Costs are Expensive!

Patricia Bouyer[1,2,⋆] Nicolas Markey[1]

[1] LSV, CNRS & ENS Cachan, France
[2] Oxford University Computing Laboratory, UK
{bouyer,markey}@lsv.ens-cachan.fr

Abstract. We study the model-checking problem for WMTL, a cost-extension of the linear-time timed temporal logic MTL, that is interpreted over weighted timed automata. We draw a complete picture of the decidability for that problem: it is decidable only for the class of one-clock weighted timed automata with a restricted stopwatch cost, and any slight extension of this model leads to undecidability. We finally give some consequences on the undecidability of linear hybrid automata.

1 Introduction

Since a couple of years, the verification technology for timed automata has evolved in several interesting directions, to answer new challenges posed by modern real-life systems, like the control of resource (e.g. energy) consumption. In that direction, weighted (or priced) timed automata [ALP01, BFH$^+$01] have been designed as an extension of the timed automaton formalism, which uses observer variables to measure the performance of executions of the system. This model raises numerous interesting optimization problems. A number of them have been shown decidable, including optimal cost reachability [ALP01, BFH$^+$01, BBBR07], optimal reachability in a multi-cost setting [LR05] or mean-cost optimal schedules [BBL04]. Note that the first and third problems even induce no extra complexity compared to the classical problems without optimization constraints (they are PSPACE-Complete).

Unfortunately, in general, adding resource consumption information is far from being free-of-charge! Indeed, to now, two main branches have been explored, which both lead either to negative results, or to complex algorithms. The first branch concerns the control problems, where a controller tries to minimize resource consumption, whatever an environment does: computing optimal cost is undecidable in general [BBR05], and this result holds even if the models have no more than three clocks [BBM06]. Similarly, the model checking of WCTL, a natural cost-extension of the branching-time logic CTL, has been investigated, and very similar undecidability results have been obtained [BBR04, BBM06], even when strong hypotheses are made on the cost [BBR06].

Recently, restriction of timed models to one clock has raised some interest in the community, with interesting complexity or decidability results, like

⋆ Partly supported by a Marie Curie fellowship, a program of the European Commission.

the NLOGSPACE-Completeness of reachability checking in one-clock timed automata [LMS04], or the decidability of emptiness checking in one-clock alternating timed automata over finite timed words [LW05, OW05, LW07, OW07]. In the context of weighted timed systems, this restriction also leads to nice improvements. Indeed, optimal timed games have been proven decidable under that restriction [BLMR06]. The same holds for the model checking of WCTL [BLM07], which even remains PSPACE (*i.e.*, has the same complexity as the model checking of TCTL, the classical timed extension of CTL, even under the single-clock restriction). However, not everything is decidable in this one-clock framework, and for instance, the model checking of WCTL* is still undecidable [BLM07].

In this paper, we tackle an obvious continuation of this literature, by considering the cost extension of the linear-time logic LTL, which we call WMTL (MTL, for "Metric Temporal Logic", is one of the classical timed extensions of LTL introduced by [Koy90], and WMTL can also be viewed as a weighted extension of MTL, hence the name). Indeed, it is known since [OW05] that the model-checking problem for MTL over timed automata accepting finite timed words is a decidable problem, whereas it becomes undecidable for infinite timed words [OW06]. We hence investigate the model checking problem for WMTL over weighted timed automata recognizing finite timed words, and draw a complete picture of the decidability results by proving that only the restriction to one-clock weighted timed automata using a single stopwatch-cost variable[3] is decidable, and that any single extension (like having a non-stopwatch cost variable, or two clocks, or two stopwatch cost variables) leads to undecidability. The decidability proof relies on technics developed in [OW05, OW07] (notice however that in our precise case, it is only valid for one-clock automata). The undecidability proofs rely on a reduction from the halting problem of two-counter machines and push ideas developed in [BBM06, BLM07] much further to get an undecidability result with only one clock in the model and a single cost variable. Finally, these undecidability results have some consequences on the undecidability landscape for linear hybrid automata.

2 Definitions

2.1 Weighted Timed Automata

In the whole paper, AP is a fixed, non-empty set of atomic propositions. In this section, we introduce the notion of *weighted timed automata* [ALP01] (also called *priced timed automata* [BFH$^+$01]), which is an extension of timed automata with a cost information (or weight) on both locations and edges. We first introduce usual notations and definitions for timed automata.

Given a finite set $\mathcal{X}$ of clocks, the set of clock valuations over $\mathcal{X}$ (that is, of applications $v\colon \mathcal{X} \to \mathbb{R}_{\geq 0}$) is denoted $\mathbb{R}_{\geq 0}^{\mathcal{X}}$. Given a valuation v and a nonnegative real $\tau \in \mathbb{R}_{\geq 0}$, the valuation $v + \tau$ is defined by $(v + \tau)(x) = v(x) + \tau$ for every $x \in \mathcal{X}$. The set $\mathcal{G}(\mathcal{X})$ denotes the set of *guards* over $\mathcal{X}$ which are finite

[3] A cost is stopwatch if it behaves like a clock that can be stopped and restarted.

conjunctions of atomic guards of the form $x \sim n$ where $x \in \mathcal{X}$ is a clock, $n \in \mathbb{N}$ is an integer constant, and $\sim$ is one of the symbols $\{<, \leq, =, \geq, >\}$. Notation $v \models g$ means that the valuation v satisfies the guard g (which is defined in the natural way). A *reset* $r \subseteq \mathcal{X}$ is a subset of $\mathcal{X}$ indicating which clocks are to be reset to 0; we write $v' = v[r \leftarrow 0]$ for the resulting valuation, *i.e.*, $v'(x) = 0$ if $x \in r$, and $v'(x) = v(x)$ otherwise.

