofs of the decidability result rely on non trivial adaptations of the main ideas underlying the study of drift in the rate of clocks made by Puri.

The algorithm that is used to check implementability manipulates strongly connected components of the region graph. It can be seen as defining exact accelerations of cycles of the timed automaton.

We will work in the future on making those accelerations practical and as a consequence, we will work on to turn the theoretical algorithm proposed in this paper into a practical one. If we succeed in this task, the results of this paper and of [DDR05] will allow us to propose a practical procedure to produce provably correct code from proved correct controller modeled as timed automata.

References

[AAB00] Aurore Annichini, Eugene Asarin, and Ahmed Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In *Proc. 12th Int. Conf. Computer Aided Verification (CAV 2000)*, pages 419–434, 2000.

[ACD+92] Rajeev Alur, Costas Courcoubetis, David L. Dill, Nicolas Halbwachs, and Howard Wong-Toi. An implementation of three algorithms for timing verification based on automata emptiness. In *Proc. 13th IEEE Real-time Systems Symposium*, pages 157–166. IEEE Comp. Soc. Press, 1992.

Algorithm 2: Algorithm for computing $R^*_\varepsilon(A)$ for a timed automaton A.

```
Data: A timed automaton $A = \langle \mathsf{Loc}, \mathsf{Var}, q_0, \mathsf{Lab}, \mathsf{Edg} \rangle$
Result: The set $J^* = R^*_\varepsilon$
begin
    1. Construct the region graph $G = (R_A, \longrightarrow_A)$ of $A$ ;
    2. Compute $\mathsf{SCC}(G) = \{$strongly connected components of $G\}$;
    3. $J^* \leftarrow \mathsf{Reach}(G, [q_0])$ ;
    4. while for some $S = p_0\, p_1 \ \ldots\ p_k \in \mathsf{SCC}(G)$, $[p_0] \not\subseteq J^*$ and $J^* \cap [p_0] \neq \varnothing$
    do
        if S is a progres cycle then
            $J^* \leftarrow J^* \cup [p_0]$
        else
            $J^* \leftarrow J^* \cup ([p_0] \cap ([p_0] \cap J^*)^{\nearrow}_{X_S})$
        $J^* \leftarrow \mathsf{Reach}(G, J^*)$ ;
end
```

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AFM+02] Tobias Amnell, Elena Fersman, Leonid Mokrushin, Paul Pettersson, and Wang Yi. Times: A tool for modelling and implementation of embedded systems. In Joost-Pieter Katoen and Perdita Stevens, editors, *Proc. 8th Int. Conference Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02)*, volume 2280 of *Lecture Notes In Computer Science*, pages 460–464. Springer-Verlag, 2002.

[AFP+03] Tobias Amnell, Elena Fersman, Paul Pettersson, Hongyan Sun, and Wang Yi. Code synthesis for timed automata. *Nordic Journal of Computing*, 9, 2003.

[AIK+03] Rajeev Alur, Franjo Ivancic, Jesung Kim, Insup Lee, and Oleg Sokolsky. Generating embedded software from hierarchical hybrid models. In *Proc. 2003 Conf. Languages, Compilers, and Tools for Embedded Systems (LCTES'03)*, pages 171–182, 2003.

[AMPS98] Eugène Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proc. System Structure and Control*. Elsevier, 1998.

[AT05] Karine Altisen and Stavros Tripakis. Implementation of timed automata: an issue of semantics or modeling? In *Proc. 3rd Int. Conf. Formal Modelling and Analysis of Timed Systems (FORMATS'05)*, Lecture Notes in Computer Science. Springer, 2005.

[BC05] Patricia Bouyer and Fabrice Chevalier. On conciseness of extensions of timed automata. *Journal of Automata, Languages and Combinatorics*, 2005. To appear.

[Bou01] Patricia Bouyer. Updatable timed automata, an algorithmic approach. Technical Report LSV-01-12, ENS Cachan, Cachan, France, 2001.

[CGP99] Edmund Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press, 1999.

[CHR02] Franck Cassez, Thomas A. Henzinger, and Jean-François Raskin. A comparison of control problems for timed and hybrid systems. In *Proc. 5th Int.*

Workshop Hybrid Systems: Computation and Control (HSCC'02), volume 2289 of *Lecture Notes in Computer Science*, pages 134–148. Springer-Verlag, 2002.

[CY91] Costas Courcoubetis and Mihalis Yannakakis. Minimum and maximum delay problems in real-time systems. In *Proc. 3rd Int. Workshop Computer Aided Verification (CAV'91)*, volume 575 of *Lecture Notes in Computer Science*, pages 399–409. Springer-Verlag, 1991.

[DDMR04] Martin De Wulf, Laurent Doyen, Nicolas Markey, and Jean-François Raskin. Robustness and implementability of timed automata. In Y. Lakhnech and S. Yovine, editors, *Proceedings of the Joint Conferences Formal Modelling and Analysis of Timed Systems (FORMATS'04) and Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'04)*, volume 3253 of *Lecture Notes in Computer Science*, pages 118–133, Grenoble, France, September 2004. Springer.

[DDR05] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Almost ASAP semantics: From timed models to timed implementations. *Formal Aspects of Computing*, 17(3):319–341, October 2005.

[Dil90] David Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proc. 1st Int. Workshop Automatic Verification Methods for Finite State Systems (CAV'89)*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer-Verlag, 1990.

[GHJ97] Vineet Gupta, Thomas A. Henzinger, and Radha Jagadeesan. Robust timed automata. In O. Maler, editor, *Proc. Int. Workshop Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *Lecture Notes in Computer Science*, pages 331–345. Springer Verlag, March 1997.

[HKSP03] Thomas A. Henzinger, Christoph M. Kirsch, Marco A. Sanvido, and Wolfgang Pree. From control models to real-time code using GIOTTO. *IEEE Control Systems Magazine*, 23(1):50–64, 2003.

[HRSV01] Thomas Hune, Judi Romijn, Marielle Stoelinga, and Frits W. Vaandrager. Linear parametric model checking of timed automata. In *Proc. 7th Int. Conf. Tools and Algorithms for Construction and Analysis of Systems (TACAS'01)*, pages 189–203, 2001.

[Pur98] Anuj Puri. Dynamical properties of timed automata. In *Proc. 5th Int. Symposium Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98)*, volume 1486 of *Lecture Notes in Computer Science*, pages 210–227. Springer, 1998.

[Yov96] Sergio Yovine. Model checking timed automata. In *European Educational Forum: School on Embedded Systems*, pages 114–152, 1996.