Improved Undecidability Results on Weighted Timed Automata

Patricia Bouyer [a], Thomas Brihaye [b], Nicolas Markey [a]

[a] *Lab. Spécification & Vérification, CNRS & ENS de Cachan, France*

[b] *Univ. Mons-Hainault, Centre Fédéré en Vérification, Belgium*

Abstract

In this paper, we strengthen two recent undecidability results of [9,10] about weighted timed automata, an extension of timed automata with cost variables. More precisely, we propose new encodings of a Minsky machine that only require three clocks and one stopwatch cost, while they required five clocks and one stopwatch cost in [9,10].

Key words: Weighted timed automata, weighted timed games, undecidability

1 Introduction

Weighted timed automata (WTA for short), or *priced timed automata*, have been defined in 2001 independently by Alur *et al.* [5] and Larsen *et al.* [6] for modelling resource consumption in timed systems. They extend classical timed automata (TA) [3] with cost information on both locations and edges. These costs increase while time elapses, but are never tested in the automaton. An interesting problem is then to compute the optimal cost for reaching a given state. In [5,6], this problem (called *optimal reachability*) is proved decidable.

In order to express more involved properties, the logic WCTL has been proposed as an extension of TCTL [2] in which cost variables can be constrained [11]. Model-checking this logic is undecidable in general for the classical dense-time semantics [11,9]. Games played on WTAs with an optimality criterion have been considered in [12,1,7,10] and though partial decidability results have been obtained in [1,7], it has recently been proved in [10] that the general problem of finding optimal strategies in such a game is undecidable. In this paper, we improve both undecidability results mentioned above: our encodings are simpler, and above all, the reductions use only three clocks, instead of five in [9,10].

Email addresses: `bouyer@lsv.ens-cachan.fr` (Patricia Bouyer), `thomas.brihaye@umh.ac.be` (Thomas Brihaye), `markey@lsv.ens-cachan.fr` (Nicolas Markey).

2 Preliminaries

In the sequel, X is a finite set of clocks and AP is a finite set of atomic propositions. We adopt standard notations for TAs and refer to [3] for classical definitions. A WTA is a tuple $(\mathsf{Loc}, \mathsf{Edg}, \mathsf{Lab}, \mathsf{cost})$ where $(\mathsf{Loc}, \mathsf{Edg}, \mathsf{Lab})$ is a TA and $\mathsf{cost}\colon \mathsf{Loc} \cup \mathsf{Edg} \to \mathbb{N}^d$ is an extra *cost function*. A *run* of a WTA is a run of the underlying TA, having both *continuous* (time-elapsing) and *discrete* transitions. Given a run ρ and a state p along that run, we write $\rho[p]$ for the finite prefix of ρ ending at p. Let ρ be the finite run $(\ell_0, \alpha_0) \xrightarrow{\gamma_0} (\ell_1, \alpha_1) \xrightarrow{\gamma_1} (\ell_2, \alpha_2) \cdots \xrightarrow{\gamma_{k-1}} (\ell_k, \alpha_k)$ where $\gamma_i \in \mathbb{R}_+$ for continuous transitions and $\gamma_i \in \mathsf{Edg}$ for discrete ones. We define

$$\mathsf{cost}(\rho) = \sum_{i \leq k, \gamma_i \in \mathbb{R}_+} \mathsf{cost}(\ell_i) \cdot \gamma_i + \sum_{i \leq k, \gamma_i \in \mathsf{Edg}} \mathsf{cost}(\gamma_i).$$

Our constructions only involve one cost, we thus always have $d = 1$ in the sequel. A *stopwatch cost* is a cost cost s.t. $\mathsf{cost}(\mathsf{Loc}) \subseteq \{0, 1\}$.

We consider the logic WCTL [1], a branching-time logic close to TCTL [2] and ICTL [4]. It is built over AP with boolean combinators, and with two families of modalities $\mathbf{E}\ \mathbf{U}_{\sim c}$ and $\mathbf{A}\ \mathbf{U}_{\sim c}$, where $\sim\, \in \{<, \leq, =, >, \geq\}$, and $c \in \mathbb{N}$. Let $\xi \in \mathsf{WCTL}$, $\mathcal{A}$ be a WTA and q be a state of $\mathcal{A}$. That ξ holds in $\mathcal{A}$ at state q, denoted by $\mathcal{A}, q \models \xi$, is defined in the standard way for atomic propositions and boolean combinators, and by:

