Model-Checking for Weighted Timed Automata[⋆]

Thomas Brihaye[1], Véronique Bruyère[1], and Jean-François Raskin[2]

[1] Institut d'Informatique, Université de Mons-Hainaut,
Avenue du Champ de Mars 6, B-7000 Mons, Belgium

[2] Département d'Informatique, Université Libre de Bruxelles,
Boulevard du Triomphe CP 212, B-1050-Bruxelles, Belgium

Abstract. We study the model-checking problem for weighted timed automata and the weighted CTL logic by the bisimulation approach. Weighted timed automata are timed automata extended with costs on both edges and locations. When the costs act as stopwatches, we get stopwatch automata with the restriction that the stopwatches cannot be reset nor tested. The weighted CTL logic is an extension of TCTL that allow to reset and test the cost variables. Our main results are *(i)* the undecidability of the proposed model-checking problem for discrete and dense time, *(ii)* its PSpace-Completeness in the discrete case for a slight restriction of the logic, *(iii)* the precise frontier between finite and infinite bisimulations in the dense case for the subclass of stopwatch automata.

1 Introduction

During the last decade, hybrid automata have been widely studied and especially the reachability problem for hybrid automata. In this article, we study a model-checking problem for a particular class of hybrid automata. Our motivation is the important open problem of model-checking timed automata extended with stopwatches used as observers [1].

We consider the model of *weighted timed automata*, which is an extension of timed automata with *tuples of costs* on both edges and locations. This model has been independently introduced in [6] and [7] (with single costs instead of tuples of costs).

The properties of weighted timed automata that we want to check are formalized by formulas of the *weighted CTL logic*, WCTL for short. This logic is close to the DTL logic of [8] and the ICTL logic of [2].

Our approach is a systematic study of the tool *bisimulation* as done in the works [10] and [11]. Indeed when the transition system of an hybrid automaton has a finite bisimulation that can be constructed effectively, the reachability problem and the model-checking problem are decidable. For instance this technique has been successfully applied to timed automata thanks to the region graph. However the converse does not hold in general.

[⋆] Supported by the FRFC project "Centre Fédéré en Vérification" funded by the Belgian National Science Foundation (FNRS) under grant nr 2.4530.02

Related works. There are few results on the model-checking of hybrid automata. Indeed the wide study of the particular case of the reachability problem has identified a frontier between decidability and undecidability. Among the numerous results about this problem, let us mention the following ones. The important class of *initialized rectangular automata* has a decidable reachability problem; however several slight generalizations of these automata lead to an undecidable reachability problem, in particular for timed automata augmented with one stopwatch [14]. The reachability problem is already undecidable for the simple class of *constant slope hybrid systems* which are timed automata augmented with integrators; the reachability problem becomes decidable when the integrators are used as *observers* (they are neither reset nor tested) [15]. The latter case has also been studied in [1]. Of course the well-known class of timed automata has a decidable reachability problem [5]. Recently the *minimum-cost* reachability problem has been introduced, that is, determine the minimum cost of runs of a weighted timed automaton from an initial location to a target location. This problem has been proved decidable independently in [6] and [7].

Concerning the model-checking problem of hybrid systems, let us mention two references. In [2], a model-checking procedure and its implementation in the HyTech tool are proposed for linear hybrid automata and the ICTL logic. This procedure is not guaranteed to terminate. In [8], the model-checking problem is proved to be decidable for some fragments of the DTL logic and a restrictive class of weighted timed automata.

Our contribution. In this paper, we investigate the WCTL model-checking problem for weighted timed automata. The weighted timed automata can be seen as constant slope hybrid systems where the integrators are used as observers and the edges have been enriched with costs. We have chosen this class of hybrid automata since they have a decidable reachability problem, even in the case of minimum cost. We also focus on the subclass of *automata with stopwatch observers*, which are weighted timed automata such that every integrator is a stopwatch. The WCTL logic is similar to the ICTL logic. Formulas allow the two actions forbidden in the weighted timed automata : to reset integrators and to test them. This logic is a natural extension of the TCTL logic to formulate properties about integrators instead of the total elapsed time.

Our first result is the *undecidability* of the model-checking problem. This proves that there are situations where the model-checking procedure of [2] will never terminate, even for classes of hybrid automata with a decidable reachability problem. What is surprising is that the undecidability holds even for the *discrete* time, a case where positive results usually happen. The proof is based on the halting problem for 2-counter machines, with its reduction distributed to *both* a weighted timed automaton and a WCTL formula. To the best of our knowledge, this approach is new[1]. This proof works for automata with stopwatch observers equipped with 1 clock and 3 integrators and for WCTL formulas where two integrators are compared.

[1] with the exception of reference [9] where we have followed the same approach.

In the sequel of the paper, we limit our study to the WCTL$_r$ logic, that is, WCTL where integrators can only be compared with *constants*. One way to prove that the model-checking problem is decidable is the effective construction of a finite bisimulation for weighted timed automata. This is the approach already proposed in [10] and [11]. The effectiveness is always guaranteed as our automata are particular linear hybrid automata. It should be noted that the existence of a finite bisimulation is sufficient but not necessary for decidability of the model-checking problem.

For *discrete* time, when working with the WCTL$_r$ logic, we show that the bisimulations are always finite. It follows that the WCTL$_r$ model checking problem for weighted timed automata is PSpace-Complete.

However for *dense* time, the panorama completely changes. In this case, we identify the *precise frontier* between finite and infinite bisimulations for automata with *stopwatch observers*. Our results are the following. There exist automata with stopwatch observers that have no finite bisimulations already with 2 clocks and 1 integrator, or with 1 clock and 2 integrators. This is no longer true with 1 clock and 1 integrator. It was a difficult task to find automata with stopwatch observers with a small number of clocks and integrators for which no finite bisimulation exists; our proofs are involved. The reason is that stopwatches cannot be reset nor tested in these automata.

2 Weighted Timed Automata

In this section, we introduce the notion of weighted timed automaton, which is an extension of timed automata with costs on both locations and edges. We begin with the usual notations on timed automata.

