Logical Methods in Computer Science
Vol. 4 (2:9) 2008, pp. 1–28
www.lmcs-online.org

Submitted Sep. 20, 2007
Published Jun. 20, 2008

MODEL CHECKING ONE-CLOCK PRICED TIMED AUTOMATA *

PATRICIA BOUYER [a], KIM G. LARSEN [b], AND NICOLAS MARKEY [c]

[a] LSV, CNRS & ENS de Cachan, France and Oxford University Computing Laboratory, UK
e-mail address: bouyer@lsv.ens-cachan.fr

[b] Aalborg University, Denmark
e-mail address: kgl@cs.aau.dk

[c] LSV, CNRS & ENS de Cachan, France
e-mail address: markey@lsv.ens-cachan.fr

Abstract. We consider the model of priced (a.k.a. weighted) timed automata, an extension of timed automata with cost information on both locations and transitions, and we study various model-checking problems for that model based on extensions of classical temporal logics with cost constraints on modalities. We prove that, under the assumption that the model has only one clock, model-checking this class of models against the logic WCTL, CTL with cost-constrained modalities, is PSPACE-complete (while it has been shown undecidable as soon as the model has three clocks). We also prove that model-checking WMTL, LTL with cost-constrained modalities, is decidable only if there is a single clock in the model and a single stopwatch cost variable (*i.e.*, whose slopes lie in $\{0, 1\}$).

An interesting direction of real-time model-checking that has recently received substantial attention is the extension and re-targeting of timed automata technology towards optimal scheduling and controller synthesis [AAM06, RLS04, BBL08].

In particular, scheduling problems can often be reformulated in terms of reachability questions with respect to behavioural models where tasks and resources relevant for the scheduling problem in question are modelled as interacting timed automata [BLR05a]. Although there exists a wide body of literature and established results on (optimal) scheduling in the fields of real-time systems and operations research, the application of model-checking has proved to provide a novel and competitive technology. In particular, model-checking has the advantage of offering a generic approach, going well beyond most classical scheduling solutions, which have good properties only for scenarios satisfying specific assumptions that may or, quite often, may not apply in actual practical circumstances. Of course, model-checking comes with its own restrictions and stumbling blocks, the most notorious being

1998 ACM Subject Classification: F.1.1,F.3.1.
Key words and phrases: priced timed automata, model-checking.

* This article is a long version of [BLM07]. It has been extended with recent related results from [BM07].

[a] Partly supported by project DOTS (ANR-06-SETI-003), and by a Marie-Curie fellowship.
[b] Partly supported by an invited professorship from ENS Cachan.
[c] Partly supported by project DOTS (ANR-06-SETI-003).

DOI:10.2168/LMCS-4 (2:9) 2008

the state-space explosion. A lot of research has thus been devoted to "guide" and "prune" the reachability search [BFH+01a].

As part of the effort on applying timed automata technology to scheduling, the notion of priced (or weighted) timed automata [BFH+01b, ALP01] has been promoted as a useful extension of the classical model of timed automata allowing continuous consumption of resources (e.g. energy, money, pollution, etc.) to be modelled and analyzed. In this way one may distinguish different feasible schedules according to their consumption of resources (*i.e.*, accumulated cost) with obvious preference for the *optimal* schedule with minimal resource requirements.

Within the model of priced timed automata, the cost variables serve purely as *evaluation functions* or *observers*, *i.e.*, the behaviour of the underlying timed automata may in no way depend on these cost variables. As an important consequence of this restriction —and in contrast to the related models of constant slope and linear hybrid automata— a number of optimization problems have been shown decidable for priced timed automata including minimum-cost reachability [BFH+01b, ALP01, BBBR07], optimal (minimum and maximum cost) reachability in multi-priced settings [LR05] and cost-optimal infinite schedules [BBL04, BBL08] in terms of minimal (or maximal) cost per time ratio in the limit. Moreover UPPAAL Cora [BLR05b] provides an efficient tool for computing cost-optimal or near-optimal solutions to reachability questions, implementing a symbolic A^* algorithm based on a new data structure (so-called priced zones) allowing for efficient symbolic state-representation with additional cost-information.

Cost-extended versions of temporal logics such as CTL (branching-time) and LTL (linear-time) appear as a natural "generalizations" of the above optimization problems. Just as TCTL and MTL provide extensions of CTL and LTL with time-constrained modalities, WCTL and WMTL are extensions with *cost*-constrained modalities interpreted with respect to priced timed automata. Unfortunately, the addition of cost now turns out to come with a price: whereas the model-checking problems for timed automata with respect to TCTL and MTL are decidable, it has been shown in [BBR04] that model-checking priced timed automata with respect to WCTL is undecidable. Also, in [BBR05] it has recently been shown that the problem of determining cost-optimal winning strategies for priced timed games is not computable. In [BBM06] it has been shown that these negative results hold even for priced timed (game) automata with no more than three clocks.

Recently, the restriction of timed systems to a single clock has raised some attention, as it leads to much nicer decidability and complexity results. Indeed, the emptiness problem in single-clock timed automata becomes NLOGSPACE-Complete [LMS04] instead of PSPACE-Complete in the general framework [AD94]. Also, the emptiness problem is decidable for single-clock alternating timed automata and is undecidable for general alternating timed automata [LW05, OW05, LW08, OW07]. Even more recently, cost-optimal timed games have been proved decidable for one-clock priced timed games [BLMR06], and construction of almost-optimal strategies can be done.

In this paper we focus on model-checking problems for priced timed automata with a single clock. On the one hand, we show that the model-checking problem with respect to WCTL is PSPACE-Complete under the "single clock" assumption. This is rather surprising as model-checking TCTL (the only cost variable is the time elapsed) under the same assumption is already PSPACE-Complete [LMS04]. On the other hand, we prove that the model-checking problem with respect to WMTL, the linear-time counterpart of WCTL, is decidable if we add the extra requirements that there is only one cost variable which

is stopwatch (*i.e.*, with slopes in $\{0,1\}$). We also prove that those two conditions are necessary to get decidability, by proving that any slight extension of that model leads to undecidability.

The paper is organized as follows: In Section 1, we present the model of priced timed automata. Section 2 is devoted to the definition of WCTL, and to the proof that it is decidable when the model has only one clock. We propose an EXPTIME algorithm, which we then slightly modify so that it runs in PSPACE. Section 3 then handles the linear-time case: we first define WMTL, prove that it is decidable under the single-clock and single-stopwatch-cost assumptions, and that it is undecidable if we lift any of these restrictions.

1. Preliminaries

1.1. **Priced Timed Automata.** In the sequel, $\mathbb{R}_+$ denotes the set of nonnegative reals. Let $\mathcal{X}$ be a set of clock variables. The set of clock constraints (or guards) over $\mathcal{X}$ is defined by the grammar "$g ::= x \sim c \mid g \wedge g$" where $x \in \mathcal{X}$, $c \in \mathbb{N}$ and $\sim \in \{<, \leq, =, \geq, >\}$. The set of all clock constraints is denoted $\mathcal{B}(\mathcal{X})$. That a valuation $v\colon \mathcal{X} \to \mathbb{R}_+$ satisfies a clock constraint g is defined in a natural way (v satisfies $x \sim c$ whenever $v(x) \sim c$), and we then write $v \models g$. We denote by v_0 the valuation that assigns zero to all clock variables, by $v+t$ (with $t \in \mathbb{R}_+$) the valuation that assigns $v(x)+t$ to all $x \in \mathcal{X}$, and for $R \subseteq \mathcal{X}$ we write $[R \leftarrow 0]v$ to denote the valuation that assigns zero to all variables in R and agrees with v for all variables in $\mathcal{X} \setminus R$.

Definition 1.1. A *priced timed automaton* (PTA for short) is a tuple $\mathcal{A} = (Q, q_0, \mathcal{X}, T, \eta, (\mathsf{cost}_i)_{1\leq i\leq p})$ where Q is a finite set of *locations*, $q_0 \in Q$ is the *initial* location, $\mathcal{X}$ is a set of *clocks*, $T \subseteq Q \times \mathcal{B}(\mathcal{X}) \times 2^{\mathcal{X}} \times Q$ is the set of *transitions*, $\eta\colon Q \to \mathcal{B}(\mathcal{X})$ defines the *invariants* of each location, and each $\mathsf{cost}_i\colon Q \cup T \to \mathbb{N}$ is a *cost (or price) function*.

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

The semantics of a PTA $\mathcal{A}$ is given as a labeled timed transition system $\mathcal{T}_\mathcal{A} = (S, s_0, \to)$ where $S \subseteq Q \times \mathbb{R}_+^{\mathcal{X}}$ is the set of states, $s_0 = (q_0, v_0)$ is the initial state, and the transition relation $\to \subseteq S \times (T \cup \mathbb{R}_+) \times S$ is composed of delay and discrete moves defined as follows:

(1) *(discrete move)* $(q,v) \xrightarrow{e} (q',v')$ if $e = (q,g,R,q') \in E$ is s.t. $v \models g$, $v' = [R \leftarrow 0]v$, $v' \models \eta(q')$. The i-th cost of this discrete move is $\mathsf{cost}_i\big((q,v) \xrightarrow{e} (q',v')\big) = \mathsf{cost}_i(e)$.

(2) *(delay move)* $(q,v) \xrightarrow{t} (q,v+t)$ if $\forall 0 \leq t' \leq t$, $v+t' \models \eta(q)$. The i-th cost of this delay move is $\mathsf{cost}_i\big((q,v) \xrightarrow{t} (q,v+t)\big) = t \cdot \mathsf{cost}_i(q)$.

A discrete move or a delay move will be called a *simple move*. A *mixed move* $(q,v) \xrightarrow{t,e} (q',v')$ corresponds to the concatenation of a delay move and a discrete move. For technical reasons, we only consider non-blocking PTAs, because we will further interpret logical formulas over infinite paths. The i-th cost of this mixed move is the sum of the i-th costs of the two moves.

A finite (resp. infinite) *run* of a PTA is a finite (resp. infinite) sequence of mixed moves in the underlying transition system. A run of $\mathcal{A}$ will thus be distinguished from a path in $\mathcal{T}_\mathcal{A}$, which is composed of simple moves and where stuttering of delay moves is allowed. Note however that a path in $\mathcal{T}_\mathcal{A}$ is naturally associated with a run in $\mathcal{A}$. The i-th cost of a run

ϱ in $\mathcal{A}$ (resp. path ϱ in $\mathcal{T}_{\mathcal{A}}$) is the sum of the i-th costs of the mixed (resp. simple) moves composing the run (resp. path), and is denoted $\mathsf{cost}_i(\varrho)$. The length $|\varrho|$ of a finite run $\varrho = s_0 \xrightarrow{t_1,e_1} s_1 \xrightarrow{t_2,e_2} \cdots \xrightarrow{t_n,e_n} s_n$ is n. A *position* along ϱ is a nonnegative integer $\pi \leq |\varrho|$. Given a position π, $\varrho[\pi]$ denotes the corresponding state s_π, whereas $\varrho_{\leq\pi}$ denotes the finite prefix of ϱ ending at position π, and $\varrho_{\geq\pi}$ is the suffix starting in π.