- $\mathcal{A}, q \models \mathbf{E}\,(\varphi\, \mathbf{U}_{\sim c}\, \psi)$ iff there exists a run $\rho = (q_i)_{i \geq 0}$ in $\mathcal{A}$ with $q = q_0$, and a position p in ρ such that $\mathcal{A}, p \models \psi$ and $\mathsf{cost}(\rho[p]) \sim c$ and $\mathcal{A}, p' \models \varphi$ for all positions $p' \neq p$ of $\rho[p]$;
- $\mathcal{A}, q \models \mathbf{A}\,(\varphi\, \mathbf{U}_{\sim c}\, \psi)$ iff for any infinite run $\rho = (q_i)_{i \geq 0}$ in $\mathcal{A}$ with $q = q_0$, there exists a position p in ρ such that $\mathcal{A}, p \models \psi$ and $\mathsf{cost}(\rho[p]) \sim c$ and $\mathcal{A}, p' \models \varphi$ for all positions $p' \neq p$ of $\rho[p]$.

In the sequel, we might omit to mention $\mathcal{A}$ when it is clear from the context, and simply write $q \models \varphi$.

3 Encoding the Counters

We now explain the undecidability proof for WCTL model-checking. Let $\mathcal{M}$ be a two-counter machine [13]. We build a WTA $\mathcal{A}_\mathcal{M}$ (with three clocks and one stopwatch cost) and a WCTL-formula Φ such that given q_0, a well-chosen state of $\mathcal{A}_\mathcal{M}$, we have that $\mathcal{M}$ halts if, and only if, $q_0 \models \Phi$. The two counters c_1 and c_2 will be encoded alternately by three clocks x, y and z. The value of c_1 is encoded by $x_1 = 1/2^{c_1}$ (with $x_1 \in \{x, y, z\}$) whereas the value of c_2 is encoded by $x_2 = 1/3^{c_2}$ (with $x_2 \in \{x, y, z\}$). To each instruction will be associated six modules, one for each injective function $\{x_1, x_2\} \to \{x, y, z\}$.

[1] WCTL stands for "Weighted CTL".

3.1 Incrementation of a counter

We consider the following instruction of the two-counter machine:

$$p_i : \ c_1 := c_1 + 1; \ \texttt{goto} \ p_j.$$

We also assume that the initial value of c_1 is stored in clock x whereas that of c_2 is stored in y. To p_i, we associate the automaton $\mathsf{Aut}^i_{1,+}(x, y, z)$ as in Fig. 1. In that figure (and in all the other ones), costs which are omitted are equal to zero. The subscript $1, +$ is a remainder that instruction p_i deals with counter stored in the first clock (x here) and is an incrementation (we might omit it when it is not necessary), the tuple (x, y, z) indicates which clocks encode counters c_1 and c_2: here, c_1 is initially stored in x and c_2 is initially stored in y. At the end of this module, the new values of c_1 and c_2 are stored in z and y, resp.; that's why we swap x and z when leaving the module (transition from $D^i_{x,y,z}$ to $A^j_{z,y,x}$).

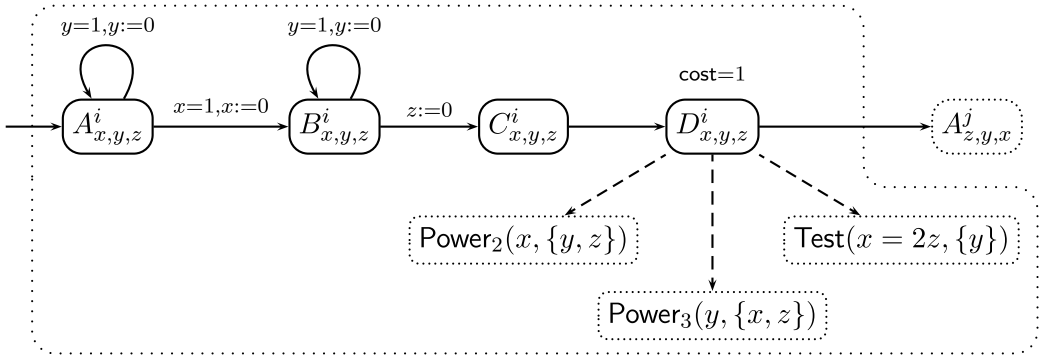


Fig. 1. Automaton $\mathsf{Aut}^i_{1,+}(x, y, z)$

For that automaton to really increment the first counter, we will enforce the following requirements (see Section 5): (1) the delay between arrival in $A^i_{x,y,z}$ and arrival in $D^i_{x,y,z}$ is 1 t.u., (2) when entering $D^i_{x,y,z}$, z equals $x/2$ and (3) the delay elapsed in $D^i_{x,y,z}$ is 0.

