On Model-Checking Timed Automata with Stopwatch Observers ⋆

Thomas Brihaye [a] Véronique Bruyère [a] Jean-François Raskin [b]

[a] *Faculté des Sciences, Université de Mons-Hainaut, Avenue du Champ de Mars 6, B-7000 Mons, Belgium*

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

Abstract

In this paper we study the model-checking problem for weighted timed automata and the weighted CTL logic ; we also study the finiteness of bisimulations of weighted timed automata. 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 allows 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 in general, *(ii)* its PSpace-Completeness in the discrete case, and its undecidability in the dense case, for a slight restriction of the weighted CTL Logic, *(iii)* the precise frontier between finite and infinite bisimulations in the dense case for the subclass of stopwatch automata.

Key words: Weighted timed automata, model-checking, bisimulations.

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 [2].

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

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 [7] and [8] (with single costs instead of tuples of costs).

The properties of weighted timed automata that we want to check are formalised by formulas of the *weighted CTL logic*, WCTL for short. This logic is close to the DTL logic of [9] and the ICTL logic of [5].

Our approach is a systematic study of the tool *bisimulation* as done in the works [12], [13] and [19]. 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 of branching logics are decidable. For instance this technique has been successfully applied to timed automata thanks to the region graph (see [4]). However the converse does not hold in general.

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 generalisations of these automata lead to an undecidable reachability problem, in particular for timed automata augmented with one stopwatch [16]. The reachability problem is also 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) [17]. The latter case has also been studied in [2]. Of course the well-known class of timed automata has a decidable reachability problem [4]. 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 [7] and [8]. Lately an interesting extension of the minimum cost-reachability problem, namely the *optimal conditional reachability problem*, has been introduced and proved to be decidable in [18].

Concerning the model-checking problem of hybrid systems, let us mention two references. In [5], 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 [9], 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. It 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 [5] 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. This proof works for automata with stopwatch observers equipped with 1 clock and 3 stopwatches and for WCTL formulas where two integrators are compared.

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 [12], [13] and [19]. 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 first prove that the WCTL_r model-checking problem becomes undecidable. As before for the WCTL logic, the proof is based on the halting problem for 2-counter machines, and it works for automata with stopwatch observers using 5 clocks and 1 stopwatch. In the case of dense time, we also 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 stopwatch, or with 1 clock and 2 stopwatches. This is no longer true with 1 clock and 1 stopwatch and in this particular case the WCTL_r model-checking problem is decidable.

A part of these results has been published in [11], namely the undecidability of the WCTL model-checking problem and the precise frontier between finite and infinite bisimulations for automata with stopwatch observers. Additionally in this paper we give proofs of the previous results and we completely study the WCTL_r model-checking problem.

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) $E \subseteq L \times \mathcal{G} \times 2^X \times L$ 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-tuple 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 [7] and [8] (with single costs instead of m-tuples 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* [1]. Note that the variables $z_1, \ldots, z_m$ cannot be reset nor tested in

[1] This notation comes from automata with integrators, the variables $z_1, \ldots, z_m$ being the integrators, see for instance [17].

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*. On figures, if a cost is not indicated, it is assumed to be null. When an edge has no cost, no reset and a guard that is always true, it is called an *empty* edge.

Definition 2 *The* semantics *of a weighted timed automaton* $\mathcal{A} = (L, E, \mathcal{I}, \mathcal{L}, \mathcal{C})$ *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 *is totally ordered in a natural way.*

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

Example 4 *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.*

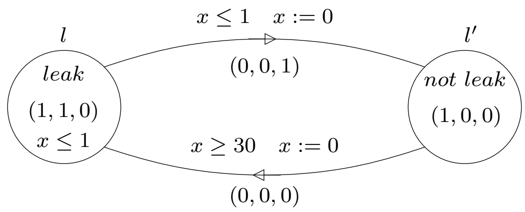

Fig. 1. The gas burner system.

3 Weighted CTL Logic

In this section, we introduce the weighted CTL logic, WCTL logic for short (close to the ICTL logic of [5] and to the DTL logic of [9]). Two logics, discrete and dense, are proposed according to discrete or dense time.

Notations. Let $Z = \{z_1, \ldots, z_m\}$ be a set of m cost variables. As done previously for clocks, the same notation $z = (z_1, \ldots, z_m)$ 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.

Definition 5 *The* syntax *of the* discrete *WCTL logic is given by the following grammar*

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

where $\sigma \in \Sigma$, π is a cost constraint and $z \in Z$. 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 [1] is a particular case of WCTL when each cost variable z_i describes the total elapsed time.

We 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 $\forall\Box\varphi \equiv \neg\exists\Diamond\neg\varphi$.[2]

The formulae of WCTL are evaluated on a given weighted timed automaton $\mathcal{A}$. The sets Σ and Z are supposed to be the same for both $\mathcal{A}$ and WCTL.

We now give the semantics of WCTL.

[2] Notation $\top$ means TRUE and $\perp$ means FALSE.

Definition 6 *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* $\mathcal{A}, q \models \varphi$ *is defined inductively as indicated below.*

- $\mathcal{A}, q \models \sigma$ *iff* $\sigma \in \mathcal{L}(l)$*;*
- $\mathcal{A}, q \models \pi$ *iff* $z \models \pi$*;*
- $\mathcal{A}, q \models \neg\varphi$ *iff* $\mathcal{A}, q \not\models \varphi$*;*
- $\mathcal{A}, q \models \varphi \vee \psi$ *iff* $\mathcal{A}, q \models \varphi$ *or* $\mathcal{A}, q \models \psi$*;*
- $\mathcal{A}, q \models \exists\bigcirc\varphi$ *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* $\mathcal{A}, q_1 \models \varphi$*;*
- $\mathcal{A}, q \models \varphi \exists U \psi$ *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* $\mathcal{A}, p \models \psi$ *and* $\mathcal{A}, p' \models \varphi$ *for all* $p' < p$*;*
- $\mathcal{A}, q \models \varphi \forall U \psi$ *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* $\mathcal{A}, p \models \psi$ *and* $\mathcal{A}, p' \models \varphi$ *for all* $p' < p$*;*
- $\mathcal{A}, q \models z_i \cdot \varphi$ *iff* $\mathcal{A}, (l, x, [z_i := 0]z) \models \varphi$.

In case $\mathbb{T} = \mathbb{R}^+$ and φ is a dense WCTL formula, the satisfaction relation is defined in the same way, except that $\mathcal{A}, q \models \exists\bigcirc\varphi$ does not exist. When $\mathcal{A}$ is clear from the context, we simply write $q \models \varphi$ instead of $\mathcal{A}, q \models \varphi$.

Let us come back to the gas burner system of Example 4 and formalise some properties by WCTL formulas.

Example 7 *Consider the first property "there exists a run with an average leaking time always bounded by* 0.5*" (which formalises* $2z_2 \leq z_3$*). 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 formalise 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 formalised as*

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

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

- 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** .