Remark 1.2. In the model of priced timed automata, the cost variables only play the role of *observers* (they are *history variables* in the sense of [OG76, AL88]): the values of these variables don't constrain the behaviour of the system (the behaviours of a priced timed automaton are those of the underlying timed automaton), but can be used as evaluation functions. For instance, problems such as "optimal reachability" [BFH+01b, ALP01], "optimal infinite schedules" [BBL04] or "optimal reachability timed games" [ABM04, BCFL04, BBR05, BBM06] have recently been investigated. The problem we consider in this paper is closely related to these kinds of problems: we will use temporal logics as a language for evaluating the performances of a system.

1.2. **Example.** The PTA of Figure 1 models a never-ending process of repairing problems, which are bound to occur repeatedly with a certain frequency. The repair of a problem has a certain cost, captured in the model by the cost variable c. As soon as a problem occurs (modeled by the Problem location) the value of c grows with rate 3, until actual repair is taking place in one of the locations Cheap (rate 2) or Expensive (rate 4). At most 20 time units after the occurrence of a problem it will have been repaired one way or another.

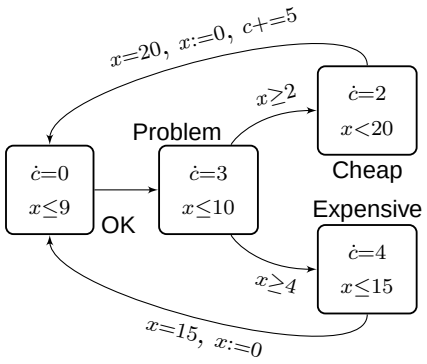


Figure 1: Repair problem as a PTA

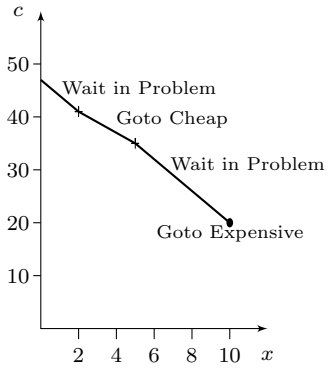


Figure 2: Minimum cost of repair and associated strategy in location Problem

In this setting we are interested in properties concerning the cost of repairs. For instance, we would like to express that whenever a problem occurs, it *may* be repaired (*i.e.* reach the location OK) within a total cost of 47. In fact Figure 2 gives the minimum cost of repair —as well as an optimal strategy— for any state of the form $(\mathsf{Problem}, x)$ with $x \in [0, 10]$. Correspondingly, the minimum cost of reaching OK from states of the form (Cheap, x) (resp. $(\mathsf{Expensive}, x)$) is given by the expression $45 - 2x$ (resp. $60 - 4x$). Symmetrically, we would like to express properties on the worst cost to repair, or to link the uptime with the (best, worst) cost of repairing. As will be illustrated later, extending temporal logics with cost informations provides a nice setting for expressing such properties.

2. Model Checking Branching-Time Logics

We first focus on the case of branching-time logics. From this point on, AP denotes a fixed, finite, non-empty set of atomic propositions. We first define the cost-extended version of CTL.

2.1. The Logic WCTL.

The logic WCTL[1] [BBR04] extends CTL with cost constraints. Its syntax is given by the following grammar:

$$\text{WCTL} \ni \varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}\,\varphi \mathsf{U}_{\mathsf{cost}\sim c}\varphi \mid \mathsf{A}\,\varphi \mathsf{U}_{\mathsf{cost}\sim c}\varphi$$

where $a \in \mathsf{AP}$, cost is a cost function, c ranges over $\mathbb{N}$, and $\sim \in \{<, \leq, =, \geq, >\}$.

We interpret formulas of WCTL over labeled PTA, *i.e.* PTA having a labeling function ℓ which associates with every location q a subset of AP. We identify each cost appearing in the WCTL formulas with the cost having the same name in the model (which is assumed to exist).

Definition 2.1. Let $\mathcal{A}$ be a labeled PTA. The satisfaction relation of WCTL is defined over configurations (q, v) of $\mathcal{A}$ as follows:

$$\begin{array}{rcl}
(q,v) \models a & \Leftrightarrow & a \in \ell(q) \\
(q,v) \models \neg\varphi & \Leftrightarrow & (q,v) \not\models \varphi \\
(q,v) \models \varphi \vee \psi & \Leftrightarrow & (q,v) \models \varphi \text{ or } (q,v) \models \psi \\
(q,v) \models \mathsf{E}\,\varphi \mathsf{U}_{\mathsf{cost}\sim c}\psi & \Leftrightarrow & \text{there is an infinite run } \varrho \text{ in } \mathcal{A} \\
 & & \text{from } (q,v) \text{ s.t. } \varrho \models \varphi \mathsf{U}_{\mathsf{cost}\sim c}\psi \\
(q,v) \models \mathsf{A}\,\varphi \mathsf{U}_{\mathsf{cost}\sim c}\psi & \Leftrightarrow & \text{any infinite run } \varrho \text{ in } \mathcal{A} \text{ from } (q,v) \\
 & & \text{satisfies } \varrho \models \varphi \mathsf{U}_{\mathsf{cost}\sim c}\psi \\
\varrho \models \varphi \mathsf{U}_{\mathsf{cost}\sim c}\psi & \Leftrightarrow & \text{there exists a position } \pi > 0 \text{ along } \varrho \text{ s.t.} \\
 & & \varrho[\pi] \models \psi, \text{ for every position } 0 < \pi' < \pi, \\
 & & \varrho[\pi'] \models \varphi, \text{ and } \mathsf{cost}(\varrho_{\leq\pi}) \sim c
\end{array}$$

If $\mathcal{A}$ is not clear from the context, we may write $\mathcal{A}, (q, v) \models \varphi$ instead of simply $(q, v) \models \varphi$.

As usual, we will use shorthands such as "$\mathtt{true} \overset{\text{def}}{\Leftrightarrow} a \vee \neg a$", "$(\varphi \Rightarrow \psi) \overset{\text{def}}{\Leftrightarrow} \neg\varphi \vee \psi$", "$\mathsf{E}\,\mathsf{F}_{\mathsf{cost}\sim c}\varphi \overset{\text{def}}{\Leftrightarrow} \mathsf{E}\,\mathtt{true}\,\mathsf{U}_{\mathsf{cost}\sim c}\varphi$", and "$\mathsf{A}\,\mathsf{G}_{\mathsf{cost}\sim c}\varphi \overset{\text{def}}{\Leftrightarrow} \neg\mathsf{E}\,\mathsf{F}_{\mathsf{cost}\sim c}\neg\varphi$". Moreover, if the cost function cost is unique or clear from the context, we may write $\varphi\mathsf{U}_{\sim c}\psi$ instead of $\varphi\mathsf{U}_{\mathsf{cost}\sim c}\psi$. Finally, we omit to mention the subscript "$\sim c$" when it is equivalent to "≥ 0" (thus imposing no real constraint).

Example 2.2. We go back to our example of Section 1.2. That it is always possible to repair a problem with cost at most 47 can be expressed in WCTL with the following formula:

$$\mathsf{A}\,\mathsf{G}\big(\mathsf{Problem} \Rightarrow \mathsf{E}\,\mathsf{F}_{c\leq 47}\mathsf{OK}\big).$$

We can also express that the worst cost to repair is 56, in the sense that state Repair can always be reached within this cost:

$$\mathsf{A}\,\mathsf{G}\big(\mathsf{Problem} \Rightarrow \mathsf{A}\,\mathsf{F}_{c\leq 56}\mathsf{OK}\big).$$

[1]WCTL stands for "Weighted CTL", following [BBR04] terminology. It would have been more natural to call it "Priced CTL" (PCTL) in our setting, but this would have been confusing with "Probabilistic CTL" [HJ94].

Now, considering time as a special case of a cost (with constant slope 1), we can express properties relating the time elapsed in the OK state and the cost to repair:

$$\mathsf{A}\,\mathsf{G}\big(\neg\mathsf{E}\,(\mathsf{OK}\;\mathsf{U}_{t\geq 8}(\mathsf{Problem}\wedge\neg\mathsf{E}\,\mathsf{F}_{c<30}\mathsf{OK}))\big).$$

This expresses that if the system spends at least 8 (consecutive) time units in the OK state, then the next Problem can be repaired with cost at most 30.

The main result of this section is the following theorem:

Theorem 2.3. *Model-checking WCTL on one-clock PTA is PSPACE-Complete.*

The PSPACE lower bound can be proved by a direct adaptation of the PSPACE-Hardness proof for the model-checking of TCTL, the restriction of WCTL to time constraints, over one-clock timed automata [LMS04].

The PSPACE upper bound is more involved, and will be done in two steps:

(1) first we will exhibit a set of regions which will be correct for model-checking WCTL formulas, see Section 2.2;
(2) then we will use this result to propose a PSPACE algorithm for model-checking WCTL, see Section 2.3.

Finally, it is worth reminding here that the model-checking of WCTL over priced timed automata with three clocks is undecidable [BBM06].

2.2. **Sufficient Granularity for WCTL.** The proof of Theorem 2.3 partly relies on the following proposition, which exhibits, for every WCTL formula Φ, a set of *regions* within which the truth of Φ is uniform. Note that these are not the classical regions as defined in [AD94, ACD93], because their granularity needs to be refined in order to be correct. Computing a sufficient granularity was already a key step for checking duration properties in simple timed automata [BES93].

Proposition 2.4. *Let Φ be a WCTL formula and let $\mathcal{A}$ be a one-clock PTA. Then there exist a finite set of constants $\{a_0, ..., a_n\}$ satisfying the following conditions:*

- $0 = a_0 < a_1 < \ldots < a_n < a_{n+1} = +\infty$;
- *for every location q of $\mathcal{A}$, for every $0 \leq i \leq n$, the truth of Φ is uniform over $\{(q, x) \mid a_i < x < a_{i+1}\}$;*
- *$\{a_0, ..., a_n\}$ contains all the constants appearing in clock constraints of $\mathcal{A}$;*
- *the constants are integral multiples of $1/C^{\hbar(\Phi)}$ where $\hbar(\Phi)$ is the* constrained temporal height *of Φ, i.e., the maximal number of nested constrained modalities[2] in Φ, and C is the lcm of all positive costs labeling a location of $\mathcal{A}$;*
- *a_n equals the largest constant M appearing in the guards of $\mathcal{A}$;*

In particular, we have $n \leq M \cdot C^{\hbar(\Phi)} + 1$.

