Lower-Bound Constrained Runs in Weighted Timed Automata

Patricia Bouyer
LSV – ENS Cachan & CNRS
Email: bouyer@lsv.ens-cachan.fr

Kim G. Larsen
Dept. Computer Science – Aalborg U.
Email:kgl@cs.aau.dk

Nicolas Markey
LSV – ENS Cachan & CNRS
Email: markey@lsv.ens-cachan.fr

***Abstract*—We investigate a number of problems related to infinite runs of weighted timed automata, subject to lower-bound constraints on the accumulated weight. Closing an open problem from [10], we show that the *existence* of an infinite lower-bound-constrained run is—for us somewhat unexpectedly—undecidable for weighted timed automata with four or more clocks.**

This undecidability result assumes a fixed and known initial credit. We show that the related problem of *existence of an initial credit* for which there exists a feasible run is decidable in PSPACE. We also investigate the variant of these problems where only bounded-duration runs are considered, showing that this restriction makes our original problem decidable in NEXPTIME. Finally, we prove that the *universal* versions of all those problems (i.e, checking that *all* the considered runs satisfy the lower-bound constraint) are decidable in PSPACE.

I. Introduction

Weighted (or priced) timed automata [3], [2], [9] have emerged as a useful formalism for formulating a wide range of resource-allocation and optimization problems [4], [14], with applications in areas such as embedded systems [18]. In [10], a new class of resource-allocation problems was introduced, namely that of constructing infinite schedules subject to boundary constraints on the accumulation of resources.

More specifically, we proposed weighted timed automata with positive as well as negative weight-rates in locations, allowing for the modeling of systems where resources (e.g. energy) are not only consumed but also possibly produced. As a basic example, consider the two-clock weighted timed automaton $\mathcal{A}$ in Fig. 1 with infinite behaviours repeatedly delaying in ℓ_0, ℓ_1, ℓ_2 and ℓ_3 for a total duration of two time units, with one time unit spent in ℓ_0 and ℓ_3 and one time unit spent in ℓ_1 and ℓ_2. The weights (+2, +3 and +4) in the four locations indicate the rate by which energy is produced (or consumed, when negative), and the weights (-2 and -3) on the edges indicate instantaneous updates to the energy level. Clearly, the energy remaining after a given iteration will depend not only on the initial energy but also highly on the distribution of the two time units over the four locations.

In this paper we consider a number of problems related to infinite runs subject to lower-bound constraints on the accumulated weight, i.e., infinite runs where the energy level never goes below zero. In the absence of an upper bound, it suffices to consider runs along which the accumulated weight is maximized. Fig. 2 illustrates three such energy-maximizing behaviours of $\mathcal{A}$. For initial energy 8, the maximum energy left

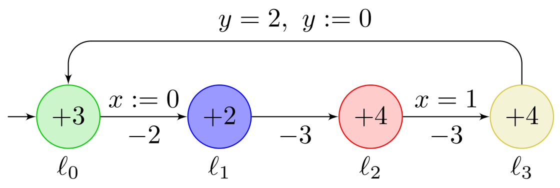


Fig. 1. A weighted timed automaton $\mathcal{A}$ (with implicit global invariant $y \leq 2$)

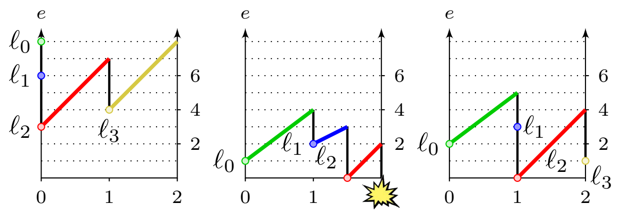


Fig. 2. Three possible behaviours in $\mathcal{A}$ (with initial credits 8, 1 and 2, resp.)

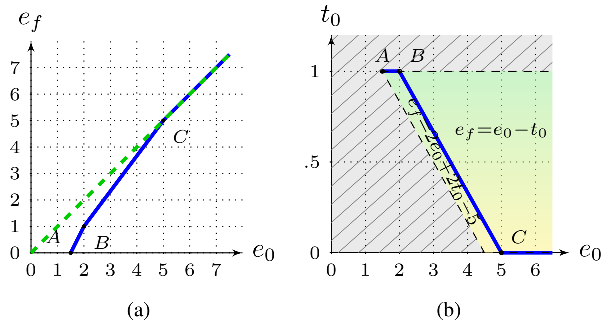


Fig. 3. Maximum final energy in ℓ_3 and optimal delay t_0, depending on initial energy e_0

after one iteration is 8, thus providing an infinite lower-bound schedule. In contrast, an initial energy-level of 1 does not even permit a single iteration (let alone an infinite schedule), and an initial energy-level 2 leaves at maximum 1, for which we already know that no infinite lower-bound schedule exists. In fact, from Fig. 3(a)—which gives the maximum final energy e_f as a function of the initial energy-level e_0—it follows that 5 is the minimum initial energy-level which permits a lower-bounded infinite schedule (notice that the optimal schedule with initial credit 8 can be played with initial credit 5). Moreover, Fig. 3(b) indicates optimal schedules (the solid line) in terms of the optimal delay t_0 to be spent in ℓ_0 given the initial energy-level e_0. The dashed area corresponds to pairs (e_0, t_0) for which no feasible run exists. Inside the feasible area, the optimal final credit is $e_f = e_0 - t_0$ on the right of the solid line, while it is $e_f = 2e_0 + 2t_0 - 5$ on the left.

For weighted timed automata with a single clock and a single weight variable, the existence of a lower-bound constrained infinite run has been shown decidable in polynomial time [10] with the restriction that no discrete updates of the accumulated weight occur on transitions. In [8], it is shown that the problem remains decidable if this restriction is lifted and even if the accumulated weight grows not only linearly but also exponentially. In contrast, the existence of *interval-constrained* infinite runs—where a simple energy-maximizing strategy does not suffice—have recently been proven undecidable for weighted timed automata with varying numbers of clocks and weight variables: *e.g.* two clocks and two weight variables [17], one clock and two weight variables [13], and two clocks and one weight variable [15]. Also, the interval-constrained problem is undecidable for weighted timed automata with one clock and one weight variable in the *game* setting [10].