Fig. 2. Instructions of a 2-counter machine

4 Undecidability of WCTL Model-Checking

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

Problem 8 *Given a weighted timed automaton $\mathcal{A}$ and a state q of $T_{\mathcal{A}}$, given a WCTL formula φ, does $\mathcal{A}, q \models \varphi$ 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 9 *In both cases of discrete and dense time, the WCTL model-checking problem for automata with stopwatch observers is undecidable.*

Corollary 10 *Problem 8 is undecidable.*

PROOF. (of Theorem 9) 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 basic instructions given on Figure 2. [3]

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.

[3] We assume that there is an **if** instruction before each decrementation instruction such that in the case the counter has a value zero, the counter value is not modified, otherwise it is decremented.

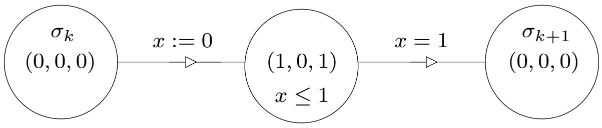

Fig. 3. Incrementing counter C_1.

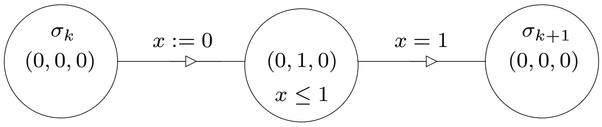

Fig. 4. Decrementing counter C_1.

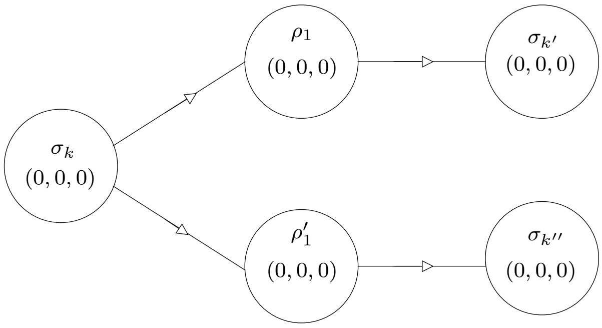

Fig. 5. **If** instruction with test on C_1.

The **goto** and **stop** instructions are easily encoded in $\mathcal{A}$.

The instruction for incrementing counter C_1 is encoded by the subautomaton given on Figure 3. The subautomaton for incrementing C_2 is similar except that the cost of the central state is $(1,1,0)$.

Considering the previous footnote, the instruction for decrementing counter C_1 is encoded in Figure 4. A similar subautomaton is given for counter C_2 with the cost of the central state equal $(0,0,1)$.

The **if** instruction is encoded as indicated on Figure 5. 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.

Let us now give formula φ:

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

Clearly, the 2-counter machine halts on the **stop** instruction if and only if $q \models \varphi$ with the following state

$$q = (l, x, z_1, z_2, z_3) = (l_0, 0, 0, 0, 0)$$

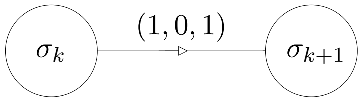

Fig. 6. Incrementing counter C_1 with no cost in the locations.

such that l_0 is the location labeled by σ_{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 $z_i - z_j \sim 0$. It does not use any freeze quantifier. The later comment implies that the model-checking for automata with stopwatch observers is already undecidable for the fragment of WCTL where the freeze operator is forbidden.

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 6 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 Section 6 we will restrict to a fragment of WCTL which can not compare between two cost variables.

5 Bisimulations

We recall in this section useful notions on time abstracting bisimulations (see [12] or [6]). Indeed in the sequel of the article we want to study the relations between finite bisimulations and Problem 16.

Definition 11 *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 $q_1, q_2 \in Q$ satisfying $q_1 \approx q_2$,*

- *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* $\mathcal{P}_0$ of the set Q if any $P \in \mathcal{P}_0$ 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 12 [6] [12] *Let $\mathcal{A}$ be a weighted timed automaton. The procedure* `Bisim` *terminates if and only if the coarsest bisimulation of $\mathcal{A}$ that respects a partition $\mathcal{P}_0$ 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 [1] for timed automata and the TCTL logic.

Proposition 13 *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 $\mathcal{P}_0$ 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 16 is decidable. Note that the effectiveness hypothesis does not need to be proved since weighted timed automata are linear hybrid automata for which the effectiveness of Procedure `Bisim` is known [12].

Corollary 14 *Let $\mathcal{A}$ be a weighted timed automaton and φ a WCTL$_r$ formula. If $\mathcal{A}$ has a finite bisimulation respecting the partition of Proposition 13, then the WCTL$_r$ model-checking problem is decidable.* [4]

To conclude this section, let us recall the classical bisimulation $\approx_t$ for timed automata [4].

Definition 15 *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. Two states $q = (l, x)$, $q' = (l', x')$ of $T_{\mathcal{A}}$ are equivalent, $q \approx_t q'$, if and only if 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 $C^+ = \{n \in \mathbb{N} \mid n > C\}$.

6 Model-Checking for WCTL$_r$

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, for discrete and dense time.

Problem 16 *Given a weighted timed automaton $\mathcal{A}$ and a state q of $T_{\mathcal{A}}$, given a WCTL$_r$ formula φ, does $\mathcal{A}, q \models \varphi$ hold ? ($\mathbb{T} = \mathbb{N}$ or $\mathbb{T} = \mathbb{R}^+$)*

Example 17 *For the gas burner system of Example 4, the property "if the number of leaks is less than 5, then the leaking time is strictly bounded by 5" is formalised 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

[4] The same result holds for WCTL (instead of WCTL$_r$) if the cost constraints in Condition 2 of Proposition 13 are general constraints $z_i \sim c$ or $z_i - z_j \sim c$.

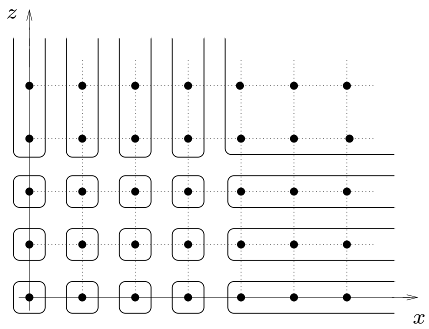

Fig. 7. Example of a finite bisimulation in the discrete case.

exceed 2 in any time interval less than 100 time units" is formalised by

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

Finally, the property "as soon as a leak is detected, the gas burner stops leaking after at most 1 time unit" is formalised by

$$\forall\Box(leak \Rightarrow z_1 \cdot \forall\Diamond(\neg leak \wedge z_1 \leq 1)).$$

This section is devoted to the study of Problem 16. We begin with the simple case of discrete time before studying the more complex case of dense time.

6.1 Discrete Time

In the case of discrete time, the model-checking problem for WCTL_r is decidable thanks to Corollary 14.

Theorem 18 *Let $\mathbb{T} = \mathbb{N}$. Let $\mathcal{A}$ be a weighted timed automaton and φ be a $WCTL_r$ formula. Then $\mathcal{A}$ has a finite bisimulation respecting the partition of Proposition 13.*

PROOF. (Sketch) This result is proved in [15] 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 15) 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. $\Box$

Figure 7 indicates an example of the finite bisimulation discussed in the previous proof for 1 clock x and 1 cost variable z.