The last point will be ensured through a global WCTL-formula stating that no cost is accumulated in location $D^i_{x,y,z}$. The second point is obtained by a module $\mathsf{Test}(x = 2z, \{y\})$, together with a WCTL-formula φ_1 (see Section 4.2 for details on that module). Finally, according to Lemma 1 below, the first point is enforced by checking that the values of x and y when entering $D^i_{x,y,z}$ are $1/2^n$ and $1/3^m$ for some integers n and m. Those conditions are ensured by modules Power_2 and Power_3 and the associated formulas φ_2 and φ_3, whose construction is explained in Section 4.3.

Lemma 1 *If a run enters location $A^i_{x,y,z}$ with $x = 1/2^{c_1}$, $y = 1/3^{c_2}$ and enters location $D^i_{x,y,z}$ t time units later with the value of x of the form $1/2^n$ for some n, and the value of y of the form $1/3^m$ for some m, then $t = 1$, $n = c_1$ and $m = c_2$.*

This lemma can easily be proved using elementary arithmetical manipulations. It plays a crucial role in our reduction: it explains how comparing clocks to powers of 1/2 and 1/3 gives a way to measure exactly 1 t.u., and thus why we encode the counters as powers of 1/2 and 1/3. Note that 2 and 3 could be replaced by any two relatively prime numbers.

Similar ideas can be used for designing an automaton $\mathsf{Aut}^i_{2,+}(x,y,z)$ that increments the second counter (*i.e.* ends up with $z = y/3$, while x returns to its original value), involving module $\mathsf{Test}(x = 3z, \{y\})$.

3.2 Decrementation of a counter

We now treat instruction: $p_i:$ `if` $(c_1 > 0)$ `then` $c_1 := c_1 - 1;$ `goto` p_j `else goto` p_k. We only give the construction of automaton $\mathsf{Aut}^i_{1,-}(x,y,z)$, which is a slight variation of the previous construction. This automaton implements the decrementation of the first counter, initially stored in x, unless it equals zero.

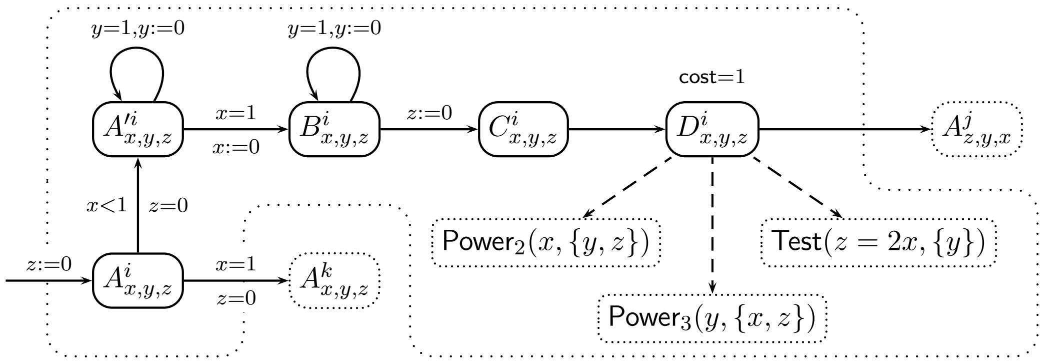


Fig. 2. Automaton $\mathsf{Aut}^i_{1,-}(x,y,z)$

In the global reduction, we will enforce the following properties: (1) the values of x and y when entering $D^i_{x,y,z}$ are $1/2^n$ and $1/3^m$ for some n and m, (2) when entering $D^i_{x,y,z}$, z equals $2x$ and (3) the delay in $D^i_{x,y,z}$ is 0. As previously, we can prove that these three conditions express correctness of the construction. Lemma 1 clearly also holds for $\mathsf{Aut}^i_{1,-}(x,y,z)$. Automaton $\mathsf{Aut}^i_{2,-}(x,y,z)$ is built in the same way.