Still, the general problem of *existence* of infinite lower-bound runs for weighted timed automata has remained unsettled since [10]. In this paper we close this open problem showing that it is undecidable for weighted timed automata with four or more clocks. Given that this problem looks rather simple (it suffices to consider energy-maximizing runs), we find this result quite surprising and somewhat disappointing. Thus, we consider a number of related problems for which we show decidability and settle complexity. In particular, the undecidability result assumes a fixed and known initial energy-level. We show that the related problem of *existence of an initial energy-level* allowing an infinite lower-bound constrained run is decidable in PSPACE. We also investigate the variant of these problems, where only the existence of *time-bounded* runs is required: for instance, for the weighted timed automaton in Fig. 1 and initial energy level of 4, we may want to settle the existence of a run along which the energy level remains non-negative during the first 4.7 time-units, say. Note that the time-bounded paradigm has recently emerged as a pertinent restriction option for the verification of real-time systems [16] (in quite the same way as bounded model checking has been used for untimed systems [6]). We show that this restriction makes our original problem decidable and NEXPTIME-complete. Our result has to be compared with rectangular hybrid automata, for which time-bounded reachability has recently been shown decidable in EXPSPACE (no matching lower bound is provided, though), under the hypothesis that all rates are non-negative (if rates can be negative, the problem is undecidable) [11]. Our model of weighted timed automata is a special case of rectangular hybrid automata, in which all variables are clocks (rate 1) and one variable can have non-negative as well as negative rates. Therefore none of the two decidability results implies the other. Finally, we prove that the *universal* versions of all the above problems (i.e., checking that *all* the considered runs satisfy the lower-bound constraint) are decidable in PSPACE.

All technical details can be found in the Appendix.

II. Definitions

A. Basic definitions

We assume X is a finite set of variables called clocks. A valuation v of the clocks is a mapping $X \to \mathbb{R}_{\geq 0}$. If v is a valuation and $t \in \mathbb{R}_{\geq 0}$, we write $v + t$ for the valuation which assigns $v(x) + t$ to clock x. If $R \subseteq X$, we write $v[R \to 0]$ for the valuation which assigns 0 to clocks in R and $v(x)$ to $x \in X \setminus R$. We write $\Phi(X)$ for the set of formulas (called clock constraints) defined by $\phi ::= \mathtt{true} \mid \phi \wedge \phi \mid x \leq c \mid x \geq c$ with $x \in X$ and $c \in \mathbb{N}$. The semantics of such formulas is given by sets of valuations and defined in a natural one.

A *weighted timed automaton* is a tuple $\mathcal{A} = \langle L, L_0, X, \mathsf{inv}, E, \mathsf{disc}, \mathsf{rate}\rangle$ consisting of a finite set L of locations, a finite set $L_0 \subseteq L$ of initial locations, a finite set X of clocks, a location invariant mapping $\mathsf{inv}\colon L \to \Phi(X)$, a finite set $E \subseteq L \times \Phi(X) \times 2^X \times L$ of edges, and functions $\mathsf{disc}\colon L \to \mathbb{Z}$ and $\mathsf{rate}\colon E \to \mathbb{Z}$ which assigns a value to each location and each edge of $\mathcal{A}$.

The semantics of a weighted timed automaton is defined as an infinite-state transition system $G_{\mathcal{A}} = \langle S, T\rangle$ with

- $S = \{(\ell, v, c) \in L \times (\mathbb{R}_{\geq 0})^X \times \mathbb{R} \mid v \models \mathsf{inv}(\ell)\}$,
- $T \subseteq S \times S$ contains two types of transitions:
  - delay transitions, which do not involve state change:
$$\{(\ell, v, c) \to (\ell, v', c') \mid \exists t \in \mathbb{R}_{\geq 0}.\ v' = v + t \text{ and } c' = c + \mathsf{rate}(\ell) \times t\}$$
  - action transitions:
$$\{(\ell, v, c) \to (\ell', v', c') \mid \exists e = (\ell, g, R, \ell') \in E.\ v \models g \text{ and } v' = v[R \to 0] \text{ and } c' = c + \mathsf{disc}(e)\}.$$

The above semantics is that of a timed automaton with an extra weight variable, which evolves with rate $\mathsf{rate}(\ell)$ in location ℓ and in a discrete manner (following the disc function) when firing transitions. Notice that if $(\ell, v, c) \to (\ell', v', c') \in T$, then for all γ, there exists γ' such that $(\ell, v, \gamma) \to (\ell', v', \gamma') \in T$. That is, the weight variable does not constrain the behaviour of the automaton. A (finite or infinite) run $\varrho = (\ell_0, v_0, c_0) \to (\ell_1, v_1, c_1) \to \ldots$ of $G_{\mathcal{A}}$ will be called a *weighted run*, and we will sometimes use the underlying standard *timed run* $(\ell_0, v_0) \to (\ell_1, v_1) \to \ldots$, which abstracts from the weight information. Since two consecutive delay transitions can be merged, we require that along any run, delay and action transitions alternate (by possibly inserting zero-delay transitions between two consecutive action transitions). We define the *length* of a run as its number of action transitions. If (ℓ_0, v_0, c_0) is the first state of run ϱ, c_0 is the *initial credit*; ϱ is said to be *initial* if $\ell_0 \in L_0$ and $v_0 = \mathbf{0}$, the valuation which assigns 0 to every clock. If ϱ is a timed run, then for every initial credit c_0 there is a unique corresponding weighted run with initial credit c_0.

B. The lower-bound constrained problems

1) The existential and universal* L*-problems. Fix an initial credit $c_0 \in \mathbb{Q}_{\geq 0}$. An infinite timed run $\varrho = (\ell_0, v_0) \to$

$(\ell_1, v_1) \to (\ell_2, v_2) \dots$ of $\mathcal{A}$ satisfies the lower-bound constraint $L(c_0)$ (which we write $\varrho \models L(c_0)$) if the corresponding weighted run $(\ell_0, v_0, c_0) \to (\ell_1, v_1, c_1) \to (\ell_2, v_2, c_2) \dots$ with initial credit c_0 has $c_i \geq 0$ for every i. In that case we say that the run is *feasible* with initial credit c_0 or simply that the corresponding weighted run is feasible.

We say that $\mathcal{A} \models \exists_\infty L(c_0)$ (resp. $\mathcal{A} \models \forall_\infty L(c_0)$) whenever there exists an initial infinite run ϱ s.t. $\varrho \models L(c_0)$ (resp. for every initial infinite run ϱ with initial credit c_0, it holds $\varrho \models L(c_0)$). The first (resp. second) problem is called the *existential (resp. universal)* L*-problem*, and denoted with $\exists_\infty L(c_0)$ (resp. $\forall_\infty L(c_0)$).