Corollary 19 *In the case of discrete time, the WCTL$_r$ model-checking problem for weighted 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 [1]. The PSPACE-EASINESS is established using classical arguments, see [1]. First note that the number of equivalence classes of the bisimulation given in the proof of Theorem 18 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. □

6.2 Dense Time

For dense time, the panorama is completely different since the model-checking becomes undecidable, already for automata with stopwatch observers.

Theorem 20 *Let $\mathbb{T} = \mathbb{R}^+$. The WCTL$_r$ model-checking problem for automata with stopwatch observers is undecidable.*

Corollary 21 *In the case of dense time, Problem 16 is undecidable.*

PROOF. (of Theorem 20) As for Theorem 9, the proof is based on a reduction of the halting problem for 2-counter machines. The emulation of the 2-counter machine M is done partly by an automaton with stopwatch observers $\mathcal{A}$ and partly by a WCTL$_r$ formula φ.

We refer to Figure 2 for the basic instructions used by the 2-counter machine M. Let us denote by K the list of instructions of M. A *configuration* of M is given by a triple $(k, c_1, c_2) \in K \times \mathbb{N}^2$ which represents the (label of the) current instruction and the value of the two counters C_1 and C_2. The first instruction of M is supposed to be labeled by k_0 and the **stop** instruction for which M halts, is supposed to be labeled by k_t. The *initial* configuration of M is thus $(k_0, 0, 0)$.

The automaton $\mathcal{A}$ contains a special clock τ which is reset to 0 whenever it reaches the value 1. The ith configuration of the machine M is encoded by the state of the transition system $T_{\mathcal{A}}$ of $\mathcal{A}$ at time i (i.e. at the ith reset of τ).

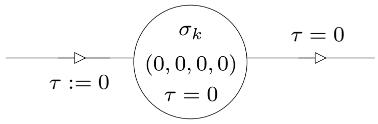

Fig. 8. location labeled by σ_k

First we explain how to encode the value of the counters C_1, C_2 of M. Let us consider *pairs* (x, z), where x is a clock and z is a cost variable, whose values are of the form $(2^{-n}, 1-2^{-n})$, $n \geq 1$, when $\tau = 0$. We will explain later how we obtain those values. By means of 4 pairs (x_1, z_1), (x_2, z_2), (x_3, z_3) and (x_4, z_4), we encode the 2 counters C_1 and C_2 as follows:

$$C_1 = c_1 \;\Leftrightarrow\; (x_1 = \tfrac{1}{2^{n_1}}) \text{ and } (x_2 = \tfrac{1}{2^{n_2}}) \text{ and } n_1 - n_2 = c_1,$$
$$C_2 = c_2 \;\Leftrightarrow\; (x_3 = \tfrac{1}{2^{n_3}}) \text{ and } (x_4 = \tfrac{1}{2^{n_4}}) \text{ and } n_3 - n_4 = c_2. \tag{1}$$

We can already notice that incrementing the counter C_1 corresponds to divide the clock x_1 by 2, and decrementing the counter C_1 corresponds to divide the clock x_2 by 2 (similarly for the counter C_2). We will explain how to proceed in detail later in the proof.

The automaton $\mathcal{A} = (L, E, \mathcal{I}, \mathcal{L}, \mathcal{C})$ has 5 clocks (the special clock τ and the clocks x_1, x_2, x_3, x_4), 4 cost variables (z_1, z_2, z_3, z_4) 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 instructions of K. It also contains additional atomic propositions ρ_i, ρ'_i, ς_i, ς'_i, for $i = 1, 2$, and μ_j, μ'_j for $j = 1, 2, 3, 4$. The set L contains a location for each label k of the machine M, which is labeled by σ_k. For each such k, the related location is as depicted in Figure 8, i.e. with an invariant $\tau = 0$ and an outgoing edge labeled by the guard $\tau = 0$. So the transition system $T_{\mathcal{A}}$ spends no time in these locations. This means that the ith configuration (k, c_1, c_2) of M is encoded by the state of $T_{\mathcal{A}}$ at time i exactly. The set L also contains additional locations that will be described later.

Formula φ will be constructed in parallel with $\mathcal{A}$ in a way that M starting with the initial configuration $(k_0, 0, 0)$ halts with the **stop** instruction if and only if $q_0 \models \varphi$ for the state q_0 of $T_{\mathcal{A}}$ given by

$$q_0 = (l, \tau, x_1 x_2, x_3, x_4, z_1, z_2, z_3, z_4) = \left(l_0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$$

where l_0 is the location labeled by σ_{k_0}. Notice that the pair (x_i, z_i) appearing in q_0 are of the desired form $(2^{-n}, 1 - 2^{-n})$.

We are now ready to encode the instructions of M with $\mathcal{A}$ and φ. The **stop** instruction is trivially implemented by a location labeled σ_{k_t}.

The **goto** instruction is encoded by the subautomaton of $\mathcal{A}$ given on Figure 9.

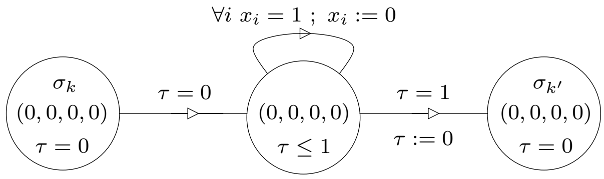

Fig. 9. k: **goto** k'.

We do not use formula φ in this case. The values of the 4 pairs (x_i, z_i) have to be kept unchanged since the values of the 2 counters are not changed. To let the value of each z_i unchanged is simple, it suffices to assign a null cost to all the locations of Figure 9 (i.e. $\mathcal{C}(l) = (0,0,0,0)$). To keep the value of the clocks x_i unchanged, we use a classical trick (see for example [3]). Since the emulation of the **goto** instruction takes exactly one unit of time, guaranteed by the clock τ, it suffices to reset to 0 each clock x_i whenever it reaches value 1. Considering the central location of Figure 9, this requires to add the 4 invariants $x_i \leq 1$, and several loops labeled by the guards $x_i = 1$ and the resets $x_i := 0$ (taking into account that 2 or more resets could be simultaneous). This is indicated on Figure 9 with notation $\forall i\ x_i = 1\ ;\ x_i := 0$. Hence we can conclude that if the 4 pairs (x_i, z_i) have the desired form $(2^{-n_i}, 1 - 2^{-n_i})$ in the location labeled σ_k, they will recover the same value when $\mathcal{A}$ enters the location labeled $\sigma_{k'}$. This ends the emulation of the **goto** instruction.

This construction, that allows to keep the value of the pairs (x_i, z_i) unchanged, will be applied again in the sequel of the proof. However we will not give an explicit construction but only refer to the *widget*. This widget takes exactly one time unit, and ensures that the value of the clocks x_i are kept constant by adding loops coupled with guards, resets and invariants in order to reset x_i whenever it reaches 1 (this will be indicated on the next figures by using notation $\forall i\ x_i = 1\ ;\ x_i := 0$).