Definition 1. *A* weighted timed automaton *(WTA in short) with k cost functions is a tuple $\mathcal{A} = (Q, Q_0, \lambda, \mathcal{X}, T, (c_i)_{1 \leq i \leq k})$ such that: Q is a finite set of locations, $Q_0 \subseteq Q$ is the set of initial locations, $\lambda\colon Q \rightarrow \mathsf{AP}$ is the labelling function, $\mathcal{X}$ is a finite set of clocks, $T \subseteq Q \times \mathcal{G}(\mathcal{X}) \times 2^{\mathcal{X}} \times Q$ is a finite set of edges, and for each $1 \leq i \leq k$, $c_i\colon Q \cup T \rightarrow \mathbb{N}$ assigns a* cost *to locations and edges.*

For $S \subseteq \mathbb{N}$, a cost c_i is said to be S-sloped *if $c_i(Q) \subseteq S$. If $S = \{0, 1\}$, it is said* stopwatch. *If $|S| = n$, we say that the cost c_i is n-sloped.*

The semantics of a weighted timed automaton $\mathcal{A}$ corresponds to the semantics of its underlying timed automaton (*i.e.*, forgetting about cost functions). It is given as a transition system $T_{\mathcal{A}} = (S, S_0, \rightarrow)$ where:

- the set of states S is $Q \times \mathbb{R}_{\geq 0}^{\mathcal{X}}$,
- the initial states are $S_0 = \{(q_0, v_0) \mid q_0 \in Q_0 \text{ and } v_0(x) = 0 \text{ for every } x \in \mathcal{X}\}$,
- the transition relation $\rightarrow$ is composed of delay and discrete moves:
  - *(delay move)* $(q, v) \xrightarrow{\tau} (q, v + \tau)$ for $(q, v) \in S$ and $\tau \in \mathbb{R}_{\geq 0}$,
  - *(discrete move)* $(q, v) \xrightarrow{e} (q', v')$ if there exists an edge $e = (q \xrightarrow{g,r} q')$ in T such that $v \models g$ and $v' = v[r \leftarrow 0]$.

  A *mixed move* $(q, v) \xrightarrow{\tau, e} (q', v')$ corresponds to the concatenation of a delay and a discrete moves $(q, v) \xrightarrow{\tau} (q, v + \tau) \xrightarrow{e} (q', v')$.

A *run* (or *execution*) in $\mathcal{A}$ is a finite path in $T_{\mathcal{A}}$, composed of mixed moves.

Let $\varrho = (q_0, v_0) \xrightarrow{\tau_1, e_1} (q_1, v_1) \xrightarrow{\tau_2, e_2} (q_2, v_2) \cdots \xrightarrow{\tau_p, e_p} (q_p, v_p)$ be a finite run in $\mathcal{A}$. For every $i \leq k$, the *i-th cost* of ϱ, denoted by $\mathsf{cost}_i(\varrho)$, is defined as:

$$\mathsf{cost}_i(\varrho) = \sum_{1 \leq j \leq p} c_i(q_{j-1}) \cdot \tau_j + \sum_{1 \leq j \leq p} c_i(e_j)$$

Informally, the cost of a run is the accumulated cost of each move along that run: delaying in some state q during d time units costs $c(q) \cdot d$, and firing an edge e costs $c(e)$. Hence, if q is a location, $c(q)$ gives the derivative of the cost function for waiting in q: we thus write $\dot{c} = 6$ on pictures. For discrete costs on transitions, we write $c{+}{=}\,3$.

Example 1. Fig. 1 models the energy consumption of a device. Due to overheating, this device cannot be left on for more than half an hour: it must either be turned to a "sleep" mode, or completely off. The model has two costs functions: cost c represents energy consumption, while cost w measures the duration of this device being on. The sleeping state consumes slightly more energy than the off state, but it is cheaper and quicker to turn the machine back on. Of course, being on consumes much more energy, but this is the only way of using the device.

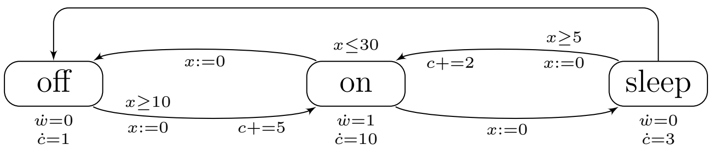


Fig. 1. Measuring the uptime of a device

2.2 The Logic WMTL

The logic WMTL is a weighted extension of LTL, but can also be viewed as an extension of MTL [Koy90], hence its name WMTL holding for "Weighted MTL".

Let $\mathcal{C}$ be a set of cost functions. We define the logic WMTL over $\mathcal{C}$ as the set of formulas defined inductively as:

$$\mathsf{WMTL} \ni \varphi \ ::= \ \sigma \ \mid \ \varphi \vee \varphi \ \mid \ \neg\varphi \ \mid \ \varphi \, \mathbf{U}_{\mathsf{cost}\sim n} \, \varphi$$

where $\sigma \in \mathsf{AP}$, cost is a cost-function symbol in $\mathcal{C}$, $\sim \in \{<, \leq, =, \geq, >\}$, and $n \in \mathbb{N}$. If there is a single cost function or if the cost function cost is clear from the context, we simply write $\varphi_1 \, \mathbf{U}_{\sim n} \, \varphi_2$ instead of $\varphi_1 \, \mathbf{U}_{\mathsf{cost}\sim n} \, \varphi_2$.

We interpret WMTL formulas over (finite) runs of weighted timed automata with $|\mathcal{C}|$ cost functions, identifying cost cost_i of $\mathcal{C}$ with the i-th cost cost_i of the runs of the automaton. Let $\mathcal{A}$ be such a weighted timed automaton, and let $\varrho = (q_0, v_0) \xrightarrow{\tau_1, e_1} (q_1, v_1) \xrightarrow{\tau_2, e_2} (q_2, v_2) \cdots \xrightarrow{e_p, \tau_p} (q_p, v_p)$ be a finite run in $\mathcal{A}$. We write $\varrho_{\geq i}$ for its suffix starting from (q_i, v_i), and $\varrho_{\leq i}$ for its prefix ending in (q_i, v_i). The satisfaction relation for WMTL is then defined inductively as follows:

$$\begin{aligned}
\varrho \models \sigma &\iff \sigma \subseteq \lambda(q_0) \\
\varrho \models \varphi_1 \vee \varphi_2 &\iff \varrho \models \varphi_1 \text{ or } \varrho \models \varphi_2 \\
\varrho \models \neg\varphi &\iff \varrho \not\models \varphi \\
\varrho \models \varphi_1 \, \mathbf{U}_{\mathsf{cost}\sim n} \, \varphi_2 &\iff \exists k > 0 \text{ s.t. } \varrho_{\geq k} \models \varphi_2, \ \forall 0 < i < k, \ \varrho_{\geq i} \models \varphi_1, \\
&\qquad \text{and } \mathsf{cost}(\varrho_{\leq k}) \sim n
\end{aligned}$$

We use classical shorthands like $\mathsf{true} \stackrel{\text{def}}{=} \sigma \vee \neg\sigma$, $\mathsf{false} \stackrel{\text{def}}{=} \neg\mathsf{true}$, $\mathbf{X}\,\varphi \stackrel{\text{def}}{=} \mathsf{false}\,\mathbf{U}\,\varphi$, $\mathbf{F}_{\mathsf{cost}\sim n}\,\varphi \stackrel{\text{def}}{=} \mathsf{true}\,\mathbf{U}_{\mathsf{cost}\sim n}\,\varphi$, and $\mathbf{G}_{\mathsf{cost}\sim n}\,\varphi \stackrel{\text{def}}{=} \neg\Big(\mathbf{F}_{\mathsf{cost}\sim n}\,\neg\varphi\Big)$.