Notations. Let $X = \{x_1, \ldots, x_n\}$ be a set of n clocks. The same notation $x = (x_1, \ldots, x_n)$ is used for the clock *variables* and for an *assignment* of values to these variables. Depending on whether the time is *dense* or *discrete*, the values are taken in domain $\mathbb{T}$ equal to the set $\mathbb{R}^+$ of nonnegative reals or to the set $\mathbb{N}$ of natural numbers. Given a clock assignment x and $\tau \in \mathbb{T}$, $x + \tau$ is the clock assignment $(x_1 + \tau, \ldots, x_n + \tau)$. The set $\mathcal{G}$ denotes the set of *guards* which are finite conjunctions of atomic guards of the form $x_i \sim c$ where x_i is a clock, $c \in \mathbb{N}$ is an integer constant, and $\sim$ is one of the symbols $\{<, \leq, =, >, \geq\}$. Notation $x \models g$ means that the clock assignment x satisfies the guard g. A *reset* $r \in 2^X$ indicates which clocks are reset to 0, that is, $x' = [x_i := 0]_{x_i \in r} x$. We use notation Σ for the set of *atomic propositions*.

Definition 1. *A* weighted timed automaton $\mathcal{A} = (L, E, \mathcal{I}, \mathcal{L}, \mathcal{C})$ *has the following components: (i) L is a finite set of* locations*, (ii) $E \subseteq L \times \mathcal{G} \times \mathcal{P}(X) \times L$ is a finite set of* edges*, (iii) $\mathcal{I} : L \rightarrow \mathcal{G}$ assigns an* invariant *to each location, (iv) $\mathcal{L} : L \rightarrow 2^\Sigma$ is the* labeling *function and (v) $\mathcal{C} : L \cup E \rightarrow \mathbb{N}^m$ assigns a m-uple of* costs *to both locations and edges.*
An automaton with stopwatch observers *is a weighted timed automaton such that for every location l, $\mathcal{C}(l) \in \{0, 1\}^m$ (instead of $\mathbb{N}^m$).*

The concept of weighted timed automata has been independently introduced in [6] and [7] (with single costs instead of m-uples of costs). In the previous definition, we say that $\mathcal{C}(l)$ (resp. $\mathcal{C}(e)$) is the cost of location l (resp. edge e). We will sometimes use the notation $\dot{z}_1 = d_1, \ldots, \dot{z}_m = d_m$ at location l instead of $\mathcal{C}(l) = (d_1, \ldots, d_m)$; the variables $z = (z_1, \ldots, z_m)$ are called *cost variables*[2]. Note that the variables $z_1, \ldots, z_m$ cannot be reset nor tested in weighted timed automata, they are just *observers.* When an edge e or a location l has null costs, that is, $\mathcal{C}(e) = (0, \ldots, 0)$ or $\mathcal{C}(l) = (0, \ldots, 0)$, we say that it has *no cost.* When an edge has no cost, nor reset and a guard that is always true, it is called an *empty* edge.

Definition 2. *The* semantics *of a weighted timed automaton $\mathcal{A}$ is defined as a* transition system $T_{\mathcal{A}} = (Q, \rightarrow)$ *with a set of* states Q *equal to* $\{(l, x, z) \mid l \in L, x \in \mathbb{T}^n, x \models \mathcal{I}(l), z \in \mathbb{T}^m\}$ *and a* transition relation $\rightarrow \; = \; \bigcup_{\tau \in \mathbb{T}} \xrightarrow{\tau}$ *defined as follows*

$$(l, x, z) \xrightarrow{\tau} (l', x', z')$$

- *case* $\tau > 0$ *(*elapse of time *at location l) :* $l = l'$, $x' = x + \tau$ *and* $z' = z + \mathcal{C}(l) \cdot \tau$,
- *case* $\tau = 0$ *(*instantaneous switch*) :* $(l, g, r, l') \in E$, $x \models g$, $x' = [x_i := 0]_{x_i \in r} x$ *and* $z' = z + \mathcal{C}(e)$.

In the previous definition, note that the value of τ (strictly positive, or null) indicates an elapse of time or an instantaneous switch. The m-tuple z of a state (l, x, z) indicates global *costs* that accumulate the individual costs described by the function $\mathcal{C}$: either the cost rate of staying in a location (per time unit), or the cost of an edge. A transition $(l, x, z) \xrightarrow{\tau} (l', x', z')$ is shortly denoted by $q \rightarrow q'$ (given q and q', it is easy to compute the unique τ such that $q \xrightarrow{\tau} q'$). When $\tau > 0$, we also shortly denote by $q + \tau$ the state q' of the transition $q \xrightarrow{\tau} q'$.

Definition 3. *Given a transition system $T_{\mathcal{A}}$, a* run $\rho = (q_i)_{i \geq 0}$ *is an infinite path in $T_{\mathcal{A}}$*

$$\rho = q_0 \xrightarrow{\tau_0} q_1 \xrightarrow{\tau_1} q_2 \cdots q_i \xrightarrow{\tau_i} q_{i+1} \cdots$$

such that $\Sigma_{i \geq 0} \tau_i = \infty$ (divergence of time). A finite run $\rho = (q_i)_{0 \leq i \leq j}$ *is any finite path in $T_{\mathcal{A}}$. A* position *in ρ is any state q_i or $q_i + \tau$ with $0 < \tau < \tau_i$. The set of positions in ρ can be totally ordered.*

We illustrate the definitions with the classical example of the gas burner system.

Example 1. The weighted timed automaton of Figure 1 represents a gas burner system with two locations l and l', one where the system is leaking and the other where it is not leaking. There is 1 clock variable x to express that a continuous leaking period cannot exceed 1 time unit and two consecutive leaking periods are separated by at least 30 time units. There are 3 costs variables z_1, z_2, z_3 such that z_1 describes the total elapsed time, z_2 the accumulated leaking time and z_3 the number of leaks.

[2] This notation comes from automata with integrators, the variables $z_1, \ldots, z_m$ being the integrators, see for instance [15].