As a corollary, we recover the partial decidability result of [BBR04], stating that the model-checking of one-clock PTA with a *stopwatch cost*[3] against WCTL formulas is decidable using classical one-dimensional regions of timed automata (*i.e.*, with granularity 1).

[2] With "constrained modality" we mean a modality decorated with a constraining interval different from $(0, +\infty)$.

[3] *I.e.*, cost with rates in $\{0, 1\}$.

Proof. The proof of this proposition is by structural induction on Φ. The cases of atomic propositions and boolean combinations are straightforward; unconstrained modalities require no refinement of the granularity (the basic CTL algorithm is correct and does not need to refine the granularity); we will thus focus on constrained modalities.

2.2.1. *We first assume that $\mathcal{A}$ has no discrete costs.* (*i.e.* $\mathsf{cost}(T) = \{0\}$), the extension to the general case will be presented at the end of the proof.

▶ **We first focus on the case when $\Phi = \mathsf{E}\,\varphi \mathsf{U}_{\mathsf{cost}\sim c}\psi$** (we simply write $\Phi = \mathsf{E}\,\varphi \mathsf{U}_{\sim c}\psi$, and assume that cost is the only cost of $\mathcal{A}$, as its other costs play no role in the problem). Assume that the result has been proved for the WCTL subformulas φ and ψ, and that we have merged all constants for φ and ψ: we thus have constants $0 = a_0 < a_1 < \ldots < a_n < a_{n+1} = +\infty$ such that for every location q of $\mathcal{A}$, for every $0 \leq i \leq n$, the truth of φ and that of ψ are both uniform over $\{(q,x) \mid a_i < x < a_{i+1}\}$. By induction hypothesis, the granularity of these constants is $1/C^{\max(\hbar(\varphi),\hbar(\psi))} = 1/C^{\hbar(\Phi)-1}$. We will exhibit extra constants such that the above proposition then also holds for the formula Φ. For the sake of simplicity, we will call *regions* all elementary intervals (a_i, a_{i+1}) and singletons $\{a_i\}$.

In order to compute the set of states satisfying $\mathsf{E}\,\varphi \mathsf{U}_{\sim c}\psi$, for every state (q,x) we compute all costs of paths from (q,x) to some region (q',r), along which φ always holds after a discrete action has been done, and such that a ψ-state can immediately be reached via a discrete action from (q',r). We then check whether we can achieve a cost satisfying "$\sim c$" for the mentioned ψ-state. We thus first explain how we compute the set of possible costs between a state (q,x) and a region (q',r) in $\mathcal{A}$. Indeed, for checking the existence of a run satisfying $\varphi \mathsf{U}_{\sim c}\psi$, we will first remove discrete transitions leading to states not satisfying φ, and then compute all possible costs of runs from (q,x) to some (q',r), where (q',r) is a ψ-state just reached by a discrete action, in the restricted graph.

For each index i, we restrict the automaton $\mathcal{A}$ to transitions whose guards contain the interval (a_i, a_{i+1}), and that do not reset the clock. We denote by $\mathcal{A}_i$ this restricted automaton. Let q and q' be two locations of $\mathcal{A}_i$. As stated by the following lemma, the set of costs of paths between (q,a_i) and (q',a_{i+1}) is an interval that can be easily computed:

Lemma 2.5. *We assume $a_{i+1} \neq +\infty$. Let $S_i(q,q')$ be the set of locations that are reachable from (q,a_i) and co-reachable from (q',a_{i+1}) in $\mathcal{T}_{\mathcal{A}_i}$, and assume it is non-empty (*i.e.*, there is a path joining those two states). Let $c_{\min}^{i,q,q'}$ and $c_{\max}^{i,q,q'}$ be the minimum and maximum costs among the costs of locations in $S_i(q,q')$. Then the set of all possible costs of paths in $\mathcal{T}_{\mathcal{A}_i}$ going from (q,a_i) to (q',a_{i+1}) is an interval $\langle (a_{i+1}-a_i)\cdot c_{\min}^{i,q,q'}, (a_{i+1}-a_i)\cdot c_{\max}^{i,q,q'}\rangle$. The interval is left-closed iff there exist two locations r and s (with possibly $r=s$) in $S_i(q,q')$ with cost $c_{\min}^{i,q,q'}$ such that*[4] *$(q,a_i) \rightsquigarrow_{\mathcal{A}_i}^* (r,a_i)$, $(r,a_i) \rightsquigarrow_{\mathcal{A}_i}^* (s,a_{i+1})$, and $(s,a_{i+1}) \rightsquigarrow_{\mathcal{A}_i}^* (q',a_{i+1})$. The interval is right-closed iff there exists two locations r and s in $S_i(q,q')$ with cost $c_{\max}^{i,q,q'}$ such that $(q,a_i) \rightsquigarrow_{\mathcal{A}_i}^* (r,a_i)$, $(r,a_i) \rightsquigarrow_{\mathcal{A}_i}^* (s,a_{i+1})$, and $(s,a_{i+1}) \rightsquigarrow_{\mathcal{A}_i}^* (q',a_{i+1})$.*

The conditions on left/right-closures characterize the fact that it is possible to instantaneously reach/leave a location with minimal/maximal cost, or if a small positive delay has to elapse (due to a strict guard).

[4]The notation $\alpha \rightsquigarrow_{\mathcal{A}_i}^* \alpha'$ means that there is a path in $\mathcal{T}_{\mathcal{A}_i}$ from α to α'.

Proof. Obviously the costs of all paths in $\mathcal{T}_{\mathcal{A}_i}$ from (q, a_i) to (q', a_{i+1}) belong to the interval $(a_{i+1} - a_i) \cdot [c_{\min}^{i,q,q'}, c_{\max}^{i,q,q'}]$. We will now prove that the set of costs is an interval containing $(a_{i+1} - a_i) \cdot (c_{\min}^{i,q,q'}, c_{\max}^{i,q,q'})$.

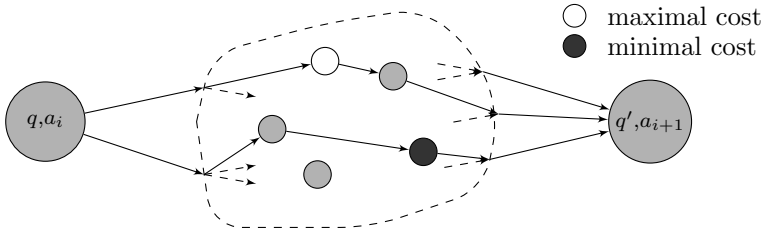


Figure 3: The set of costs between two states is an interval.

Let $\tau_{\min}$ (resp. $\tau_{\max}$) be a sequence of transitions in $\mathcal{A}_i$ leading from (q, a_i) to (q', a_{i+1}) and going through a location with minimal (resp. maximal) cost (see Figure 3). Easily enough, the possible costs of the paths following $\tau_{\min}$ (resp. $\tau_{\max}$) form an interval whose left (resp. right) bound is $c_{\min}^{i,q,q'} \cdot (a_{i+1} - a_i)$ (resp. $c_{\max}^{i,q,q'} \cdot (a_{i+1} - a_i)$).

Now, if c and c' are the respective costs of q and q', then $\frac{1}{2} \cdot (c + c') \cdot (a_{i+1} - a_i)$ is in both intervals. Indeed, the path following $\tau_{\min}$ (resp. $\tau_{\max}$) which delays $\frac{1}{2} \cdot (a_{i+1} - a_i)$ time units in q, then directly goes to q' and waits there for the remaining $\frac{1}{2} \cdot (a_{i+1} - a_i)$ time units achieves the above-mentioned cost. This implies that the set of all possible costs is an interval.

The bound $c_{\min}^{i,q,q'} \cdot (a_{i+1} - a_i)$ is reached iff there is a path from (q, a_i) to (q', a_{i+1}) which delays only in locations with cost $c_{\min}^{i,q,q'}$. This is precisely the condition expressed in the lemma. The same holds for the upper bound $c_{\max}^{i,q,q'} \cdot (a_{i+1} - a_i)$. □

Similar results clearly hold for other kinds of regions:

- between a state (q, a_i) and a region $(q', (a_i, a_{i+1}))$ with $a_{i+1} \neq +\infty$, the set of possible costs is an interval $\langle 0, c_{\max}^{i,q,q'} \cdot (a_{i+1} - a_i)\rangle$, where 0 can be reached iff it is possible to go from (q, a_i) to some state (q'', a_i) co-reachable from (q', x) for some $x \in (a_i, a_{i+1})$, and $\mathsf{cost}(q'') = 0$.
- between a state (q, x), with $x \in (a_i, a_{i+1})$, and (q', a_{i+1}), the set of costs is $(a_{i+1} - x) \cdot \langle c_{\min}^{i,q,q'}, c_{\max}^{i,q,q'}\rangle$, with similar conditions as above for the bounds of the interval.
- between a state (q, x), with $x \in (a_i, a_{i+1})$, and region $(q', (a_i, a_{i+1}))$ (assuming $a_{i+1} \neq +\infty$), the set of possible costs is $[0, c_{\max}^{i,q,q'} \cdot (a_{i+1} - x))$;
- between a state (q, a_n) and a region $(q', (a_n, +\infty))$, the set of possible costs is either $[0, 0]$, if no positive cost rate is reachable and co-reachable, or $\langle 0, +\infty)$ otherwise. If the latter case, 0 can be achieved iff it is possible to reach a state (q'', a_n) with $\mathsf{cost}(q'') = 0$;
- between a state (q, x) with $x \in (a_n, +\infty)$ and a region $(q', (a_n, +\infty))$, the set of costs is either $[0, 0]$ or $[0, +\infty)$, with the same conditions as previously.

We use these computations and build a graph G labeled by intervals which will store all possible costs between symbolic states (*i.e.*, pairs (q, r), where q is a location and r a region) in $\mathcal{T}_{\mathcal{A}}$. Vertices of G are pairs $(q, \{a_i\})$ and $(q, (a_i, a_{i+1}))$, and tuples $(q, x, \{a_i\})$ and $(q, x, (a_i, a_{i+1}))$, where q is a location of $\mathcal{A}$. Their roles are as follows: vertices of the

form (q, x, r) are used to initiate a computation, they represent a state (q, x) with $x \in r$. States $(q, \{a_i\})$ are "regular" steps in the computation, while states $(q, (a_i, a_{i+1}))$ are used either for finishing a computation, or just before resetting the clock (there will be no edge from $(q, (a_i, a_{i+1}))$ to any $(q', \{a_{i+1}\})$).

