P. Bouyer, N. Markey and
P.A. Reynier

Robust Model-Checking of Timed Automata

Research Report LSV-05-08
June 2005

Laboratoire
Spécification
et
Vérification

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Robust Model-Checking of Timed Automata

Patricia Bouyer, Nicolas Markey, Pierre-Alain Reynier

LSV – CNRS & ENS de Cachan – France
emails: {bouyer,markey,reynier}@lsv.ens-cachan.fr

Abstract. Recent works by Raskin *et al.* have brought out a model of parameterized timed automata which can be used for proving that timed systems are *implementable*. This is strongly connected to *robustly* verifying timed automata, *i.e.* verifying whether a property still holds even if the transitions may be slightly delayed or expedited. In those works, they have proved decidability of robust model-checking for simple safety objectives like "avoid a set of bad states". We extend here these results by providing PSPACE algorithms for the robust model-checking of Büchi-like and LTL properties. We also solve the case of the bounded-response-time property.

1 Introduction

Verification and control of real-time systems. In the last thirty years, formal verification of systems has become a very active field of research in computer science, with numerous success stories. Formal verification aims at checking that (the model of) a system satisfies (a formula expressing) its specifications. The importance of taking real-time constraints into account in verification has quickly been understood, and the model of timed automata (defined by Alur & Dill [2, 3]) has become one of the most established models for real-time systems, with well-studied underlying theory and development of mature model-checking tools, such as Uppaal [17] and Kronos [9].

Implementation of real-time systems. Though theoretical results about the verification of real-time systems are now well-understood, implementing those systems is more problematic: timed automata are governed by an idealized, theoretical semantics, and their implementation on real hardware could fail to satisfy their mathematically proved properties. More precisely, timed automata assume that clocks are continuous and infinitely precise, while CPU have a finite frequency and are digital. This is not bound to the non-zenoness property since, for instance, [10] shows an example of a timed automaton that performs transitions at dates n and $n + 1/n$. Any implementation of such an automaton will eventually fail to satisfy that property.

Another difference between the theoretical and real-life frameworks is the immediacy of communications. This is especially true in controller synthesis,

Work supported by ACI SI CORTOS (Control and Observation of Real-Time Open Systems).

where the aim is to build a controller (a timed automaton) that "tells" the system what to do in order to continuously ensure a given property. Communications are modelled as being immediate, while this is unrealistic in practice.

A semantical point of view. In [13], a new semantics, called the AASAP-semantics (AASAP stands for "Almost ASAP"), has been introduced for timed automata. It takes into account both the inherent digital aspect of hardware and the non-instancy of hardware communication. In this semantics, the controller has a non-null reaction delay, synchronization of actions is not instantaneous, and several characteristics of a real processor are taken into account. The point is then to decide whether a given classical controller correctly supervises the system under the AASAP-semantics. In [13], solving this problem is reduced to that of checking whether there exists a delay reaction Δ for the controller to supervise the system: given a system Sys and a controller Cont, their interaction is denoted $[\![\mathsf{Sys} \parallel \mathsf{Cont}]\!]_\Delta$ where Δ is the parameter representing the reaction delay of the controller (and in practice this is the classical parallel composition where clock constraints are enlarged by Δ).

The problem is then to decide, given a property $\mathcal{P}$ to be satisfied, whether there exists some $\Delta \in \mathbb{Q}_{\geq 0}$ s.t. $[\![\mathsf{Sys} \parallel \mathsf{Cont}]\!]_\Delta$ satisfies $\mathcal{P}$. It is thus a problem of *robust model-checking*. The special case of safety properties (stating that a set of bad configurations is never reached) has been solved in [11] through a region-based algorithm.

Our contribution. In this paper, we solve the robust model-checking problem for more general specifications like Büchi and LTL properties. The algorithm we propose is based on an extension of the classical region automaton construction which has been inspired by the algorithm proposed in [11] for safety properties. Our algorithms remain in PSPACE, which we prove is optimal.

We also develop a PSPACE algorithm for verifying simple timed properties (namely, the bounded-response-time and bounded-invariance properties). Our algorithm is *ad hoc*, but it is a first step towards the verification of more general timed specifications.

Related work.

- Our approach contrasts with other modeling solutions proposed, for example, in [1], where the authors propose to model the behavior of the platform as a timed automaton. This framework is very expressive, but suffers from not verifying the "faster is better" property ("if an automaton can be implemented on some hardware, it can also be implemented on faster hardwares").
- A notion of robust timed automata has been proposed and studied in [14, 18], where not all traces are accepted, but only those belonging to an accepting tube. This approach is topological, and is not related to ours (in fact, it drops some behaviors of the system while we add some), though this is also a semantical proposition for robustness.

– In [20, 6], a small perturbation on slopes of clocks is allowed. Such a perturbation is closely related to the model we consider, as safety properties are satisfied in a similar way with such a perturbation and with the enlargement on clock constraints like in our model, as proved in [20].
– Our work is also related to works on code generation from timed automata: the tool Times [7] automatically generates an executable code corresponding to a timed automaton. However, they do not deal with the problem of hardware imprecision. Henzinger *et al.* propose a model, called Giotto, as a model for embedded software that can be used as an intermediate step between theoretical models and real platforms [15].

Outline of the paper. In Section 2, we introduce classical definitions on timed automata. In Section 3, we define the notion of robust satisfiability, and the problem of robust model-checking, and make clear the link between our work and the results of [13, 11]. Then, we provide in Section 4 our model-checking algorithm for co-Büchi properties, and in Section 5 its application to LTL properties. Finally, we present in Section 6 our first results for timed properties, and conclude in Section 7 with a large landscape on possible future work.

2 Definitions

Timed automata. Let $\mathcal{C}$ be a finite set of variables, named *clocks*. We denote by $\mathcal{G}$ the set of *clock contraints* generated by the following grammar:

$$\mathcal{G} \ni g ::= g \wedge g \mid c \sim n$$

where c ranges over $\mathcal{C}$, n ranges over the set of natural numbers $\mathbb{N}$ and[1] $\sim \in \{\leq, \geq\}$. Given a parameter $\Delta \in \mathbb{Q}_{\geq 0}$, whether a clock valuation $v\colon \mathcal{C} \to \mathbb{R}^+$ satisfies a constraint g within Δ, written $v \models_\Delta g$, is defined inductively as follows:

$$\begin{cases} v \models_\Delta c \leq n & \text{iff } \ v(c) \leq n + \Delta \\ v \models_\Delta c \geq n & \text{iff } \ v(c) \geq n - \Delta \\ v \models_\Delta g_1 \wedge g_2 & \text{iff } \ v \models_\Delta g_1 \text{ and } v \models_\Delta g_2 \end{cases}$$

A *timed automaton* is a 5-tuple $\mathcal{A} = (L, \ell_0, \mathcal{C}, \Sigma, \delta)$ where L is a finite set of *locations*, $\ell_0 \in L$ is the initial location, $\mathcal{C}$ is a finite set of *clocks*, Σ is a finite set of *actions*, and $\delta \subseteq L \times \mathcal{G} \times \Sigma \times 2^{\mathcal{C}} \times L$ is the set of transitions. In the sequel, w.l.o.g. we assume that transitions are labeled by their name, and we identify Σ with δ.

We define a parameterized semantics for $\mathcal{A}$ which we denote by $[\![\mathcal{A}]\!]_\Delta$. Notice that, in the definitions below, the standard semantics of timed automata can be recovered by letting $\Delta = 0$. In that case, we omit to mention the subscript Δ in the notations.