4 Modules

4.1 Adding x or $1-x$ to the cost variable, where x is a clock

Following [10], we build modules $\mathsf{Add}^+(x,\{z\})$ and $\mathsf{Add}^-(x,\{z\})$, displayed on Fig. 3 and 4. Those automata clearly satisfy the following Lemma:

Lemma 2 *If a run enters location ℓ_0 of $\mathsf{Add}^+(x,\{z\})$ (resp. $\mathsf{Add}^-(x,\{z\})$) with $x = \alpha_0 \in [0,1]$, $y = \beta_0 \in [0,1]$ and $\mathsf{cost} = \gamma_0$, it then leaves location ℓ_1 with the same values for x and y, and with $\mathsf{cost} = \gamma_0 + \alpha_0$ (resp. $\mathsf{cost} = \gamma_0 + 1 - \alpha_0$).*

4.2 Checking $y = 2x$

Module $\mathsf{Test}(y = 2x, \{z\})$ is the (deterministic) automaton displayed on Fig. 5. It sets the cost to $2x + 1 - y$. Let $\varphi_1 = S \wedge \mathbf{E}\,\mathbf{F}_{\leq 1}\, T \wedge \mathbf{E}\,\mathbf{F}_{\geq 1}\, T$. The following Lemma clearly holds:

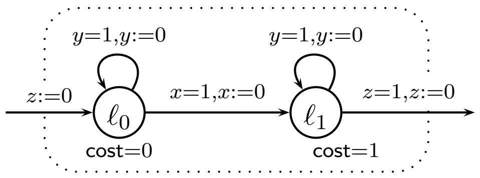


Fig. 3. Automaton $\mathsf{Add}^+(x, \{z\})$

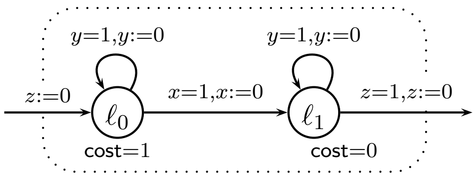


Fig. 4. Automaton $\mathsf{Add}^-(x, \{z\})$

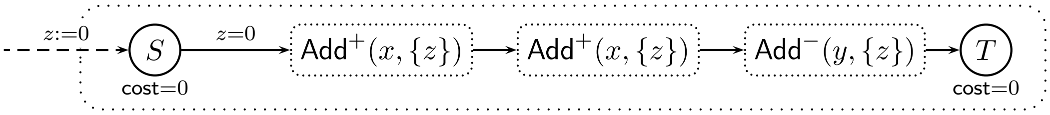


Fig. 5. Automaton $\mathsf{Test}(y = 2x, \{z\})$

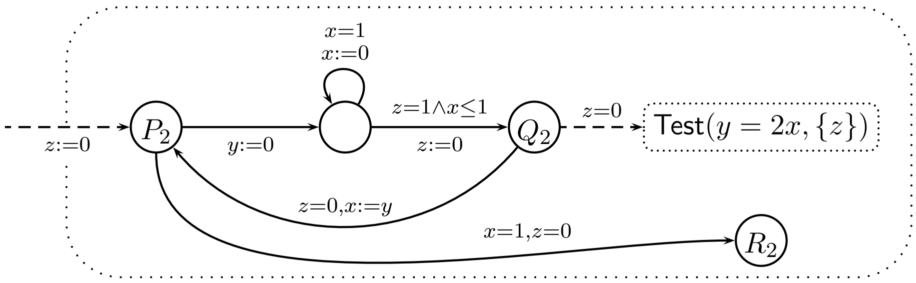


Fig. 6. Automaton $\mathsf{Power}_2(x, \{y, z\})$

Lemma 3 *Formula φ_1 holds in S along module* $\mathsf{Test}(y = 2x, \{z\})$ *with $x = \alpha_0 \in [0, 1]$ and $y = \beta_0 \in [0, 1]$ if, and only if, $\beta_0 = 2\alpha_0$.*

This construction can easily be adapted for other tests, especially for building a module $\mathsf{Test}(y = 3x, \{z\})$ testing if $y = 3x$.

4.3 Checking that the value of x is of the form $1/2^d$

Module $\mathsf{Power}_2(x, \{y, z\})$ is displayed on Fig. 6. Note that it requires two auxiliary clocks. Note also that it uses an update "$x := y$", instead of classical resets. This is for the sake of simplicity, as the module could be adapted (by duplicating the periodic part, involving no extra clock) in order to only have standard resets [8].