Edges of G are defined as follows:

- $(q, \{a_i\}) \to (q', \{a_{i+1}\})$ if there is a path from (q, a_i) to (q', a_{i+1}). This edge is then labeled with an interval $\langle (a_{i+1} - a_i) \cdot c_{\min}^{i,q,q'}, (a_{i+1} - a_i) \cdot c_{\max}^{i,q,q'} \rangle$, the nature of the interval (left-closed and/or right-closed) depending on the criteria exposed in Lemma 2.5.
- $(q, \{a_i\}) \to (q', \{a_i\})$ if there is an instantaneous path from (q, a_i) to (q', a_i) in $\mathcal{A}$, the edge is then labeled with the interval $[0, 0]$ (because we assumed there are no discrete costs on transitions of $\mathcal{A}$).
- $(q, \{a_i\}) \to (q', \{a_0\})$ if there is a transition in $\mathcal{A}$ enabled when the value of the clock is a_i and resetting the clock. It is labeled with $[0, 0]$.
- $(q, (a_i, a_{i+1})) \to (q', \{a_0\})$ if there is a transition in $\mathcal{A}$ enabled when the value of the clock is in (a_i, a_{i+1}) and resetting the clock. It is labeled with $[0, 0]$.
- $(q, \{a_i\}) \to (q', (a_i, a_{i+1}))$ if there is a path from (q, a_i) to some (q', α) with $a_i < \alpha < a_{i+1}$. This edge is labeled with the interval $\langle 0, (a_{i+1} - a_i) \cdot c_{\max}^{i,q,q'} \rangle$.
- $(q, x, \{a_i\}) \to (q, \{a_i\})$ labeled with $[0, 0]$.
- $(q, x, (a_i, a_{i+1})) \to (q', \{a_{i+1}\})$ if there is a path from some (q, α) with $a_i < \alpha < a_{i+1}$ to (q', a_{i+1}). This edge is labeled with $(a_{i+1} - x) \cdot \langle c_{\min}^{i,q,q'}, c_{\max}^{i,q,q'} \rangle$.
- $(q, x, (a_i, a_{i+1})) \to (q', (a_i, a_{i+1}))$ labeled with $[0, (a_{i+1} - x) \cdot c_{\max}^{i,q,q'})$.

Figure 4 represents one part of this graph. Note that each path π of this graph is naturally associated with an interval $\iota(\pi)$ (possibly depending on variable x if we start from a node $(q, x, (a_i, a_{i+1}))$) by summing up all intervals labeling transitions of π.

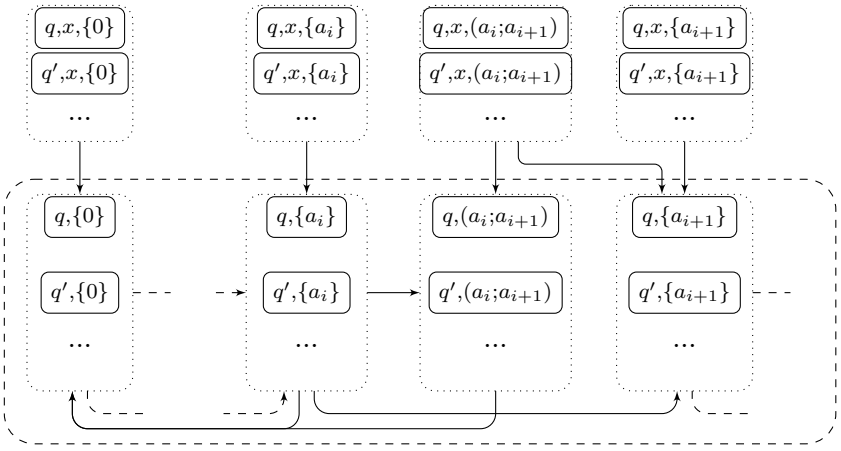


Figure 4: (Schematic) representation of the graph G (intervals labeling transitions have been omitted to improve readability)

The correctness of graph G w.r.t. costs is stated by the following lemma, which is a direct consequence of the previous investigations.

Lemma 2.6. *Let q and q' be two locations of $\mathcal{A}$. Let r and r' be two regions, and let $\alpha \in r$. Let $d \in \mathbb{R}^+$. There exists a path π in G from a state (q, x, r) to (q', r') with $\iota(\pi)(\alpha) \ni d$*

if, and only if, there is a path in $\mathcal{T}_{\mathcal{A}}$ with total cost d, and going from (q,α) to some (q',β) with $\beta \in r'$.

Corollary 2.7. *Fix two regions r and r'. Then the set of possible costs of paths in G from (q,x,r) to (q',r') is of the form*

$$\bigcup_{m\in\mathbb{N}} \langle \alpha_m - \beta_m \cdot x, \alpha'_m - \beta'_m \cdot x\rangle$$

(possibly with β_m and/or $\beta'_m = 0$, and/or $\alpha'_m = +\infty$). Moreover,

- *all constants α_m and α'_m are either integral multiples of $1/C^{\max(\hbar(\varphi),\hbar(\psi))}$ or $+\infty$, and constants β_m and β'_m are either costs of the automaton or 0;*
- *if $r = (a_n, +\infty)$, then $\beta_m = \beta'_m = 0$ for all m.*

Proof. Applying Lemma 2.6, the union of the costs of all paths in G from (q,x,r) to (q',r') represents the set of all possible costs of paths in $\mathcal{T}_{\mathcal{A}}$ from (q,α) with $\alpha \in r$ to some (q',β) with $\beta \in r'$. This set can be written as the countable union, for each $m \in \mathbb{N}$, of the costs of paths of length m in G, thus a countable union of (a finite union of) intervals. Now, any path in G contains at most one transition issued from a state (q,x,r). Thus, coefficients β_m are either 0, or the cost of some location of $\mathcal{A}$.

Coefficients α_m are then integral combinations of terms of the form $c\cdot(a_{i+1}-a_i)$ where c is the cost of some location. As all a_i's are integral multiples of $1/C^{\max(\hbar(\varphi),\hbar(\psi))}$, we get what we expected. The special form for the unbounded region is obvious from the construction of G. □

Lemma 2.8. *For every location q, and for $\Phi = \mathsf{E}\,\varphi \mathsf{U}_{\sim c}\psi \in$ WCTL, the set of clock values x such that (q,x) satisfies Φ is a finite union of intervals. Moreover,*

- *the bounds of those intervals are integral multiples of $1/C^{\hbar(\Phi)}$;*
- *the largest finite bound of those intervals is at most the maximal constant appearing in the guards of the automaton.*

Proof. The set of clock values x such that (q,x) satisfies $\mathsf{E}\,\varphi\mathsf{U}_{\sim c}\psi$ can be written as

$$\bigcup_{r \text{ region}} \{x \in r \mid (q,x) \models \mathsf{E}\,\varphi\mathsf{U}_{\sim c}\psi\}.$$

There is a finite number of regions. For the unbounded region, the set of possible costs does not depend on the initial value of x, and thus either the whole region satisfies the formula, or no point in that region does. Fix a bounded region r, and $x \in r$. Then, $(q,x) \models \mathsf{E}\,\varphi\mathsf{U}\psi$ if, and only if there exists a path in $\mathcal{T}_{\mathcal{A}}$ from (q,x) to some (q',r') such that (i) a ψ-state is immediately reachable from (q',r') by a discrete move, and (ii) along that path, all states traversed just after a discrete move satisfy φ. For each pair (q,r) leading to a ψ-state, we can applying Corollary 2.7 on the graph just obtained after having removed discrete transitions not leading to a φ-state. The set of possible costs of paths satisfying $\varphi\mathsf{U}\psi$ is then a (countable) union of the form $\bigcup_{m\in\mathbb{N}}\langle \alpha_m - \beta_m\cdot x, \alpha'_m - \beta'_m \cdot x\rangle$ with the constraints on constants described in the previous corollary. We assume that $r = (a_i, a_{i+1})$ and that the constraint $\sim c$ is either $\leq c$, or $< c$, or $= c$ (the other cases would be handled in a similar way). If $\alpha_m - \beta_m \cdot a_i > c$, then the interval $\langle \alpha_m - \beta_m\cdot x, \alpha'_m - \beta'_m\cdot x\rangle$ plays no role for the satisfaction of formula $\mathsf{E}\,\varphi\mathsf{U}_{\sim c}\psi$ in the region r, we can thus remove this interval from the union. Now, β_m is an integer which is either null or divides C. Thus as α_m is an integral multiple of $1/C^{\max(\hbar(\varphi),\hbar(\psi))} = 1/C^{\hbar(\Phi)-1}$, left-most bounds of interesting intervals

can be $\alpha_m - \beta_m \cdot x$ for finitely many α_m's and β_m's (with the further option closed or open). Fix some α and β, and also fix some β'. For two intervals $\langle \alpha - \beta \cdot x, \alpha'_1 - \beta' \cdot x \rangle$ and $\langle \alpha - \beta \cdot x, \alpha'_2 - \beta' \cdot x \rangle$ in the above union, it is sufficient to keep only the one with the largest α' (because the other is included in this interval). Thus, in the above countable union of intervals, we can select a finite union of intervals which will be sufficient for checking property $\mathsf{E}\,\varphi \mathsf{U}_{\sim c} \psi$ in region r.

We thus assume that the set of costs of paths which may witness formula $\varphi \mathsf{U}_{\sim c} \psi$ is a finite union $\bigcup_{m=1}^{k} \langle \alpha_m - \beta_m \cdot x, \alpha'_m - \beta'_m \cdot x \rangle$ with α_m and α'_m in $\mathbb{N}/C^{\hbar(\Phi)-1}$ and β_m and β'_m in $(C/\mathbb{N}^* \cap \mathbb{N}) \cup \{0\}$. Now, the bounds a'_i of the intervals of positions where Φ holds should correspond to values of x where one of the bounds $\alpha_m - \beta_m \cdot x$ or $\alpha'_m - \beta'_m \cdot x$ exactly equals c. It easily follows that those bounds a'_i are integral multiples of $1/C^{\hbar(\Phi)}$, as required.

This proves that we get only finitely many new intervals, and that the largest constant is the same as for φ and ψ (because of the initial remark on the unbounded region), thus it is the largest constant appearing in the automaton. □

This concludes the induction step for formula $\mathsf{E}\,\varphi \mathsf{U}_{\sim c} \psi$ when the automaton has no discrete cost. We will now handle the cases of the formulas $\mathsf{E}\,\mathsf{G}_{\geq c}\mathtt{false}$ and $\mathsf{E}\,\mathsf{G}_{=c}\mathtt{false}$ before giving several equivalences to handle all the other cases.

▶ **We now consider the formulas $\Phi = \mathsf{E}\,\mathsf{G}_{=c}\mathtt{false}$ and $\Phi = \mathsf{E}\,\mathsf{G}_{\geq c}\mathtt{false}$:** handling those modalities is sufficient for our proof, as we explain later.