2) The time-bounded* L*-problems. Fix an initial credit $c_0 \in \mathbb{Q}_{\geq 0}$ and a time bound $T \in \mathbb{Q}_{\geq 0}$. A timed run $\varrho = (\ell_0, v_0) \to (\ell_1, v_1) \to (\ell_2, v_2) \dots$ satisfies the T-time-bounded lower-bound constraint $L(c_0)$ (which we write $\varrho \models_T L(c_0)$) whenever the following holds[1]:

- the total duration of ϱ is at most T, and can be strictly less than T only if ϱ is infinite. We let p_0 be the number of transitions along ϱ;
- if $(\ell_0, v_0, c_0) \to (\ell_1, v_1, c_1) \to (\ell_2, v_2, c_2) \dots$ is the weighted run corresponding to ϱ with initial credit c_0, then for every $i \leq p_0$, $c_i \geq 0$.

We then say that $\mathcal{A} \models \exists_T L(c_0)$ (resp. $\mathcal{A} \models \forall_T L(c_0)$) whenever there exists an initial run ϱ such that $\varrho \models_T L(c_0)$ (resp. for all initial finite run of duration T and for all initial infinite run of duration at most T, it holds $\varrho \models_T L(c_0)$). The first (resp. second) problem is called the *existential (resp. universal) time-bounded* L*-problem* . In short we write $\exists_T L(c_0)$ (resp. $\forall_T L(c_0)$).

3) Existence of an initial credit. The above four problems assume a fixed and known initial credit. We will be interested also in the existence (and synthesis) of an initial credit for which the previous problems can be answered positively. Formally, for $Q \in \{\exists, \forall\}$ and $\alpha \in \{\infty, T\}$, we write $\mathcal{A} \models \exists c_0 . Q_\alpha L(c_0)$ whenever there exists an initial credit $c_0 \in \mathbb{Q}_{\geq 0}$ such that $\mathcal{A} \models Q_\alpha L(c_0)$. In short we denote this problem by $\exists c.\ Q_\alpha L(c)$.

In this paper we solve the various above-mentioned problems. Our results are summarized in Table I (the previously known result is displayed in gray). Parameter Q is for existential or universal problem, whereas parameter α is for unbounded (∞) or bounded (T) time. In this table, constraints such as ≥ 3 refer to the number of clocks. When unspecified, we mean that the result holds for arbitrarily many clocks.

III. Undecidability of $\mathcal{A} \models \exists_\infty L(e_0)$

We first prove the undecidability of the existential L-problem. While its proof is not of the most difficult, the result is quite surprising (at least to us) as the problem looks rather simple (it amounts to checking that, by maximizing the accumulated weight, we can keep it non-negative).

[1]Various similar definitions could be considered instead of this one, for instance requiring that the duration is always (at least) T. Such variants could be handled by our techniques with minor amendments.

Theorem 1. *The existential* L*-problem is undecidable for the class of weighted timed automata with at least four clocks (and rates in* $\{0, 1\}$*).*

We only give a proof sketch here, which is simple to understand but requires five clocks. Another proof can be obtained by using the encoding of the hardness proof in Section IV-B.

Sketch of proof: Consider the automata depicted on Fig. 4. Writing x_0 for the value of clock x when entering ℓ_0, the

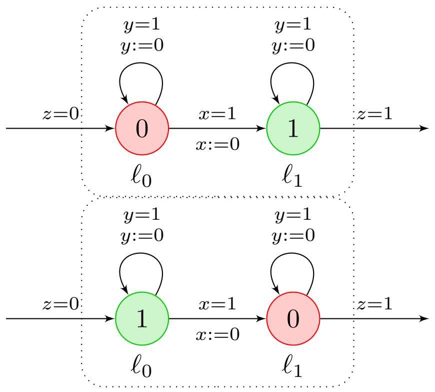


Fig. 4. Automata for crediting x_0 and $1 - x_0$ respectively

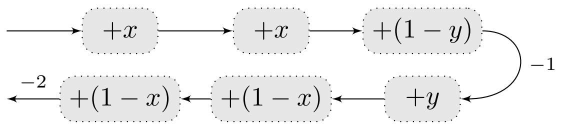


Fig. 5. Automaton for checking $y \leq 2x$

effect of these automata is to add x_0 (resp. $1 - x_0$) to the weight, while preserving the value of all the clocks (provided that these values are in $[0, 1)$). Using these modules, it is then easy to enforce linear constraints between clocks: Fig. 5 is an example, in which each box is a copy of one of the automata of Fig. 4 (after possibly exchanging the role of x and y). Discrete values between boxes represent a discrete increase or decrease of the weight variable. It is easily checked that for any run in this module with initial credit 0 and clock values x_0 and y_0 for clocks x and y, the final weight is 0 and the final values of the clocks are unchanged (assuming they are in $[0, 1)$). Moreover, such a run exists if, and only if, $y_0 \leq 2x_0$. One can obviously adapt this module to check any linear constraint between clocks, which entails undecidability (for instance by encoding additive constraints in timed automata, for which reachability is undecidable [5]). This reduction uses five clocks, but the above automata for checking additive constraints can also be used to encode Turing machines, using only four clocks (cf. Section IV-B). ■

IV. Decidability of $\mathcal{A} \models \exists_T L(c_0)$

In this section, we show that we recover decidability when we restrict to the time-bounded L-problem. We also characterize the exact complexity of this problem.

	Fixed initial credit		Existence of initial credit	
α \ Q	$\exists$	$\forall$	$\exists$	$\forall$
∞	≤ 1: decidable [8] ≥ 4: undecidable	in PSPACE ≥ 3: PSPACE-c.	in PSPACE ≥ 3: PSPACE-c.	
T	in NEXPTIME ≥ 5: NEXPTIME-c.			

TABLE I
SUMMARY OF OUR RESULTS

Theorem 2. *The existential time-bounded* L*-problem is* NEXPTIME*-complete (resp.* NP*-complete) for weighted timed automata with five clocks or more, when the time bound is given in binary (resp. unary).*

A. Upper bound

We sketch the proof of the upper bound. Full details are given in Appendix A.

We fix a weighted timed automaton $\mathcal{A}$ and assume that it has an extra clock u that is never reset. We fix a time bound $T \in \mathbb{Q}_{\geq 0}$ and an initial credit $c_0 \in \mathbb{Q}_{\geq 0}$. Our approach consists in building a finite (exponential-size) witness from any run witnessing $\mathcal{A} \models \exists_T L(c_0)$; our algorithm will then non-deterministically pick a path and check that it is a witness. As stated in Lemma 3, the new witness may have the same duration as the original one, or it may end in a configuration from which there is a feasible zero-delay cycle with non-negative effect on the accumulated weight. Such a cycle will be called a profitable zero-delay cycle (see Appendix A1).

