Robust Model-Checking of Linear-Time Properties in Timed Automata*

Patricia Bouyer, Nicolas Markey, Pierre-Alain Reynier

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

Abstract. Formal verification of timed systems is well understood, but their *implementation* is still challenging. Raskin *et al.* have recently brought out a model of parameterized timed automata in which the transitions might be slightly delayed or expedited. This model is used to prove that a timed system is *implementable* with respect to a safety property, by proving that the parameterized model *robustly* satisfies the safety property. We extend here the notion of implementability to the broader class of linear-time properties, and provide PSPACE algorithms for the robust model-checking of Büchi-like and LTL properties. We also show how those algorithms can be adapted in order to verify bounded-response-time properties.

Keywords: Implementability, robust verification, timed systems.

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]) 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 [13] and KRONOS [7].

Implementation of real-time systems. Implementing mathematical models on physical machines is an important step for applying theoretical results on practical examples. This step is well understood for many untimed models that have been studied (*e.g.* finite automata, pushdown automata). In the timed setting, while timed automata are widely-accepted as a framework for modelling the real-time aspects of timed systems, it is known that they cannot be faithfully

* Work supported by ACI "Sécurité Informatique" CORTOS (Control and Observation of Real-Time Open Systems), a program of the French Ministry of research.

implemented on finite-speed CPUs (*e.g,*. the authors of [8] provide an example of a timed automaton that performs transitions exactly at dates n and $n+1/n$).

Studying the "implementability" of timed automata is thus a challenging question of obvious theoretical and practical interest.

A semantical point of view. In [10], a new semantics, called the AASAP-semantics (AASAP stands for "Almost ASAP"), has been introduced for timed automata. It takes into account the inherent digital aspect of hardware, the non-instancy of hardware communication, and several characteristics of a real processor. The point is then to decide whether a given classical controller correctly supervises the system under the AASAP-semantics. In [10], solving this problem is reduced to that of checking whether there exists a delay reaction $\varDelta$ for the controller to supervise the system: given a system Sys and a controller Cont, their interaction is denoted $[\![\mathsf{Sys} \parallel \mathsf{Cont}]\!]_\varDelta$ where $\varDelta$ 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 $\varDelta$).

The problem is then to decide, given a property $\mathcal{P}$ to be satisfied, whether there exists some $\varDelta \in \mathbb{Q}_{\geq 0}$ s.t. $[\![\mathsf{Sys} \parallel \mathsf{Cont}]\!]_\varDelta$ 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 cannot be reached) has been solved in [9] 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 (*e.g.*, that "something occurs infinitely often", or that "a request is eventually granted"). The algorithm we propose is based on an extension of the classical region automaton construction which roughly captures all behaviors of the system, even those which may deviate due to constraint enlargement. Our algorithm is in PSPACE, which appears to be 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 another modeling-based solution [1], where the behavior of the platform is modeled 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 [11, 14], 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 approach to robustness. Finally, in [16, 4, 9], a small perturbation on slopes of clocks is allowed. In the case of safety properties and under some natural assumptions, this approach is equivalent to constraint enlargement, as proved in [9].

Outline of the paper. In Section 2, we introduce basic definitions, we define the problem of robust model-checking, and we make clear the link between our work and the results of [10, 9]. Then, we provide in Section 3 our model-checking algorithm for co-Büchi properties, and in Section 4 its application to LTL properties. Finally, we present in Section 5 our first results for timed properties, and conclude with a landscape on possible future works.

Only sketches of the proofs are done in this paper. The full proofs are available in the associated technical report [6].

2 Definitions

2.1 Timed Automata

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 $\mathbb{N}$ and [1] $\sim \in \{\leq, \geq\}$.

A *timed automaton* is a 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. We assume w.l.o.g. 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 the subscript Δ.

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 *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 an *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 $c \in \mathcal{C}$.

[1] We simplify the notations by assuming that all inequalities are non-strict. As argued in [9], this does not change the expressive power of the model under the enlarged semantics.

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 consecutive transitions $(\delta_i)_{i \in I}$.