To handle those two formulas, we will extend the graph G defined previously for the initial automaton (with non-refined classical regions). We add to the graph G new "final" states which are triples $\overline{(q, y, r)}$ (we overline it to distinguish it from the initial states). Such a state has the same incoming transitions as the state (q, r), except that we will enforce the final value of the clock be y, and not any value in r. For instance, a transition $(q, \{a_i\}) \to \overline{(q', y, (a_i, a_{i+1}))}$ will be labeled by the interval $\langle 0, (y - a_i) \cdot c_{\max}^{i,q,q'}]$ (remember the construction of the graph on page 9). From each of these new final states, we add an outgoing transition labeled by a finite union of intervals corresponding to all the costs of a single mixed move leading to a state from which infinite runs are possible. These intervals are either of the form $\langle 0, \gamma \cdot (b - y) \rangle$, or of the form $\langle \gamma \cdot (a - y), \gamma \cdot (b - y) \rangle$ where γ is the cost rate of the corresponding state, and a, b are constants of the automaton.

Now, we omit the details, but they are very similar to those for the original graph G. In this extended graph, the set of possible costs of paths in $\mathcal{T}_{\mathcal{A}}$ from (q, x) to (q', y) corresponds to the set of costs of paths in the new graph from (q, x, r) to $\overline{(q', y, r')}$ and is a countable union

$$\bigcup_{m \in \mathbb{N}} \langle \alpha_m - \beta_m \cdot x + \gamma_m \cdot y, \alpha'_m - \beta'_m \cdot x + \gamma'_m \cdot y \rangle$$

where α_m and α'_m are integers (or $+\infty$), and β_m, β'_m, γ_m and γ'_m are costs of the automaton or 0 (result similar to Corollary 2.7). We can even be more precise: β_m is either 0 or the cost rate of q, whereas β'_m is the cost rate of q. Similarly, γ_m is either 0 or the cost rate of q', and γ'_m is the cost rate of q'.

A state (q, x) will satisfy the formula $\Phi = \mathsf{E}\,\mathsf{G}_{=c}\mathtt{false}$ whenever there is a run ϱ in $\mathcal{A}$ such that it can be decomposed into $\varrho = \varrho_1 \cdot \varrho_2 \cdot \varrho_3$ such that the cost of ϱ_1 is strictly less than c, the cost of $\varrho_1 \cdot \varrho_2$ is strictly larger than c and ϱ_2 corresponds to a single mixed move. That is, whenever there exists a path from (q, x, r) to $\overline{(q', y, r')}$ of cost less than c s.t., when adding up the outgoing cost of a single mixed move, we get a cost larger than c. As in

Lemma 2.8, we can restrict the above union to a finite union, and we thus only need to solve finitely many linear systems of inequations. Then, we can analyze all possible cases for the bounds where the truth of Φ changes, and as previously, we see that the granularity needs only to be refined by $1/C$, hence the granularity which is required is $1/C$ (since we started from the classical region automaton, with non-refined constants).

A state (q, x) satisfies $\mathsf{E}\,\mathsf{G}_{\geq c}\mathtt{false}$ whenever there is an infinite run from (q, x) for which the cost of all its prefixes is strictly less than c (though the limit of these costs can be c itself). In such a run, there is a prefix of cost strictly less than c and from that point on, the cost of each mixed move is very close to 0 (and indeed as close as we want to 0). We thus proceed as follows: we fix a location q and a region r. For every x and y, we compute the set of possible costs between (q, x) and (q, y) for $x, y \in r$. This is a countable union

$$\bigcup_{m\in\mathbb{N}} \langle \alpha_m - \beta_m \cdot x + \gamma_m \cdot y, \alpha'_m \rangle$$

after having simplified the previous union in which β'_m and γ'_m were both equal to the cost of location q. For each of the terms of the union, we distinguish between several cases:

- if $\beta_m = \gamma_m = \alpha_m = 0$, then there is a cycle which can be iterated from (q, r), and the global cost will be as small as we want. If the left-most bound of the interval is closed, then we can ensure a zero-cost, otherwise we cannot ensure a zero-cost.
- if $\beta_m = \gamma_m = 0$ but $\alpha_m > 0$, then there is no corresponding cycle that can be iterated without the cost to diverge.
- if $\beta_m = 0$ but $\gamma_m > 0$ is the cost of q, then the only chance to be able to iterate a cycle without paying too much is to choose y be the left-most point a of the region r. Then, either $\alpha_m + \gamma_m \cdot a = 0$, in which case we can iterate a cycle, or $\alpha_m + \gamma_m \cdot a > 0$, in which case we cannot iterate a cycle.
- if $\beta_m = 0$ but $\gamma_m > 0$ is the cost of q, a similar reasoning can be done, but with the right-most bound b of r.
- if $\beta_m = \gamma_m > 0$ is the cost of location q, then it is not difficult to check that α_m is then not smaller than $\beta_m \cdot (b - a)$ (this can be checked on the graph G). Hence, a corresponding cycle can only be iterated if $a = b$, and thus if r is a punctual region.

The analysis of all these cases show that we only need to look at terms of the union such that $\alpha_m - \beta_m \cdot b + \gamma_m \cdot a = 0$, and either $a = b$, or the $\alpha_m \cdot \beta_m \cdot \gamma_m = 0$. Moreover, for each such constraint, it is only necessary to look at one of the witnessing intervals. We see that this set of states is a set of regions (we do not need to refine the region: a whole region either satisfies the property, or does not satisfy the property).

That way, we can compute the set of states S_0 from which there exists an infinite run with a cost as small as possible (though possibly not zero).

It remains to describe the set of states from which there is a finite path of cost strictly less than c and reaching a state of S_0. This can easily be done using the extended graph G we have presented above.

► **We now explain how we reduce all the other cases to the previous ones.** We consider the case of formula $\mathsf{A}\,\varphi\mathsf{U}_{\sim c}\psi$, still assuming that the automaton has no discrete costs. We prove this result by reducing to the previous case. We consider the region automaton of $\mathcal{A}$ w.r.t. constants $(a_i)_{0\leq i\leq n+1}$ mentioned earlier (correct for subformulas φ and ψ), we assume it is still a timed automaton (truth of formulas in the original automaton and in this region automaton is then equivalent).

We moreover assume that we have two copies of each state, labeled with two extra atomic proposition $\mathsf{has\_paid}$ and $\mathsf{can\_have\_not\_paid}$ which characterize when the last move had a positive cost, and when it could have no cost (for instance an instantaneous transition or a transition from a location where the cost rate is null). We denote the new automaton by $\mathcal{A}_{\text{ext}}$, and give now a list of equivalences, not difficult to check, and useful for proving the induction step for formulas of the form $\mathsf{A}\,\varphi\mathsf{U}_{\sim c}\psi$.

- $(q,x),\mathcal{A} \models \mathsf{A}\,\varphi\mathsf{U}_{\geq c}\psi$ iff $(q,x),\mathcal{A} \models \mathsf{A}\,\varphi\mathsf{U}\psi \wedge \mathsf{A}\,\mathsf{G}_{<c}(\mathsf{A}\,\varphi\mathsf{U}\psi) \wedge \mathsf{A}\,\mathsf{F}_{\geq c}\mathtt{true}$;
- $(q,x),\mathcal{A} \models \mathsf{A}\,\varphi\mathsf{U}_{>c}\psi$ iff $(q,x),\mathcal{A} \models \mathsf{A}\,\varphi\mathsf{U}\psi \wedge \mathsf{A}\,\mathsf{G}_{\leq c}(\mathsf{A}\,\varphi\mathsf{U}\psi) \wedge \mathsf{A}\,\mathsf{F}_{>c}\mathtt{true}$;
- $(q,x),\mathcal{A} \models \mathsf{E}\,\mathsf{G}_{>c}\mathtt{false}$ iff $(q,x),\mathcal{A} \models \mathsf{E}\,\mathsf{G}_{\geq c}\mathtt{false} \vee \mathsf{E}\,\mathsf{F}_{\leq c}\mathsf{E}\,\mathsf{G}(\mathsf{can\_have\_not\_paid})$;
- $(q,x),\mathcal{A} \models \mathsf{A}\,\varphi\mathsf{U}_{\leq c}\psi$ iff $(q,x),\mathcal{A} \models \mathsf{A}\,\varphi\mathsf{U}\psi \wedge \mathsf{A}\,\mathsf{F}_{\leq c}\psi$;
- $(q,x),\mathcal{A} \models \mathsf{E}\,\mathsf{G}_{\leq c}\psi$ iff $(q,x),\mathcal{A}_{\text{ext}} \models \mathsf{E}\,\mathsf{G}\psi \vee \mathsf{E}\,\psi\mathsf{U}_{>c}\mathtt{true}$;
- $(q,x),\mathcal{A} \models \mathsf{A}\,\varphi\mathsf{U}_{<c}\psi$ iff $(q,x),\mathcal{A} \models \mathsf{A}\,\varphi\mathsf{U}\psi \wedge \mathsf{A}\,\mathsf{F}_{<c}\psi$;
- $(q,x),\mathcal{A} \models \mathsf{E}\,\mathsf{G}_{<c}\psi$ iff $(q,x),\mathcal{A}_{\text{ext}} \models \mathsf{E}\,\mathsf{G}\psi \vee \mathsf{E}\,\psi\mathsf{U}_{\geq c}\mathtt{true}$;
- $(q,x),\mathcal{A} \models \mathsf{A}\,\varphi\mathsf{U}_{=c}\psi$ iff $(q,x),\mathcal{A} \models \mathsf{A}\,\varphi\mathsf{U}_{\geq c}\psi \wedge \mathsf{A}\,\mathsf{F}_{=c}\psi$;
- $(q,x),\mathcal{A} \models \mathsf{E}\,\mathsf{G}_{=c}\psi$ iff
  $(q,x),\mathcal{A}_{\text{ext}} \models (\mathsf{E}\,\mathsf{G}_{=c}\mathtt{false}) \vee (\mathsf{E}\,\mathsf{F}_{=c}(\mathsf{has\_paid} \wedge \psi \wedge (\mathsf{E}\,\mathsf{G}\psi \vee \mathsf{E}\,\psi\mathsf{U}\mathsf{has\_paid})))$;

Those transformations (which do not increase $\hbar(\Phi)$) are sufficient to lift the result to all the modalities of WCTL (under the assumption that we have no discrete costs).