We let $\varphi_2 = P_2 \wedge \mathbf{E}\,((Q_2 \rightarrow \mathbf{E}\,(Q_2\,\mathbf{U}\,\varphi_1))\,\mathbf{U}\,R_2)$. We have the following Lemma:

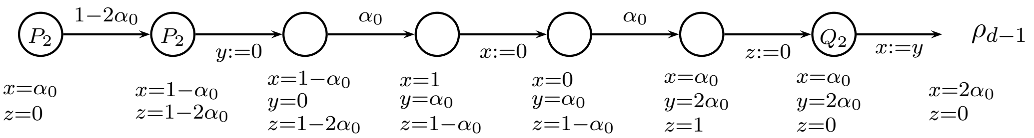
Lemma 4 *Formula φ_2 holds in P_2 in module* $\mathsf{Power}_2(x, \{y, z\})$ *with $x = \alpha_0 \in (0, 1]$ if, and only if, there exists a non-negative integer d s.t. $\alpha_0 = 1/2^d$.*

PROOF. Let q_0 be the configuration $(P_2, \langle \alpha_0, -, 0 \rangle)$ when entering P_2 for the first time. Assume that $\mathsf{Power}_2(x, \{y, z\}), q_0 \models \varphi_2$, and pick a run ρ witnessing this property, *i.e.*, starting from P_2, reaching R_2, and s.t. intermediate positions satisfy $Q_2 \rightarrow \mathbf{E}\,(Q_2\,\mathbf{U}\,\varphi_1)$ in $\mathsf{Power}_2(x, \{y, z\})$. Since time cannot elapse in Q_2, Lemma 3 ensures that, when entering Q_2, the value of y is always twice the value of x.

Let n be the number of times ρ enters the location Q_2. If $n = 0$, then $\alpha_0 = 1 = 2^0$, as required. Now, assume $n > 0$. Clearly, clock x has the same value when ρ enters Q_2

as the previous time it entered P_2, provided this value is in $(0,1]$. Since $y = 2x$ when entering Q_2, it can easily be proved by induction that, when entering Q_2 for the k-th time, with $k \leq n$, then $x = 2^{k-1}\alpha_0$. Thus, the last time run ρ enters Q_2, we have $x = 2^{n-1}\alpha_0$, and $y = 2^n\alpha_0$. From that point on, ρ must go to location R_2 without entering Q_2 any more. This requires that the last value of y in Q_2 is 1. Thus $\alpha_0 = 1/2^n$.

Conversely, if there exists a non-negative integer d s.t. $\alpha_0 = 1/2^d$ we have to prove that $\mathsf{Power}_2(x, \{y, z\}), q_0 \models \varphi_2$. We build by induction a run ρ_d witnessing this fact. If $d = 0$, we take $\rho_d = (P_2, \langle 1, -, 0\rangle) \rightarrow (R_2, \langle 1, -, 0\rangle)$. Otherwise, assume we can build a run ρ_{d-1} from $(P_2, \langle 1/2^{d-1}, -, 0\rangle)$ to $(R_2, \langle 1, -, 0\rangle)$. We build ρ_d as follows:



Clearly, the paths ρ_d are paths of $\mathsf{Power}_2(x, \{y, z\})$ and satisfy φ_2. $\square$

It is easy to adapt this construction in order to build a module $\mathsf{Power}_3(x, \{y, z\})$ and a formula φ_3 that check if x is of the form $1/3^d$, for some integer d.

5 Global Reduction

We build the global automaton $\mathcal{A}_{\mathcal{M}}$ inductively using sub-automata $\mathsf{Aut}^i_{c,+}$ and $\mathsf{Aut}^i_{c,-}$ as explained previously. To the halting instruction corresponds a unique location A_{Halt}, labelled with Halt. The initial location is the state $A^1_{x,y,z}$ denoted A^1 for short. In the sequel, a state of $\mathcal{A}_{\mathcal{M}}$ is written $(A, \langle x, y, z\rangle)$, where A is the location and $\langle x, y, z\rangle$ is the valuation of x, y and z, in that order. Thus, the initial configuration of $\mathcal{M}$ is encoded by $q_0 = (A^1, \langle 1, 1, 0\rangle)$. We set a new atomic proposition D which is true in all states $D^i_{\sigma(x,y,z)}$, for any permutation σ.

As explained in Section 4, our modules (Test, Power_2 and Power_3) require that some WCTL formulas (namely φ_1, φ_2 and φ_3) hold in their initial state in order to really play their roles. This will be ensured in $\mathcal{A}_{\mathcal{M}}$ through the following formula: $\Phi = \mathbf{E}\ [(D \rightarrow \varphi)\ \mathbf{U}_{\leq 0}\ \mathsf{Halt}]$ where $\varphi = \bigwedge_{i=1,2,3} \mathbf{E}\,(D\,\mathbf{U}_{\leq 0}\,\varphi_i)$.