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 $\mathsf{trace}(\pi)$.

Let $T = (\delta_i)_{i \in I}$ be a trace of $\mathcal{A}$. A state (ℓ', v') is said to be *reachable* from a set of states S following T in $[\![\mathcal{A}]\!]_\Delta$ if there exists a path over T in $[\![\mathcal{A}]\!]_\Delta$ starting in some $(\ell, v) \in S$ and containing (ℓ', v'). We write $\mathsf{Reach}^T_\Delta(S)$ for the set of states that are reachable from S following trace T. We note $\mathsf{Reach}_\Delta(S)$ for the union over all possible traces T of $\mathsf{Reach}^T_\Delta(S)$. This set represents all states that are reachable in $[\![\mathcal{A}]\!]_\Delta$ from S.

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

- for all $c \in \mathcal{C}$, either $v(c)$ 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 the 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 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 $(\ell, v) \xrightarrow{\sigma} (\ell', v')$ 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. $(\ell, v) \xrightarrow{t} (\ell', v')$ 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 [2] for more details.

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

2.2 Robust Verification of Linear-Time Properties

In this section, after several remarks on the implementability of timed systems, we present the problem of robust verification for linear-time properties.

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 [10], 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 behaves correctly (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 model-checking. 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* $\equiv\!\!\!\!|$ 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 [10],

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

This result holds for properties quantifying universally over paths, and thus holds for LTL properties, but not for CTL properties.

In the sequel we address the *robust model-checking problem*: "given a timed automaton $\mathcal{A}$ and a path property $\mathcal{P}$, can we decide whether $\mathcal{A} \models\!\!\!\equiv \mathcal{P}$?" This problem has been solved in [9] for basic safety properties of the type "avoid bad states", with several restrictions on timed automata.

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 hypotheses on timed automata:

Restriction 1 *We assume timed automata $\mathcal{A}$ satisfy 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 bounded timed automata are as expressive as standard timed automata (see for example [5]). Note that it entails that any time-divergent path contains infinitely many action transitions. In the following we will only consider such infinite time-divergent paths. The second point is a classical restriction [16], and in the framework of bounded timed automata, it is less restrictive than classical strong non-*Zenoness* assumptions.

Robust model-checking of safety properties. The following result has then been proved in [9]: let $\mathcal{A}$ be a timed automaton (satisfying Restriction 1) with initial state (ℓ_0, v_0), let Bad be a set of bad locations of $\mathcal{A}$, and define the set $\mathsf{Reach}^*(S) = \bigcap_{\Delta>0} \mathsf{Reach}_\Delta(S)$, where S denotes a set of states, then:

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

These results rely on the classical region automaton construction where a strongly connected component (SCC for short) of the region automaton is added to the set $\mathsf{Reach}^*(\ell_0, v_0)$ as soon as it can be reached: indeed, if an SCC can be partly reached, then by iterating the SCC, all points of the SCC can also be reached.

Example 1 ([16, 9]). Consider the automaton depicted on Figure 1. For this automaton, 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.

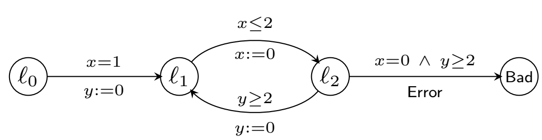

Fig. 1. A timed automaton $\mathcal{A}$.

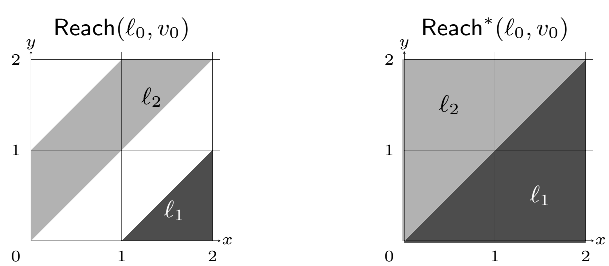

Fig. 2. Differences between $\mathsf{Reach}(\ell_0, v_0)$ and $\mathsf{Reach}^*(\ell_0, v_0)$.