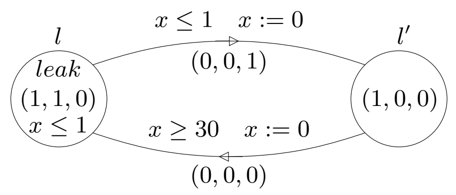


Fig. 1. The gas burner system.

3 Weighted CTL Logic and Model-Checking

In this section, we introduce the weighted CTL logic, WCTL logic for short (close to the ICTL logic of [2]). Two logics, discrete and dense, are proposed according to discrete or dense time.

Notations. As done previously for clocks, the same notation $z = (z_1, \dots, z_m)$ is used for the cost variables and for an assignment of values to these variables. A *cost constraint* π is of the form $z_i \sim c$ or $z_i - z_j \sim c$ where z_i, z_j are cost variables and $c \in \mathbb{N}$ is an integer constant. Notation $z \models \pi$ means that the cost assignment z satisfies the cost constraint π. Notation σ means any atomic proposition $\sigma \in \Sigma$.

Definition 4. *The* syntax *of the* discrete *WCTL logic is given by the following grammar*

$$\varphi ::= \sigma \mid \pi \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists\bigcirc\, \varphi \mid \varphi\exists U\varphi \mid \varphi\forall U\varphi \mid z_i \cdot \varphi$$

Dense *WCTL formulae are defined in the same way, except that operator* $\exists\bigcirc$ *is forbidden.*

The WCTL logic uses freeze quantifiers "$z_i\ \cdot$" on the cost variables z_i, $1 \leq i \leq m$. This logic allows to reset such variables and to test them. These actions are forbidden in weighted timed automata, where the cost variables are only observers. Note that the TCTL logic [4] is a particular case of WCTL when each cost variable z_i describes the total elapsed time.

We restrict ourselves to *closed* WCTL formulas, i.e. formulas φ such that every occurrence of a cost variable z_i in φ is bound by a freeze quantifier. We also impose that different freeze quantifiers bind different cost variables, i.e. two occurrences of the freeze quantifier $z_i\cdot$ are forbidden in the same formula. For convenience, we use the following abbreviations: $\exists\Diamond\varphi \equiv \top\exists U\varphi$, $\forall\Diamond\varphi \equiv \top\forall U\varphi$, $\exists\Box\varphi \equiv \neg\forall\Diamond\neg\varphi$, and $\forall\Box\varphi \equiv \neg\exists\Diamond\neg\varphi$.

We now give the semantics of WCTL.

Definition 5. *Suppose* $\mathbb{T} = \mathbb{N}$. *Let* $\mathcal{A}$ *be a weighted timed automaton and* $q = (l, x, z)$ *be a state of the transition system* $T_\mathcal{A}$ *of* $\mathcal{A}$. *Let* φ *be a discrete WCTL formula. Then the* satisfaction *relation* $q \models \varphi$ *is defined inductively as indicated below. In case* $\mathbb{T} = \mathbb{R}^+$ *and* φ *is a dense WCTL formula, the satisfaction relation is defined in the same way, except that* $q \models \exists\bigcirc\,\varphi$ *does not exist.*

- $q \models \sigma$ *iff* $\sigma \in \mathcal{L}(l)$;
- $q \models \pi$ *iff* $z \models \pi$;
- $q \models \neg\varphi$ *iff* $q \not\models \varphi$;
- $q \models \varphi \vee \psi$ *iff* $q \models \varphi$ *or* $q \models \psi$;
- $q \models \exists\bigcirc\, \varphi$ *iff there exists a run* $\rho = (q_i)_{i\geq 0}$ *in* $T_{\mathcal{A}}$ *with* $q = q_0$ *and* $q_0 \xrightarrow{\tau} q_1$ *satisfying* $\tau = 0$ *or* $\tau = 1$, *such that* $q_1 \models \varphi$;
- $q \models \varphi \exists U \psi$ *iff there exists a run* $\rho = (q_i)_{i\geq 0}$ *in* $T_{\mathcal{A}}$ *with* $q = q_0$, *there exists a position* p *in* ρ *such that* $p \models \psi$ *and* $p' \models \varphi$ *for all* $p' < p$;
- $q \models \varphi \forall U \psi$ *iff for any run* $\rho = (q_i)_{i\geq 0}$ *in* $T_{\mathcal{A}}$ *with* $q = q_0$, *there exists a position* p *in* ρ *such that* $p \models \psi$ *and* $p' \models \varphi$ *for all* $p' < p$;
- $q \models z_i \cdot \varphi$ *iff* $(l, x, [z_i := 0]z) \models \varphi$.

Let us come back to the gas burner system of Example 1 and formalize some properties by WCTL formulas.

Example 2. Consider the first property "there exists a run with an average leaking time always bounded by 0.5" (in other words, $2z_2 \leq z_3$). Since the cost constraints π allowed in WCTL are of the form $z_i \sim c$ or $z_i - z_j \sim c$, we replace the cost $\mathcal{C}(l) = (1,1,0)$ by $(1,2,0)$ in the automaton of Figure 1. The WCTL formula for the given property is therefore

$$z_2 \cdot z_3 \cdot (\exists\Box z_2 \leq z_3).$$

The next property we want to formalize is "in any time interval longer than 60 time units, the accumulated leaking time is at most 5% of the interval length" (that is, $z_1 \geq 60 \Rightarrow 20z_2 \leq z_1$). Again we have to modify the automaton by replacing $\mathcal{C}(l)$ by $(1,20,0)$. The related WCTL formula is

$$z_1 \cdot z_2 \cdot (\forall\Box(z_1 \geq 60 \Rightarrow z_2 \leq z_1)).$$

Finally, the property "there exists a run such that the accumulated leaking time is at most 5% of the time interval length and the average leaking time is bounded by 0.5, until the system never leaks" is formalized as

$$z_1 \cdot z_2 \cdot z_3 \cdot ((z_2 \leq z_1 \wedge z_2 \leq z_3)\ \exists\mathrm{U}\ (\forall\Box\neg leak))$$

if $\mathcal{C}(l)$ is replaced by $(1,20,0)$ and $\mathcal{C}(l, x \leq 1, x := 0, l')$ by $(0,0,10)$.