Remark 2. Classically, there are two possible semantics for timed temporal logics [Ras99]: the continuous semantics, where the system is observed continuously, and the point-based semantics, where the system is observed only when the state of the system changes. We have chosen the latter, because the model checking problem for MTL under the continuous semantics is already undecidable [AH90].

Example 2. We continue with our previous example. Assume that the battery has been charged for a total of 1300 cost units (for cost c). We would like to know whether it is possible to use the device for at least two hours within a period of three hours. This is equivalent to the existence of a finite path in our model satisfying the following WMTL formula:

$$\mathbf{F}_{c\leq 1300}\,\texttt{end} \wedge \mathbf{F}_{w\geq 120}\,\texttt{end} \wedge \mathbf{F}_{t\leq 180}\,\texttt{end}$$

where t is a special cost measuring time, *i.e.*, $\dot{t}$ equals 1 in every location, and $\texttt{end}$ characterizes the ending state of the finite path (for instance, $\texttt{end} = \neg\mathbf{X}\,\mathsf{true}$).

2.3 The Problem and Our Results

In the following, we focus on the *existential* model-checking problem for WMTL over weighted timed automata, stated as: given $\mathcal{A}$ a weighted timed automaton and φ a WMTL formula, decide whether there exists a finite run ϱ in $\mathcal{A}$ starting in an initial state and such that $\varrho \models \varphi$. Since WMTL is closed under negation, our results obviously extend to the dual problem of *universal* model-checking.

We prove that the model-checking problem against WMTL properties is decidable for:

(1) one-clock WTAs with one stopwatch cost variable.

Any extension to that model leads to undecidability. Indeed, we prove that the model-checking problem against WMTL properties is undecidable for:

(2) one-clock WTAs with one cost variable,
(3) one-clock WTAs with two stopwatch cost variables,
(4) two-clock WTAs with one stopwatch cost variable.

We present our results as follows. In Section 3, we explain the positive result (1) using an abstraction proposed in [OW05] for proving the decidability of MTL model checking over timed automata. Then, in Section 4, we present all our undecidability results, starting with the proof for result (2), and then slightly modifying the construction for proving results (3) and (4). We conclude with some corollaries for linear hybrid automata.

3 Decidability Result

Theorem 3. *Model checking one-clock weighted timed automata with one stopwatch cost against* WMTL *properties is decidable, and non-primitive recursive.*

Proof. Time can be viewed as a special $\{1\}$-sloped cost. Hence, the non-primitive recursive lower bound follows from that of MTL model checking over finite timed words, see [OW05, OW07].

The decidability then relies on the same encoding as [OW05]. We present the construction, but don't give all details, especially when there is nothing new compared with the above-mentioned papers.

Let φ be a WMTL formula, and $\mathcal{A}$ be a single-clock weighted timed automaton with a stopwatch cost. Classically, from formula φ, we construct an "equivalent" one-variable alternating timed automaton[4] $\mathcal{B}_\varphi$. Fig. 2 displays an example of such

[4] We use the *eager* semantics [BMOW07] for alternating automata, where configuration of the automaton always have the same sets of successors.

an automaton, corresponding to formula $\mathbf{G}\,[a \Rightarrow (\mathbf{F}_{\leq 3}\, b \vee \mathbf{F}_{\geq 2}\, c)]$ (see [OW05] for more details on alternating timed automata).

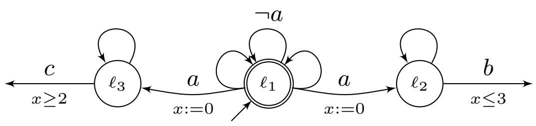


Fig. 2. A timed alternating automaton for formula $\mathbf{G}\,[a \Rightarrow (\mathbf{F}_{\leq 3}\, b \vee \mathbf{F}_{\geq 2}\, c)]$

However, note that in that case, the unique variable of the alternating automaton is not a clock but a cost variable, whose rate will depend on the location of $\mathcal{A}$ which is being visited. However, as for MTL, we have the property that $\mathcal{A} \models \varphi$ iff there is an accepting joint execution of $\mathcal{A}$ and $\mathcal{B}_\varphi$.

In the following, we write q for a generic location of $\mathcal{A}$ and ℓ for a generic location of $\mathcal{B}_\varphi$. Similarly, Q denotes the set of locations of $\mathcal{A}$ and L the set of locations of $\mathcal{B}_\varphi$.

An $\mathcal{A}/\mathcal{B}_\varphi$*-joint configuration* is a finite subset of $Q \times \mathbb{R}_{\geq 0} \cup L \times \mathbb{R}_{\geq 0}$ with exactly one element of $Q \times \mathbb{R}_{\geq 0}$ (the current state in automaton $\mathcal{A}$). The joint behaviour of $\mathcal{A}$ and $\mathcal{B}_\varphi$ is made of time evolutions and discrete steps in a natural way. Note that, from a given joint configuration γ, the time evolution is given by the current location q_γ of $\mathcal{A}$: if the cost rate in q_γ is 1, then all variables behave like clocks, *i.e.*, grow with rate 1, and if the cost rate in q_γ is 0, then all variables in $\mathcal{B}_\varphi$ are stopped, and only the clock of $\mathcal{A}$ grows with rate 1.

We encode configurations with words over the alphabet $\Gamma = 2^{(Q \times \mathsf{Reg} \cup L \times \mathsf{Reg})}$, where $\mathsf{Reg} = \{0, 1, \ldots, M\} \cup \{\top\}$ (M is an integer above the maximal constant appearing in both $\mathcal{A}$ and $\mathcal{B}_\varphi$). A state (ℓ, c) of $\mathcal{B}_\varphi$ will for instance be encoded by $(\ell, \mathit{int}(c))$ [5] if $c \leq M$, and it will be encoded by $(\ell, \top)$ if $c > M$.

Now given a joint configuration $\gamma = \{(q, x)\} \cup \{(\ell_i, c_i) \mid i \in I\}$, partition γ into a sequence of subsets $\gamma_0, \gamma_1, \ldots, \gamma_p, \gamma_\top$, such that $\gamma_\top = \{(\alpha, \beta) \in \gamma \mid \beta > M\}$, and if $i, j \neq \top$, for all $(\alpha, \beta) \in \gamma_i$ and $(\alpha', \beta') \in \gamma_j$, $\mathit{frac}(\beta) \leq \mathit{frac}(\beta')$ [6] iff $i \leq j$ (so that (α, β) and (α', β') are in the same block γ_i iff β and β' are both smaller than or equal to M and have the same fractional part). We assume in addition that the fractional part of elements in γ_0 is 0 (even if it means that $\gamma_0 = \varnothing$), and that all γ_i for $1 \leq i \leq p$ are non-empty.