In the next sections, we solve the robust model-checking for co-Büchi, LTL, and bounded-response-time properties.

3 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 2.2, this immediately defines the notion of robust satisfaction for a co-Büchi condition in a timed automaton. We also recall the notion of satisfying a co-Büchi condition for 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 often.

Extended region automaton $\mathcal{R}^*$. We build 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 $\mathcal{R}(\mathcal{A})$

The γ-transitions which are added to the classical region automaton indicate that an SCC can be reached and iterated, and then, as already written in Subsection 2.2, all configurations along the SCC can be reached.

Decidability of the robust model-checking for co-Büchi conditions. The following result is the main result of this paper. The extension from simple reachability to repeated reachability is not trivial since the method used in [9], based on the distance between new reachable states, is not sufficient in our context. Instead, we prove that the extended region automaton roughly recognizes all paths of the system, even those which deviate from standard semantics.

Theorem 2. *Let $\mathcal{A}$ be a timed automaton and B a set of locations of $\mathcal{A}$. Then*

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

Proof (Sketch). We first prove the left-to-right implication by contradiction. Assume that $\mathcal{A} \models\!\!\!\equiv \mathsf{co\text{-}Büchi}(B)$, and that $\mathcal{R}^*(\mathcal{A}) \not\models \mathsf{co\text{-}Büchi}(B)$. We can thus pick some $\Delta > 0$ s.t. every path of $[\![\mathcal{A}]\!]_\Delta$ starting in (ℓ_0, v_0) satisfies the co-Büchi condition, and pick a path π in $\mathcal{R}^*(\mathcal{A})$ not satisfying the co-Büchi condition. We will build from π a path in $[\![\mathcal{A}]\!]_\Delta$ not satisfying the co-Büchi condition, and thus obtain a contradiction. To this aim we state the following Lemma:

Lemma 3. *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'').*

Splitting the path π into subpaths not containing γ-transitions, we can apply the first point of the above lemma to each subpath. We thus obtain real paths in $[\![\mathcal{A}]\!]_\Delta$, which we can glue together using the second point.

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 location $f \in B$ that witnesses the Büchi condition for paths π_Δ for arbitrarily small Δ. We will build a path of $\mathcal{R}^*(\mathcal{A})$ satisfying the Büchi condition $\{f\}$, using the following lemma:

Lemma 4. *Given a timed automaton $\mathcal{A}$, there exists a positive value Δ s.t. for any (finite) path ρ in $[\![\mathcal{A}]\!]_\Delta$, there exists a path in $\mathcal{R}^*(\mathcal{A})$ whose trace, when removing γ-transitions, is the same as the trace of ρ.*

The proof is done by induction on the length of ρ. Using this lemma, we can fix such a value of Δ and apply it to any prefix of the corresponding path π, which satisfies the Büchi condition $\{f\}$. We can take a prefix of π containing $k+1$ times the discrete state f, which leads to a path of $\mathcal{R}^*(\mathcal{A})$ satisfying the Büchi condition $\{f\}$. $\square$

As a corollary, and using the PSPACE-hardness of the robust model-checking of safety properties [9], we get:

Corollary 5. *The robust model-checking for co-Büchi acceptance conditions is* PSPACE*-complete.*

Remark 1. We prove Theorem 2 for co-Büchi conditions, because we need those conditions for verifying LTL properties (see Section 4). 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.

4 Robust Model-Checking of LTL

We now 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 6 (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)*

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

***Semantics of* LTL.** We define the semantics of LTL over traces of timed automata, which naturally induces a semantics over paths of timed automata: a path π will satisfy an LTL formula if and only if its trace $\mathsf{trace}(\pi)$ satisfies this formula. We thus assume we are given an infinite path π and denote by $T = (\delta_i)_{i\in\mathbb{N}} \in \Sigma^\omega$ its trace. 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 (we omit the semantics of standard boolean operators):

$$
\begin{array}{lcl}
T \models a & \iff & \delta_0 = a \\
T \models \mathbf{X}\,\varphi & \iff & T^1 \models \varphi \\
T \models \varphi_1 \,\mathbf{U}\, \varphi_2 & \iff & \exists\, i \geq 0 \text{ s.t. } T^i \models \varphi_2 \text{ and } \forall\, 0 \leq j < i, T^j \models \varphi_1
\end{array}
$$

In the following, we equivalently write $\pi \models \varphi$ for $\mathsf{trace}(\pi) \models \varphi$ 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 2. 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 [17, 15] for a discussion on these semantics).