Lemma 5 $\mathcal{A}_{\mathcal{M}}, q_0 \models \Phi$ *iff the two-counter machine* $\mathcal{M}$ *has a halting computation.*

PROOF. First assume $q_0 \models \Phi$, and pick a run ρ, starting in q_0 and witnessing Φ, *i.e.*, reaching state Halt with cost 0, and such that intermediate positions satisfy $D \rightarrow \varphi$ in $\mathcal{A}_{\mathcal{M}}$. As the cost rate in all D-states is 1, and the overall cost of ρ is 0, no time can elapse in D-state. Also, each time ρ is in a D-state, φ holds. Consider a D-state $(D^i_{y,x,z}, \langle x_D, y_D, z_D\rangle)$ along ρ (the case of other permutations of (x, y, z) would be treated similarly). In that state:

- formula $\mathbf{E}\,(D\,\mathbf{U}_{\leq 0}\,\varphi_2)$ holds, where $\varphi_2 = P_2 \wedge \mathbf{E}\,((Q_2 \rightarrow \mathbf{E}\,(Q_2\,\mathbf{U}\,\varphi_1))\,\mathbf{U}\,R_2)$ (see Section 4.3). This means that this is possible to immediately enter module

$\mathsf{Power}_2(y, \{x, z\})$ and satisfy φ_2. Lemma 4 then ensures that y_D equals $1/2^n$, for some integer n.
- similarly, formula $\mathbf{E}\,(D\,\mathbf{U}_{\leq 0}\,\varphi_3)$ and module $\mathsf{Power}_3(x, \{y, z\})$ ensure that x_D equals $1/3^m$ for some integer m.
- formula $\mathbf{E}\,(D\,\mathbf{U}_{\leq 0}\,\varphi_1)$ holds, where $\varphi_1 = S \wedge \mathbf{E}\,\mathbf{F}_{\leq 1}\,T \wedge \mathbf{E}\,\mathbf{F}_{\geq 1}\,T$ (see Section 4.2). Thus, it is possible to immediately (with clock values y_D, x_D and z_D) enter module $\mathsf{Test}(y = 2z, \{x\})$ (or another Test module, depending on instruction i) and satisfy φ_1. According to Lemma 3, this ensures that the corresponding test holds, *i.e.* $y_D = 2z_D$ (for the case of $\mathsf{Test}(y = 2z, \{x\})$).

It follows that:

- if ρ enters $\mathsf{Aut}^i_{1,+}(y, x, z)$, say, with clock valuation $\langle 1/3^m, 1/2^n, -\rangle$, then, according to Lemma 1 (whose hypotheses hold, according to the remarks above), it reaches $D^i_{y,x,z}$ (and then the next sub-automaton $\mathsf{Aut}^j(z, x, y)$) after exactly 1 t.u. with clock valuation $\langle 1/3^m, -, 1/2^{n+1}\rangle$.
- if ρ enters $\mathsf{Aut}^i_{1,-}(y, x, z)$ with $y = 1$, then it immediately enters the next sub-automaton, without letting time elapse.
- if ρ enters $\mathsf{Aut}^i_{1,-}(y, x, z)$ with valuation $\langle 1/3^m, 1/2^n, -\rangle$, assuming $n \neq 0$, then we can again apply Lemma 1, which claims that ρ reaches $D^i_{y,x,z}$ (and then the next automaton $\mathsf{Aut}^j(z, x, y)$) after exactly 1 t.u. and with valuation $\langle 1/3^m, -, 1/2^{n-1}\rangle$.

By induction, whenever ρ enters the first location of a sub-automaton $\mathsf{Aut}^i(\alpha, \beta, \gamma)$, for any permutation (α, β, γ) of (x, y, z), then $\alpha = 1/2^n$ and $\beta = 1/3^m$, for some integers n and m. According to Φ, ρ eventually enters state A_{Halt}. In the meantime, it traverses a (finite) sequence $(\mathcal{A}_k)_k$ of sub-automata of the form $\mathsf{Aut}^i(\alpha, \beta, \gamma)$. Thus, to ρ, we can associate a sequence of tuples $p_k = (i_k, c_{1,k}, c_{2,k})$ as follows: (a) i_k is the index i of the sub-automaton $\mathcal{A}_k$, (b) $c_{1,k}$ is the integer s.t. $\alpha = 1/2^{c_{1,k}}$ when ρ enters $\mathcal{A}_k$, and (c) $c_{2,k}$ is the integer s.t. $\beta = 1/2^{c_{2,k}}$ when ρ enters $\mathcal{A}_k$. Quite obviously, our construction ensures that the values of the counters between p_k and p_{k+1} are updated according to instruction i_k of $\mathcal{M}$. The sequence $(p_k)_k$ thus corresponds to a halting computation of $\mathcal{M}$.