If γ is a joint configuration, we define its encoding $H(\gamma)$ as the word (over Γ) $\mathsf{reg}(\gamma_0)\mathsf{reg}(\gamma_1)\ldots\mathsf{reg}(\gamma_p)\mathsf{reg}(\gamma_\top)$ where $\mathsf{reg}(\gamma_i) = \{(\alpha, \mathsf{reg}(\beta)) \mid (\alpha, \beta) \in \gamma_i\}$ with $\mathsf{reg}(\beta) = \mathit{int}(\beta)$ if $\beta \leq M$, and $\mathsf{reg}(\beta) = \top$ otherwise.

Example 3. Consider the configuration

$$\gamma = \{(q, 1.6)\} \cup \{(\ell_1, 5.2), (\ell_2, 2.2), (\ell_2, 2.6), (\ell_3, 1.5), (\ell_3, 4.5)\}.$$

---

[5] *int* represents the integral part.

[6] *frac* represents the fractional part.

Assuming that the maximal constant (on both $\mathcal{A}$ and $\mathcal{B}_\varphi$) is 4, the encoding is then

$$H(\gamma) = \{(\ell_2, 2)\} \cdot \{(\ell_3, 1)\} \cdot \{(q, 1), (\ell_2, 2)\} \cdot \{(\ell_1, \top), (\ell_3, \top)\}$$

We define a discrete transition system over encodings of $\mathcal{A}/\mathcal{B}_\varphi$-joint configurations: there is a transition $W \Rightarrow W'$ if there exists $\gamma \in H^{-1}(W)$ and $\gamma' \in H^{-1}(W')$ such that $\gamma \to \gamma'$ (that can be either a time evolution or a discrete step).

Lemma 4. *The equivalence relation $\equiv$ defined as $\gamma_1 \equiv \gamma_2 \overset{def}{\Leftrightarrow} H(\gamma_1) = H(\gamma_2)$ is a time-abstract bisimulation over joint configurations.*

Proof. We assume that $\gamma_1 \to \gamma_1'$ and that $\gamma_1 \equiv \gamma_2$. We write $H(\gamma_1) = H(\gamma_2) = w_0 w_1 \dots w_p w_\top$ where $w_i \neq \emptyset$ if $1 \leq i \leq p$. We distinguish between the different possible cases for the transition $\gamma_1 \to \gamma_1'$.

- assume $\gamma_1 \to \gamma_1'$ is a time evolution, and the cost rate in the corresponding location of $\mathcal{A}$ is 0. If $\gamma_1 = \{(q_1, x_1)\} \cup \{(\ell_{i,1}, c_{i,1}) \mid i \in I_1\}$, then $\gamma_1' = \{(q_1, x_1 + t_1)\} \cup \{(\ell_{i,1}, c_{i,1}) \mid i \in I_1\}$ for some $t_1 \in \mathbb{R}_{\geq 0}$. We assume in addition that $\gamma_2 = \{(q_2, x_2)\} \cup \{(\ell_{i,2}, c_{i,2}) \mid i \in I_2\}$.
  We set γ_1^i the part of configuration γ_1 which corresponds to letter w_i, and we write α_1^i for the fractional part of the clock values corresponding to γ_1^i. We have $0 = \alpha_1^0 < \alpha_1^1 < \dots < \alpha_1^p < 1$. We define similarly $(\alpha_2^i)_{0 \leq i \leq p}$ for configuration γ_2. We then distinguish between several cases:
  - either $x_1 + t_1 > M$, in which case it is sufficient to choose $t_2 \in \mathbb{R}_{\geq 0}$ such that $x_2 + t_2 > M$.
  - or $x_1 + t_1 \leq M$ and $\mathit{frac}(x_1 + t_1) = \alpha_1^i$ for some $0 \leq i \leq p$. In that case, choose $t_2 = x_1 + t_1 - \alpha_1^i + \alpha_2^i - x_2$. As $\gamma_1 \equiv \gamma_2$, it is not difficult to check that $t_2 \in \mathbb{R}_{\geq 0}$. Moreover, $\mathit{frac}(x_2 + t_2) = \alpha_2^i$ and $\mathit{int}(x_2 + t_2) = \mathit{int}(x_1 + t_1)$.
  - or $x_1 + t_1 \leq M$ and $\alpha_1^i < \mathit{frac}(x_1 + t_1) < \alpha_1^{i+1}$ for some $0 \leq i \leq p$ (setting $\alpha_1^{p+1} = 1$). As previously, in that case also, we can choose $t_2 \in \mathbb{R}_{\geq 0}$ such that $\alpha_2^i < \mathit{frac}(x_2 + t_2) < \alpha_2^{i+1}$ and $\mathit{int}(x_2 + t_2) = \mathit{int}(x_1 + t_1)$.

  In all cases, defining $\gamma_2' = \{(q_2, x_2 + t_2)\} \cup \{(\ell_{i,2}, c_{i,2}) \mid i \in I_2\}$, we get that $\gamma_2 \to \gamma_2'$ and $\gamma_1' \equiv \gamma_2'$, which proves the inductive case.
- there are two other cases (time evolution with rate of all variables being 1, and discrete step), but they are similar to the case of MTL, and we better refer to [OW07]. □

Hence, from the previous lemma, we get:

Corollary 5. *$W \Rightarrow^* W'$ iff there exist $\gamma \in H^{-1}(W)$ and $\gamma' \in H^{-1}(W')$ such that $\gamma \to^* \gamma'$.*

The set $\Gamma = 2^{(Q \times \mathsf{Reg} \cup L \times \mathsf{Reg})}$ is naturally ordered by inclusion $\subseteq$. We extend the classical subword relation for words over Γ as follows: Given two words $a_0 a_1 \dots a_n$ and $a_0' a_1' \dots a_{n'}'$ in Γ^*, we say that $a_0 a_1 \dots a_n \sqsubseteq a_0' a_1' \dots a_{n'}'$ whenever there exists an increasing injection $\iota : \{0, 1, \dots, n\} \to \{0, 1, \dots, n'\}$ such that for every $i \in \{0, 1, \dots, n\}$, $a_i \subseteq a_{\iota(i)}'$. Following [AN00, Theorem 3.1], the preorder $\sqsubseteq$ is a well-quasi-order.