[1] We simplify the notations by assuming that all inequalities are non-strict. As argued in [11], this does not change the expressive power of the model under the enlarged semantics.

A *state* of $[\![\mathcal{A}]\!]_\Delta$ is a pair (ℓ, v) where $\ell \in L$ and $v\colon \mathcal{C} \to \mathbb{R}^+$ assigns to each clock its current value. Intuitively, in a given position (ℓ, v), there are two possible behaviors for $[\![\mathcal{A}]\!]_\Delta$:

- it can either perform a *action transition*, namely a transition of δ. This requires that there exists $(\ell, g, \sigma, r, \ell') \in \delta$ s.t. $v \models_\Delta g$. In that case, the automaton ends up in state $(\ell', v[r \leftarrow 0])$, where $v[r \leftarrow 0]$ is the valuation mapping clocks in r to 0 and the other clocks to their valuation given by v;
- or it can perform a *delay transition*, *i.e.* let a certain amount of time t elapse. In that case, the automaton ends up in state $(\ell, v+t)$ where $v+t$ represents the valuation $c \mapsto v(c) + t$ for all $c \in \mathcal{C}$.

In the first case we write $(\ell, v) \xrightarrow{\sigma}_\Delta (\ell', v[r \leftarrow 0])$, whereas we write $(\ell, v) \xrightarrow{t}_\Delta (\ell, v+t)$ in the second case. The graph $[\![\mathcal{A}]\!]_\Delta$ is thus an infinite transition system.

Paths in timed automata. A *trace* in a timed automaton $\mathcal{A} = (L, \ell_0, \mathcal{C}, \Sigma, \delta)$ is a (finite or infinite) sequence of transitions $(\delta_i)_{i\in I} \in \delta^I$ (if $\delta_i = (\ell_i, g_i, \sigma_i, r_i, \ell'_i)$, then $\ell'_i = \ell_{i+1}$).

A *path* of $[\![\mathcal{A}]\!]_\Delta$ over a trace $(\delta_i)_{i\in I}$ is a sequence $(\ell_0, v_0) \xrightarrow{d_0}_\Delta (\ell_0, v_0 + d_0) \xrightarrow{\delta_0}_\Delta (\ell_1, v_1) \xrightarrow{d_1}_\Delta (\ell_1, v_1 + d_1) \ldots$ where for each $i \in I$, $d_i \in \mathbb{R}^+$. The (unique) trace corresponding to a path π is referred to as $\mathrm{trace}(\pi)$. Note that a trace does not necessarily admit a path. Traces that do are said to be *consistent*. Also note that a path is a sequence of consecutive transitions of $[\![\mathcal{A}]\!]_\Delta$ which is *stutter-free* in the sense that delay and action transitions alternate. We define the distance $\mathrm{d}((\ell, v), (\ell', v'))$ between two states (ℓ, v) and (ℓ', v') to be $\|v - v'\|_\infty$ if $\ell = \ell'$ and $+\infty$ otherwise. The distance between two paths $\pi = (\pi_i)$ and $\pi' = (\pi'_i)$ of the same length is defined accordingly as the maximum of the distances $\mathrm{d}(\pi_i, \pi'_i)$.

Let $T = (\delta_i)_{i\in I}$ be a trace of $\mathcal{A}$. A state ℓ' is said to be *reachable* from a set of states R following T in $[\![\mathcal{A}]\!]_\Delta$ if there exists a path over T in $[\![\mathcal{A}]\!]_\Delta$ starting in some $r \in R$ and containing q'. We write $\mathsf{Reach}^T_\Delta(R)$ for the set of states that are reachable from R following trace T. We note $\mathsf{Reach}_\Delta(R)$ for the union over all possible traces T of $\mathsf{Reach}^T_\Delta(R)$. This set represents all locations that are reachable in $[\![\mathcal{A}]\!]_\Delta$. We also define $\mathsf{Reach}^*(R) = \bigcap_{\Delta>0} \mathsf{Reach}_\Delta(R)$. Since $\Delta \mapsto \mathsf{Reach}_\Delta(R)$ is increasing, this definition makes sense, and, in the end, $\mathsf{Reach}^*(R)$ contains the set of positions that are reachable whatever the positive value of Δ.

Region automaton. In order to symbolically reason about the infinite state space of timed automata, [3] defines an equivalence relation (of finite index) as follows. Let $\mathcal{A}$ be a timed automaton, and M be the largest integer constant occuring in $\mathcal{A}$. Two valuations v and v' are equivalent iff the following conditions hold on valuations v and v':[2]

[2] $\lfloor v(c) \rfloor$ represents the integer part of $v(c)$ and $\langle v(c) \rangle$ represents its fractional part.

- for all $c \in \mathcal{C}$, either both $v(c)$ and $v'(c)$ are greater than M, or $\lfloor v(c) \rfloor = \lfloor v'(c) \rfloor$;
- for all $c, c' \in \mathcal{C}$, if both $v(c)$ and $v(c')$ are lower than M, then
  - $\langle v(c) \rangle \leq \langle v(c') \rangle$ iff $\langle v'(c) \rangle \leq \langle v'(c') \rangle$;
  - $\langle v(c) \rangle = 0$ iff $\langle v'(c) \rangle = 0$.

This defines an equivalence relation, whose equivalence classes are referred to as *regions*. We write $[v]$ for the region containing v, and $\overline{r}$ for the topological closure of a region r.

The set of regions is finite, and exponential in the size of the timed automaton. We define the *region automaton* as the finite automaton $\mathcal{R}(\mathcal{A}) = (\Gamma, \gamma_0, \rightarrow)$ where

- Γ is the set $\{(\ell, r) \mid \ell \in L,\ r \text{ region}\}$,
- γ_0 is the set initial state (ℓ_0, r_0) where r_0 is the region which contains the valuation v_0 with $v_0(c) = 0$ for every $c \in \mathcal{C}$
- $\rightarrow \subseteq \Gamma \times (\Sigma \cup \{\tau\}) \times \Gamma$ and $((\ell, r), \sigma, (\ell', r')) \in \rightarrow$ iff $(\ell, r) \neq (\ell', r')$ and
  - either $\sigma \in \Sigma$ and $((\ell, v), \sigma, (\ell', v'))$ is a transition of $[\![\mathcal{A}]\!]$ for some $v \in r$ and $v' \in r'$,
  - or σ is the symbol τ, and there exists $t \in \mathbb{R}^+$ s.t. $((\ell, v), t, (\ell', v'))$ is a transition of $[\![\mathcal{A}]\!]$ for some $v \in r$ and $v' \in r'$.

The notions of path in the region automaton, trace of a path, ... are defined in the usual way. It is well known that this automaton is *time-abstract bisimilar* to the original timed automaton, which implies that, under the standard semantics, all reachability and Büchi-like properties can be checked equivalently on the original timed automaton or on the region automaton. We assume that classical properties of region automata are known, and refer to [3] for more details.

Restrictions on timed automata. A *progress cycle* in the region automaton of $\mathcal{A}$ is a cyclic path along which all the clocks are reset, and that does not only contain the initial region (*i.e.* the region where all the clocks are set to 0). We do the following hypothesis on timed automata:

Restrictions 1 *Following [11], we restrict our study to timed automata $\mathcal{A}$ satisfying the following requirements:*

- *clocks are bounded by some constant M,*
- *all the cycles in the region automaton $\mathcal{R}(\mathcal{A})$ are* progress cycles*.*

The first hypothesis is not really restrictive since every timed automaton can be transformed into such a bounded timed automaton (see for example [8]). Note that it entails that any time-divergent path contains infinitely many action transitions. The second point is a classical restriction [20]. As mentioned in [11], it is less restrictive than classical strong-non-Zenoness assumptions.