Conversely, if $\mathcal{M}$ has a halting computation, we can exactly mimic this computation with a run in $\mathcal{A}_{\mathcal{M}}$. The arguments are similar to the ones above in order to prove that this run satisfies Φ. □

Theorem 6 *Model-checking* WCTL *on WTAs with three clocks and one stopwatch cost is undecidable.*

Note that our reduction holds for a restriction of WCTL not involving equality-constraints, and involves only a *stopwatch* cost.

6 Application to Optimal Reachability Timed Games

Optimal reachability timed games have been first introduced in [12] and further studied in [1,7,10]. We refer to the above papers for formal definitions.

A *weighted timed game* (WTG) is a WTA with a distinguished set of winning states, and where the set of actions is split into controllable actions (played by the *controller*) and uncontrollable actions (played by the *environment*). We assume a classical definition of strategies, and the aim of a game is, for the controller, to enforce winning states and to minimize the cost of the plays, whatever does the environment. To illustrate these notions, we better give an example.

Example 7 ([7]) *We consider the WTG in Fig. 7. Dashed (resp. plain) arrows are for uncontrollable (resp. controllable) actions. Depending on the choice of the environment (going to location ℓ_2 or ℓ_3), the accumulated cost along plays of the game is either $5t + 10(2 - t) + 1$ (through ℓ_2) or $5t + (2 - t) + 7$ (through ℓ_3) when t is the delay in state ℓ_0. The optimal cost the controller can ensure is thus $\inf_{t\leq 2} \max(5t + 10(2 - t) + 1, 5t + (2 - t) + 7) = 14 + 1/3$, and the optimal delay is then $t = 4/3$. The optimal strategy for the controller is thus to wait in state ℓ_0 until $x = 4/3$, and then enter state ℓ_1. Then, the environment chooses to go either to ℓ_2 or to ℓ_3, and finally as soon as $x = 2$, the controller goes to state* Win.

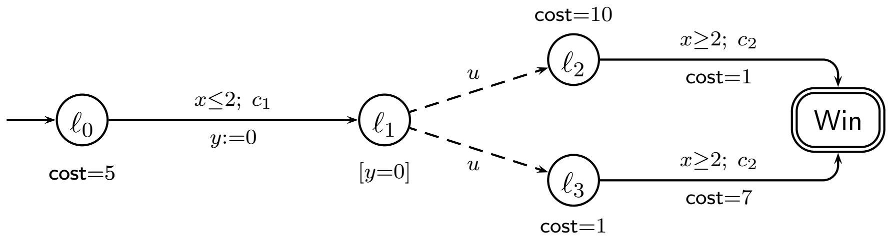


Fig. 7. A weighted timed game

This example indicates that the region partitioning (of [3]) is not sufficient for solving optimal WTGs. Restricted decidability results have however been obtained in [1,7], but the general problem has been recently proved undecidable [10]. This result relies on a reduction which uses five clocks. A construction similar to that of WCTL can be used to get a reduction with only three clocks, we sketch it now.

Given a two-counter machine $\mathcal{M}$ we construct a WTG $\mathcal{G}_{\mathcal{M}}$ with one cost variable cost. The shape of the automaton is similar to the one described in Section 3, we only point out the few differences:

- When arriving in state A_{Halt} we add a discrete cost 3;
- All arrows leading to a test module (dashed on the figures) are uncontrollable;
- The module for checking that $y = 2x$ is split into two branches, one setting the cost to $2 + 2x + (1 - y)$, and the other to $1 + 2(1 - x) + y$ (these two branches are slight adaptations of Fig. 5 and transitions leading to one or the other branch are uncontrollable). If the relation $y = 2x$ does not hold, the environment has a strategy to set the cost up to a value strictly greater than 3 (if $y < 2x$, he takes the branch storing $2 + 2x + (1 - y)$ in the cost, otherwise he takes the other branch). If the relation $y = 2x$ holds, then whatever branch chooses the environment, the accumulated cost will be exactly 3, and the controller will win the game;
- The module for checking that $y = 3x$ is similar, and has two branches, one setting the cost to $2+3x+(1-y)$, and the other one setting the cost to $3(1-x)+y$. Thus, if the relation $y = 3x$ does not hold, the environment has a strategy to set the

cost up to a value strictly greater than 3 (if $y < 3x$, he takes the branch storing $2 + 3x + (1 - y)$ in the cost, otherwise he takes the other branch). If the relation $y = 3x$ holds, then whatever branch chooses the environment, the accumulated will be exactly 3, and the controller will win the game;