Lemma 6. *Assume that $W_1 \sqsubseteq W_2$, and that $W_2 \Rightarrow^* W_2'$. Then, there exists $W_1' \sqsubseteq W_2'$ such that $W_1 \Rightarrow^* W_1'$.*

The algorithm then proceeds as follows: we start from the encoding of the initial configuration, say W_0, and then generate the tree unfolding of the implicit graph $(\Gamma^*, \Rightarrow)$, stopping a branch when the current node is labelled by W such that there already exists a node of the tree labelled by W' with $W' \sqsubseteq W$ (note that by Lemma 6, if there is an accepting path from W, then so is there from W', hence it is correct to prune the tree after node W). Note that this tree is finitely branching. Hence, if the computation does not terminate, then it means that there is an infinite branch (by König lemma). This is not possible as $\sqsubseteq$ is a well-quasi-order. Hence, the computation eventually terminates, and we can decide whether there is a joint accepting computation in $\mathcal{A}$ and $\mathcal{B}_\varphi$, which implies that we can decide whether $\mathcal{A}$ satisfies φ or not. □

Remark 7. In the case of MTL, the previous encoding can be used to prove the decidability of model checking for timed automata with any number of clocks. In our case, it cannot: Lemma 4 does not hold for two-clock weighted timed automata, even with a single stopwatch cost. Consider for instance two clocks x and z, and a cost variable cost. Assume we are in location q of the automaton with cost rate 0 and that there is an outgoing transition labelled by the constraint $x = 1$. Assume moreover that the value of z is 0, whereas the value of x is 0.2. We consider two cases: either the value of cost is 0.5, or the value of cost is 0.9. In both cases, the encoding[7] of the joint configuration is $\{(q, z, 0)\} \cdot \{(q, x, 0)\} \cdot \{(\mathsf{cost}, 0)\}$. However, in the first case, the encoding when firing the transition will be $\{(q, x, 1)\} \cdot \{(\mathsf{cost}, 0)\} \cdot \{(q, z, 0)\}$, whereas in the second case, it will be $\{(q, x, 1)\} \cdot \{(q, z, 0)\} \cdot \{(\mathsf{cost}, 0)\}$. Hence the relation $\equiv$ is not a time-abstract bisimulation.

4 Undecidability Results

4.1 One-Clock WTAs With One Cost Variable

Theorem 8. *Model checking one-clock weighted timed automata with one cost variable against* WMTL *properties is undecidable.*

We push some ideas used in [BBM06, BLM07] further to prove this new undecidability result. We reduce the halting problem for a two-counter machine $\mathcal{M}$ to that problem. The unique clock of the automaton will store both values of the counters. If the first (resp. second) counter has value c_1 (resp. c_2), then the value of the clock will be $2^{-c_1}3^{-c_2}$. Our machine $\mathcal{M}$ has two kinds of instructions. The first kind increments one of the counter, say $\mathtt{c}$, and jumps to the next instruction:

$$\mathtt{p_i} : \ \mathtt{c} := \mathtt{c} + 1; \ \mathtt{goto} \ \mathtt{p_j}. \tag{1}$$

[7] We extend the encoding we have presented above to several clocks, as originally done in [OW05].

The second kind decrements one of the counter, say $\mathtt{c}$, and goes to the next instruction, except if the value of the counter was zero:

$$\mathtt{p_i}: \mathtt{if}\ (\mathtt{c} == 0)\ \mathtt{then\ goto}\ \mathtt{p_j}\ \mathtt{else}\ \mathtt{c} := \mathtt{c} - 1;\ \mathtt{goto}\ \mathtt{p_k}. \qquad (2)$$

Our reduction consists in building a weighted timed automaton $\mathcal{A}_{\mathcal{M}}$ and a WMTL formula φ such that the two-counter machine $\mathcal{M}$ halts iff $\mathcal{A}_{\mathcal{M}}$ has an execution satisfying φ. Each instruction of $\mathcal{M}$ is encoded as a module, all the modules are then plugged together.

Module for instruction (1)**.** Consider instruction (1), which increments the first counter. To simulate this instruction, we need to be able to divide the value of the clock by 2. The corresponding module, named Mod_i, is depicted on Fig. 3.[8]

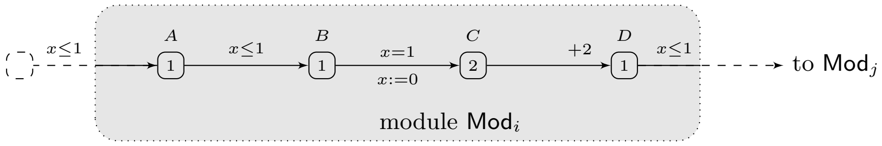


Fig. 3. Module for incrementing c_1

The following lemma is then easy to prove:

Lemma 9. *Assume that there is an execution ϱ entering module* Mod_i *with $x = x_0 \le 1$, exiting with $x = x_1$, and such that no time elapses in A and D and the cost between A and D equals* 3*. Then $x_1 = x_0/2$.*

A similar result can be obtained for a module incrementing c_2: it simply suffices to replace the cost rate in C by 3 instead of 2.

Module for instruction (2)**.** Consider instruction (2). The simulation of this instruction is much more involved than the previous instruction. Indeed, we first have to check whether the value of x when entering the module is of the form 3^{-c_2} (*i.e.*, whether $c_1 = 0$). This is achieved, roughly, by multiplying the value of x by 3 until it reaches (or exceeds) 1. Depending on the result, this module will then branch to module Mod_j or decrement counter c_1 and go to module Mod_k. The difficult point is that clock x must be re-set to its original value between the first and the second part. We consider the module Mod_i depicted on Fig. 4.

Lemma 10. *Assume there exists an execution ϱ entering module* Mod_i *with $x = x_0 \le 1$, exiting to module* Mod_j *with $x = x_1$, and such that*

[8] As there is a unique cost variable, we write its rate within the location, and add a discrete incrementation (*eg* +2) on edges, when the edge has a positive cost.