3 Robust Verification

In this section, after several remarks on the implementability of timed systems, we present the problem of robust verification. We also briefly recall results obtained in [11] concerning safety properties, which are the bases of this work.

Implementability of timed systems. Controllers of programs built using a classical synthesis algorithm may be seen as idealized controllers which are difficult to implement. We should be able to guarantee that a controller built for satisfying some property $\mathcal{P}$ can be implemented in such a way that an implementation of the controller also satisfies the property $\mathcal{P}$. In [13], a simplified model of hardware is given, with specifications (the frequency of the clock and the speed of the CPU) given as characteristic parameters of the platform on which the controller will be implemented. Two important properties are then proved: 1) first, "faster is better", which means that if a program correctly behaves (w.r.t. the property $\mathcal{P}$) on a given hardware, then it will also behave correctly on a faster hardware, 2) for a program $\mathcal{A}$ to be correctly implemented on a platform as the one described above, it is sufficient to prove its correctness on $[\![\mathcal{A}]\!]_\Delta$ for some $\Delta > 0$. This naturally leads to the definition of robust satisfaction below.

Robust satisfaction. We assume that we are given a property $\mathcal{P}$ for paths of timed automata, and we note $\models$ the classical satisfaction relation for $\mathcal{P}$. Given a timed automaton $\mathcal{A}$, with initial state (ℓ_0, v_0), we define the *robust satisfaction relation* $\models\!\!\!\equiv$ as follows:

$$\mathcal{A} \models\!\!\!\equiv \mathcal{P} \overset{\text{def}}{\iff} \exists \Delta > 0 \text{ s.t. for all paths } \pi \text{ of } [\![\mathcal{A}]\!]_\Delta \text{ starting in } (\ell_0, v_0),\ \pi \models \mathcal{P}.$$

Intuitively, if the property $\mathcal{P}$ holds *robustly*, then it is possible to find a sufficiently fast hardware (somehow given by the parameter Δ) to implement the automaton $\mathcal{A}$ correctly w.r.t. $\mathcal{P}$, because, as explained above and proved in [13],

$$\mathcal{A} \models\!\!\!\equiv \mathcal{P} \implies \mathcal{A} \text{ implementable w.r.t. } \mathcal{P}.$$

Robust model-checking. In the sequel we address the following problem:

Definition 2 (Robust model-checking). *Given a timed automaton $\mathcal{A}$ and a path property $\mathcal{P}$, can we decide whether $\mathcal{A} \models\!\!\!\equiv \mathcal{P}$?*

The robust model-checking problem has been solved in [11] for safety properties of the type "avoid bad states". More precisely, the authors prove the following theorem:

Theorem 3 ([11]). *Let $\mathcal{A}$ be a timed automaton with initial state (ℓ_0, v_0), let* Bad *be a set of bad locations of $\mathcal{A}$, then:*

1. *$\exists \Delta > 0$ s.t. $\mathsf{Reach}_\Delta(\ell_0, v_0) \cap \mathsf{Bad} = \emptyset$ is equivalent to $\mathsf{Reach}^*(\ell_0, v_0) \cap \mathsf{Bad} = \emptyset$,*
2. *checking whether $\mathsf{Reach}^*(\ell_0, v_0) \cap \mathsf{Bad} = \emptyset$ is decidable, and* PSPACE*-complete.*

The proof of this theorem relies on the classical region automaton construction where strongly connected components (SCC for short) of the region automaton are added to the set $\mathsf{Reach}^*(\ell_0, v_0) = \bigcap_{\Delta>0} \mathsf{Reach}_\Delta(\ell_0, v_0)$ as soon as it can be reached: indeed, if an SCC can be partly reached, then by iterating the SCC, it can be proved (see [11]) that all points of the region can also be reached.

Example 1 ([20, 11]). Consider automaton depicted on Figure 1. For this au-

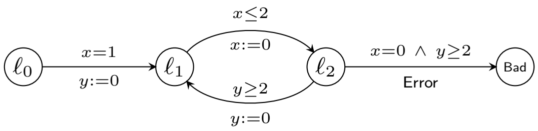

Fig. 1. A timed automaton $\mathcal{A}$.

tomaton, it is possible to compute the sets $\mathsf{Reach}(\ell_0, v_0)$ and $\mathsf{Reach}^*(\ell_0, v_0)$. We obtain, for locations ℓ_1 and ℓ_2, the two sets described on Figure 2. The difference is due to the iteration of the cycle around ℓ_1 and ℓ_2.

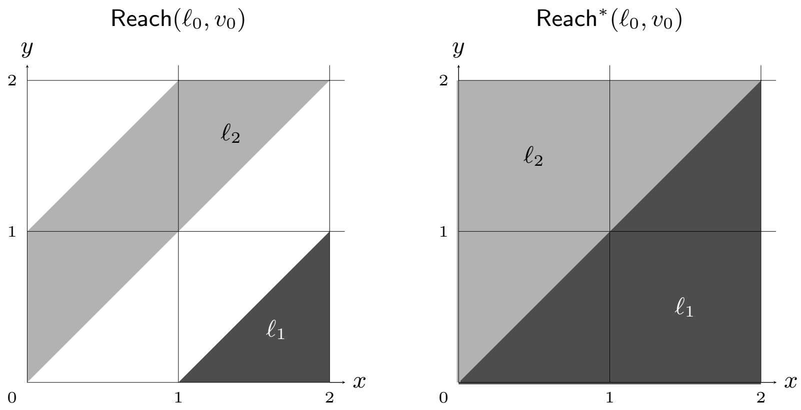

Fig. 2. Differences between $\mathsf{Reach}(\ell_0, v_0)$ and $\mathsf{Reach}^*(\ell_0, v_0)$.

Remark 1. It is important to notice that this semantics may be difficult to understand. In particular it is possible for a system to not satisfy robustly a property $\mathcal{P}$, and to not satisfy robustly the property $\neg\mathcal{P}$, which means that the sentence "$\mathcal{A} \not\equiv \mathcal{P}$" is not equivalent to the sentence "$\mathcal{A} \equiv \neg\mathcal{P}$". For example, let consider the automaton $\mathcal{A}$ of Example 1, and the safety property $\mathcal{P} = \mathbf{G}\,(\neg\mathsf{Error})$ (its negation is $\neg\mathcal{P} = \mathbf{F}\,(\mathsf{Error})$). We can observe that in $[\![\mathcal{A}]\!]$, transition Error is never enabled, whereas for any $\Delta > 0$, whatever how small, the valuation

$v = (0, 2)$ is reachable in $[\![\mathcal{A}]\!]_\Delta$ (see Figure 2), and thus the transition Error will be enabled. Finally, we will have both $\mathcal{A} \not\models \mathcal{P}$ and $\mathcal{A} \not\models \neg\mathcal{P}$.

In the next sections, we extend results of [11] to more general properties than the simple safety properties, and prove that the robust model-checking is decidable for Büchi-like properties, LTL properties, and for bounded-response-time properties.

4 Robust Model-Checking of co-Büchi Conditions

In this section, we are interested in co-Büchi conditions: given a set B of locations in the timed automaton, a path π satisfies $\mathsf{co\text{-}Büchi}(B)$ iff its trace contains finitely many transitions entering a location in B. Following Section 3, this immediately defines the notions of satisfaction and robust satisfaction of a co-Büchi condition in a timed automaton. We also extend the notion of satisfying a co-Büchi condition to the region automaton: it satisfies a co-Büchi condition B iff every path starting in γ_0 (the initial state) only runs in states of B finitely many times.