The problem that we want to study in this article is the following *model-checking* problem, for discrete and dense time.

Problem 1. Given a weighted timed automaton $\mathcal{A}$ and a state q of $T_{\mathcal{A}}$, given a WCTL formula φ, does $q \models \varphi$ hold ? ($\mathbb{T} = \mathbb{N}$ or $\mathbb{T} = \mathbb{R}^+$)

The next theorem states that this problem is undecidable, already for automata with stopwatch observers.

Theorem 1. *In both cases of discrete and dense time, the WCTL model-checking problem for automata with stopwatch observers is undecidable.*

Corollary 1. *Problem 1 is undecidable.*

Proof. (of Theorem 1) The proof is based on a reduction of the halting problem for a 2-counter machine. We recall that a machine with 2 counters C_1 and C_2 can be described by a linear labeled program allowing the following basic instructions:

- k : **goto** k' ;
- k : **if** $C_i > 0$ **then goto** k' **else goto** k'' ;
- k : $C_i := C_i + 1$;
- k : $C_i := C_i - 1$ (this operation is not defined if $C_i = 0$) ;
- k : **stop** .

The emulation of the 2-counter machine is done partly by an automaton with stopwatch observers $\mathcal{A}$ and partly by a WCTL formula φ. Suppose that the first label of the program is k_0 and the last instruction is a **stop** instruction labeled by k_t. The 2 counters are encoded by 3 cost variables as follows

$$C_1 = z_1 - z_2, \quad C_2 = z_1 - z_3.$$

The automaton $\mathcal{A} = (L, E, \mathcal{I}, \mathcal{L}, \mathcal{C})$ has 1 clock x and no cost on its edges. The set Σ of atomic propositions labeling L contains an atomic proposition σ_k for each label k of the program and 4 additional atomic propositions ρ_1, ρ'_1, ρ_2 and ρ'_2. The set L contains a location for each label k of the program, which is labeled by σ_k; it contains additional locations.

The **goto** and **stop** instructions are easily encoded in $\mathcal{A}$. The instruction for incrementing counter C_1 is encoded by the subautomaton given in Figure 2. The

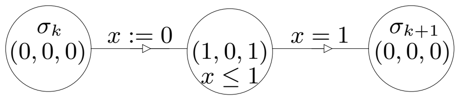


Fig. 2. Incrementing counter C_1.

subautomaton for incrementing C_2 is similar except that the cost of the central state is $(1, 1, 0)$.

The instruction for decrementing counter C_1 is encoded in Figure 3. The atomic proposition ρ_1 is a witness that $C_1 > 0$ while ρ'_1 is a witness that $C_1 = 0$. Since the automaton $\mathcal{A}$ is not allowed to test its cost variables, the formula φ will check if $C_1 = 0$ or $C_1 > 0$ depending on the values of z_1 and z_2. A similar subautomaton is given for counter C_2 with atomic propositions ρ_2 and ρ'_2.

The **if** instruction is encoded similarly to the decrementation instruction, see Figure 4. Again φ will check which case occurs.

Let us now give formula φ :

$$\sigma_{k_0} \ \wedge \ z_1 \cdot z_2 \cdot z_3 \cdot \left(\begin{pmatrix} \rho_1 \Rightarrow z_1 - z_2 > 0 \wedge \rho'_1 \Rightarrow z_1 - z_2 = 0 \\ \wedge\, \rho_2 \Rightarrow z_1 - z_3 > 0 \wedge \rho'_2 \Rightarrow z_1 - z_3 = 0 \end{pmatrix} \exists \mathrm{U} \ \sigma_{k_t} \right).$$

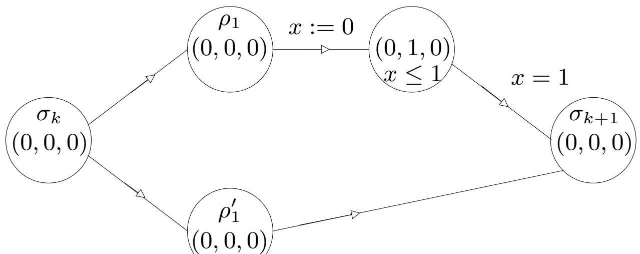


Fig. 3. Decrementing counter C_1.

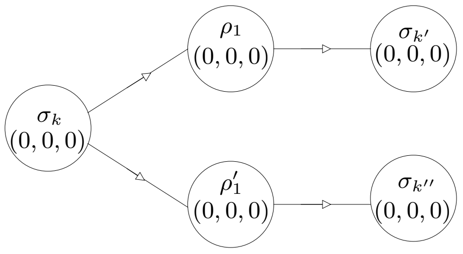


Fig. 4. If instruction with test on C_1.

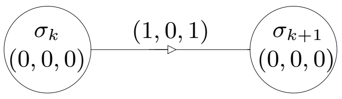


Fig. 5. Incrementing counter C_1 with no cost in the locations.

Clearly, the 2-counter machine halts on the instruction **stop** labeled by k_t iff $q \models \varphi$ for the state q equal to $(l_0, 0, 0, 0, 0)$ where l_0 is the location labeled by k_0. It follows that the model-checking problem is undecidable. $\square$

Comments. The previous proof works for discrete or dense time. The automaton $\mathcal{A}$ is an automaton with stopwatch observers using 1 clock x and 3 cost variables z_1, z_2, z_3. All its edges have no cost. The formula φ uses cost constraints of the form $z_i - z_j \sim 0$.

The proof can be easily adapted if one prefers an automaton with all its locations having no cost. In this case, $\mathcal{A}$ has no clock and again 3 cost variables. In Figure 5 an incrementation of counter C_1 is depicted. The formula φ remains identical. One can imagine a third proof with 1 clock and 3 cost variables, as a mix of both previous approaches, such that there exist non null costs on certain locations and on certain edges.