Below, we explain the proof for the case where the initial run is finite. The case where the witness is infinite reduces to that case (see Appendix A6).

1) Reduction scheme. Theorem 2 is a consequence of the following lemma, which we explain below.

Lemma 3. *If $\mathcal{A}$ has a feasible run ϱ from some (ℓ_0, v_0, c_0) to some (ℓ, v, c) of duration at most T, then it also contains a feasible run ϱ' of length*[2] *N in $O(T \cdot |X|^3 \cdot |L|^2)$, starting from (ℓ_0, v_0, c_0) and ending in (ℓ', v', c') such that*

- *either $(\ell', v') = (\ell, v)$ and $c' \geq c$ and $v'(u) = v(u)$,*
- *or from (ℓ', v', c') there is a profitable zero-delay cycle.*

The general idea for proving this lemma is to split the witness into a small number of chunks, which we prove (in Lemma 5) can be made short while (broadly speaking) preserving the total time elapse and and increasing the accumulated weight.

For the rest of the proof we assume we are given a finite weighted run ϱ: $(\ell_0, v_0, c_0) \xrightarrow{\delta(t_0)} (\ell_0, v_0 + t_0, c_0') \xrightarrow{e_1} (\ell_1, v_1, c_1) \xrightarrow{\delta(t_1)} (\ell_1, v_1 + t_1, c_1') \ldots$ witnessing the fact that $\mathcal{A} \models \exists_T L(c_0)$.

The $[i..j]$-segment of ϱ, denoted $\varrho[i..j]$, is defined as $(\ell_i, v_i, c_i) \to \cdots \to (\ell_j, v_j, c_j)$. Segment $\varrho[i..j]$ is said *flexible* whenever for every clock $x \in X \setminus \{u\}$, there exist an integer $0 \leq d_x \leq M$ (where M is the maximal constant of $\mathcal{A}$) and an index $h_x > i$ such that

- clock x is not reset along $\varrho[i..h_x - 1]$;
- for every $i \leq k < h_x$, it holds $d_x \leq v_k(x) \leq d_x + 1$ (we assume $M + 1 = \infty$),
- and for every $h_x \leq k \leq j$, it holds $0 \leq v_k(x) \leq 1$.

The terminology *flexible* comes from the fact that we can modify delays along a flexible segments, while keeping the global time elapsed. The following lemma states that a segment of duration less than 1 is made of few flexible sub-segments.

Lemma 4. *Let $\varrho[i..j]$ be a segment of duration less than* 1*. Then $\varrho[i..j]$ can be split into at most $|X| + 1$ many flexible segments. The sum of the lengths of the sub-runs which do not take part in a flexible segment is bounded by $|X|$.*

Consider a flexible segment $\varrho[i..j] = (\ell_i, v_i, c_i) \xrightarrow{\delta(t_i)} \ldots \xrightarrow{e_j} (\ell_j, v_j, c_j)$ along which clock u has constant integral part. We define, for every $i < h \leq j$, $\delta_j = \sum_{k=i}^{h-1} t_k$. For every clock x that is reset along $\varrho[i..j]$ we write m_x (resp. n_x) for the index of the first (resp. last) edge along which x is reset. The run $\varrho[i..j]$ is depicted on Fig. 6.

We transform $\varrho[i..j]$ into a (shorter) run $\varrho'_{i \to j}$ that starts in (ℓ_i, v_i, c_i) and fires edges e_j (resp. e_{m_x}, e_{n_x}) after exactly δ (resp. δ_{m_x}, δ_{n_x}) time units. Furthermore that run will not reset clock x before firing e_{m_x} after δ_{m_x} time units, and it will not reset clock x after having fired e_{n_x} after δ_{n_x} time units (see Fig. 6). This enforces that the clock constraints are satisfied along the new run, and that the final state of ϱ' is the same as that of ϱ. Edges e_{m_x}, e_{n_x} (for $x \in X$) and e_j act as checkpoints. Between two checkpoints we "optimize" the accumulated weight (which ensures that $c'_j \geq c_j$) and shorten the path (which proves the existence of a short witness).

2) Transformation of a segment between two checkpoints. We first focus on one segment between two checkpoints. It is characterized by a constraint described by a tuple $\gamma = ((\ell, v, c), \alpha, e, B, R, c')$ where:

(a) (ℓ, v, c) is the initial state of that segment,
(b) the duration $\alpha \in \mathbb{R}_{>0}$ of that segment (with the assumption that $\mathsf{frac}(v(u)) + \alpha < 1$),
(c) the final edge e of that segment,
(d) the set $B \subseteq X$ of clocks that should not be reset along that segment,
(e) the set $R \subseteq X$ of clocks that should have value 0 at the end of the segment (when firing e),

[2]Precisely, $N = \big(1 + (|X| + 1) \cdot (1 + (|L| + 1) \cdot |L| \cdot (|X| + 1)^2)\big) \cdot (\lfloor T \rfloor + 1)$, where $\lfloor T \rfloor$ is the integral part of T.

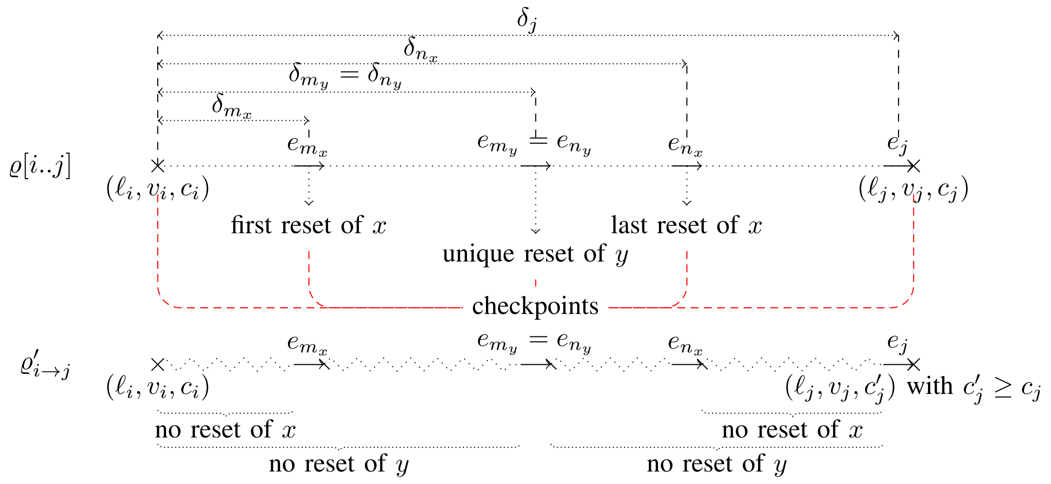


Fig. 6. A flexible segment

(f) a lower bound $c' \in \mathbb{R}_{\geq 0}$ for the accumulated weight at the end of the segment.