Extended region automaton $\mathcal{R}^*$. We adapt the algorithm proposed in [11] by building an extension of the region automaton that takes into account the possible "deviations" of the underlying timed automaton.

Let $\mathcal{A}$ be a timed automaton, and $\mathcal{R}(\mathcal{A})$ be its corresponding region automaton. We define the extended region automaton $\mathcal{R}^*(\mathcal{A})$ as follows:

- states of $\mathcal{R}^*(\mathcal{A})$ are states of $\mathcal{R}(\mathcal{A})$, *i.e.* pairs (ℓ, r) where ℓ is a location of $\mathcal{A}$ and r is a region for automaton $\mathcal{A}$
- transitions of $\mathcal{R}^*(\mathcal{A})$ are transitions of $\mathcal{R}(\mathcal{A})$ (we assume labels of transitions are names of transitions in $\mathcal{A}$), and transitions $(\ell, r) \xrightarrow{\gamma} (\ell, r')$ when $\overline{r} \cap \overline{r'} \neq \emptyset$ and (ℓ, r') is in an SCC of $\mathcal{R}(\mathcal{A})$

Decidability of the robust model-checking for co-Büchi conditions.

Theorem 4. *Let $\mathcal{A}$ be a timed automaton and B a set of locations of $\mathcal{A}$. Then*

$$\mathcal{A} \models\!\!\!\equiv \mathsf{co\text{-}Büchi}(B) \iff \mathcal{R}^*(\mathcal{A}) \models \mathsf{co\text{-}Büchi}(B)$$

The proof of this result relies on the two following lemmae:

Lemma 5 ([12, Theorem 27]). *Let $\mathcal{A}$ be a timed automaton with n clocks, let $\alpha < \frac{1}{n}$, let k be an integer. There exists some $\Delta > 0$ such that for every path π in $[\![\mathcal{A}]\!]_\Delta$ over trace T of length less than k, there exists a path π' in $[\![\mathcal{A}]\!]$ over T such that the distance between π and π' is strictly less than α.*

Lemma 6 ([12, Theorem 25]). *Let $\mathcal{A}$ be a timed automaton, let τ be the trace of a cyclic path of $\mathcal{R}(\mathcal{A})$, and let (ℓ, r) be a state of τ. Then for all v and v' in $\overline{r}$, for every $\Delta > 0$, there exists a path in $[\![\mathcal{A}]\!]_\Delta$ over trace τ^k for some $k \in \mathbb{N}$, starting in (ℓ, v) and ending in (ℓ, v').*

The intuition behind the first lemma is that if we give a bound on the length of paths, it is possible to take $\Delta > 0$ small enough so that paths of $[\![\mathcal{A}]\!]_\Delta$ can be approached by real paths of $[\![\mathcal{A}]\!]$. The intuition behind the second lemma is that two states of $\mathcal{A}$ belonging to some cyclic path of $\mathcal{R}(\mathcal{A})$ can be connected (using a path of $[\![\mathcal{A}]\!]_\Delta$) just by iterating the cyclic path of $\mathcal{R}(\mathcal{A})$ (of course the number of times the cycle needs to be iterated depends on the choice of Δ, the smaller is Δ, the larger is the number of iterations which is needed).

Proof (of Theorem 4). We first assume that $\mathcal{A} \models\!\!\!\equiv \mathsf{co\text{-}B\ddot{u}chi}(B)$, and pick some $\Delta > 0$ s.t. every path of $[\![\mathcal{A}]\!]_\Delta$ starting in (ℓ_0, v_0) satisfies the co-Büchi condition.

Assume that $\mathcal{R}^*(\mathcal{A}) \not\models \mathsf{co\text{-}B\ddot{u}chi}(B)$, and pick a path π not satisfying the co-Büchi condition. There is a state $f \in B$ in which the path enters infinitely many times. We split that path in a sequence of subpaths not containing any γ-transitions: $\pi = \pi_0 \xrightarrow{\gamma} \pi_1 \xrightarrow{\gamma} \cdots$. Then each π_i is a non-empty finite (except possibly the last one) path in $\mathcal{R}(\mathcal{A})$ over trace T_i. We let r_i be the first region of each π_i, and r'_i be the last one. By construction of $\mathcal{R}^*(\mathcal{A})$, we have $\overline{r'_i} \cap \overline{r_{i+1}} \neq \emptyset$. We use the following lemma to build a path in $[\![\mathcal{A}]\!]_\Delta$ starting in the initial state (ℓ_0, u_0) of $\mathcal{A}$ and not satisfying the co-Büchi condition.

Lemma 7. *Let π be a path in $\mathcal{R}(\mathcal{A})$ labelled by T, starting in (ℓ, r), and ending in (ℓ', r') such that there is a transition $(\ell', r') \xrightarrow{\gamma} (\ell', r'')$ in $\mathcal{R}^*(\mathcal{A})$ (due to a cyclic path over some trace τ). Then, for every $\Delta > 0$,*

1. *for every valuation $v' \in \overline{r'}$, there exists a valuation $v \in \overline{r}$ and a path in $[\![\mathcal{A}]\!]_\Delta$ from (ℓ, v) to (ℓ', v') over trace T;*
2. *for every valuation $v' \in \overline{r'} \cap \overline{r''}$, for every valuation $v'' \in \overline{r''}$, there exists a path in $[\![\mathcal{A}]\!]_\Delta$ over trace τ^k (for some $k \geq 0$) from (ℓ', v') to (ℓ', v'').*

Proof (of the lemma). The set $\bigcap_{\Delta>0} \mathsf{Reach}^T_\Delta(\ell, \overline{r})$ is closed, and contains the region r' by hypothesis; It thus contains $\overline{r'}$. Thus, for every $\Delta > 0$, $(\ell', \overline{r'}) \subseteq \mathsf{Reach}^T_\Delta(\ell, \overline{r})$, which concludes the first point of the lemma.

The second point is a direct consequence of Lemma 6: the cycle over trace τ can be iterated and connect any two states of r''. $\square$

Applying this lemma, for each i such that π_i is finite, for all $v'_i \in \overline{r'_i} \cap \overline{r_{i+1}}$, there exists $v_i \in \overline{r_i}$ and a real path in $[\![\mathcal{A}]\!]_\Delta$ over trace T_i from configuration (ℓ_i, v_i) to (ℓ'_i, v'_i). To glue everything together, we can apply the second point of the lemma: we can construct a real path of $[\![\mathcal{A}]\!]_\Delta$ over trace $\tau_i^{k_i}$, for some $k_i \geq 0$, from (ℓ'_i, v'_i) to (ℓ_{i+1}, v_{i+1}). Notice that for the first subpath π_0, since the initial region r_0 equals to $\{u_0\}$, we have $v_0 = u_0$ and thus the corresponding real path starts in (ℓ_0, u_0) as required.

If there are infinitely many γ-transitions along π, it is sufficient to glue everything together. We get a real path of $[\![\mathcal{A}]\!]_\Delta$ over trace $T_0 \tau_1^{k_1} T_1 \ldots \tau_i^{k_i} T_i \ldots$ which goes infinitely often through location f.