We now turn to the **if** instruction. We treat the test of the counter C_1, the other case is similar. To test whether the counter C_1 is equal to 0 is equivalent to test whether x_1 is equal to x_2, see (1). But testing equality between two clocks is not allowed in the automaton. We need to introduce a more tricky encoding which uses both the automaton $\mathcal{A}$ and the formula φ. Let us consider the subautomaton of $\mathcal{A}$ given on Figure 10. The atomic proposition ρ_1 is a witness for $x_1 = x_2$ and the atomic proposition ρ'_1 is a witness for $x_1 < x_2$.[5] Since $\mathcal{A}$ is not allowed to compare its clocks, we use instead the *branching* power of WCTL$_r$ through φ. To check if $x_1 = x_2$ in the location labeled by ρ_1 is equivalent to check later on that $x_1 = x_2 = 1$ (letting time elapse), that is to check with a subformula ψ_1 of φ that the location labeled ς_1 can be reached from it. We proceed in a similar way to check if $x_1 < x_2$ in the location labeled

[5] The index 1 in ρ_1 and ρ'_1 is used to recall that it is counter C_1 which is tested.

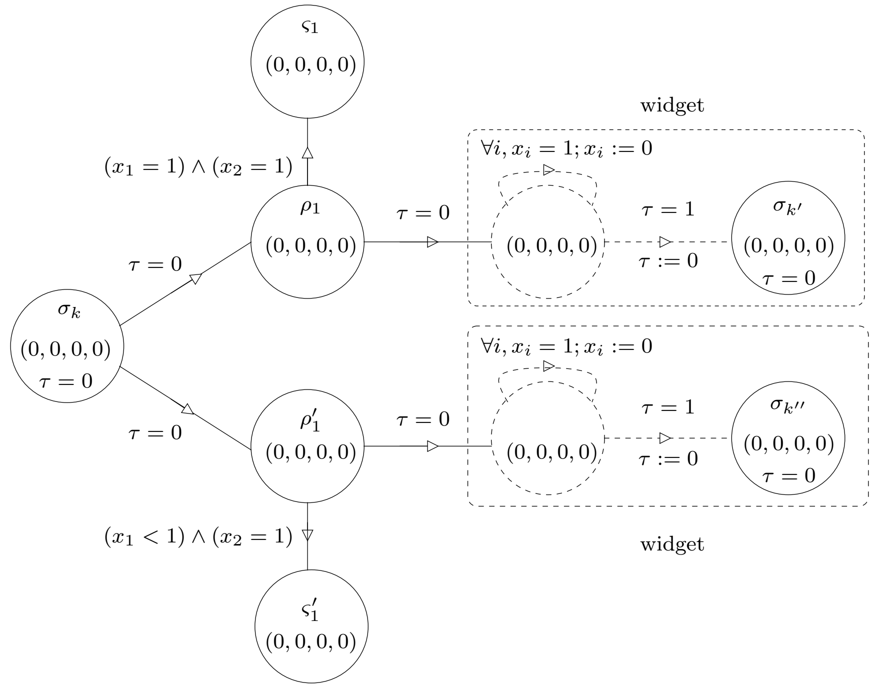

Fig. 10. k: **if** $C_1 = 0$ **then goto** k' **else goto** k''.

ς'_1. This subformula ψ_1 is defined as follows:

$$\psi_1 \;\equiv\; (\rho_1 \Rightarrow \rho_1 \exists \mathrm{U} \varsigma_1) \wedge (\rho'_1 \Rightarrow \rho'_1 \exists \mathrm{U} \varsigma'_1).$$

In the **if** instruction, depending on whether $C_1 = 0$ or $C_1 > 0$, there is a **goto** k' or a **goto** k''. This is encoded in the automaton of Figure 10 by using two widgets such that the value of the pairs (x_i, z_i) are left unchanged.

The **if** instruction for counter C_2 is treated similarly. The subautomaton is the same except that atomic propositions ρ_2, ρ'_2, ς_2 and ς'_2 are used instead of ρ_1, ρ'_1, ς_1 and ς'_1, and clocks x_3, x_4 are used instead of x_1, x_2. The subformula is the following one:

$$\psi_2 \;\equiv\; (\rho_2 \Rightarrow \rho_2 \exists \mathrm{U} \varsigma_2) \wedge (\rho'_2 \Rightarrow \rho'_2 \exists \mathrm{U} \varsigma'_2).$$

It remains to emulate the incrementation and decrementation instructions. In both cases, it suffices to divide the value of a clock by 2 while the value of the other clocks remain unchanged. We only go into detail for the instruction $C_1 := C_1 + 1$, the other cases being similar. Let us consider the subautomaton of $\mathcal{A}$ given on Figure 11. In order to increment C_1, if $\mathcal{A}$ enters the location labeled σ_k with $(x_1, z_1) = (2^{-n}, 1 - 2^{-n})$, it has to reach the location labeled $\sigma_{k'}$ with $(x_1, z_1) = (2^{-(n+1)}, 1 - 2^{-(n+1)})$, the values of the 3 other pairs (x_i, z_i) being unchanged. To force $\mathcal{A}$ to adopt this behaviour, we again use the branching aspect of the logic through the following subformula[6]:

$$\phi_1 \;\equiv\; \mu_1 \Rightarrow \mu_1 \exists\mathrm{U}(\mu_1' \wedge z_1 = 1) \tag{2}$$

where the atomic propositions μ_1 and μ_1' are witness that the pair (x_1, z_1) is modified.

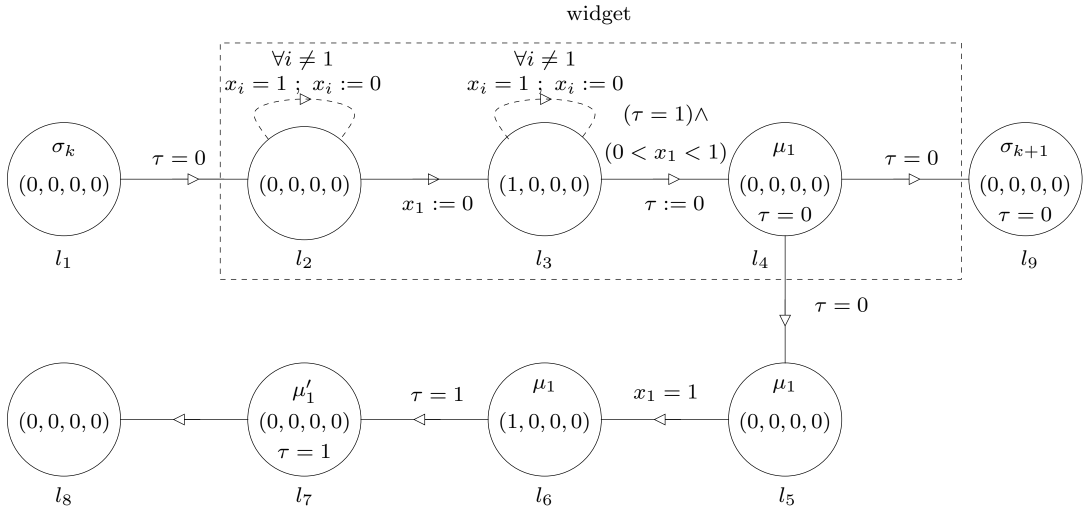

Fig. 11. k: $C_1 := C_1 + 1$.