We write $R_\gamma(\mathcal{A})$ for the set of runs satisfying constraint γ (defined in an obvious way). Intuitively, if we replace a segment of a feasible run with another segment satisfying the same constraint γ, then the newly constructed run is still feasible. The following lemma shows that any constraint γ can be fulfilled by a short segment.

Lemma 5. *Let $K = (|L|+1) \cdot (|L| \cdot (|X|+1))$. The following two properties are equivalent:*

(i) *there is a finite run ϱ in $R_\gamma(\mathcal{A})$;*

(ii) *there is a run of length at most K that either is in $R_\gamma(\mathcal{A})$ or reaches a state (ℓ', v', c') from which there is a profitable zero-delay cycle.*

Roughly, the idea is to postpone any delay in a location with negative rate to the latest appearance of that location along the run, and to transfer all delays in a location with positive rate to its first appearance along the run. This transformed run is in $R_\gamma(\mathcal{A})$, delays in at most $|L|$ locations, and still satisfies the lower-bound constraint on the rate. Then we can remove the zero-delay cycles (unless they are profitable).

3) Completing the study of a flexible segment. We glue runs we have constructed for all segments between two checkpoints (this requires some technical details that can be found in Appendix A5), and we get a run $\varrho'_{i\to j}$, which starts in (ℓ_i, v_i, c_i), has length at most $(2|X|+1) \cdot K$, satisfies the lower-bound constraint, and

- either it ends in (ℓ_j, v_j, c'_j) with $c'_j \geq c_j$, and has duration that of $\varrho[i..j]$;
- or it ends in a state from which there is a profitable zero-delay cycle, and its duration is at most that of $\varrho[i..j]$.

4) Conclusion and complexity. We apply the previous constructions: we first split the run into segments along which the integral part of u is constant (there are T such segments). Applying Lemma 4 we chunk each of these segments into at most $|X|+1$ flexible segments. We can then change these segments into short segments of length at most $(2|X|+1) \cdot K$ each. The maximal length of the short witness is therefore in $O(T \cdot |X|^3 \cdot |L|^2)$.

By guessing a sequence of consecutive edges of that length in $\mathcal{A}$ and then doing some linear programming resolution, we can check whether there exists a feasible witness along that sequence of edges. This is an NEXPTIME procedure when T is given in binary. It runs in NP if T is encoded in unary.

B. Lower bound

We reuse the modules defined in the proof of Theorem 1 for checking linear constraints between clocks. These modules can be used to check constraints of the form $\alpha x + \beta y + \gamma \geq 0$, for integers α, β and γ. If we allow integer rates (instead of only 0 and 1), we can assume the total time spent to check such a linear constraint is constant (2 time units).

1) Encoding a rule of a Turing machine. We encode a (non-deterministic) Turing machine over alphabet $\{0,1\}$ as follows. Assume the tape content is $w_1(q,a)w_2$ where $w_1, w_2 \in \{0,1\}^*$, q is a state of the Turing machine and $a \in \{0,1\}$ is the content of the cell which is being read. We encode this with a location (q,a) and with the values of two clocks x_1 and x_2, whose binary representation will be $\mathsf{enc}(x_1) = 0.\overline{w_1}$ (we write $\overline{w_1}$ for the word obtained from w_1 by reading from right to left) and $\mathsf{enc}(x_2) = 0.w_2$.

We assume that $(b, \to, q') \in \delta(q,a)$ is a possible transition from (q,a) in the Turing machine. In order to be able to mimic that transition, we want to go to state (q', c) with two clocks[3] y_1 and y_2 such that $\mathsf{enc}(y_1) = 0.b \cdot \overline{w_1}$ and $\mathsf{enc}(y_2) = 0.w'_2$ where $w_2 = c \cdot w'_2$. The symbol $c \in \{0,1\}$ is the content of the new cell which is pointed. In terms of value, we have to enforce:

$$y_1 = \frac{1}{2}x_1 + \frac{1}{2}b \qquad \text{and} \qquad y_2 = 2(x_2 - \frac{1}{2}c).$$

The module is depicted on Fig. 7. We write U on a transition instead of $u = 1, u := 0$. On every locations, for every clock x that is not already used in an outgoing transition, there is

[3]For the sake of readability, our reduction involves six clocks. However, clock y_2 needs not to be fresh, and we will be able to use x_1 instead since it is not used later in the module.

a self-loop $x = 1, x := 0$, which is omitted on the figure (for readability). The module works as follows:

(a) in the first part (until module $\mathsf{Test}(2y_1 = x_1 + b)$), the automaton non-deterministically resets clock y_1, and checks that the new value has the property that after exactly two time units (i.e. after the second U-transition), it holds $y_1 = (x_1 + b)/2$.

(b) then, non-deterministically again, the automaton jumps to one of the two branches:

- the left branch corresponds to the case where $c = 1$: this is checked by the module $\mathsf{Test}(2x_2 \geq 1)$. We then have to set y_2 accordingly, which is achieved by non-deterministically resetting y_2 and checking that $y_2 = 2x_2 - 1$.
- the right branch is for the case where $c = 0$: the encoding is similar to that used in the upper branch, adapted to the fact that $c = 0$. Notice that we have to test a strict inequality, namely $2x_2 < 1$. This is achieved by crediting the variable with $1 - 2x_2$ and checking that this value is positive. This in turn can be tested by the module depicted on Fig. 8: this module checks positiveness of the weight variable while preserving the values of the weight and of all clocks, except z.[4]

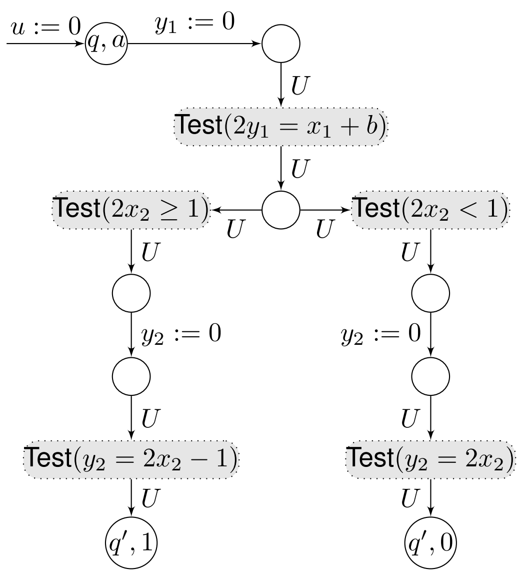
We assume that the total time needed to traverse this module is the same along both branches (even if it means adding some more states). We write d_1 for that constant.