In the sequel of the article, we will work with the WCTL logic restricted to cost constraints π of the form $z_i \sim c$. It is denoted WCTL$_r$. The related model-checking problem is the following one.

Problem 2. Given a weighted timed automaton $\mathcal{A}$ and a state q of $T_{\mathcal{A}}$, given a WCTL$_r$ formula φ, does $q \models \varphi$ hold ? ($\mathbb{T} = \mathbb{N}$ or $\mathbb{T} = \mathbb{R}^+$)

Example 3. For the gas burner system of Example 1, the property "if the number of leaks is less than 5, then the leaking time is strictly bounded by 5" is formalized in WCTL$_r$ by the next formula

$$z_2 \cdot z_3 \cdot \forall\Box(z_3 < 5 \Rightarrow z_2 < 5).$$

The next property "at each position of every run, the number of leaks does not exceed 2 in any time interval less than 100 time units" is formalized by

$$\forall\Box(z_1 \cdot z_3 \cdot \forall\Box(z_1 \leq 100 \Rightarrow z_3 \leq 2)).$$

4 Bisimulations

In the sequel of the article, we want to study Problem 2 via bisimulations. We recall in this section useful notions on time abstracting bisimulations (see [10] or [3]).

Definition 6. *Let $\mathcal{A}$ be a weighted timed automaton and $T_\mathcal{A} = (Q, \rightarrow)$ its transition system. A* bisimulation *of $\mathcal{A}$ is an equivalence relation $\approx \subseteq Q \times Q$ such that for all $q_1, q_2 \in Q$, $q_1 \approx q_2$,*

- *whenever $q_1 \xrightarrow{0} q'_1$ with $q'_1 \in Q$, there exists $q'_2 \in Q$ such that $q_2 \xrightarrow{0} q'_2$ and $q'_1 \approx q'_2$;*
- *whenever $q_1 \xrightarrow{\tau} q'_1$ with $\tau > 0$ and $q'_1 \in Q$, there exist $\tau' > 0$ and $q'_2 \in Q$ such that $q_2 \xrightarrow{\tau'} q'_2$ and $q'_1 \approx q'_2$.*

A bisimulation $\approx$ is *finite* if it has a finite number of equivalence classes. It is said to *respect a partition* $\mathcal{P}$ of the set Q if any $P \in \mathcal{P}$ is a union of equivalence classes of $\approx$. A set $P \subseteq Q$ will be sometimes called a *region*.

Given a region $P \subseteq Q$, the set $Pre(P)$ of predecessor states of P is defined as Pre_0 or $Pre_{>0}$ according to both kinds of transitions : instantaneous switch or elapse of time, by

$$Pre_0(P) = \{q \in Q \mid \exists q' \in P \; q \xrightarrow{0} q'\};$$

$$Pre_{>0}(P) = \{q \in Q \mid \exists q' \in P \; \exists \tau > 0 \; q \xrightarrow{\tau} q'\}.$$

A crucial property of a bisimulation $\approx$ is that for every equivalence class P of $\approx$, the predecessor $Pre(P)$ is a union of equivalence classes. It follows that the *coarsest* bisimulation respecting a partition $\mathcal{P}_0$ can be computed by the next procedure.

```
Procedure Bisim.
  Initially $\mathcal{P} := \mathcal{P}_0$ ;
  While there exist $P, P' \in \mathcal{P}$ such that $\varnothing \subsetneq P \cap Pre(P') \subsetneq P$, do
          $P_1 := P \cap Pre(P')$, $P_2 := P \setminus Pre(P')$
          $\mathcal{P} := (\mathcal{P} \setminus \{P\}) \cup \{P_1, P_2\}$ ;
  Return $\mathcal{P}$ .
```

Proposition 1. *Let $\mathcal{A}$ be a weighted timed-automaton. The procedure* `Bisim` *terminates iff the coarsest bisimulation of $\mathcal{A}$ that respects a partition $\mathcal{P}$ is finite.*

An important property of bisimulations is that they preserve WCTL$_r$ formulas if they respect a well-chosen initial partition. We omit the proof since it is similar to the proof given in [4] for timed automata and the TCTL logic.

Proposition 2. *Let $\mathcal{A}$ be a weighted timed automaton and φ be a WCTL$_r$ formula. If $\mathcal{A}$ has a bisimulation $\approx$ that respects the partition $\mathcal{P}$ induced by*

1. *the atomic propositions σ labeling the locations of $\mathcal{A}$,*
2. *the cost constraints π appearing in φ,*
3. *the reset of the cost variables in φ (operator $z\cdot$),*

then for any states q, q' of $T_{\mathcal{A}}$ such that $q \approx q'$, we have $q \models \varphi$ iff $q' \models \varphi$.

As a consequence of this proposition, it can be proved that if each step of Procedure `Bisim` is *effective* and if this procedure *terminates*, then Problem 2 is decidable. Note that the effectiveness hypothesis is not necessary since weighted timed automata are linear hybrid automata for which the effectiveness of Procedure `Bisim` is known [10].

Corollary 2. *If a weighted timed automaton $\mathcal{A}$ has a finite bisimulation respecting the partition of Proposition 2, then the WCTL$_r$ model-checking problem is decidable.*[3]

To conclude this section, let us recall the classical bisimulation $\approx_t$ for timed automata [5]. Let $T_{\mathcal{A}}$ be the transition system of a timed automaton $\mathcal{A}$. Let $C \in \mathbb{N}$ be the supremum of all constants c used in guards of $\mathcal{A}$. For $\tau \in \mathbb{T}$, $\overline{\tau}$ denotes its fractional part and $\lfloor \tau \rfloor$ its integral part.