The proof that the evolution of the pair (x_1, z_1) is done correctly is rather technical and is formalised in Lemma 22. The other pairs are left unchanged using the widget (see locations l_2, l_3 and l_4 of Figure 11).

We have a similar subautomaton and subformula for decrementing C_1 such that x_1, z_1, μ_1, μ_1' and ϕ_1 are replaced respectively by x_2, z_2, μ_2, μ_2' and ϕ_2. (Similarly for the incrementation and the decrementation of counter C_2 by using indexes 3 and 4).

We are now able to give the whole formula φ:

$$\varphi \;\equiv\; (\psi_1 \wedge \psi_2 \wedge \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4)\, \exists\mathrm{U}\; \sigma_{k_t}. \tag{3}$$

Clearly M halts on the **stop** instruction if and only if $q_0 \models \varphi$. It follows that the model-checking problem for automata with stopwatch observers is undecidable. $\square$

Lemma 22 *Let us consider Figure 11. If $\mathcal{A}$ enters location l_1 with $(x_1, z_1) = (2^{-n}, 1-2^{-n})$ and if formula ϕ_1 is satisfied at location l_4, then $\mathcal{A}$ enters location l_9 with the value of (x_1, z_1) equal to $(2^{-(n+1)}, 1 - 2^{-(n+1)})$.*

[6] The index 1 in μ_1 and μ_1' is used to recall that the pair (x_1, z_1) is modified.

PROOF. By hypothesis, $\mathcal{A}$ enters location l_1 with $(x_1, z_1) = (2^{-n}, 1 - 2^{-n})$ and $\tau = 0$. By construction, we can see that $(x_1, z_1) = (0, 1 - 2^{-n})$ when entering location l_3.

Since ϕ_1 is satisfied at location l_4, we have $z_1 = 1$ in location l_7. This implies that $z_1 = \tau = 1$ in location l_7 and so $z_1 = \tau$ when leaving location l_5 with $x_1 = 1$.

We have to show that the value of (x_1, z_1) in l_4 is $(2^{-(n+1)}, 1 - 2^{-(n+1)})$. Let us notice that the value of (x_1, z_1) entering l_5 is equal to its value in l_4.

Figure 12 represents the evolution of the variables x_1, z_1 and τ along the path from l_3 to l_5. It indicates in bold face a quantity α kept constant along the lines. In the first line, recall that (x_1, z_1) has value $(0, 1 - 2^{-n})$. In the second line, it has value (α, β) with $\beta = 1 - 2^{-n} + \alpha$. In the third line, we have $\alpha + \beta = 1$ showing that $\alpha = 2^{-(n+1)}$. Thus (x_1, z_1) has value $(2^{-(n+1)}, 1 - 2^{-(n+1)})$ at location l_4. $\square$

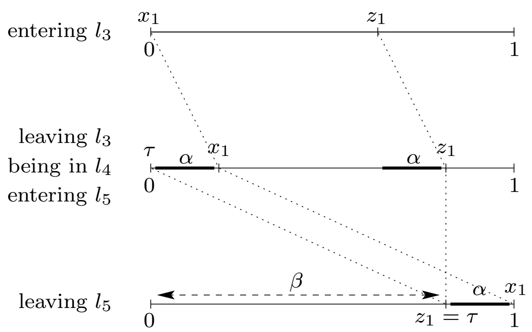

Fig. 12. Evolution of the variables from l_3 to l_5.

Comments. The previous proof uses an automaton $\mathcal{A}$ with stopwatch observers and a WCTL_r formula φ. The automaton has 5 clocks and 4 cost variables (clock τ and pairs (x_i, z_i), $1 \le i \le 4$). It has no cost on its edges. The formula does not use the freeze operator. In particular, the model-checking problem for automata with stopwatch observers is already undecidable for the fragment of WCTL_r where the freeze operator is forbidden.

In the next corollary, we show that the WCTL_r model-checking problem is already undecidable for automata with stopwatch observers using 5 clocks and 1 cost variable only. The proof will now use the freeze operator.

The fact that we were able to reduce the number of cost variables to only one is very interesting, when one recalls that the minimum-cost reachability problem has been proved to be decidable for weighted timed automata with 1 cost variable [7] [8].

Corollary 23 *Let $\mathbb{T} = \mathbb{R}^+$. The WCTL$_r$ model-checking problem is undecidable for automata with stopwatch observers using 5 clocks and 1 cost variable.*

PROOF. Let us show how to modify the proof of Theorem 20 in a way to use only 1 cost variable.

We first recall the role of the 4 cost variables z_i in the proof of Theorem 20. In addition to the special clock τ, the clocks x_i, $1 \leq i \leq 4$, are used to encode the 2 counters as indicated in (1). Each clock x_i is coupled with the cost variable z_i such that (x_i, z_i) has values of the form $(2^{-n}, 1 - 2^{-n})$, $n \geq 1$, when $\tau = 0$. Looking at the encoding of each basic instruction of the 2-counter machine, we notice that the cost variables z_i are useful only for the incrementation and decrementation instructions (see Figure 11).

We are now going to show that the 4 cost variables z_i can be replaced by 1 cost variable z. The encoding of the **stop**, **goto** and **if** instructions is done exactly as in the proof of Theorem 20, except that the 4-tuple $(0, 0, 0, 0)$ appearing in the locations of Figures 8, 9 and 10 is replaced by $\dot{z} = 0$.

It remains to detail the encoding of the incrementation and decrementation instructions. We explain the idea for the incrementation of counter C_1. Considering Figure 11, we have shown in the proof of Theorem 20 that if the automaton $\mathcal{A}$ enters location l_1 with $(x_1, z_1) = (2^{-n}, 1 - 2^{-n})$, it will reach location l_9 with $(x_1, z_1) = (2^{-(n+1)}, 1 - 2^{-(n+1)})$, the values of the 3 other pairs (x_i, z_i) being unchanged. Figure 13 is now used instead of Figure 11 such that z is the only cost variable and μ, μ' are the witness that z is correctly used to modify the pair (x_1, z).

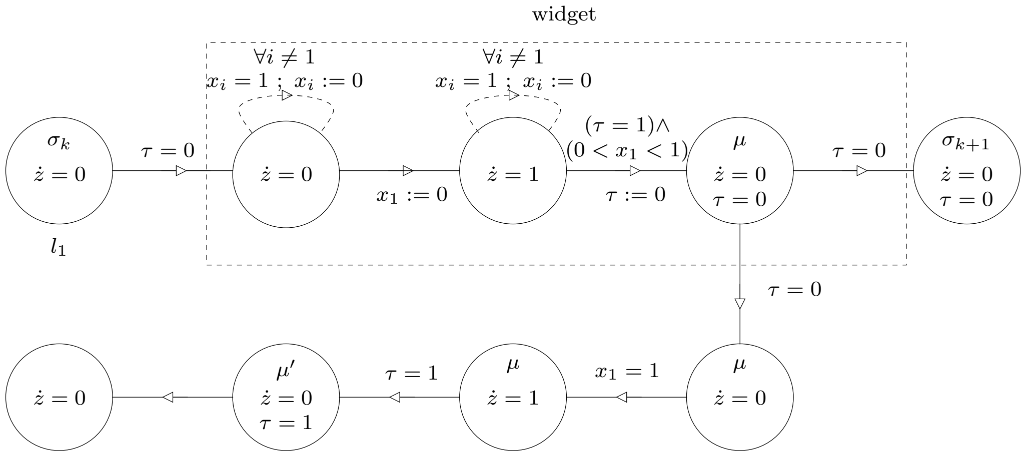

Fig. 13. k: $C_1 := C_1 + 1$ (with the cost variable z)

Assume that in Figure 13, one enters l_1 with $x_1 = 2^{-n}$ and z equal to 0. Then it is easy to replace location l_1 of Figure 13 by a subautomaton in a way that if one enters it with $(x_1, z) = (2^{-n}, 0)$, one leaves it with $(x_1, z) = (2^{-n}, 1 - 2^{-n})$.