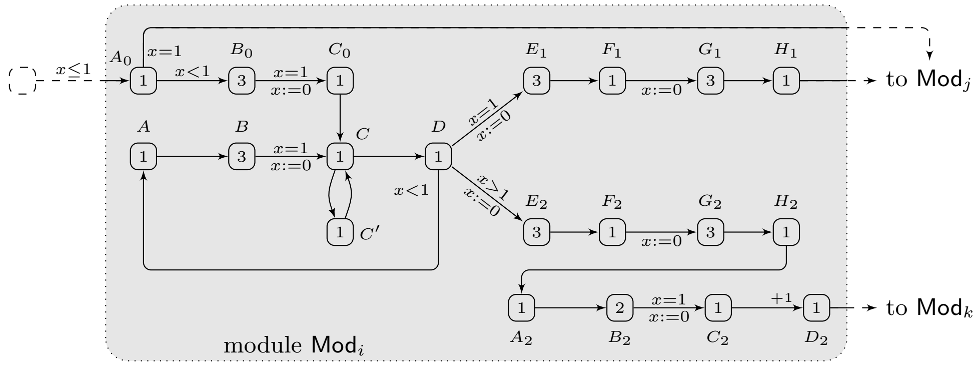


Fig. 4. Module testing/decrementing c_1

- *no time elapses in A_0, C_0, D, A, C', F_1 and H_1;*
- *any visit to C_0 or C' is eventually followed (strictly) by a visit to C' or F_1;*
- *the cost exactly equals 3 along each part of ϱ between A or A_0 and the next visit in D, between C_0 or C' and the next visit in C' or F_1, and between the last visit to D and H_1.*

Then $x_1 = x_0$ and there exists $n \in \mathbb{N}$ s.t. $x_0 = 3^{-n}$.

Proof. Let ϱ be such an execution. First, if $x_0 = 1$ and ϱ goes directly to module Mod_j, then the result immediately follows.

Otherwise, ϱ visits D at least once. We prove inductively that, at the k-th visit in D, the value of x equals $3^k x_0$ (remember that no time can elapse in D). The first part of ϱ between A_0 and D is as follows[9] (the labels on the arrows represent the cost of the corresponding transition):

$$(A_0, x_0) \xrightarrow{0} (B_0, x_0) \xrightarrow{3(1-x_0)} (B_0, 1) \xrightarrow{0} (C_0, 0) \xrightarrow{0} (C, 0) \xrightarrow{\alpha} (C, \alpha) \xrightarrow{0} (D, \alpha).$$

The total cost, $3(1-x_0)+\alpha$, must equal 3. Thus $\alpha = 3x_0$. A similar argument shows that one turn in the loop (from D back to itself) also multiplies clock x by 3, hence the result. Since ϱ eventually fires the transition from D to E_1, it must be the case that $x_0 = 3^{-n}$ for some $n \in \mathbb{N}$.

We now prove that $x_1 = x_0$. The proof follows a similar line: we prove that at the k-th visit to C_0 or C', the value of x is $(3^k - 3)x_0$. This clearly holds when $k = 1$ (*i.e.*, when we visit C_0). Assuming that ϱ eventually visits C', we consider the part of ϱ between C_0 and the first visit to C':

$$\begin{aligned}(C_0, 0) \xrightarrow{0} (C, 0) \xrightarrow{3x_0} (C, 3x_0) \xrightarrow{0} (D, 3x_0) \xrightarrow{0} (A, 3x_0) \xrightarrow{0} (B, 3x_0)\\ (B, 3x_0) \xrightarrow{3(1-3x_0)} (B, 1) \xrightarrow{0} (C, 0) \xrightarrow{\beta} (C, \beta) \xrightarrow{0} (C', \beta).\end{aligned}$$

[9] By contradiction, it can be proved that C' cannot be visited along that part of ϱ, since the cost between C_0 and C' must be exactly 3.

The cost of this part is $3-6x_0+\beta$, and must equal 3. Thus $\beta = 6x_0$ as required. A similar computation (considering each part of ϱ between two consecutive visits to C') proves the inductive case.

Now, consider the part from the last visit of C' to H_1:

$$(C', (3^n-3)x_0) \xrightarrow{0} (C, (3^n-3)x_0) \xrightarrow{3x_0} (C, 3^n x_0) \xrightarrow{0} (D, 3^n x_0) \xrightarrow{0} (E_1, 0)$$
$$(E_1, 0) \xrightarrow{3\gamma} (E_1, \gamma) \xrightarrow{0} (F_1, \gamma) \xrightarrow{0} (G_1, 0) \xrightarrow{3\delta} (G_1, \delta) \xrightarrow{0} (H_1, \delta).$$

Remember that $3^n x_0 = 1$, which explains why the computation goes to E_1 instead of E_2). The cost between C' and F_1 is $3x_0 + 3\gamma$, and equals 3. Thus $\gamma = 1-x_0$. Similarly, the cost between D and H_1 is $3\gamma+3\delta$ and must equal 3, which proves that δ, which is precisely x_1, equals x_0. □

We have a similar result for a trajectory going to module Mod_k:

Lemma 11. *Assume there exists an execution ϱ entering module Mod_i with $x = x_0 \leq 1$, exiting to module Mod_k with $x = x_1$, and such that*

- *no time elapses in A_0, C_0, D, A, C', F_2 H_2, A_2 and D_2;*
- *any visit to C_0 or C' is eventually followed (strictly) by a visit to C' or F_2;*
- *the cost exactly equals 3 along each part of ϱ between A or A_0 and the next visit in D, between C_0 or C' and the next visit in C' or F_2, between D and H_2, and between H_2 and D_2.*

Then $x_1 = 2x_0$ and for every $n \in \mathbb{N}$, $x_0 \neq 3^{-n}$.

Proof. The arguments of the previous proof still apply: the value of x at the k-th visit to D is $3^k x_0$. If x_0 had been of the form 3^{-n}, then ϱ would not have been able to fire the transition to E_2. Also, the value of x when ϱ visits H_2 is precisely x_0. The part from H_2 to D is then as follows:

$$(H_2, x_0) \xrightarrow{0} (A_2, x_0) \xrightarrow{0} (B_2, x_0) \xrightarrow{2(1-x_0)} (B_2, 1) \xrightarrow{0} (C_2, 0) \xrightarrow{\kappa} (C_2, \kappa) \xrightarrow{1} (D_2, \kappa).$$

The cost of this part is $2(1-x_0) + \kappa + 1$, so that $x_1 = \kappa = 2x_0$. □

Again, these results can easily be adapted to the case of an instruction testing and decrementing c_2: it suffices to

- set the costs of states B_0, B, E_1, E_2, G_1 and G_2 to 2,
- set the cost of B_2 to 3,
- set the discrete cost of $C_2 \to D_2$ to 0
- set the discrete costs of $C \to D$, $G_1 \to H_1$ and $G_2 \to H_2$ to +1.

Global reduction. We now explain the global reduction: the automaton $\mathcal{A}_\mathcal{M}$ is obtained by plugging the modules above following the instructions of $\mathcal{M}$. There is one special module for instruction Halt, which is made of a single Halt state. We also add a special initial state that lets 1 t.u. elapse (so that $x = 1$) before entering the first module.