2) The global reduction. We fix a non-deterministic Turing machine $\mathcal{M}$ and an input u of length n. We use the encoding of instructions of a non-deterministic Turing machine as in the previous section, and we glue all modules together to get automaton $\mathcal{A}$. We add an initialization module which leads to (q_0, u_1), set x_1 to 0 and x_2 to $\sum_{i=2}^{n} u_i 2^{-(i-1)}$; this can easily be achieved using module $\mathsf{Test}(2^{n-1}x_2 = \sum_{i=2}^{n} u_i 2^{n-i})$, in total time $d_2 = n - 1$ (using rates 2^{n-1}, 2^{n-2}). We parametrize $\mathcal{A}$ with an integer K, yielding $\mathcal{A}[K]$. In $\mathcal{A}[K]$, a new clock t is reset when reaching the initial state of $\mathcal{M}$ (after the initialization phase), and is used in an invariant $t \leq d_1 K$ in all locations (except the initialization phase and the halting location). Furthermore from every location (except locations of the initialization phase and the halting location) we can go to a sink location with discrete weight -4 when $t \geq d_1 K$ (we call these transitions the bad transitions). Notice that the maximal value of the accumulated weight within a module while mimicking the instructions of $\mathcal{M}$ never exceeds 3: hence, if this transition to the sink state is taken, then the lower-bound constraint will be violated. The halting location has no outgoing transition, and we can wait there as long as we want, with rate zero.

Note that automaton $\mathcal{A}$ has five clocks (four clocks for simulating the instructions of the Turing machine, and the extra clock t).

Proposition 6. *$\mathcal{M}$ halts on input u with a computation of at most K steps if, and only if, $\mathcal{A}[K] \models \exists_{(d_1 K + d_2)+1} L(0)$.*

[4]Similarly as for y_2, clock z needs not be fresh, and it can be replaced by clock x_1, which is not re-used later in the module.



$u := 0$
q, a
$y_1 := 0$
U
$\mathsf{Test}(2y_1 = x_1 + b)$
U
$\mathsf{Test}(2x_2 \geq 1)$
U
U
$\mathsf{Test}(2x_2 < 1)$
U
U
$y_2 := 0$
$y_2 := 0$
U
U
$\mathsf{Test}(y_2 = 2x_2 - 1)$
$\mathsf{Test}(y_2 = 2x_2)$
U
U
$q', 1$
$q', 0$

Fig. 7. Mimicking instruction $(b, \rightarrow, q') \in \delta(q, a)$ (self-loops omitted)

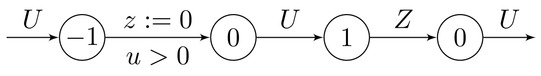


Fig. 8. Automaton for testing that the energy level is positive

Proof: Assume $\mathcal{M}$ has a halting computation of length k (with $k \leq K$) on u. In $\mathcal{A}$, we can first safely initialize the two clocks and reach the initial state of the Turing machine. The initialization phase takes d_2 time units. Then all instructions can also be safely mimicked, and each instruction takes d_1 time units. Hence globally the execution takes then $d_2 + d_1 k$ time units before reaching the halting location, and we can wait there enough time to meet the requirement.

Assume that $\mathcal{A}[K] \models \exists_{(d_1 K + d_2)+1} L(0)$. Then it means that we have first safely initialized the clocks, and then mimicked properly instructions. And then as we have not violated the lower-bound constraint, it means that we have reached a halting location at the latest when $t = pK$ (since otherwise we should have taken the bad transition to the sink state, violating the lower-bound constraint). ■

If K is exponential (in n), the binary encoding of K, and therefore the binary encoding of $d_1 K + d_2$, is polynomial. The above reduction implies $\mathsf{NEXPTIME}$-hardness for the existential time-bounded L-problem when the time bound is given in binary. If K is polynomial in n, the unary encoding of $d_1 K + d_2$ is polynomial (since d_1 is a constant and $d_2 = n - 1$). This implies NP-hardness for the existential time-bounded L-problem when the time bound is given in unary.

Remark 7. *Notice that if we lift the time bound (hence we remove clock t), we encode a (generic) non-deterministic Turing machine. This provides another proof of Theorem 1, which uses only four clocks.*

V. The universal L-problems

In this section we show that the universal L-problems are much easier to solve than the existential L-problems.

Theorem 8. *The universal and the universal time-bounded L-problems are* PSPACE*-complete.*

The PSPACE lower bounds are straightforward reductions from the reachability problem in timed automata: if there is a run reaching the final location, then there is a run of duration at most $T \overset{\text{def}}{=} |\mathcal{R}(\mathcal{A})|$, where $\mathcal{R}(\mathcal{A})$ is the region automaton of $\mathcal{A}$ (see [1]). Therefore by giving a negative rate to accepting locations and rate zero to other locations, the non reachability problem is equivalent to the universal L-problem, time-bounded (by T) or not.

The PSPACE upper bound can be proven using the corner-point abstraction $\mathcal{R}_{cp}(\mathcal{A})$ of $\mathcal{A}$ (see [7]). The result then relies on the following lemma.

Lemma 9. $\mathcal{A} \models \forall_\infty L(c_0)$ *if, and only if,* $\mathcal{R}_{cp}(\mathcal{A}) \models \forall_\infty L(c_0)$.

Proof: Pick an infinite run $\varrho = (\ell_0, v_0, c_0) \to (\ell_1, v_1, c_1) \dots$ which does not satisfy the lower-bound constraint. Take a prefix of ϱ, say $\varrho[0..n]$, which violates the lower-bound constraint: there is $0 \le i \le n$ such that $c_i < 0$. Then applying [7, Prop. 3] we get that there is a finite path π in $\mathcal{R}_{cp}(\mathcal{A})$ with the cost being smaller: this path violates the constraint as well.