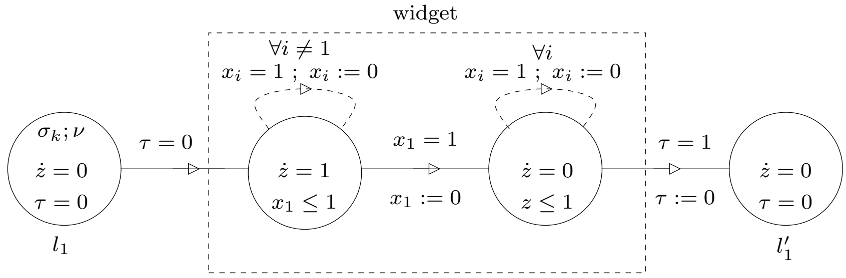

Fig. 14. Modification of the value of (x_1, z) from $(2^{-n}, 0)$ to $(2^{-n}, 1-2^{-n})$

This subautomaton is given on Figure 14. On the later figure, one can verify that if one enters l_1 with $(x_1, z) = (2^{-n}, 0)$, then z is equal to $1-2^{-n}$ when the guard $x_1 = 1$ is satisfied, and thus one reaches l'_1 with $(x_1, z) = (2^{-n}, 1-2^{-n})$. Finally, to impose that z is equal to 0 at location l_1 of Figure 14 is done thanks to the logic, since this is impossible inside the automaton. This means that formula ϕ_1 of (2) is replaced by

$$\phi' \;\equiv\; \nu \Rightarrow z \cdot (\mu \Rightarrow \mu \exists \mathrm{U} (\mu' \wedge z = 1))$$

where ν is a witness that the cost variable z must be reset to 0.

Subautomata for decrementing C_1, incrementing and decrementing C_2 are constructed in a similar way. The same formula ϕ' can be used in each of these cases since it concerns the unique cost variable z. Notice that whereas incrementing or decrementing a counter requires one time unit for their encoding in the proof of Theorem 20, it here requires two time units.

To complete the proof, the final formula φ given in (3) must be replaced by:

$$\varphi \;\equiv\; (\psi_1 \wedge \psi_2 \wedge \phi') \,\exists \mathrm{U}\; \sigma_{k_t}.$$

□

7 Bisimulations of Automata with Stopwatch Observers

In the previous section, we have shown that in the case of dense time, the WCTL_r model-checking problem for automata with stopwatch observers is undecidable (Theorem 20). Looking at the proof of this result, it follows by Corollary 14 that there exist an automaton with stopwatch observers using 5 clocks and 1 cost variable and a WCTL_r formula φ for which any bisimulation respecting the partition $\mathcal{P}_0$ of Proposition 13 is infinite.

In this section, we will identify the *precise frontier* between finite and infinite bisimulations for the class of automata with stopwatch observers. The next

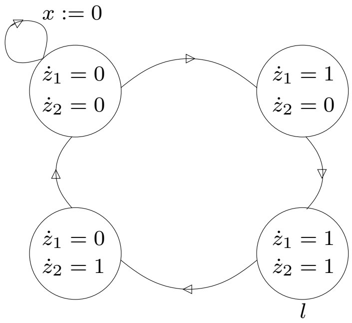

Fig. 15. 1 clock and 2 cost variables.

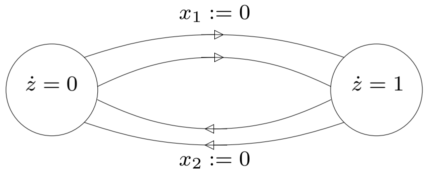

Fig. 16. 2 clocks and 1 cost variable.

theorem states that there are already infinite bisimulations in the case of 1 clock and 2 cost variables, as well as of 2 clocks and 1 cost variable.

Theorem 24 *Let $\mathbb{T} = \mathbb{R}^+$. There exist an automaton with stopwatch observers $\mathcal{A}$ using either 1 clock and 2 cost variables, or 2 clocks and 1 cost variable, and a WCTL$_r$ formula φ, such that no bisimulation respecting the partition $\mathcal{P}_0$ of Proposition 13 is finite.*

PROOF. The two automata that we are going to consider are given in Figures 15 and 16. Note that these automata have several empty edges and no labeling of the locations by atomic propositions.

The proof is based on Procedure `Bisim` and Proposition 12 with the initial partition $\mathcal{P}_0$ given in Proposition 13. Note that Condition 1 of Proposition 13 is trivially satisfied.

Let us begin with the case of 1 clock variable x and 2 cost variables z_1, z_2.

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

As initial partition, instead of the partition $\mathcal{P}_0$ of Proposition 13, we take the partition $\mathcal{P}$ induced by the bisimulation given in Definition 15. The following discussion justifies this choice.

At location of Figure 15 where $\dot z_1 = \dot z_2 = 1$ (we denote this location by l), the behaviour 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 [4], if x, z_1 and z_2 are compared with constant 1, then Procedure `Bisim` leads to the bisimulation $\approx_t$ of Definition 15 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 15 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 $\approx_t$.

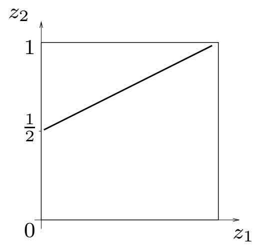

Fig. 17. Region S ($x = 0$).

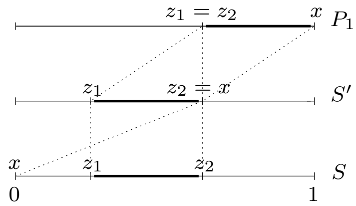

Fig. 18. Its construction.

Let us now show that Procedure `Bisim` applied on partition $\mathcal{P}$ does not terminate because it generates an infinite number of regions R_n, $n \geq 1$ [7], each containing exactly one triple (x, z_1, z_2) such that [8]

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

(a) We need to work with a particular region generated by the procedure (see Figure 17)

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

It is constructed as (see Figure 18)

- $S' = Pre_{>0}(P_1) \cap P_2$ 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 $\dot{z}_1 = \dot{z}_2 = 0$.

Looking at the bold intervals in Figure 18, 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 [9].