Definition 7. *Two states $q = (l, x)$, $q' = (l', x')$ of $T_{\mathcal{A}}$ are equivalent, $q \approx_t q'$, iff the following conditions hold*

- *$l = l'$;*
- *For any i, $1 \leq i \leq n$, either $\lfloor x_i \rfloor = \lfloor x'_i \rfloor$ or $x_i, x'_i > C$;*
- *For any $i \neq j$, $1 \leq i, j \leq n$ such that $x_i, x_j \leq C$, $\overline{x}_i \leq \overline{x}_j$ iff $\overline{x}'_i \leq \overline{x}'_j$;*
- *For any i, $1 \leq i \leq n$ such that $x_i \leq C$, $\overline{x}_i = 0$ iff $\overline{x}'_i = 0$.*

Note that for discrete time, only the first two conditions have to be considered in this definition. Thus given a clock x_i, its possible values in an equivalence class are 1, 2, …, C and $C^+ = \{n \in \mathbb{N} \mid n > C\}$.

5 Frontier between Finite and Infinite Bisimulations

In this section, we study Problem 2 with the approach of Corollary 2. We begin with the simple case of discrete time before studying the more complex case of dense time.

[3] The same result holds for WCTL (instead of WCTL$_r$) if the cost constraints in Condition 2 of Proposition 2 are general constraints $z_i \sim c$ or $z_i - z_j \sim c$.

5.1 Discrete Time

Theorem 2. *Let $\mathbb{T} = \mathbb{N}$. Any weigthed timed automaton has a finite bisimulation respecting the partition $\mathcal{P}$ of Proposition 2.*

Proof. (Sketch) This result is proved in [13] for more general automata which are the discrete-time rectangular automata, but without costs on the edges. However, the proposed bisimulation remains valid for weighted timed automata. It is the usual bisimulation of timed automata (see Definition 7) adapted as follows : the cost variables are treated as clock variables, and constant C is the supremum of the constants used in the guards of $\mathcal{A}$ and in the cost constraints of φ. □

Corollary 3. *Let $\mathbb{T} = \mathbb{N}$. The WCTL$_r$ model-checking problem for weigthed timed automata is* PSpace-Complete.

Proof. (Sketch). The PSpace-Hardness is a direct consequence of the fact that TCTL model-checking on timed automata is PSpace-Complete [4]. The PSpace-Easiness is established using classical arguments, see [4]. First note that the number of equivalence classes of the bisimulation given in the proof of Theorem 2 is bounded by an exponential in the size of the input of the model-checking problem (sum of the sizes of the automaton and the formula). We can turn the usual labeling algorithm used for CTL-like logics into a nondeterministic algorithm that uses polynomial space and computes the labels of regions as they are required. By Savitch's theorem, we know that there also exists a deterministic version of this algorithm that uses polynomial space. □

5.2 Dense Time

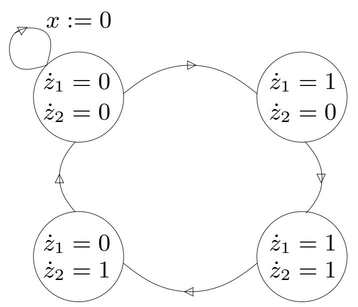
For dense time, the panorama is completely different. We will identify the *precise frontier* between finite and infinite bisimulations for the subclass of *automata with stopwatch observers*. We will conclude with some comments on the entire class of weighted timed automata.

Automata with stopwatch observers. By the proof of Theorem 1, for WCTL, we know that there exist automata with stopwatch observers using 1 clock and 3 cost variables for which any bisimulation respecting the partition $\mathcal{P}$ of Proposition 2 is infinite. The next theorem states that, for WCTL$_r$, it is already the case with 1 clock and 2 cost variables, as well as with 2 clock and 1 cost variables[4].

Theorem 3. *Let $\mathbb{T} = \mathbb{R}^+$. There exists an automaton with stopwatch observers $\mathcal{A}$ using either 1 clock and 2 cost variables, or 2 clock and 1 cost variables, and a WCTL$_r$formula φ such that no bisimulation respecting the partition $\mathcal{P}$ of Proposition 2 is finite.*

[4] We were able to establish this result partly with experiments performed with the HyTech tool [12].

Proof. The two automata that we are going to consider are given in Figures 6 and 7. Note that these automata have empty edges and no labeling of the loca-



$x := 0$
$\dot z_1 = 0$ $\dot z_2 = 0$
$\dot z_1 = 1$ $\dot z_2 = 0$
$\dot z_1 = 0$ $\dot z_2 = 1$
$\dot z_1 = 1$ $\dot z_2 = 1$

Fig. 6. 1 clock and 2 cost variables.

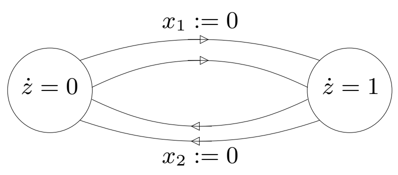


Fig. 7. 2 clocks and 1 cost variables.

tions by atomic propositions.

The proof is based on Procedure Bisim and Proposition 1 with the initial partition $\mathcal{P}$ given in Proposition 2. Note that Condition 1 of Proposition 2 is trivially satisfied.

Let us begin with the case of 1 clock variable x and 2 cost variables z_1, z_2. As initial partition $\mathcal{P}$, we take the partition induced by the bisimulation given in Definition 7. The following discussion justifies this choice.

At location of Figure 6 where $\dot z_1 = \dot z_2 = 1$ (we denote this location by l), the behavior of z_1, z_2 is the one of a clock. We have thus 3 clocks x, z_1, z_2 at location l. As shown in [5], if x, z_1 and z_2 are compared with constant 1, then Procedure Bisim leads to the bisimulation $\approx_t$ of Definition 7 in the cube $[0,1]^3$ and in location l. A way to get these comparisons with constant 1 is simply to add some guard or invariant $x = 1$ in the automaton of Figure 6 and to consider some WCTL$_r$ formula φ with the two cost constraints π_1 and π_2 respectively equal to $z_1 = 1$ and $z_2 = 1$. Again by Procedure Bisim, the bisimulation $\approx_t$ is transfered to the other locations by applying Pre_0 on the empty edges of the automaton. Therefore, as announced before, we can take as partition $\mathcal{P}$ the partition of the cube $[0,1]^3$ induced by $\approx_t$.