If there are only finitely many (p, say) γ-transitions along π, then path π_p is an infinite path of $\mathcal{R}(\mathcal{A})$, we can fix a valuation $v_p \in r_p$ and build a real path of $\mathcal{A}$ (and thus of $[\![\mathcal{A}]\!]_\Delta$) starting in (ℓ_p, v_p) and which follows π_p. That way, we get a real path of $[\![\mathcal{A}]\!]_\Delta$ labelled by $T_0\tau_1^{k_1}T_1\dots T_{p-1}\tau_p^{k_p}T_p$ which runs infinitely often through location f.

In both cases, we get a contradiction with the initial assumption.

Conversely, assume that $\mathcal{A} \not\models \mathsf{co\text{-}Büchi}(B)$. This entails that, for any positive Δ, there is a path in $[\![\mathcal{A}]\!]_\Delta$ entering infinitely many times a state in B. Since B is finite, there exists a state $f \in B$ s.t., for all positive Δ, there exists $0 < \Delta' < \Delta$ and a path $\pi_{\Delta'}$ in $[\![\mathcal{A}]\!]_{\Delta'}$ entering infinitely many times location f. We fix such a location f.

Following lemma 5, we pick some $\alpha < 1/n$, and fix $k = 2W+1$ where W is the number of states of $\mathcal{R}(\mathcal{A})$. Lemma 5 ensures the existence of some positive Δ s.t., for all $0 < \Delta' < \Delta$, any path in $[\![\mathcal{A}]\!]_{\Delta'}$ of length less than k can be approximated by a path in $[\![\mathcal{A}]\!]$ over the same trace and s.t. the distance between both paths is strictly less than $\frac{\alpha}{2}$.

For this precise Δ, we pick $0 < \Delta' < \Delta$ and a path π of $[\![\mathcal{A}]\!]_{\Delta'}$ which satisfies the Büchi condition $\{f\}$.

We prove that, for each (finite) path in $[\![\mathcal{A}]\!]_{\Delta'}$, we can build a path in $\mathcal{R}^*(\mathcal{A})$ following the same transitions (except that it can possibly perform extra γ-transitions). This is sufficient as we can then take a prefix of π which contains $k+1$ times the discrete state f, its corresponding path in $\mathcal{R}^*(\mathcal{A})$ will contain $k+1$ times the discrete state f, there will thus be a reachable cycle in $\mathcal{R}^*(\mathcal{A})$ which contains state f, which means that some path of $\mathcal{R}^*(\mathcal{A})$ satisfies the Büchi condition $\{f\}$.

Let ρ be a path in $[\![\mathcal{A}]\!]_{\Delta'}$. We prove, by induction on the length of the path, that we can build a path ν in $\mathcal{R}^*(\mathcal{A})$ whose trace, when removing γ-transitions, is the same as the trace of ρ, and that remains at distance less than $\frac{\alpha}{2}$ from ρ. In the sequel, by abuse of notation, for two clock valuations u and v, we write $\mathrm{d}(u, v)$ for $\|v-u\|_\infty$. The distance between a valuation u and a region r, denoted by $\mathrm{d}(u, r)$, is the infimum of the distances between u and the valuations $v \in r$.

- if ρ has length less than k, then we can directly apply Lemma 5: there is a corresponding path in $[\![\mathcal{A}]\!]$ which follows the same transitions as π, and thus a path in $\mathcal{R}(\mathcal{A})$ (and thus in $\mathcal{R}^*(\mathcal{A})$) taking the same transitions;
- assume that path ρ has length N and we have solved the problem for paths of length strictly less than N. We fix the notations:

$$\rho = (\ell_0, v'_0) \xrightarrow{\sigma_1} (\ell_1, v'_1) - \cdots \xrightarrow{\sigma_{N-1}} (\ell_{N-1}, v'_{N-1}) \xrightarrow{\sigma_N} (\ell_N, v'_N).$$

  By induction hypothesis, there is a path ν_{N-1} in $\mathcal{R}^*(\mathcal{A})$ s.t.

$$\nu_{N-1} = (\ell_0, u_0) \xrightarrow{\gamma^*\sigma_1\gamma^*} (\ell_1, u_1) - \cdots \xrightarrow{\gamma^*\sigma_{N-1}\gamma^*} (\ell_{N-1}, u_{N-1})$$

  where u_i are regions s.t., for every $0 \le i < N$, $\mathrm{d}(v'_i, u_i) < \frac{\alpha}{2}$. From Lemma 5, we can construct a path ρ' in $\mathcal{A}$

$$\rho' = (\ell_{N-k}, v_{N-k}) \xrightarrow{\sigma_{N-k+1}} (\ell_{N-k+1}, v_{N-k+1}) - \cdots \xrightarrow{\sigma_N} (\ell_N, v_N)$$

such that for each $N-k \le i \le N$, $\mathrm{d}(v_i, v'_i) < \frac{\alpha}{2}$.We consider the corresponding run in $\mathcal{R}(\mathcal{A})$:

$$\nu' = (\ell_{N-k}, r_{N-k}) \xrightarrow{\sigma_{N-k+1}} (\ell_{N-k+1}, r_{N-k+1}) - \cdots \xrightarrow{\sigma_N} (\ell_N, r_N)$$

where $r_i = [v_i]$.
For each $N-k \le i \le N$, we get that there exists $z_i \in u_i$ such that $d(z_i, v_i) < \alpha$, which entails that $\overline{r_i} \cap \overline{u_i} \neq \emptyset$ (see [12, Lemma 12]).
As the length of ν' is k, there exists $N-k \le p < q \le N$ such that $(\ell_p, r_p) = (\ell_q, r_q)$, thus (ℓ_p, r_p) lies in an SCC of $\mathcal{R}(\mathcal{A})$. Since $\overline{r_p} \cap \overline{u_p} \neq \emptyset$, there is a transition $(\ell_p, u_p) \xrightarrow{\gamma} (\ell_p, r_p)$ in $\mathcal{R}^*(\mathcal{A})$. This will allow us to glue ν_{N-1} and ν' together. The resulting path

$$\nu = (\ell_0, u_0) \xrightarrow{\gamma^*\sigma_1\gamma^*} (\ell_1, u_1) - \cdots \xrightarrow{\gamma^*\sigma_p\gamma^*} (\ell_p, u_p) -\!\!- \\ \xrightarrow{\gamma} (\ell_p, r_p) \xrightarrow{\sigma_{p+1}} (\ell_{p+1}, r_{p+1}) - \cdots \xrightarrow{\sigma_N} (\ell_N, r_N)$$

is a path of $\mathcal{R}^*(\mathcal{A})$ satisfying the requirements about its trace and its distance to ρ. □

As a corollary, we get:

Corollary 8. *The robust model-checking for co-Büchi acceptance conditions is* PSPACE*-complete.*

Proof (sketch).

- **PSPACE-easiness:** the extended region automaton $\mathcal{R}^*(\mathcal{A})$ does not need to be first constructed, but the co-Büchi condition can be checked on-the-fly as follows. Indeed we better checked the dual Büchi condition by checking whether there is a state f in B which is reachable and which belongs to a cycle of $\mathcal{R}^*(\mathcal{A})$. This is a classical reachability check in the region automaton, apart from the fact that we have to check when there are γ-transitions, in which case we add in one go the entire corresponding SCC. We do this non-deterministically by first guessing that we enter an SCC of $\mathcal{R}(\mathcal{A})$, and then checking that it is really an SCC.
- **PSPACE-hardness:** we do the proof by reduction to the robust model-checking of simple safety properties [12]. Let us take a timed automaton $\mathcal{A}$ with set of bad states B. We define timed automaton $\mathcal{B}$ as $\mathcal{A}$ where we add a self-loop on states B, and where the acceptance condition is a co-Büchi condition for states in B. Then it is easy to get that $\mathcal{A}$ robustly avoids B iff $\mathcal{B}$ robustly satisfies the co-Büchi condition B. □

Remark 2. We prove Theorem 4 for co-Büchi conditions, because we need those conditions for verifying LTL conditions (see below). However, we could have adapted our construction to Büchi conditions (this would require to unfold once the SCCs in $\mathcal{R}^*(\mathcal{A})$), or other standard acceptance conditions on infinite runs.