(b) The first region $R_1 = \{0, \frac{1}{3}, \frac{2}{3}\}$ is then constructed as (see Figures 19 and 20)

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

Looking at the bold intervals in Figure 20, 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 19, the intersection of R'_1 and S, which is nothing else than $R_1 = Pre_0(R'_1) \cap S$, is the point $(0, \frac{1}{3}, \frac{2}{3})$.

[7] We were able to discover the particular regions R_n with experiments performed with the HyTech tool [14].

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

[9] Notice that P_1, P_2 and P_3 belong to partition $\mathcal{P}$.

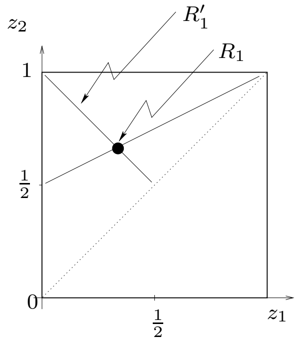

Fig. 19. Region R_1.

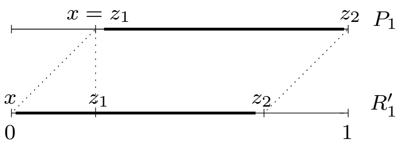

Fig. 20. Its construction.

(c) It remains to explain how to construct R_{n+1} from R_n, assuming that R_n is the point $(0, \frac{1}{3^n}, \frac{3^n+1}{2 \cdot 3^n})$. It is done as follows (see Figures 21 and 22)

- $S'_1 = Pre_0(R_n) \cap P_1$ 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 $\dot{z}_1 = 1$, $\dot{z}_2 = 0$,
- $R_{n+1} = Pre_0(R'_{n+1}) \cap S$.

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 22 (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 21, the intersection of R'_{n+1} and S, which is R_{n+1}, is therefore the point $(0, \frac{1}{3^{n+1}}, \frac{3^{n+1}+1}{2 \cdot 3^{n+1}})$.

This completes the proof of the case of 1 clock variable and 2 cost variables. We now proceed to the case of 2 clock variables and 1 cost variable.

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

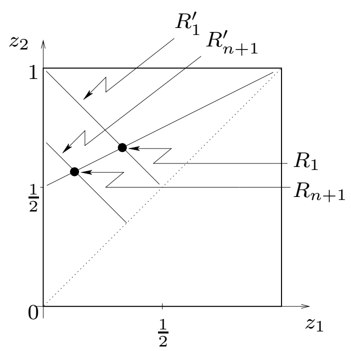

Fig. 21. Region R_{n+1}.

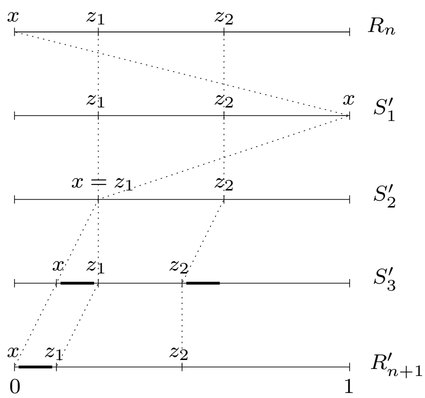

Fig. 22. Its construction from R_n.

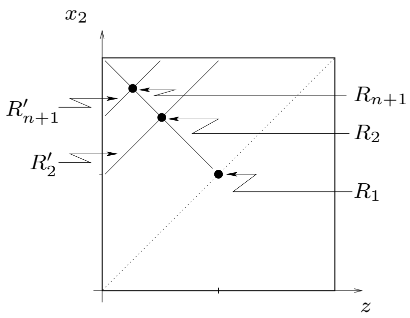

Fig. 23. Region R_{n+1}.

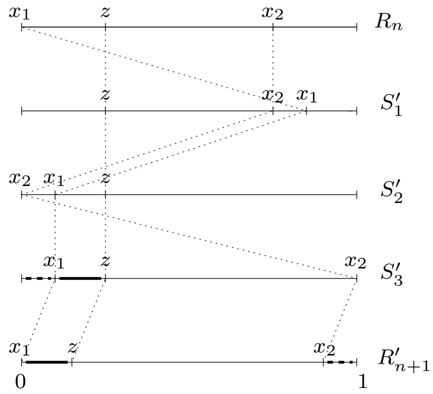

Fig. 24. Its construction from R_n.

The proof for this second case is in the same vein as before; it will be less detailed. As before, we consider the partition $\mathcal{P}$ induced by $\approx_t$ as initial partition. Let us show that Procedure `Bisim` here generates the regions R_n, $n \geq 1$, each formed by the unique triple

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

(a) We first consider the particular region

$$S \ : \quad 0 = x_1 < z < x_2 < 1, \quad x_2 + z = 1$$

constructed as $R = Pre_{>0}(P_1) \cap P_2$ with $P_1 \ : \ 0 < x_1 = z < x_2 = 1$, $P_2 : \ 0 = x_1 < z < x_2 < 1$, and $\dot{z} = 0$. This construction is the same as in Figure 20 except that x_1, z, x_2 respectively replace x, z_1, z_2.

(b) The first region $R_1 = \{0, \frac{1}{2}, \frac{1}{2}\}$ is then constructed as S except that P_2 equals $0 = x_1 < z = x_2 < 1$ (instead of $z < x_2$).

(c) The construction of R_{n+1} from R_n is performed as follows (see Figures 23 and 24)

- $S'_1 = Pre_0(R_n) \cap P_1$ with $P_1 : \ 0 < z < x_2 < x_1 < 1$,
- $S'_2 = Pre_{>0}(S'_1) \cap P_2$ with $P_2 : \ 0 = x_2 < x_1 < z < 1$, and $\dot{z} = 0$,
- $S'_3 = Pre_0(S'_2) \cap P_3$ with $P_3 : \ 0 < x_1 < z < x_2 = 1$,
- $R'_{n+1} = Pre_{>0}(S'_3) \cap P_4$ with $P_4 : \ 0 = x_1 < z < x_2 < 1$, and $\dot{z} = 1$,
- $R_{n+1} = Pre_0(R'_{n+1}) \cap S$.

From the bold and dashed intervals of Figure 24, we see that on R'_{n+1}, we must have $z + (1 - x_2) = \frac{1}{2^n}$. Thus on R_{n+1}, the intersection of this equality with S is the point $(0, 1 - \frac{1}{2^{n+1}}, \frac{1}{2^{n+1}})$. $\square$

From the previous theorem, it follows that the remaining case to fix the precise

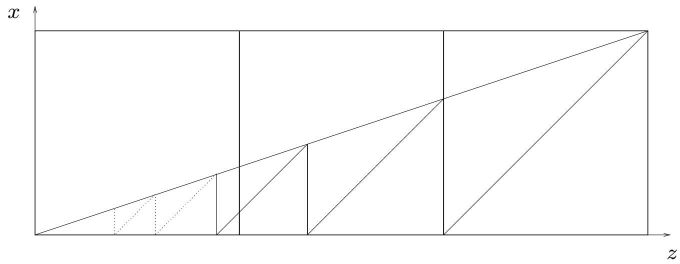

Fig. 25. Infinite bisimulation when $d_1 = 1, d_2 = 3$.

frontier between finite and infinite bisimulations is the case of 1 clock variable and 1 cost variable. Indeed for the case of no cost variable, i.e. the case of timed automata, it is known that they have a finite bisimulation (see Definition 15).