(1) *1 clock variable x and 2 cost variables z_1, z_2.*

Let us show that Procedure Bisim does not terminate because it generates an infinite number of regions R_n, $n \geq 1$, each containing exactly one triple (x, z_1, z_2) such that[5]

$$(x, z_1, z_2) = (0, \frac{1}{3^n}, \frac{3^n + 1}{2 \cdot 3^n}).$$

(a) We need to work with a particular region generated by the procedure (see Figure 8)

$$S \; : \quad 0 = x < z_1 < z_2 < 1, \;\; 2z_2 - z_1 = 1.$$

[5] When speaking about the constructed regions, we can omit the locations since the empty edges transfer the information to each location.

It is constructed as (see Figure 9)

- $S' = Pre_{>0}(P_1) \cap P_2$ with $P_1 : \; 0 < z_1 = z_2 < x = 1$, $P_2 : \; 0 < z_1 < z_2 = x < 1$, and $\dot{z}_1 = 1$, $\dot{z}_2 = 0$,
- $S = Pre_{>0}(S') \cap P_3$ with $P_3 : \; 0 = x < z_1 < z_2 < 1$, and $\dot{z}_1 = \dot{z}_2 = 0$.

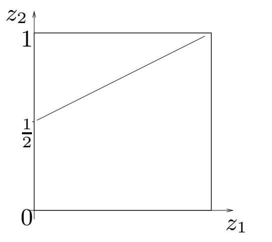


Fig. 8. Region S $(x = 0)$.

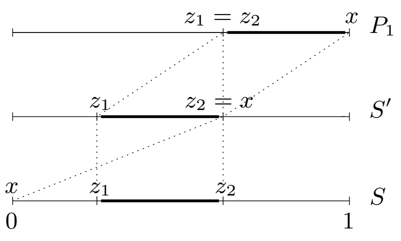


Fig. 9. Its construction.

Looking at the bold intervals in Figure 9, we see that on line S, we have $z_2 - z_1 = 1 - z_2$. It follows that $2z_2 - z_1 = 1$ must be satisfied in S.
(b) The first region $R_1 = \{0, \frac{1}{3}, \frac{2}{3}\}$ is then constructed as (see Figures 10 and 11)

- $R'_1 = Pre_{>0}(P_1) \cap P_2$ with $P_1 : \; 0 < x = z_1 < z_2 = 1$, $P_2 : \; 0 = x < z_1 < z_2 < 1$, and $\dot{z}_1 = 0$, $\dot{z}_2 = 1$,
- $R_1 = Pre_0(R'_1) \cap S$.

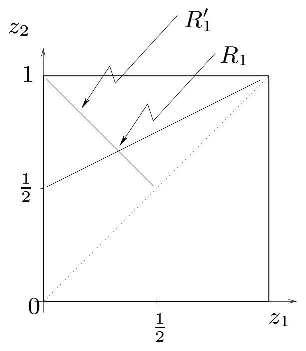


Fig. 10. Region R_1.

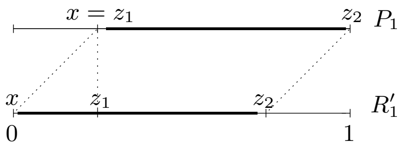


Fig. 11. Its construction.

Looking at the bold intervals in Figure 11, one verifies that R'_1 is the region

$$R'_1 \; : \quad 0 = x < z_1 < z_2 < 1, \;\; z_1 + z_2 = 1.$$

In Figure 10, the intersection of R'_1 and S, which is nothing else than $R_1 = Pre_0(R'_1) \cap S$, is the point $(0, \frac{1}{3}, \frac{2}{3})$.
(c) It remains to explain how to construct R_{n+1} from R_n. It is done as follows (see Figures 12 and 13)

- $S'_1 = Pre_0(R_n) \cap P_1$ with $P_1 : \ 0 < z_1 < z_2 < x = 1$,
- $S'_2 = Pre_{>0}(S'_1) \cap P_2$ with $P_2 : \ 0 < x = z_1 < z_2 < 1$, and $\dot{z}_1 = 0$, $\dot{z}_2 = 0$,
- $S'_3 = Pre_{>0}(S'_2) \cap P_3$ with $P_3 : \ 0 < x < z_1 < z_2 < 1$, and $\dot{z}_1 = 0$, $\dot{z}_2 = 1$,
- $R'_{n+1} = Pre_{>0}(S'_3) \cap P_4$ with $P_4 : \ 0 = x < z_1 < z_2 < 1$, and $\dot{z}_1 = 1$, $\dot{z}_2 = 0$,
- $R_{n+1} = Pre_0(R'_{n+1}) \cap S$.

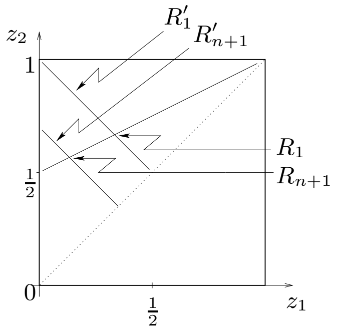


Fig. 12. Region R_{n+1}.

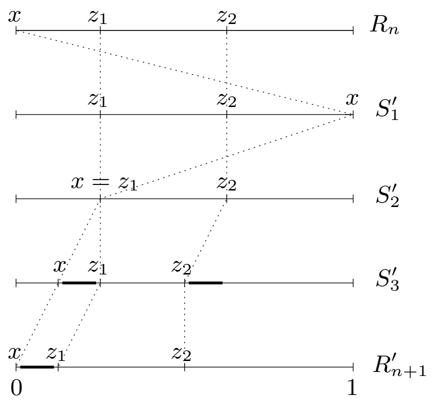


Fig. 13. Its construction from R_n.