Conversely take an infinite path π in $\mathcal{R}_{cp}(\mathcal{A})$ which violates the lower-bound constraint, and take a violating prefix. Then applying [7, Prop. 6], for every $\varepsilon > 0$, we get a real path whose cost is ε-close to that of π, hence becomes negative (for ε small enough). ■

Lemma 10. $\mathcal{R}_{cp}(\mathcal{A}) \not\models \forall_\infty L(c_0)$ *if, and only if, one of the following conditions is satisfied:*

(1) *there is a reachable cycle with a negative effect;*
(2) *there is an acyclic path from the initial state, which can be extended into an infinite path, that yields a negative cost.*

Proof: Take a path which violates the lower-bound constraint. If it contains a cycle with negative effect, then (1) holds. Otherwise consider a prefix which violates the lower-bound constraint. We can remove from that prefix all cycles (since they have non-negative effect), and we still get a counter-example, which furthermore satisfies (2). ■

The above conditions can be checked in PSPACE (the size of $\mathcal{R}_{cp}(\mathcal{A})$ is exponential, and a path in $\mathcal{R}_{cp}(\mathcal{A})$ can be guessed using polynomial space).

This algorithm can be adapted to handle the time-bounded problem, by adding a clock u, which is never reset, but is used in an invariant $u \le T$ on every location. From every location we add a transition constrained by $u = T$ leading to a sink location where cost remains constant. This yields an automaton $\mathcal{B}$ such that $\mathcal{A} \models \forall_T L(c_0)$ if, and only if, $\mathcal{B} \models \forall_\infty L(c_0)$. We apply the previous algorithm to $\mathcal{B}$. The size of $\mathcal{R}(\mathcal{B})$ is exponential in the size of $\mathcal{A}$ and T. This yields a PSPACE algorithm for deciding $\mathcal{A} \models \forall_\infty L(c_0)$.

VI. When the initial credit is not known

We prove here that if the initial credit is not known, then all the problems can be solved in PSPACE. More precisely we prove the following theorem.

Theorem 11. *The existence of an initial credit is* PSPACE*-complete for all the L-problems (existential or universal, time-bounded or not).*

The complete proof of this theorem can be found in Appendix B. The lower bounds are straightforward by reduction from the reachability problem in timed automata, which is known to be PSPACE-hard already when there are three clocks [12].

a) ***The time-bounded case.*** In order to prove the upper bounds for the time-bounded problems, the idea is to reduce the size of possible witnesses using a construction similar to that in Lemma 3, which allows to bound the cost variation along a possible witness and therefore gives information on how much the initial credit needs be for satisfying the lower-bound constraint along the witness. We give the details only for the time-bounded universal problem. It relies on the following characterisation, which can be checked in polynomial space.

Lemma 12. $\mathcal{A} \not\models \exists c.\forall_T L(c)$ *if, and only if, there is a finite initial timed run of duration at most T, which ends in some configuration* (ℓ, v) *s.t. from* (ℓ, v)*, there is a zero-delay cycle with negative accumulated weight.*

Proof: The right-to-left implication is obvious.

Assume $\mathcal{A} \not\models \exists c.\forall_T L(c)$ and fix a timed run ϱ of duration at most T (those will be the only possible witnesses contradicting $\forall_T L(c)$). We transform ϱ as in the proof of Lemma 3 but we minimize the weight instead of maximizing it (and we do not require the lower-bound on the weight be satisfied). This yields an alternative finite run ϱ' which satisfies the following conditions (N s the uniform bound of Lemma 3):

- its duration is at most T and its length is at most N;
- at any point in time, its accumulated weight is smaller than that of ϱ;
- additionally, either (i) it reaches a state (ℓ, v), from which there is a zero-delay cycle with a negative accumulated weight, or (ii) its duration is that of ϱ.

Assume we are in case (ii). If we start with initial credit 0, the cost never goes below $-(N+T) \cdot R$ where R is the maximal absolute value for a rate or a weight decorating a location or an edge of the automaton. Therefore by setting $c_0 = (N+T) \cdot R$, we get that $\varrho' \models L(c_0)$. By construction, ϱ' has smaller accumulated weight than ϱ at any point in time, which implies $\varrho \models L(c_0)$.

Assume towards a contradiction that case (i) never happens. Then, for all runs ϱ of duration at most T, $\varrho \models L(c_0)$. This contradicts the assumption that $\mathcal{A} \not\models \exists c.\forall_T L(c)$. Therefore there is a run ϱ such that its corresponding ϱ' satisfies (i). ■

b) ***The time-unbounded case.*** For proving the upper bound for the standard problems, we rely on the corner-point abstraction $\mathcal{R}_{cp}(\mathcal{A})$ (see [7]). The case of $\exists c.\forall_\infty L(c)$ is not very difficult.

We give the proof for $\exists c.\exists_\infty L(c)$ since it is more involved. It relies on the following lemma, which immediately yields the PSPACE upper bound.

Lemma 13. *$\mathcal{A} \models \exists c.\exists_\infty L(c)$ if, and only if, $\mathcal{R}_{cp}(\mathcal{A}) \models \exists c.\exists_\infty L(c)$.*

Proof: Let ϱ be a witness for $\mathcal{A} \models \exists c.\exists_\infty L(c)$. We project ϱ on $\mathcal{R}_{cp}(\mathcal{A})$, yielding an infinite weighted tree. For each index i there is a branch π'_i of length i of the tree, with overall weight better than that of $\varrho[0..i]$ ([7, Prop. 6]). Since the tree is finitely branching, applying König's lemma, there is an infinite branch π of the tree such that π'_i coincide with $\pi[0..i]$. Let $i_1, \ldots, i_n, \ldots$ the sequence of such positions i's. W.l.o.g. we assume that the accumulated weight along $(\pi[0..i])$ for $i \to \infty$ converges, say to γ (which might be infinite). If γ is finite, this is a stationary sequence (since $\mathcal{R}_{cp}(\mathcal{A})$ is weighted finite automaton with integral values), and we can therefore find a cycle with overall weight zero: this yields a witness for the property (we can then easily compute a bound for the initial credit). If $\gamma = \infty$, then we can find an increasing sub-sequence, and we therefore exhibit a cycle with positive weight, yielding a witness.