- The modules Power_2 and Power_3 are similar to the one for WCTL (tests $y = 2x$ (resp. $y = 3x$) are done as described above). In this module, if the controller cheats, that's because at some point of the loop he does not satisfy $y = 2x$ (resp. $y = 3x$), which can be detected by the environment going to the corresponding test module, or that's because he will not be able to reach a location labelled by R_2 (or R_3). In the first case, a state labelled by T will be reached (the play will thus be winning), but the cost will be strictly greater than 3, whereas in the second case, the play will not be winning.

Following the lines of [10] and our previous proof for WCTL: $\mathcal{M}$ halts iff the controller has a winning strategy in $\mathcal{G}_{\mathcal{M}}$ to enforce one of the states labelled by $\{\mathsf{Halt}, T, R_2, R_3\}$ with cost less than or equal to 3. This is true because if the environment does not do any uncontrollable action, then the controller will have to never wait in states with a positive cost (otherwise the global cost will be strictly greater than 3). If the environment does an uncontrollable action, it means that he wants to check that the controller has played correctly, and if (and only if) the latter has really played correctly, he will be able to reach a state labelled by T, R_2 or R_3 with cost less than or equal to 3. Thus:

Theorem 8 *The problem of deciding whether there exists a winning strategy with cost less than or equal to a given value in a WTG with three clocks and one stopwatch cost is undecidable.*

7 Conclusion

In this paper, we have improved two undecidability results of [9,10] by decreasing the number of clocks used in the reductions: both WCTL model-checking and optimal timed games are undecidable with three clocks and one stopwatch cost. These bounds are quite tight, as the same problems with only one clock are decidable. We did not manage to close the gap, with WTAs having two clocks, even when trying to encode both counter with only one clock $x = 1/(2^{c_1}3^{c_2})$. But on the other hand, for priced timed automata with two clocks and one stopwatch, the coarsest bisimulation has in general an infinite index [11].

References

[1] R. Alur, M. Bernadsky, and P. Madhusudan. Optimal reachability in weighted timed games. In *Proc. 31st Int. Coll. Automata, Languages and Programming (ICALP'04)*, vol. 3142 of *LNCS*, pages 122–133. Springer, 2004.

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

[3] R. Alur and D. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126(2):183–235, 1994.

[4] R. Alur, Th. A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. Soft. Engin.*, 22(3):181–201, 1996.

[5] R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. In *Proc. 4th Int. Workshop Hybrid Systems: Computation and Control (HSCC'01)*, vol. 2034 of *LNCS*, pages 49–62. Springer, 2001.

[6] G. Behrmann, A. Fehnker, Th. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *Proc. 4th Int. Workshop Hybrid Systems: Computation and Control (HSCC'01)*, vol. 2034 of *LNCS*, pages 147–161. Springer, 2001.

[7] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *Proc. 24th Conf. Foundations of Software Technology & Theoretical Computer Science (FST&TCS'04)*, vol. 3328 of *LNCS*, pages 148–160. Springer, 2004.

[8] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Updatable timed automata. *Theor. Comp. Sci.*, 321(2–3):291–345, 2004.

[9] Th. Brihaye, V. Bruyère, and J.-F. Raskin. On model-checking timed automata with stopwatch observers, 2005. Submitted.

[10] Th. Brihaye, V. Bruyère, and J.-F. Raskin. On optimal timed strategies. In *Proc. 3rd Int. Conference Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, vol. 3821 of *LNCS*, pages 49–64. Springer, 2005.

[11] Th. Brihaye, V. Bruyère, and J.-F. Raskin. Model-checking for weighted timed automata. In *Proc. Joint Conf. Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, vol. 3253 of *LNCS*, pages 277–292. Springer, 2004.

[12] S. La Torre, S. Mukhopadhyay, and A. Murano. Optimal-reachability and control for acyclic weighted timed automata. In *Proc. 2nd IFIP Int. Conf. Theor. Comp. Sci. (TCS 2002)*, vol. 223 of *IFIP Conf. Proc.*, pages 485–497. Kluwer, 2002.

[13] M. Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall Int., 1967.