The WMTL formula is built as follows: we first define an intermediary subformula stating that no time can elapse in some given state. It writes $\mathsf{zero}(P) = \mathbf{G}\,(P \Rightarrow (P\,\mathbf{U}_{=0}\,\neg P))$. If the local cost in state P is not zero (which is the case in all the states of $\mathcal{A}_{\mathcal{M}}$), this formula forbids time elapsing in P. We then let φ_1 be the formula requiring that time cannot elapse in a state labelled with A, D, A_0, C_0, C', F_1, F_2 H_1, H_2, A_2 and D_2. It remains to express the second and third conditions in Lemmas 9, 10 and 11. This is the role of the following formula φ_2:

$$\begin{aligned}\varphi_2 = \mathbf{G}\,[(A \vee A_0) \Rightarrow (\neg D\,\mathbf{U}_{=3}\,D)] \wedge \\ \mathbf{G}\,[(C_0 \vee C') \Rightarrow (\neg(C' \vee F_1)\,\mathbf{U}_{=3}\,(C' \vee F_1))] \wedge \\ \mathbf{G}\,[D \Rightarrow ((\neg A\,\mathbf{U}\,A) \vee (\neg(H_1 \vee H_2)\,\mathbf{U}_{=3}\,(H_1 \vee H_2)))] \wedge \\ \mathbf{G}\,[H_2 \Rightarrow (\neg D_2\,\mathbf{U}_{=3}\,D_2)].\end{aligned}$$

The following proposition is now straightforward:

Proposition 12. *The machine $\mathcal{M}$ halts iff there exists an execution in $\mathcal{A}_{\mathcal{M}}$ satisfying $\varphi_1 \wedge \varphi_2 \wedge \mathbf{F}\,\mathsf{Halt}$.*

Remark 13.
- For the sake of simplicity, our reduction uses discrete costs, so that our WMTL formulas only involve constraints "$= 0$" and "$= 3$" (and the same formula φ_2 can be used for both counters). But our undecidability result easily extends to automata without discrete costs.
- Our reduction uses a $\{1, 2, 3\}$-sloped cost variable, but it could be achieved with any $\{p, q, r\}$-sloped cost variable (with $0 < p < q < r$, and p, q and r are pairwise coprime) by encoding the values of the counters by the clock value $(p/q)^{c_1} \cdot (p/r)^{c_2}$.
- Our WMTL formula can easily be turned into a WMITL formula (whose syntax is that of MITL [AFH96], *i.e.*, with no punctual constraints). It suffices to replace formulas of the form $(\neg p)\,\mathbf{U}_{=n}\,p$ with $(\neg p)\,\mathbf{U}_{\leq n}\,p \wedge (\neg p)\,\mathbf{U}_{\geq n}\,p$.

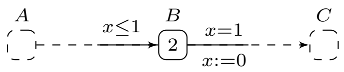
4.2 Two-Clock WTAs With One Stopwatch-Cost Variable

We now prove a similar result for WTAs with two clocks but only with a stopwatch cost (*i.e.*, a cost with slope 0 or 1).

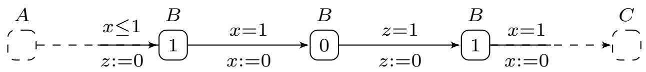
Theorem 14. *Model checking two-clock weighted timed automata with one stopwatch cost against* WMTL *properties is undecidable.*

Proof (Sketch). The proof uses the same encoding, except that states with cost 2 or 3 are replaced by sequences of states with costs 0 and 1 having the same effect. We have two different kinds of states with cost 2 (or 3):

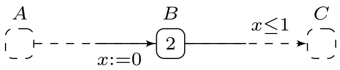
- those in which we stay until $x = 1$:



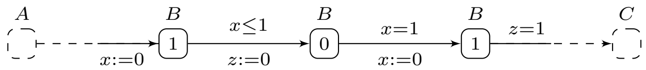
These states are replaced by the following submodule:



– those in which we enter with $x = 0$ (and exit with $x \leq 1$):



Those are replace with a slightly different sequence of states:



Those transformations are easily adapted to states with cost 3. □

4.3 One-Clock WTAs with Two Stopwatch-Cost Variables

In the above constructions, each clock can be replaced with an observer variable, *i.e.*, with a "clock cost" that is not involved in the guards of the automaton anymore. We briefly explain this transformation on an example, and leave the details to the keen reader.

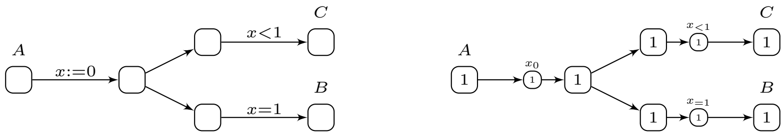


Fig. 5. Replacing a clock with an extra "clock cost"

Fig. 5 displays the transformation to be applied to the automaton. It then suffices to enforce that no time elapses in states x_0, $x_{<1}$ and $x_{=1}$, and that the following formula holds:

$$\bigwedge_{\sim n \in \{<1, =1\}} \mathbf{G} \left[\left(x_0 \wedge \neg x_0 \, \mathbf{U} \, x_{\sim n} \right) \Rightarrow \left(\neg x_0 \, \mathbf{U}_{(c_x \sim n)} \, x_{\sim n} \right) \right]$$

This precisely encodes the role of clock x in the original automaton with a clock cost, which is in particular a stopwatch cost. Note that this transformation is not correct in general, but it is here because our reduction never involves two consecutive transitions with the same guard. As a consequence:

Theorem 15. – *Model checking one-clock weighted timed automata with two stopwatch costs against* WMTL *properties is undecidable.*

– *Model checking zero-clock weighted timed automata with two costs (or three stopwatch costs) against* WMTL *properties is undecidable.*

4.4 Remarks on Hybrid Systems

It is worth making some remarks on the relation between our undecidability results and the undecidability of reachability in linear hybrid automata (LHA for short; we refer to [HKPV98] for their definition). Indeed, the construction on Fig. 5 is not surprising as MITL can capture the exact behaviour of timed automata [Ras99], and could thus be used to transfer undecidability results from LHA to WMTL model checking: from an LHA $\mathcal{H}$, we can construct a WMTL formula $\varphi_{\mathcal{H}}$ such that a given location of $\mathcal{H}$ is reachable iff the WTA resulting from the transformation satisfies formula $\varphi_{\mathcal{H}}$.

To the best of our knowledge, the tightest undecidability results for LHA hold either with five clocks and a single two-sloped variable [HKPV98], or with four stopwatch variables [Fle02].