2.2.2. *We now explain how we can prove the induction step of Proposition 2.4 for a formula* $\Phi = \mathsf{E}\,\varphi\mathsf{U}_{\sim c}\psi$ *when the automaton has discrete costs on transitions.* We will simplify the problem and reduce it to the computation of states satisfying a formula in an automaton without discrete costs. Then, applying the result proved for the automata without discrete costs, we will get the induction step. We note T the set of transitions of $\mathcal{A}$ that have a positive discrete cost. We unfold the automaton as follows: there is a copy of $\mathcal{A}$ for every integer smaller than or equal to $c+1$. Copy of location q in the i-th copy is denoted $q_{(i)}$. There is a transition from $q_{(i)}$ to $q'_{(j)}$ if: either $i=j$ and there is a transition in $\mathcal{A}$ from q to q' not in T; or $j = i+k \leq c+1$ and there is a transition in T with discrete cost k from q to q'; or $j = p+1$, $i+k > c+1$ and there is a transition in T with discrete cost k from q to q'. We note $\mathcal{A}_{\text{unf}}$ this unfolding. Then,

$$(q,x),\mathcal{A} \models \mathsf{E}\,\varphi\mathsf{U}_{\sim c}\psi \quad \text{iff} \quad (q_{(0)},x),\mathcal{A}_{\text{unf}} \models \bigvee_{i\leq p+1} \mathsf{E}\,\varphi\mathsf{U}_{\sim c-i}(\psi \wedge \mathsf{copy}_i)$$

where copy_i is an atomic proposition labeling all locations of $\mathcal{A}_i$. The correctness of this construction is obvious. Now, applying the induction hypothesis on automata with no discrete cost on transitions, the granularity of regions required for model-checking each formula is $1/C^{\max(\hbar(\varphi),\hbar(\psi))+1}$, the granularity for the original formula in $\mathcal{A}$ is thus also $1/C^{\max(\hbar(\varphi),\hbar(\psi))+1} = 1/C^{\hbar(\Phi)}$, which proves the induction step also for automata with discrete costs on transitions.

Finally, this extension to automata with discrete costs can be adapted to modalities of the form $\mathsf{A}\,\mathsf{U}$. We omit the tedious details. □

Remark 2.9. In the above proof, we have exhibited exponentially many constants a_i's at which truth of the formula can change. We will show here that the exponential number of constants is unavoidable in general. Indeed, consider the one-clock PTA $\mathcal{A}$ displayed on Figure 5. Using a WCTL formula, we will require that the cost is exactly 4 between a

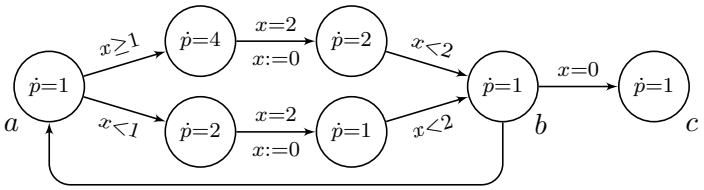


Figure 5: The one-clock PTA $\mathcal{A}$

and b. That way, if clock x equals $x_0.x_1x_2x_3\ldots x_n\ldots$ (this is the binary representation of a real in the interval $(0,2)$) when leaving a, then it will be equal to $x_1.x_2x_3\ldots x_n\ldots$ in b. We consider the WCTL formula $\varphi(X) = \mathsf{E}\Big((a \vee b)\mathsf{U}_{=0}(\neg a \wedge \mathsf{E}\,(\neg b\mathsf{U}_{=4}(b \wedge X)))\Big)$, where X is a formula we will specify. Then formula $\varphi(\mathsf{E}\,\mathsf{F}_{=0}c)$ states that we can go from a to b with cost 4, and that $x = 0$ when arriving in b (since we can fire the transition leading to c). From the remark above, this can only be true if $x = 0$ or $x = 1$ in a. Now, consider formula $\varphi(\mathsf{E}\,\mathsf{F}_{=0}c \vee \varphi(\mathsf{E}\,\mathsf{F}_{=0}c))$. If it holds in state a, then state c can be reached after exactly one or two rounds in the automaton, *i.e.*, if the value of x is in $\{0, 1/2, 1, 3/2\}$. Clearly enough, nesting φ n times characterizes values of the clocks of the form $p/2^{n-1}$ where p is an integer strictly less than 2^n.

2.3. **Algorithms and Complexity.** In this section, we provide two algorithms for model-checking WCTL on one-clock PTA. The first algorithm runs in EXPTIME, whereas the second one runs in PSPACE, thus matching the PSPACE lower bound. However, it is easier to first explain the first algorithm, and then reuse part of it in the second algorithm. Finally, we will pursue the example of Subsection 1.2 for illustrating our PSPACE algorithm.

2.3.1. *An EXPTIME Algorithm.* The correctness of the algorithm we propose for model-checking one-clock PTA against WCTL properties relies on the properties we have proved in the previous section: if $\mathcal{A}$ is an automaton with maximal constant M, writing C for the l.c.m. of all costs labeling a location, and if Φ is a WCTL formula of constrained size n (the maximal number of nested constrained modalities), then the satisfaction of Φ is uniform on the regions $(m/C^n; (m+1)/C^n)$ with $m < M \cdot C^n$, and also on $(M; +\infty)$. The idea is thus to test the satisfaction of Φ for each state of the form $(q, k/2C^n)$ for $0 \leq k \leq (M \cdot 2C^n) + 1$ (*i.e.* at the bounds and in the middle of each region).

To check the truth of $\Phi = \mathsf{E}\,\varphi\mathsf{U}_{\mathsf{cost}\sim c}\psi$ in state (q, x) with $x = k/2C^n$, we will non-deterministically guess a witness. Using graph G that we have defined in Section 2.2, we begin with proving a "small witness property":

Lemma 2.10. *Let s be the smallest positive cost in $\mathcal{A}$, and C be the lcm of all positive costs of $\mathcal{A}$. Let q be a location of $\mathcal{A}$, and $x \in \mathbb{R}^+$. Let $\Phi = \mathsf{E}\,\varphi\mathsf{U}_{\sim c}\psi$ be a WCTL formula of size n. Then $(q, x) \models \Phi$ iff there exists a run in $\mathcal{A}$, from (q, x) and satisfying $\varphi\mathsf{U}_{\sim c}\psi$, and whose projection in G visits at most $N = \lfloor c \cdot C^n/s \rfloor + 2$ times each state of G.*

Proof. Let τ be a run in $\mathcal{A}$, starting from (q, x) (with $x = k/2C^n$ for some k) and satisfying $\varphi\mathsf{U}_{\sim c}\psi$. To that run corresponds a path ϱ in the region graph, starting in (q, x). Consider a cycle in that path ϱ: either it has a global cost interval $[0, 0]$, in which case it can be removed and still yields a witnessing run; or it has a global cost interval of the form $\langle a, b\rangle$ with $b > 0$. In that case, letting s be the smallest positive cost of the automaton, we know

that $b \geq s/C^n$. Now, if some state of G is visited (strictly) more than $N = \lfloor c \cdot C^n/s \rfloor + 2$ times along ϱ, we build a path ϱ' from ϱ by removing extraneous cycles, in such a way that each state of G is visited at most N times along ϱ (and that ϱ starts and ends in the same states). Since we assumed that ϱ does not contain cycles with cost interval $[0;0]$, we know that the upper bound of the accumulated cost along ϱ' is above c. Also, the lower bound of the accumulated costs along ϱ' is less than that of ϱ. Since ϱ "contains" a run witnessing $\varphi \mathsf{U}_{\sim c} \psi$, the cost interval of ϱ contains a value satisfying $\sim c$, thus so does the cost interval of ϱ'. In other words, ϱ' still contains a path witnessing $\varphi \mathsf{U}_{\sim c} \psi$. This path can easily be lifted to a run in $\mathcal{A}$ satisfying the formula $\varphi \mathsf{U}_{\sim c} \psi$. □

Since a transition in G may correspond to a linear sequence of transitions in $\mathcal{A}$, we know that if $(q, x) \models \mathsf{E}\, \varphi \mathsf{U}_{\sim c} \psi$, then there exists a witness having at most exponentially many transitions in $\mathcal{A}$.

We now describe our algorithm: assuming we have computed, for each state q of $\mathcal{A}$, the intervals of values of x where φ (resp. ψ) holds, we non-deterministically guess the successive states of a path in $\mathcal{A}$, checking that φ holds after each action transition and that the path reaches a ψ-state after an action transition and with cost satisfying $\sim c$. This verification can be achieved in PSPACE (and can be made deterministic as PSPACE = NPSPACE). Since we apply this algorithm for each state $(q, k/2C^n)$ with $0 \leq k \leq (M \cdot 2C^n) + 1$, our global algorithm runs in deterministic exponential time.

It is immediate to design a similar algorithm for formulas $\mathsf{E}\, \mathsf{G}_{\geq c} \mathsf{false}$ and $\mathsf{E}\, \mathsf{G}_{=c} \mathsf{false}$. The other existential modalities are handled by reducing to those case, as explained in Section 2.2.

2.3.2. *A PSPACE Algorithm.* The PSPACE algorithm will reuse some parts of the previous algorithm, but it will improve on space performance by computing and storing only the minimal information required: instead of computing the truth value of each subformula in each state $(q, k/2C^n)$, it will only compute the information it really needs. Our method is thus similar in spirit to the space-efficient, on-the-fly algorithm for TCTL presented in [HKV96].

We will then need, while guessing a witness for $\mathsf{E}\, \varphi \mathsf{U}_{\mathsf{cost} \sim c} \psi$, to check that all intermediary states reached after an action transition satisfy formula φ. As φ might be itself a WCTL formula with several nested modalities, we will fork a new computation of our algorithm on formula φ from each intermediary state. The maximal number of threads running simultaneaously is at most the depth of the parsing tree of formula Φ. When a thread is preempted we only need to store a polynomial amount of information in order to be able to resume it. Indeed, it is sufficient to store for each preempted thread a triple (α, K, I) where α is a node of the region graph, K records the number of steps of the path we are guessing (we know that when $\mathsf{E}\, \varphi \mathsf{U}_{\sim c} \psi$ holds, an exponential witness exists), and I is an interval corresponding to the accumulated cost along the path being guessed.

The algorithm thus runs as follows: we start by labeling the root of the tree by $\alpha = (q, x, r)$, $K = 0$ and $I = [0;0]$. Then we guess a sequence of transitions in the region graph, starting from (q, x, r); when a new state (q', r') is added, we increment the value of K and update the value of the interval, as described in the previous section. If we just fired an action transition, then either we fork an execution for checking that φ holds, or we check that the constraint $\mathsf{cost} \sim c$ can be satisfied by the new interval and we verify that the new state satisfies ψ (by again forking a new execution).

The number of nested guesses can be bounded by the depth of the parsing tree of Φ, because when a new thread starts, it starts from a node in the parsing tree that is a child of the previous node. Thus, the memory needed in this algorithm is the parsing tree of formula Φ with each node labeled by a tuple which can be stored in polynomial space. This globally leads to a PSPACE algorithm.

Example 2.11. We illustrate our PSPACE algorithm on our initial example, with formula $\Phi = \neg \mathsf{E}\,(\mathsf{OK}\ \mathsf{U}_{t\leq 8}(\mathsf{Problem} \wedge \neg \mathsf{E}\,\mathsf{F}_{c<30}\mathsf{OK}))$. We write $g = 1/C^2$ for the resulting granularity as defined in Prop. 2.4, and consider a starting state, e.g. $(\mathsf{OK}, x = mg)$.