***Robust model-checking of* LTL.** The *robust satisfaction relation* for LTL is thus derived from the general definition given in Section 2.2:

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

We recall the following classical result on LTL:

Proposition 7 ([18]). *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 state that the robust model-checking of LTL is decidable.

Theorem 8. *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:*

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

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 3 how to robustly model-check co-Büchi properties, we get the following result:

Corollary 9. *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 $\mathsf{NLOGSPACE}$ 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.

5 Towards Robust Model-Checking of Timed Properties

The logic MTL [12, 3] 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* $\varphi = \mathbf{G}\,(a \to \mathbf{F}_{\leq c}\, b)$. where a and b denote actions (elements of Σ), c belongs to $\mathbb{Q}^+$, and $\to$ 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 (defined as in Subsection 2.2) ensures that the system, even under small perturbations, will satisfy this quantitative property given by the bounded delay c.

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:

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

In particular, if π satisfies φ then π also satisfies the LTL property $\mathbf{G}\,(a \to \mathbf{F}\, b)$.

We now state the following result:

Theorem 10. *The robust model-checking of bounded-response-time properties is decidable in* PSPACE *over timed automata.*

Proof (Sketch). Let $\varphi = \mathbf{G}\,(a \to \mathbf{F}_{\leq c}\, b)$. We assume $\mathcal{A}$ is a timed automaton which satisfies the untimed property $\mathbf{G}\,(a \to \mathbf{F}\, b)$. The proof is based on 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.

The right hand-side of the above equivalence is decidable, one solution is to use *corner-points* because paths with maximal time length always run through corner-points [5]. Such an algorithm has a PSPACE complexity. $\square$

Remark 3. The above proof is somehow *ad-hoc*, as it is very specific to the formula which is considered. However it can for example be adapted to bounded-invariance properties like $\mathbf{G}\,(a \rightarrow \mathbf{G}_{\leq c} \neg b)$.

6 Conclusion

In this paper, we have extended the results of [9] in order to decide a sufficient condition for the implementability of a timed automaton. To that aim, we have defined a notion of *robust satisfaction* for linear-time properties and provided PSPACE algorithms for the robust model-checking of Büchi-like and LTL properties (these algorithms are b.t.w. 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 [16], 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 [16, 9], and it seems quite natural to think that our proofs may be adapted to the case of drifts on clocks. Furthermore, the case of bounded-response-time properties is encouraging and we are trying to extend it to more general timed properties. 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 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? In *Proc. 3rd Int. Work. Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *LNCS*, pages 273–288. Springer, 2005.
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
3. R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.
4. 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.
5. 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.
6. P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of timed automata. Tech. Report LSV-05-08, LSV, ENS Cachan, France, 2005.

7. 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.
8. 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.
9. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. Tech. Report 2004.30, Centre Fédéré en Vérification, Belgium, Dec. 2005. Revised version.
10. M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. *Formal Aspects of Computing*, 17(3):319–341, 2005.
11. 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.
12. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
13. 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.
14. J. Ouaknine and J. B. Worrell. Revisiting digitization, robustness and decidability for timed automata. In *Proc. 18th Ann. Symp. Logic in Computer Science (LICS'03)*, pages 198–207. IEEE Comp. Soc. Press, 2003.
15. J. Ouaknine and J. B. Worrell. On the decidability of metric temporal logic. In *Proc. 19th Ann. Symp. Logic in Computer Science (LICS'05)*, pages 188–197. IEEE Comp. Soc. Press, 2005.
16. 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.
17. J.-F. Raskin. *Logics, Automata and Classical Theories for Deciding Real-Time.* PhD thesis, University of Namur, Namur, Belgium, 1999.
18. 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.