From these results and the remarks above, the model checking of WMTL is undecidable for WTAs, either with five clocks and a single two-sloped cost variable, or without clocks and four stopwatch costs. While the first result cannot be compared with any of our results, the second is subsumed by Theorem 15.

On the other hand, by using similar ideas to those we have developed in this paper, we can get noticeable undecidability results for LHA: it suffices to encode as an LHA all the constraints mentioned in Lemmas 9, 10, 11, which can be done rather easily using two variables and one clock. However, we can do better and adapt the construction to get the undecidability already with one variable and two clocks. We present this construction now, which can be interesting in itself to understand the power of LHA. The incrementation instruction follows rather easily from that of Fig. 3, we better present the modified construction

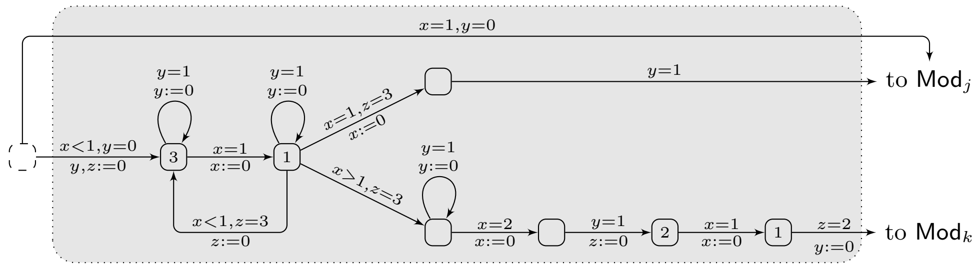


Fig. 6. Undecidability of LHA with two clocks (x, y) and a single hybrid variable (z)

for the test and decrementation instruction (see Fig. 6, where the derivative of variable z, when relevant, is indicated in each location). It differs from Fig. 4 in the way the initial value of clock x is recovered at the end of the loop which checks whether the value of clock x is of the form 3^{-n} or not: clock y is reset when entering the module and then computes a modulo 1, which implies that the initial value of clock x, say x_0, is recovered each time $y = 1$.

Corollary 16. *The reachability problem for LHA, either with two clocks and a single $\{p, q, r\}$-sloped variable (with p, q and r positive and pairwise coprime), or with three clocks and a single stopwatch variable, is undecidable.*

Finally, note that the class of LHA with one clock and one stopwatch variable can easily be proved decidable using the classical regions of [AD94].

References

[AD94] Rajeev Alur and David Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[AH90] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. In *Proc. 5th Annual Symposium on Logic in Computer Science (LICS'90)*, pages 390–401. IEEE Computer Society Press, 1990.

[ALP01] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *LNCS*, pages 49–62. Springer, 2001.

[AN00] Parosh Aziz Abdulla and Aletta Nylén. Better is better than well: On efficient verification of infinite-state systems. In *Proc. 15th Annual Symposium on Logic in Computer Science (LICS'00)*, pages 132–140. IEEE Computer Society press, 2000.

[BBBR07] Patricia Bouyer, Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On the optimal reachability problem. *Formal Methods in System Design*, 2007. To appear.

[BBL04] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Staying alive as cheaply as possible. In *Proc. 7th International Workshop on Hybrid Systems: Computation and Control (HSCC'04)*, volume 2993 of *LNCS*, pages 203–218. Springer, 2004.

[BBM06] Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. Improved undecidability results on weighted timed automata. *Information Processing Letters*, 98(5):188–194, 2006.

[BBR04] Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. Model-checking for weighted timed automata. In *Proc. Joint Conference on Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, volume 3253 of *LNCS*, pages 277–292. Springer, 2004.

[BBR05] Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On optimal timed strategies. In *Proc. 3rd International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3821 of *LNCS*, pages 49–64. Springer, 2005.

[BBR06] Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On model-checking timed automata with stopwatch observers. *Information and Computation*, 204(3):447–478, 2006.

[BFH+01] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-cost reachability for priced timed automata. In *Proc. 4th International Workshop* *on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *LNCS*, pages 147–161. Springer, 2001.

[BLM07] Patricia Bouyer, Kim G. Larsen, and Nicolas Markey. Model-checking one-clock priced timed automata. In *Proc. 10th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'07)*, volume 4423 of *LNCS*, pages 108–122. Springer, 2007.

[BLMR06] Patricia Bouyer, Kim G. Larsen, Nicolas Markey, and Jacob I. Rasmussen. Almost optimal strategies in one-clock priced timed automata. In *Proc. 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'06)*, volume 4337 of *LNCS*, pages 345–356. Springer, 2006.

[BMOW07] Patricia Bouyer, Nicolas Markey, Joël Ouaknine, and James Worrell. The cost of punctuality. In *Proc. 21st Annual Symposium on Logic in Computer Science (LICS'07)*. IEEE Computer Society Press, 2007. To appear.

[Fle02] Emmanuel Fleury. *Automates temporisés avec mises à jour*. PhD thesis, École Normale Supérieure de Cachan, Cachan, France, 2002.

[HKPV98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, 1998.

[Koy90] Ron Koymans. Specifying real-time properties with Metric Temporal Logic. *Real-Time Systems*, 2(4):255–299, 1990.

[LMS04] François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. Model checking timed automata with one or two clocks. In *Proc. 15th International Conference on Concurrency Theory (CONCUR'04)*, volume 3170 of *LNCS*, pages 387–401. Springer, 2004.

[LR05] Kim G. Larsen and Jacob Illum Rasmussen. Optimal conditional scheduling for multi-priced timed automata. In *Proc. 8th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'05)*, volume 3441 of *LNCS*, pages 234–249. Springer, 2005.

[LW05] Slawomir Lasota and Igor Walukiewicz. Alternating timed automata. In *Proc. 8th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'05)*, volume 3441 of *LNCS*, pages 250–265. Springer, 2005.

[LW07] Slawomir Lasota and Igor Walukiewicz. Alternating timed automata. *ACM Transactions on Computational Logic*, 2007. To appear.

[OW05] Joël Ouaknine and James Worrell. On the decidability of Metric Temporal Logic. In *Proc. 19th Annual Symposium on Logic in Computer Science (LICS'05)*, pages 188–197. IEEE Computer Society Press, 2005.

[OW06] Joël Ouaknine and James Worrell. On Metric Temporal Logic and faulty Turing machines. In *Proc. 9th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'06)*, volume 3921 of *LNCS*, pages 217–230. Springer, 2006.

[OW07] Joël Ouaknine and James Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science*, 3(1:8):1–27, 2007.

[Ras99] Jean-François Raskin. *Logics, Automata and Classical Theories for Deciding Real-Time*. PhD thesis, University of Namur, Namur, Belgium, 1999.