5 Robust Model-Checking of LTL

In this section we show how our results on robust model-checking of co-Büchi conditions can be used to robustly model-check LTL properties on timed automata. We use the classical construction of Büchi automata which recognize exactly the models of LTL formulae, and then apply the results of the previous section.

Definition 9 (Logic LTL). *The logic* LTL *over finite set of actions Σ is defined by the following grammar: (a ranges over the set of actions Σ)*

$$\mathsf{LTL} \ni \varphi \ ::= \ a \ \mid \ \varphi_1 \vee \varphi_2 \ \mid \ \neg\varphi \ \mid \ \mathbf{X}\,\varphi \ \mid \ \varphi_1\,\mathbf{U}\,\varphi_2$$

***Semantics of* LTL.** As explained in Section 3, we need to define the semantics of LTL over paths of timed automata. To a given path corresponds a unique trace, and since the satisfaction of a LTL formula only depends on the actions, and not on the whole path, we will define the satisfaction relation for LTL over traces and say that a path π satisfy an LTL formula if and only if its trace trace(π) satisfies this formula.

We assume we are given an infinite path π and denote by $T = (\delta_i)_{i\in\mathbb{N}} \in \Sigma^\omega$ its trace. We explained in Section 2 why w.l.o.g. we consider only infinite paths. Given a natural number $j \in \mathbb{N}$, we denote by T^j the trace $(\delta_i)_{i\geq j}$. The satisfaction relation for LTL over traces is denoted $\models$ and is defined inductively as follows:

$$\begin{array}{lll}
T \models a & \iff & \delta_0 = a \\
T \models \varphi_1 \vee \varphi_2 & \iff & T \models \varphi_1 \text{ or } T \models \varphi_2 \\
T \models \neg\varphi & \iff & T \not\models \varphi \\
T \models \mathbf{X}\,\varphi & \iff & T^1 \models \varphi \\
T \models \varphi_1\,\mathbf{U}\,\varphi_2 & \iff & \exists\, i \geq 0 \text{ s.t. } T^i \models \varphi_2 \text{ and } \forall\, 0 \leq j < i, T^j \models \varphi_1
\end{array}$$

In the following, we equivalently write $\pi \models \varphi$ for trace(π) $\models \varphi$ and use classical shortcuts like $\mathbf{F}\,\varphi$ (which holds for $\top\,\mathbf{U}\,\varphi$ where $\top$ denotes the "true" formula) or $\mathbf{G}\,\varphi$ (which holds for $\neg(\mathbf{F}\,(\neg\varphi))$.

Remark 3. It is worth noticing that the semantics we consider is the so-called *pointwise* semantics where formulae are interpreted only when an action occurs, which is quite different from the *interval-based* semantics where formulae can be interpreted at any time (see for example [21, 19] for a discussion on these semantics).

***Robust semantics of* LTL.** The *robust satisfaction relation* for LTL is thus derived from the general definition given in Section 3:

$$\mathcal{A} \models\!\!\equiv \varphi \iff \exists \Delta > 0 \text{ s.t. } \forall\pi \text{ path of } [\![\mathcal{A}]\!]_\Delta \text{ starting in } (\ell_0, v_0),\ \pi \models \varphi.$$

This definition extends the definition proposed in [11] for simple safety properties (which correspond to LTL formulae of the form $\mathbf{G}\,(\neg a)$).

***Robust model-checking of* LTL.** We first recall the following classical result on LTL:

Proposition 10 ([23, 22]). *Given an* LTL *formula* φ*, we can build a Büchi automaton* $\mathcal{B}_\varphi$ *(with initial state* q_φ *and repeated states* Q_φ*) which accepts the set* $\{T \in \Sigma^\omega \mid T \models \varphi\}$*.*

We now prove that the robust model-checking of LTL is decidable.

Theorem 11. *Given a timed automaton* $\mathcal{A}$*, and an* LTL *formula* φ*, we denote by* $\mathcal{C} = \mathcal{A} \times \mathcal{B}_{\neg\varphi}$ *the timed Büchi automaton obtained by a strong synchronization over actions of automata* $\mathcal{A}$ *and* $\mathcal{B}_{\neg\varphi}$*. We then have the following equivalence:*

$$\mathcal{A} \models\!\!\equiv \varphi \iff \mathcal{C} \models\!\!\equiv \textsf{co-Büchi}(L \times Q_{\neg\varphi}).$$

Proof. We proceed by equivalence from the negation of the first sentence:

$$\begin{aligned}
\mathcal{A} \not\models\!\!\equiv \varphi \iff & \forall\ \Delta > 0,\ \exists\ \pi \text{ path of } [\![\mathcal{A}]\!]_\Delta \text{ starting in } (\ell_0, v_0),\ \text{s.t. } \pi \not\models \varphi \\
\iff & \forall\ \Delta > 0,\ \exists\ \pi \text{ path of } [\![\mathcal{A}]\!]_\Delta \text{ starting in } (\ell_0, v_0),\ \text{s.t. } \pi \models \neg\varphi \\
\iff & \forall\ \Delta > 0,\ \exists\ \pi \text{ path of } [\![\mathcal{A}]\!]_\Delta \text{ starting in } (\ell_0, v_0),\ \text{s.t. the trace } T \\
& \text{associated to } \pi \text{ is accepted by } \mathcal{B}_{\neg\varphi} \\
\iff & \forall\ \Delta > 0,\ \exists\ \pi \text{ path of } [\![\mathcal{A}]\!]_\Delta \text{ starting in } (\ell_0, v_0),\ \text{s.t. the trace } T \\
& \text{associated to } \pi \text{ is s.t. } T \models \textsf{Büchi}(Q_{\neg\varphi}) \text{ in } \mathcal{B}_{\neg\varphi} \\
& \text{(by definition of } \mathcal{B}_{\neg\varphi} \text{ – see Proposition 10)}
\end{aligned}$$

If we note $\times$ the strong synchronization over the actions Σ, we can easily notice that the two transition systems $[\![\mathcal{A}]\!]_\Delta \times \mathcal{B}_{\neg\varphi}$ and $[\![\mathcal{A} \times \mathcal{B}_{\neg\varphi}]\!]_\Delta$ are identical. In particular, the set of accepted paths are the same. Then, we get:

$$\begin{aligned}
\mathcal{A} \not\models\!\!\equiv \varphi \iff & \forall\ \Delta > 0,\ \exists\ \pi \text{ path of } [\![\mathcal{A} \times \mathcal{B}_{\neg\varphi}]\!]_\Delta \text{ starting in } ((\ell_0, v_0), q_{\neg\varphi}),\ \text{s.t.} \\
& \text{the trace } T \text{ associated to } \pi \text{ is s.t. } T \models \textsf{Büchi}(L \times Q_{\neg\varphi}) \\
& \text{(where } L \text{ is the set of locations of } \mathcal{A}\text{)} \\
\iff & \forall\ \Delta > 0,\ \exists\ \pi \text{ path of } [\![\mathcal{C}]\!]_\Delta \text{ starting in } ((\ell_0, v_0), q_{\neg\varphi}),\ \text{s.t.} \\
& \pi \models \textsf{Büchi}(L \times Q_{\neg\varphi})
\end{aligned}$$