Theorem 25 *Let $\mathbb{T} = \mathbb{R}^+$. Let $\mathcal{A}$ be an automaton with stopwatch observers using 1 clock variable x and 1 cost variable z. Let φ be a WCTL$_r$ formula. Then $\mathcal{A}$ has a finite bisimulation respecting the partition $\mathcal{P}_0$ of Proposition 13.*

PROOF. (Sketch) The proposed bisimulation is the one of Definition 15, where z is treated as a clock. It is not difficult to verify that the conditions of Definition 11 are satisfied. □

The next result follows by Corollary 14.

Corollary 26 *In the case of dense time, the WCTL$_r$ model-checking problem for automata with stopwatch observers using 1 clock variable and 1 cost variable is decidable.* [10]

Comments. All the results of this section 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 variable x and 1 cost variable 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 [13] (see Figure 25). If for automata with 1 clock x and 1 cost variable 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 15, where z is treated as a clock and each diagonal $z - x = c$ is replaced by $z - dx = c$ (see Figure 26). Note 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 4, z_3 is such a variable. The required finite bisimulation is a direct product of the bisimulation given before for x and z with

[10] This result also holds for the WCTL logic, since when there is only 1 cost variable, the two logics WCTL and WCTL$_r$ are equivalent.

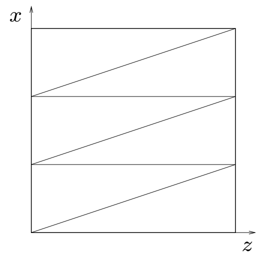

Fig. 26. Finite bisimulation when $d = 3$.

the bisimulation of Definition 15 applied to the variables $z_2, \ldots, z_m$ treated as clocks.

8 Conclusion

In this paper, we have studied the model-checking problem for weighted timed automata and the WCTL logic. We have also studied the subclass of automata with stopwatch observers and the slight restriction WCTL$_r$.

We have obtained several results, most of them for automata with stopwatch observers, that are recalled on Figure 27. The WCTL model-checking problem is undecidable in discrete and dense time, already for automata with stopwatch observers using 1 clock and 3 cost variables (Theorem 9). For WCTL$_r$ and discrete time, the model-checking problem becomes decidable with a complexity in PSpace because weighted timed automata all have finite bisimulations (Theorem 18 and Corollary 19). However, in dense time, the WCTL$_r$ model-checking problem remains undecidable. The undecidability already holds for automata with stopwatch observers using 5 clocks and 1 cost variable (Corollary 23) [11]. This later result is interesting since it indicates an undecidability result, whereas the minimum-cost reachability problem is decidable for weighted timed automata with 1 cost variable [7] [8]. In dense time, the precise frontier between finite and infinite bisimulations of automata with stopwatch observers is the following one: *(i)* finite bisimulations in the case of 1 clock and 1 cost variable [12] (Theorem 25), *(ii)* infinite bisimulations in the case of 1 clock and 2 cost variables, as well as for 2 clocks and 1 cost variable (Theorem 24). It follows that in the particular case of automata with stopwatch observers equipped with only 1 clock and 1 cost variable, the WCTL$_r$ model-checking problem is decidable (Corollary 26). It was a difficult task to obtain Theorem 20. Historically, we have first proved Theorem 24 in [11], and this was already difficult since stopwatches can be neither reset nor tested in

[11] Recently in [10] the authors were able to prove the same result with only 3 clocks and 1 cost variable.

[12] and of course in the case of any number of clocks and no cost variable, i.e. of timed automata.

Time	**Logic**	**Clocks**	**Stopw.**	**Bisim.**	**Mod.-Check.**
Discrete	WCTL	1	3	infinite	undecidable
	WCTL_r	any	any (costs)	finite	decidable
Dense	WCTL	1	3	infinite	undecidable
	WCTL_r	1	1	finite	decidable
		1	2	infinite	?
		2	1	infinite	?
		5	1	infinite	undecidable

Fig. 27. Summary of the results.

the automata. After, thanks to our knowledge of the infinite bisimulations we have constructed, we were able to prove Theorem 20.

As mentioned on Figure 27, several problems are left open in dense time. What is the precise frontier between decidability and undecidability of the model-checking problem for automata with stopwatch observers and the WCTL_r logic? Similarly for weighted timed automata and the WCTL logic? For which fragments of WCTL_r or WCTL is the model-checking problem decidable?

References

[1] R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[2] R. Alur, C. Courcoubetis, and T. A. Henzinger. Computing accumulated delays in real-time systems. In *CAV'93: Computer Aided Verification*, volume 697 of *Lecture Notes in Computer Science*, pages 181–193. Springer, 1993.

[3] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer-Verlag, 1993.

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

[5] R. Alur, T. A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.

[6] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971–984, 2000.

[7] R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. In *HSCC'01: Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2001.

[8] G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. W. Vaandrager. Minimum-cost reachability for priced timed automata. In *HSCC'01: Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

[9] A. Bouajjani, R. Echahed, and J. Sifakis. On model checking for real-time properties with durations. In *LICS'93: Logic in Computer Science*, pages 147–159. IEEE Computer Society Press, 1993.

[10] P. Bouyer, T. Brihaye, and N. Markey. Improved undecidability results on priced timed automata. Research Report LSV-05-18, Laboratoire Spécification et Vérification, ENS Cachan, France, August 2005. 16 pages.

[11] T. Brihaye, V. Bruyère, and J.-F. Raskin. Model-Checking for Weighted Timed Automata. In *FORMATS-FTRTFT'04: Joint International Conferences on Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 3253 of *Lecture Notes in Computer Science*, pages 277–292. Springer, 2004.

[12] T.A. Henzinger. Hybrid automata with finite bisimulations. In *ICALP'95: Automata, Languages, and Programming*, volume 944 of *Lecture Notes in Computer Science*, pages 324–335. Springer-Verlag, 1995.

[13] T.A. Henzinger. The theory of hybrid automata. In *LICS'96: Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.

[14] 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*, volume 1019 of *Lecture Notes in Computer Science*, pages 41–71. Springer-Verlag, 1995.

[15] T.A. Henzinger and P.W. Kopke. Discrete-time control for rectangular hybrid automata. In *ICALP'97: Automata, Languages, and Programming*, volume 1256 of *Lecture Notes in Computer Science*, pages 582–593. Springer-Verlag, 1997.

[16] 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.

[17] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Decidable integration graphs. *Information and Computation*, 150(2):209–243, 1999.

[18] K. G. Larsen and J. I. Rasmussen. Optimal conditional reachability for multi-priced timed automata. In *FoSSaCS'05: Foundations of Software Science and Computational Structures*, volume 3441 of *Lecture Notes in Computer Science*, pages 234–249. Springer-Verlag, 2005.

[19] S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Form. Methods Syst. Des.*, 18:25–68, 2001.