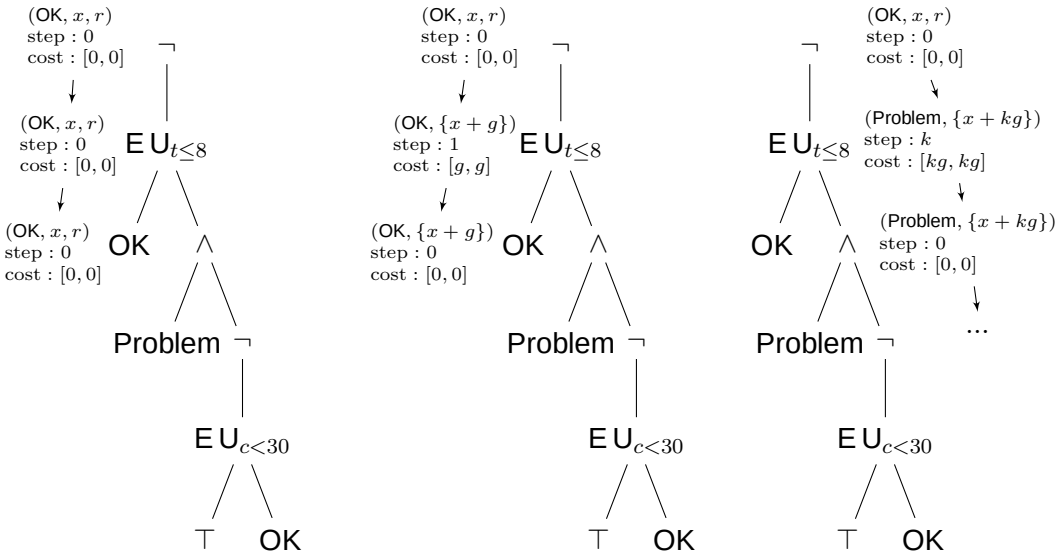


Figure 6: Execution of our PSPACE algorithm on the initial example.

Figure 6 shows three steps of our algorithm. The first step represents the first iteration, where subformula OK is satisfied at the beginning of the run. At step 2, the execution goes to $(\mathsf{OK}, x + g)$: we check that the left-hand-side formula still holds in $(\mathsf{OK}, x + g)$ (as depicted), but also in intermediary states. The third figure corresponds to k steps later, when the algorithm decides to go to the right-hand-part of $\mathsf{E}\,\mathsf{U}_{t\leq 8}$. In that case, of course, it is checked that $kg \leq 8$, and then goes on verifying the second until subformula.

3. Model-checking linear-time logics

We now turn to the case of linear-time temporal logics. We begin with the definition of our logic WMTL.

3.1. 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".

The syntax of WMTL is defined inductively as follows:

$$\text{WMTL} \ni \varphi \ ::= \ a \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \mathsf{U}_{\mathsf{cost}\sim c}\varphi$$

where $a \in \mathsf{AP}$, cost is a cost function, c ranges over $\mathbb{N}$, $\sim \in \{<, \leq, =, \geq, >\}$. If there is a single cost function or if the cost function cost is clear from the context, we simply write $\varphi \mathsf{U}_{\sim c} \psi$ for $\varphi \mathsf{U}_{\mathsf{cost} \sim c} \psi$.

We interpret WMTL formulas over (finite) runs of labeled PTA, identifying each cost of the formula with the corresponding cost in the automaton.

Definition 3.1. Let $\mathcal{A}$ be a labeled PTA, and let $\varrho = (q_0, v_0) \xrightarrow{\tau_1, e_1} (q_1, v_1) \cdots \xrightarrow{e_p, \tau_p} (q_p, v_p)$ be a finite run in $\mathcal{A}$. The satisfaction relation for WMTL is then defined inductively as follows:

$$\begin{aligned}
\varrho \models a &\Leftrightarrow a \in \ell(q_0) \\
\varrho \models \neg\varphi &\Leftrightarrow \varrho \not\models \varphi \\
\varrho \models \varphi_1 \vee \varphi_2 &\Leftrightarrow \varrho \models \varphi_1 \text{ or } \varrho \models \varphi_2 \\
\varrho \models \varphi_1 \mathsf{U}_{\mathsf{cost} \sim c} \varphi_2 &\Leftrightarrow \exists 0 < \pi \leq |\varrho| \text{ s.t. } \varrho_{\geq \pi} \models \varphi_2,\ \forall 0 < \pi' < \pi,\ \varrho_{\geq \pi'} \models \varphi_1, \\
&\qquad \text{and } \mathsf{cost}(\varrho_{\leq \pi}) \sim c.
\end{aligned}$$

Example 3.2. Back on our example of Figure 1, we can express that there is no path from OK back to itself in time less than 10 *and* cost less than 20. This is achieved by showing that no path satisfies the following formula:

$$\mathsf{OK}\, \mathsf{U}\, (\mathsf{Problem} \wedge (\neg \mathsf{OK})\, \mathsf{U}_{x \leq 10}\, \mathsf{OK} \wedge (\neg \mathsf{OK})\, \mathsf{U}_{c \leq 20}\, \mathsf{OK}).$$

As we will see, model-checking WMTL will in fact be undecidable when the automaton involves more than one cost.

Remark 3.3. 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], whereas the model-checking under the point-based semantics is decidable over finite runs [OW05].

We study existential model-checking of WMTL over priced timed automata, stated as: given a one-clock PTA $\mathcal{A}$ and a WMTL formula φ, 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 PTA 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 PTA with one cost variable,
(3) two-clock PTA with one stopwatch cost variable,
(4) one-clock PTA with two stopwatch cost variables.

We present our results as follows. In Section 3.2, 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 3.3, 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).

3.2. **Decidability of WMTL for One-Clock PTA With One Stopwatch Cost.**

Theorem 3.4. *Model checking one-clock PTA 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 do not give all details, especially when there is nothing new compared with the above-mentioned paper.

Let φ be a WMTL formula, and $\mathcal{A}$ be a single-clock PTA with a stopwatch cost. Classically, from formula φ, we construct an "equivalent" one-variable alternating timed automaton[5] $\mathcal{B}_\varphi$. Figure 7 displays an example of such an automaton, corresponding to formula $\mathsf{G}[a \Rightarrow (\mathsf{F}_{\leq 3} b \vee \mathsf{F}_{\geq 2} c)]$ (see [OW05] for more details on alternating timed automata).

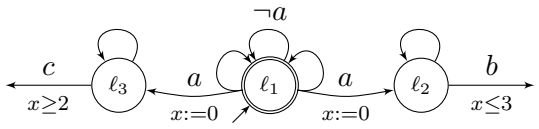


Figure 7: A timed alternating automaton for formula $\mathsf{G}[a \Rightarrow (\mathsf{F}_{\leq 3} b \vee \mathsf{F}_{\geq 2} c)]$

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 run 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 $(\ell, \mathit{int}(c))$ [6] 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 $(\alpha', \beta') \in \gamma_j$, $\mathit{frac}(\beta) \leq \mathit{frac}(\beta')$ [7] 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

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

[6] *int* represents the integral part.

[7] *frac* represents the fractional part.

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)

$$\mathsf{reg}(\gamma_0)\mathsf{reg}(\gamma_1)\dots\mathsf{reg}(\gamma_p)\mathsf{reg}(\gamma_\top)$$

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 3.5. Consider the configuration

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

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 3.6. *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 $\gamma_1 \to \gamma_1'$ and $\gamma_1 \equiv \gamma_2$. 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 3.7. *$W \Rightarrow^* W'$ 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 3.8. *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 3.8, 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 3.9. 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 3.6 does not hold for two-clock PTA, 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[8] 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.

Remark 3.10. Let $\mathcal{A}$ be a PTA with a stopwatch cost. From the construction using encodings by words we have presented above, we see that truth of WMTL formulas is invariant by classical regions (by classical regions, we mean one-dimensional regions with granularity 1): indeed, in the above construction, it suffices to change the initial configuration with the encoding of the region we want to start from, and applying the previous results, we immediately get that the truth of the formula will then not depend on the precise initial value of the clock. As a consequence, the model checking of WCTL* [9] is decidable (and non-primitive recursive) for PTA with a single stopwatch cost: it suffices to label regions (in the classical sense) with the WMTL subformulas they satisfy. Let us mention right now that the undecidability results below directly extend to WCTL*, so that again, any extension of the model leads to undecidability.

3.3. **Undecidability Results.** In this part, we prove that the above result is tight, in the sense that adding an extra stopwatch cost ot removing the "stopwatch" condition yields undecidability.

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

[9] WCTL* is the extension of CTL* [CES86] with cost constraints. We omit its definition.

3.3.1. *One-Clock PTA With One Cost Variable.*

Theorem 3.11. *Model checking one-clock PTA with one (general) cost 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:

$$\mathtt{p_i}: \ \mathtt{c} := \mathtt{c}+1; \ \mathtt{goto}\ \mathtt{p_j}. \tag{3.1}$$

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:

$$\mathtt{p_i}: \ \mathtt{if}\ (\mathtt{c}==0)\ \mathtt{then\ goto}\ \mathtt{p_j}\ \mathtt{else}\ \mathtt{c} := \mathtt{c}-1; \ \mathtt{goto}\ \mathtt{p_k}. \tag{3.2}$$

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

Module for instruction (3.1). Consider instruction (3.1), 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 Figure 8.[10]

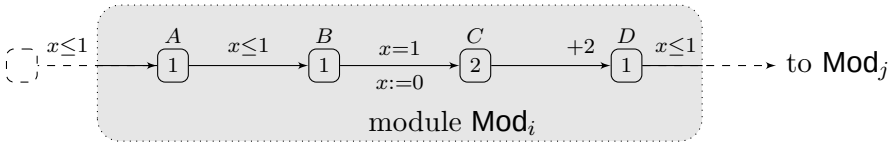


Figure 8: Module for incrementing c_1

The following lemma is then easy to prove:

Lemma 3.12. *Assume that there is a run ϱ entering module Mod_i with $x = x_0 \leq 1$, 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 (3.2). 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 Figure 9.