Recall that $R_n = (0, \frac{1}{3^n}, \frac{3^n+1}{2 \cdot 3^n})$. Thus looking at the bold intervals of Figure 13 (in particular at lines R'_{n+1}, S'_3 and R_n)), the next equality must hold on R'_{n+1}

$$z_1 + z_2 = \frac{3^n + 1}{2 \cdot 3^n}.$$

On Figure 12, the intersection of R'_{n+1} and S, which is R_{n+1}, is therefore the point $(0, \frac{1}{3^{n+1}}, \frac{3^{n+1}+1}{2 \cdot 3^{n+1}})$.

(2) *2 clock variables x_1, x_2 and 1 cost variable z.* (Sketch)

The proof for the case of 2 clock variables x_1, x_2 and 1 cost variable z is in the same vein as before. The automaton is the one of Figure 7. Procedure `Bisim` does not terminate because it generates an infinite number of regions R_n, $n \geq 1$, each formed by the unique triple

$$(x_1, x_2, z) = (0, 1 - \frac{1}{2^n}, \frac{1}{2^n}).$$

□

Any timed automaton has a finite bisimulation respecting the partition $\mathcal{P}$ of Proposition 2 (see Definition 7). Thus the remaining case to establish a precise frontier between finite and infinite bisimulations is the case of automata with stopwatch observers using 1 clock and 1 cost variables.

Theorem 4. *Let $\mathbb{T} = \mathbb{R}^+$. Let $\mathcal{A}$ be an automaton with stopwatch observers using 1 clock and 1 cost variables x and z. Then $\mathcal{A}$ has a finite bisimulation respecting the partition $\mathcal{P}$ of Proposition 2.*

Proof. (Sketch) The proposed bisimulation is the one of Definition 7, where z is treated as a clock. □

Corollary 4. *The WCTL$_r$ model-checking problem for automata with stopwatch observers using 1 clock and 1 cost variables is decidable.*

Comments on weighted timed automata. All the results of the previous paragraph are concerned with automata with stopwatch observers. If we consider weighted timed automata, the frontier between finite and infinite bisimulations is easily established. There exist weighted timed automata with 1 clock and 1 cost variables x and z such that $\dot{z} = d_1$, $\dot{z} = d_2$, with $d_1, d_2 > 0$ two integer constants, for which no finite bisimulation exists [11] (see Figure 14). If for automata with 1 clock and 1 cost variables x and z, we impose that there exists an integer constant $d > 0$ such that $\dot{z} \in \{0, d\}$ in each location, then a finite bisimulation exists. It is the bisimulation of Definition 7, where z is treated as a clock and each diagonal $z - x = c$ is replaced by $z - dx = c$ (see Figure 15). Note

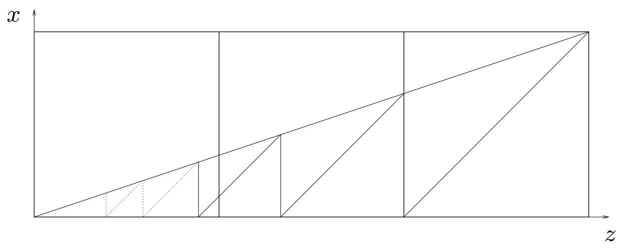


Fig. 14. Infinite bisimulation when $d_1 = 1, d_2 = 3$.

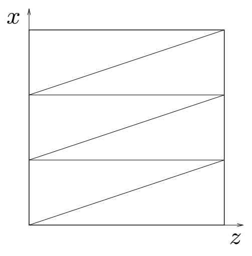


Fig. 15. Finite bisimulation when $d = 3$.

that a finite bisimulation still exists if we allow to add to the variables x and z additional cost variables $z_2, \ldots, z_m$ having a null cost on the locations and an arbitrary cost on the edges. In Example 1, z_3 is such a variable. The required finite bisimulation is a direct product of the bisimulation given before for x and z with the bisimulation of Definition 7 applied to the variables $z_2, \ldots, z_m$ treated as clocks.

Note that Problem 2 remains unanswered for dense time (except in the case of Corollary 4) since the existence of a finite bisimulation is sufficient but not necessary for decidability of the model-checking problem. For instance, in the particular case of weighted timed automata with one cost variable, the cost-bounded reachability problem is shown to be decidable in [1] while there is no finite bisimulation as mentioned above.

References

1. R. Alur, C. Courcoubetis, and T.A. Henzinger. Computing accumulated delays in real-time systems. In *CAV 93: Computer-Aided Verification*, Lecture Notes in Computer Science 697, pages 181–193. Springer-Verlag, 1993.
2. R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.
3. R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971–984, 2000.
4. Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.
5. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
6. Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In *Proceedings of the 4th International Workshop on Hybrid Systems:Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer-Verlag, 2001.
7. Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-cost reachability for priced timed automata. In *Proceedings of the 4th International Workshop on Hybrid Systems:Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer-Verlag, 2001.
8. Ahmed Bouajjani, Rachid Echahed, and Joseph Sifakis. On model checking for real-time properties with durations. In *Logic in Computer Science*, pages 147–159, 1993.
9. Véronique Bruyère and Jean-François Raskin. Real-time model-checking: Parameters everywhere. In Paritosh K. Pandya and Jaikumar Radhakrishnan, editors, *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, Proceedings*, number 2914 in Lecture Notes in Computer Science, pages 100–111. Springer, 2003.
10. T.A. Henzinger. Hybrid automata with finite bisimulations. In *ICALP 95: Automata, Languages, and Programming*, Lecture Notes in Computer Science 944, pages 324–335. Springer-Verlag, 1995.
11. T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.
12. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HyTech. In *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 1019, pages 41–71. Springer-Verlag, 1995.
13. T.A. Henzinger and P.W. Kopke. Discrete-time control for rectangular hybrid automata. In *ICALP 97: Automata, Languages, and Programming*, Lecture Notes in Computer Science 1256, pages 582–593. Springer-Verlag, 1997.
14. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proceedings of the 27th Annual Symposium on Theory of Computing*, pages 373–382. ACM Press, 1995.
15. Yonit Kesten, Amir Pnueli, Joseph Sifakis, and Sergio Yovine. Decidable integration graphs. *Information and Computation*, 150(2):209–243, 1999.