Taking again the negation, we get:

$$\begin{aligned}
\mathcal{A} \models\!\!\equiv \varphi \iff & \exists\ \Delta > 0,\ \text{s.t. } \forall\ \pi \text{ path of } [\![\mathcal{C}]\!]_\Delta \text{ starting in } ((\ell_0, v_0), q_{\neg\varphi}), \\
& \pi \not\models \textsf{Büchi}(L \times Q_{\neg\varphi}) \\
\iff & \exists\ \Delta > 0,\ \text{s.t. } \forall\ \pi \text{ path of } [\![\mathcal{C}]\!]_\Delta \text{ starting in } ((\ell_0, v_0), q_{\neg\varphi}), \\
& \pi \models \textsf{co-Büchi}(L \times Q_{\neg\varphi}) \\
\iff & \mathcal{C} \models\!\!\equiv \textsf{co-Büchi}(L \times Q_{\neg\varphi})
\end{aligned}$$

This concludes the proof. □

It remains to notice that the timed Büchi automaton $\mathcal{A} \times \mathcal{B}_{\neg\varphi}$ satisfies all Restrictions 1 (bounded clocks and only progress cycles) as soon as $\mathcal{A}$ does. Since we have shown in Section 4 how to robustly model-check co-Büchi properties, we get the following result:

Corollary 12. *The robust model-checking of* LTL *over timed automata is decidable and* PSPACE*-complete.*

Classically, the verification of LTL over finite structures is PSPACE-complete, but the complexity is only NLOGSPACE in the size of the system we analyze. In the case of timed automata, both standard and robust model-checking problems for LTL are PSPACE-complete, but they are PSPACE in both the size of the structure and the size of the formula.

6 Towards Robust Model-Checking of MTL

The logic MTL [16, 5] extends the logic LTL with time restrictions on "until" modalities. We present here a first positive step towards the robust model-checking of MTL formulae. We consider the following *bounded-response-time property*

$$\varphi = \mathbf{G}\ (a \rightarrow \mathbf{F}_{\leq c}\, b).$$

where a and b denote actions (elements of Σ), c belongs to $\mathbb{Q}^+$, and $\rightarrow$ denotes the classical "imply" operator. This formula expresses that event a is always followed in less than c time units by a b. This property thus constrains the reaction delays of the system. The robust satisfaction of such a property ensures that the system, even under small perturbations, will satisfy this quantitative property given by the bounded delay c.

Semantics. To formally define the satisfiability of φ over a path, we need timing informations about the path. We thus define the *time length* of a path between two actions as follows. Let consider an infinite path π:

$$(\ell_0, v_0) \xrightarrow{d_0} (\ell_0, v_0 + d_0) \xrightarrow{\delta_0} (\ell_1, v_1) \cdots (\ell_k, v_k) \xrightarrow{d_k} (\ell_k, v_k + d_k) \xrightarrow{\delta_k} \cdots$$

Given two indices $i_1 < i_2$, we define the *time length* of π between actions δ_{i_1} and δ_{i_2}, denoted by $\mathsf{time}(\delta_{i_1}, \delta_{i_2})$ by the value $\sum_{j=i_1+1}^{i_2} d_j$. We then say that path π satisfies the formula φ, denoted by $\pi \models \varphi$, whenever:

$$\forall\ i \geq 0,\ \text{if } \delta_i = a,\ \text{then } \exists\ j > i \text{ s.t. } \delta_j = b \text{ and } \mathsf{time}(\delta_i, \delta_j) \leq c.$$

Note in particular that if π satisfies φ then π also satisfies the LTL property $\mathbf{G}\ (a \rightarrow \mathbf{F}\, b)$.

Robust model-checking of bounded response time properties.

Theorem 13. *The robust model-checking of bounded-response-time properties is decidable in* PSPACE *over timed automata.*

Proof. Let $\varphi = \mathbf{G}\,(a \to \mathbf{F}_{\leq c}\, b)$. We assume $\mathcal{A}$ is a timed automaton which satisfies the untimed property $\mathbf{G}\,(a \to \mathbf{F}\, b)$. We prove the following equivalence:

$\mathcal{A} \not\models \varphi \iff$ there is a state α in $\mathsf{Reach}^*(\ell_0, v_0)$ s.t. there is a finite path in $[\![\mathcal{A}]\!]$ from α starting with an a, ending after the first b such that the time elapsed between these two actions is greater than c.

– We assume $\mathcal{A} \not\models \varphi$. For every $\Delta > 0$, there exists a path π_Δ in $[\![\mathcal{A}]\!]_\Delta$ which does not satisfy φ. Let (ℓ_Δ, v_Δ) be the configuration just before action a occurs, and let T_Δ be the trace from (ℓ_Δ, v_Δ) up to the first action b appearing along π_Δ after (ℓ_Δ, v_Δ) (the time length of the mentioned fragment of π_Δ is strictly greater than c). First note that there are only finitely many possible traces T_Δ as the system robustly satisfies the untimed property $\mathbf{G}\,(a \to \mathbf{F}\, b)$. Thus, we can first extract an infinite subsequence $(\Delta_i)_{i\geq 0}$ which converges down to 0 such that the corresponding trace is constant equal to T. Then, by compactness of the state space, there exists an infinite subsequence $(\Delta_i)_{i\geq 0}$ converging down to 0, such that the sequence $(\ell_{\Delta_i}, v_{\Delta_i})_{i\geq 0}$ converges to some state (ℓ, v). Moreover, the part of π_{Δ_i} from $(\ell_{\Delta_i}, v_{\Delta_i})$ to first action b has time length strictly greater than c. This implies the two following facts:
  - First, we get that $(\ell, v) \in \mathsf{Reach}^*(\ell_0, v_0)$. Indeed, using Theorem 32 of [12], we have that $\forall \alpha > 0$, there exists a natural number i such that for all $j \geq i$, we have $\mathrm{d}((\ell_{\Delta_j}, v_{\Delta_j}), \mathsf{Reach}^*(\ell_0, v_0)) < \alpha$. This entails $\mathrm{d}((\ell, v), \mathsf{Reach}^*(\ell_0, v_0)) = 0$ and since the set $\mathsf{Reach}^*(\ell_0, v_0)$ is closed, we get that the limit point (ℓ, v) belongs to $\mathsf{Reach}^*(\ell_0, v_0)$.
  - Moreover, we claim that there exists a real path over T in $[\![\mathcal{A}]\!]$ starting from (ℓ, v) with an action a, ending with a b, and with time length between these two actions greater than c. Indeed, first note that for every (closed) clock constraint g,

  $$(\ell_{\Delta_i}, v_{\Delta_i}) \underset{i\to\infty}{\longrightarrow} (\ell, v) \text{ and } (\ell_{\Delta_i}, v_{\Delta_i}) \models_{\Delta_i} g \text{ implies } (\ell, v) \models g.$$

  Since the trace T is of finite length, say k, there exists an extraction of the sequence $(\Delta_i)_{i\geq 0}$, which we still denote by $(\Delta_i)_{i\geq 0}$ such that along the corresponding path over T, each delay of the delay transitions converges: let t_i^j, with $1 \leq j \leq k$, be the delay associated to the j-th delay transition of the path corresponding to Δ_i, we have:

  $$\forall\ 1 \leq j \leq k,\ t_i^j \underset{i\to\infty}{\longrightarrow} t^j \tag{1}$$

  Applying the previous remark, we can build a path in $[\![\mathcal{A}]\!]$ as follows:

  $$(\ell, v) \xrightarrow{T_1 = a} \xrightarrow{t^1} \xrightarrow{T_2} \cdots \xrightarrow{t^{k-1}} \xrightarrow{T_k} \xrightarrow{t^k} \xrightarrow{b} .$$

  As claimed above, this path has a time length greater than c. Indeed, we have that for all number i, $\displaystyle\sum_{j=1}^{k} t_i^j > c$ which gives by (1), $\displaystyle\sum_{j=1}^{k} t^j \geq c$.