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

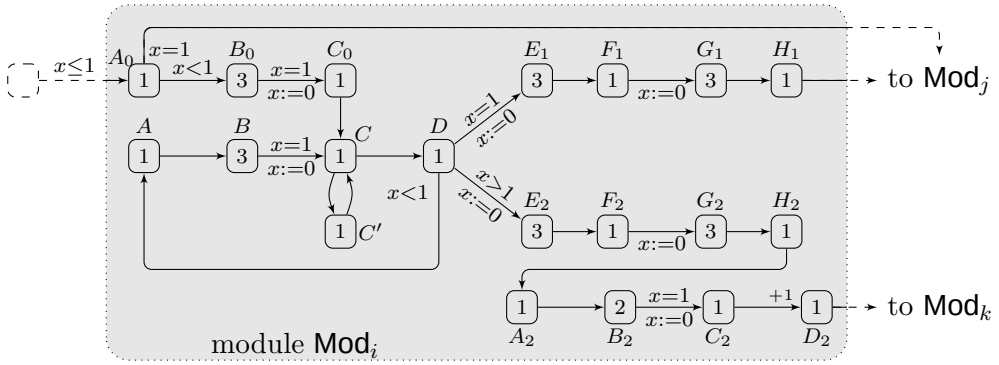


Figure 9: Module testing/decrementing c_1

Lemma 3.13. *Assume there exists a run ϱ entering module Mod_i with $x = x_0 \leq 1$, exiting to module Mod_j with $x = x_1$, and such that*

- *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 a run. 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[11] (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':

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

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.

[11] 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.

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 run going to module Mod_k:

Lemma 3.14. *Assume there exists a run 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 the last visit to 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 $\mathsf{zero}(P) = \mathsf{G}(P \Rightarrow (P \mathsf{U}_{=0} \neg P))$. 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 other conditions of Lemmas 3.12, 3.13 and 3.14. We write φ_2 for the corresponding

formula.. For instance, the conditions of Lemmas 3.13 and 3.14 would be expressed as follows[12]:

$$\mathsf{G}\left[A_0 \wedge \mathsf{Mod}_{\text{decr}} \Rightarrow \left\{ \begin{array}{l} \left(\begin{array}{l} (A \vee A_0) \Rightarrow (\neg D \mathsf{U}_{=3} D) \wedge \\ (C_0 \vee C') \Rightarrow (\neg(C' \vee F_1)\mathsf{U}_{=3}(C' \vee F_1)) \wedge \\ (D \wedge \neg D \mathsf{U} H_1) \Rightarrow (\neg H_1 \mathsf{U}_{=3} H_1) \end{array} \right) \mathsf{U} H_1 \\ \qquad \bigvee \\ \left(\begin{array}{l} (A \vee A_0) \Rightarrow (\neg D \mathsf{U}_{=3} D) \wedge \\ (C_0 \vee C') \Rightarrow (\neg(C' \vee F_2)\mathsf{U}_{=3}(C' \vee F_2)) \wedge \\ (D \wedge \neg D \mathsf{U} H_2) \Rightarrow (\neg H_2 \mathsf{U}_{=3} H_2) \wedge \\ H2 \Rightarrow (\neg D_2 \mathsf{U}_{=3} D_2) \end{array} \right) \mathsf{U} H_2 \end{array} \right\} \right]$$

The following proposition is now straightforward:

Proposition 3.15. *The machine $\mathcal{M}$ halts iff there exists a run in $\mathcal{A}_{\mathcal{M}}$ satisfying $\varphi_1 \wedge \varphi_2 \wedge$ FHalt.*

Remark 3.16. • 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.

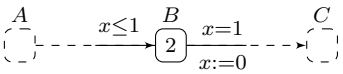
- Our reduction uses a $\{1,2,3\}$-sloped cost variable, but it could be achieved with any $\{p,q,r\}$-sloped cost variable (with $0 < p < q < r$, and p, q and r are pairwise coprime) by encoding the values of the counters by the clock value $(p/q)^{c_1} \cdot (p/r)^{c_2}$.
- Our WMTL formula can easily 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 $(\neg p)\mathsf{U}_{=n} p$ with $(\neg p)\mathsf{U}_{\leq n} p \wedge (\neg p)\mathsf{U}_{\geq n} p$.

3.3.2. *Two-Clock PTA with One Stopwatch-Cost Variable.* While this case does not fit in our "one-clock" setting, it is an interesting intermediate step between the previous and the next results.

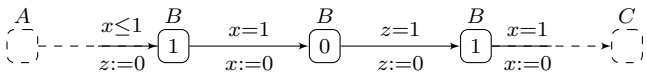
Theorem 3.17. *Model checking two-clock PTA with one stopwatch cost against WMTL properties is undecidable.*

Proof. 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 $x = 1$:



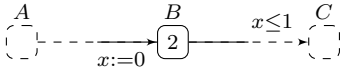
These states are replaced by the following submodule:



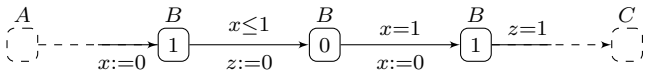
[12]The atomic proposition $\mathsf{Mod}_{\text{decr}}$ is used to indicate that we are in a module decrementing one of the counters. It implicitly labels all the states of such modules.

A simple computation shows that both sequences have the same effect on clock x and induce the same cost. Of course, the case of cost 3 is handled by adding one more pair of states with costs 0 and 1.

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



Those are replace with a slightly different sequence of states:



Again, one is easily convinced that both sequences are "equivalent", and that this transformation adapts to states with cost 3. □

3.3.3. *One-Clock PTA 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.

Figure 10: Replacing a clock with an extra "clock cost"

Figure 10 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:

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

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. Thus, we get immediately the following result:

Theorem 3.18. *Model checking one-clock PTA with two stopwatch-cost variables against WMTL properties is undecidable.*

4. Conclusion

In this paper, we have studied various model-checking problems for one-clock priced timed automata. We have proved that the model-checking of one-clock priced timed automata against WCTL properties is PSPACE-complete. This is rather surprising as model-checking TCTL over one-clock timed automata has the same complexity, though it allows much less features. For proving this result, we have exhibited a sufficient granularity such that truth of formulas over regions defined with this granularity is uniform. Based on this

result, we developed a space-efficient algorithm which computes satisfaction of subformulas on-the-fly. This result has to be contrasted with the undecidability result of [BBM06] which establishes that model-checking priced timed automata with three clocks and more against WCTL properties is undecidable.

We have also depicted the precise decidability border for WMTL model-checking, a cost-constrained extension of LTL. We have proved that the restriction to single-clock single-stopwatch cost variable leads to decidability, and that any single extension leads to undecidability.

There are several natural research directions: the decidability of WCTL model-checking for two-clocks priced timed automata is not known, we just know that these models have an infinite bisimulation [BBR04]; another interesting extension is multi-constrained modalities, e.g. $\mathsf{E}\,\varphi\mathsf{U}_{\mathsf{cost}_1\leq 5,\mathsf{cost}_2>3}\varphi$?

References

[AAM06] Yasmina Abdeddaïm, Eugene Asarin, and Oded Maler. Scheduling with timed automata. *Theoretical Computer Science*, 354(2):272–300, 2006.

[ABM04] Rajeev Alur, Mikhail Bernadsky, and P. Madhusudan. Optimal reachability in weighted timed games. In *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *Lecture Notes in Computer Science*, pages 122–133. Springer, 2004.

[ACD93] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[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.

[AL88] Martin Abadi and Leslie Lamport. The existence of refinement mappings. In *Proc. of the 3rd Annual IEEE Symp. on Logic In Computer Science (LICS'88)*, pages 165–175. IEEE Computer Society Press, 1988.

[ALP01] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In *Proc. 4th International Workshop Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, 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 on weighted timed automata. *Formal Methods in System Design*, 31(2):135–175, October 2007.

[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 *Lecture Notes in Computer Science*, pages 203–218. Springer, 2004.

[BBL08] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 32(1):2–23, February 2008.

[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 Conf. Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, volume 3253 of *Lecture Notes in Computer Science*, 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 *Lecture Notes in Computer Science*, pages 49–64. Springer, 2005.

[BCFL04] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Optimal strategies in priced timed game automata. In *Proc. 24th Conf. Found. Softw. Tech. & Theor. Comp. Science (FST&TCS'04)*, volume 3328 of *Lecture Notes in Computer Science*, pages 148–160. Springer, 2004.

[BES93] Ahmed Bouajjani, Rachid Echahed, and Joseph Sifakis. On model checking for real-time properties with durations. In *Proc. 8th Annual Symposium on Logic in Computer Science (LICS'93)*. IEEE Computer Society Press, 1993.

[BFH+01a] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, and Judi Romijn. Efficient guiding towards cost-optimality in UPPAAL. In *Proc. 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, volume 2031 of *Lecture Notes in Computer Science*, pages 174–188, 2001.

[BFH+01b] 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 *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

[BLM07] Patricia Bouyer, Kim G. Larsen, and Nicolas Markey. Model-checking one-clock priced timed automata. In *Proceedings of the 10th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'07)*, volume 4423 of *Lecture Notes in Computer Science*, pages 108–122. Springer, March 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 Conf. Found. Softw. Tech. & Theor. Comp. Science (FST&TCS'06)*, volume 4337 of *Lecture Notes in Computer Science*, pages 346–357. Springer, 2006.

[BLR05a] Gerd Behrmann, Kim G. Larsen, and Jacob I. Rasmussen. Optimal scheduling using priced timed automata. *ACM SIGMETRICS Performance Evaluation Review*, 32(4):34–40, 2005.

[BLR05b] Gerd Behrmann, Kim G. Larsen, and Jacob I. Rasmussen. Priced timed automata: Algorithms and applications. In *Revised Lectures 3rd International Symposium on Formal Methods for Components and Objects (FMCO'04)*, volume 3657 of *Lecture Notes in Computer Science*, pages 162–182. Springer, 2005.

[BM07] Patricia Bouyer and Nicolas Markey. Costs are expensive! In *Proceedings of the 5th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'07)*, volume 4763 of *Lecture Notes in Computer Science*, pages 53–68. Springer, October 2007.

[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)*, pages 109–118. IEEE Computer Society Press, 2007.

[CES86] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.

[HJ94] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[HKV96] Thomas A. Henzinger, Orna Kupferman, and Moshe Y. Vardi. A space-efficient on-the-fly algorithm for real-time model checking. In *Proc. 7th Intl. Conf. Concurrency Theory (CONCUR'96)*, volume 1119 of *Lecture Notes in Computer Science*, pages 514–529. Springer, 1996.

[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 I. Rassmussen. Optimal conditional reachability for multi-priced timed automata. In *Proc. 8th Internationl Conference on Foundations of Software Science and*

Computation Structures (FoSSaCS'05), volume 3441 of *Lecture Notes in Computer Science*, 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.

[LW08] Slawomir Lasota and Igor Walukiewicz. Alternating timed automata. *ACM Transactions on Computational Logic*, 9(2), March 2008.

[OG76] Susan Owicki and David Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, 6(4):319–340, August 1976.

[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.

[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), 2007.

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

[RLS04] Jacob I. Rasmussen, Kim G. Larsen, and K. Subramani. Resource-optimal scheduling using priced timed automata. In *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, volume 2988 of *Lecture Notes in Computer Science*, pages 220–235. Springer, 2004.