– Conversely, we assume that there exists a state α in $\mathsf{Reach}^*(\ell_0, v_0)$ s.t. there is a finite path in $[\![\mathcal{A}]\!]$ from α starting with an a, ending with a b such that the time elapsed between these two actions is greater than c. We note ρ this witness path. Since α is in $\mathsf{Reach}^*(\ell_0, v_0)$, for every $\Delta > 0$ there exists a path π_Δ from (ℓ_0, v_0) to α. Furthermore, for every $\Delta > 0$, we can modify path ρ by waiting Δ more time units before the b is performed (this is possible since in $[\![\mathcal{A}]\!]_\Delta$, the last guard has been enlarged by Δ). This becomes a path of $[\![\mathcal{A}]\!]_\Delta$ and we finally get a path ρ_Δ such that $\pi_\Delta.\rho_\Delta$ is a witness trace in $[\![\mathcal{A}]\!]_\Delta$ for proving that φ is not robustly satisfied.

The right hand-side of the above equivalence is decidable, one solution is to use *corner-points* because paths with maximal time length always go through corner-points [8]. Such an algorithm has a PSPACE complexity. □

Remark 4. The above proof is somehow *ad-hoc*, as it is very specific to the formula which is considered. However it can be adapted to bounded invariance properties like $\mathbf{G}\,(a \rightarrow \mathbf{G}_{\leq c} \neg b)$. Of course our hope is to extend all the constructions presented in this paper to robustly verify the so-called "Safety MTL" [19], or even full MTL. One hope is that the robust model-checking of MTL is much easier than the classical model-checking of MTL (which has been proved to be non-primitive recursive [19]). One of the reasons for this hope is that from the robust verification point of view, there is no real sense to check punctuality constraints, and relaxing punctuality leads to more efficient model checking algorithms (in EXPSPACE, see [4]).

7 Conclusion

In this paper, we have extended the results of [11] in order to decide a sufficient condition for the implementability a timed automaton w.r.t. some path property. To that aim, we have defined a notion of *robust satisfaction* and proposed model-checking algorithms to decide whether a timed automaton robustly satisfies a path formula. In [11], the robust model-checking of simple safety objectives was proved to be decidable, and a PSPACE algorithm was designed for solving that particular problem. We have provided in this paper PSPACE algorithms for the robust model-checking of Büchi-like ant LTL properties (these algorithms are b.t.w. proved to be optimal). We have also made a first step towards the robust model-checking of MTL formulae, through the verification of bounded-response-time property.

It is worth noticing that our results may extend easily to another case of perturbations: in [20], Puri considers drifts in the rates of clocks, instead of enlarging guards. In fact, both extensions happen to have the same impact on the set of reachable states [20, 11], and it seems quite natural to think that our proofs may be adapted to the case of drifts on clocks.

Furthermore, the case of the bounded-response-time property is encouraging and lets us think that it will be possible to robustly model-check the whole logic MTL. More interestingly, we also hope that perturbations will ease timed model-checking: MTL and MITL may have the same expressive power under the robust semantics, and this could provide a better complexity for the robust model-checking of MTL than for the classical one [4, 19].

Another direction to be studied is that of semantics: indeed we have pointed out that we consider in this paper a pointwise semantics for LTL. It could be interesting to study whether our results extend to the more involved interval-based semantics.

Finally, it could also be a great challenge to extend this approach to branching time properties. This requires to adapt the robust semantics, and also to bring new keys to make the link with implementability. This may lead to a robust model-checking of logics like CTL, or even TCTL.

References

1. K. Altisen and S. Tripakis. Implementation of timed automata: An issue of semantics or modeling? Technical Report TR-2005-12, Verimag, Grenoble, France, June 2005.
2. R. Alur and D. Dill. Automata for modeling real-time systems. In *Proc. 17th Int. Coll. Automata, Languages and Programming (ICALP'90)*, volume 443 of *LNCS*, pages 322–335. Springer, 1990.
3. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
4. R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.
5. R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.
6. R. Alur, S. La Torre, and P. Madhusudan. Perturbed timed automata. In *Proc. 8th Int. Work. Hybrid Systems: Computation and Control (HSCC'05)*, volume 3414 of *LNCS*, pages 70–85. Springer, 2005.
7. T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi. TIMES: A tool for schedulability analysis and code generation of real-time systems. In *Proc. 1st Int. Work. Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, volume 2791 of *LNCS*, pages 60–72. Springer, 2003.
8. G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *Proc. 4th Int. Work. Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *LNCS*, pages 147–161. Springer, 2001.
9. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. KRONOS: a model-checking tool for real-time systems. In *Proc. 10th Int. Conf. Computer Aided Verification (CAV'98)*, volume 1427 of *LNCS*, pages 546–550. Springer, 1998.
10. F. Cassez, T. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *Proc. 5th Int. Work. Hybrid Systems: Computation and Control (HSCC'02)*, volume 2289 of *LNCS*, pages 134–148. Springer, 2002.
11. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In *Proc. Joint Conf. Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, volume 3253 of *LNCS*, pages 118–133. Springer, 2004.

12. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. Technical Report 2004.30, Centre Fédéré en Vérification, Belgium, 2004.
13. M. De Wulf, L. Doyen, and J. Raskin. Almost ASAP semantics: From timed models to timed implementations. In *Proc. 7th Int. Work. Hybrid Systems: Computation and Control (HSCC'04)*, volume 2993 of *LNCS*, pages 296–310. Springer, 2004.
14. V. Gupta, T. A. Henzinger, and R. Jagadeesan. Robust timed automata. In *Proc. Int. Work. Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *LNCS*, pages 331–345. Springer, 1997.
15. T. A. Henzinger, B. Horowitz, and C. M. Kirsch. GIOTTO: A time-triggered language for embedded programming. *Proc. of the IEEE*, 91(1):84–99, 2003.
16. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
17. K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Journal of Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.
18. J. Ouaknine and J. B. Worrell. Revisiting digitization, robustness and decidability for timed automata. In *Proc. 18th Annual Symposium on Logic in Computer Science (LICS'03)*. IEEE Computer Society Press, 2003.
19. J. Ouaknine and J. B. Worrell. On the decidability of metric temporal logic. In *Proc. 19th Ann. Symp. Logic in Computer Science (LICS'05)*. IEEE Comp. Soc. Press, 2005. To appear.
20. A. Puri. Dynamical properties of timed automata. In *Proc. 5th Int. Symp. Formal techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98)*, volume 1486 of *LNCS*, pages 210–227. Springer, 1998.
21. J.-F. Raskin. *Logics, Automata and Classical Theories for Deciding Real-Time*. PhD thesis, University of Namur, Namur, Belgium, 1999.
22. M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Proc. Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer, 1996.
23. P. Wolper, M. Y. Vardi, and A. P. Sistla. Reasoning about infinite computation paths. In *Proc. 24th Ann. Symp. Foundations of Computer Science (FOCS'83)*, pages 185–194. IEEE Comp. Soc. Press, 1983.