Assume now that π is a witness for $\mathcal{R}_{cp}(\mathcal{A}) \models \exists c.\exists_\infty L(c)$. As clock constraints are non-strict, π is a real timed run in $\mathcal{A}$, which witnesses the property. ■

VII. Conclusion

In this paper, we have shown the surprising result that exhibiting an infinite path in a weighted timed automaton satisfying a lower-bound constraint on the cost is actually undecidable when the initial credit is fixed. On the other side, we have considered different variants of that problem, which we proved decidable. The most notable one is the existence of a time-bounded path satisfying the lower-bound constraint, which we prove can be decided in NEXPTIME.

As further work we want to investigate slightly different models for energy consumption, where for instance, the cost can be set to 0 in some places. We think the time-bounded paradigm makes sense and we would like to push it further by analyzing the decidability of various problems which are unfortunately undecidable in general. One of the problems of interest is the existence of a path that satisfies both a lower-bound and an upper-bound constraints.

References

[1] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[2] R. Alur, S. La Torre, and G. J. Pappas, "Optimal paths in weighted timed automata," in *HSCC'01*, ser. LNCS, vol. 2034. Springer, 2001, pp. 49–62.

[3] G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager, "Minimum-cost reachability for priced timed automata," in *HSCC'01*, ser. LNCS, vol. 2034. Springer, 2001, pp. 147–161.

[4] G. Behrmann, K. G. Larsen, and J. I. Rasmussen, "Optimal scheduling using priced timed automata," *SIGMETRICS Performance Evaluation Review*, vol. 32, no. 4, pp. 34–40, 2005.

[5] B. Bérard and C. Dufourd, "Timed automata and additive clock constraints," *Information Processing Letters*, vol. 75, no. 1-2, pp. 1–7, 2000.

[6] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," in *TACAS'99*, ser. LNCS, vol. 1579. Springer, 1999, pp. 193–207.

[7] P. Bouyer, E. Brinksma, and K. G. Larsen, "Staying alive as cheaply as possible," *Formal Methods in System Design*, vol. 32, no. 1, pp. 2–23, 2008.

[8] P. Bouyer, U. Fahrenberg, K. G. Larsen, and N. Markey, "Timed automata with observers under energy constraints," in *HSCC'10*. ACM Press, 2010, pp. 61–70.

[9] ——, "Quantitative analysis of real-time systems using priced timed automata," *Communications of the ACM*, vol. 54, no. 9, pp. 78–87, 2011.

[10] P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey, and J. Srba, "Infinite runs in weighted timed automata with energy constraints," in *FORMATS'08*, ser. LNCS, vol. 5215. Springer, 2008, pp. 33–47.

[11] Th. Brihaye, L. Doyen, G. Geeraerts, J. Ouaknine, J.-F. Raskin, and J. Worrell, "On reachability for hybrid automata over bounded time," in *ICALP'11*, ser. LNCS, vol. 6756. Springer, 2011, pp. 416–427.

[12] C. Courcoubetis and M. Yannakakis, "Minimum and maximum delay problems in real-time systems," *Formal Methods in System Design*, vol. 1, no. 4, pp. 385–415, 1992.

[13] U. Fahrenberg, L. Juhl, K. G. Larsen, and J. Srba, "Energy games in multiweighted automata," in *ICTAC'11*, ser. LNCS, vol. 6916. Springer, 2011, pp. 95–115.

[14] K. G. Larsen and J. I. Rasmussen, "Optimal conditional reachability for multi-priced timed automata," in *FoSSaCS'05*, ser. LNCS, vol. 3441. Springer, 2005, pp. 234–249.

[15] N. Markey, "Verification of embedded systems – algorithms and complexity," École Normale Supérieure de Cachan, France, Mémoire d'habilitation, 2011.

[16] J. Ouaknine and J. Worrell, "Towards a theory of time-bounded verification," in *ICALP'10*, ser. LNCS, vol. 6199. Springer, 2010, pp. 22–37.

[17] K. Quaas, "On the interval-bound problem for weighted timed automata," in *LATA'11*, ser. LNCS, vol. 6638. Springer, 2011, pp. 452–464.

[18] M. Woehrle, K. Lampka, and L. Thiele, "Segmented state space traversal for conformance testing of cyber-physical systems," in *FORMATS'11*, ser. LNCS, vol. 6919. Springer, 2011, pp. 193–208.

Technical appendix

A. Proof of Theorem 2

1) Zero-delay cycles. A zero-delay cycle on state (ℓ, v) is a path starting and ending in (ℓ, v) in which all delay transitions are 0-delay. In particular, such a cycle cannot visit a resetting transition, unless the reset clock already has value 0 in v.

Our first task is to detect those configurations from which a zero-delay cycle is feasible. For a state (ℓ, v) and a non-negative real c, we set the Boolean predicate $\mathsf{profit}((\ell, v), c)$ to true if, and only if, there is a zero-delay cycle from (ℓ, v, c) back to (ℓ, v, c') for some $c' \geq c$, along which the lower-bound constraint is satisfied. Such a cycle is called a profitable zero-delay cycle. Note that $\mathsf{profit}((\ell, v), c) = \mathsf{profit}((\ell, v'), c)$ whenever v and v' belong to the same region r, so that the region-based predicate $\mathsf{profit}((\ell, r), c)$ is well-defined (in the obvious way).

Detecting zero-delay cycles can be done efficiently:

Lemma 14. *Given (ℓ, r), we can compute in polynomial time the smallest c such that $\mathsf{profit}((\ell, r), c)$, if any. Furthermore it is a natural number.*

Proof: Consider the sub-graph of the region automaton where only region r appears. This is a weighted graph whose size is at most $|L|$. Using a modified Bellman-Ford algorithm [10], we can compute the least c for which $\mathsf{profit}((\ell, r), c)$, if any. This algorithm runs in polynomial time, and returns a natural number (because discrete weights are integers). ■

We use this predicate to abstract weighted runs: from a state (ℓ, v, c) satisfying $\mathsf{profit}((\ell, v), c)$, there is an infinite (Zeno) run satisfying the lower-bound constraint[5]. Furthermore as we shall prove in Section A6, any (Zeno) infinite witness will visit a state satisfying $\mathsf{profit}((\ell, v), c)$. Hence we have reduced our problem to that of finding a finite witness, either with duration T or with duration less than T but ending in a state which carries a profitable zero-delay cycle.

2) Proof of Lemma 4. Since $\varrho[i..j]$ has duration less than 1, there are at most $|X|$ many delay transitions along which some clock reaches the upper bound of the unit interval to which it belongs (that is, changes integral part with no reset operation). Each segment between any two consecutive such transitions is clearly flexible. ∎

3) Equivalence between constraints. We need to know how the fractional parts of the clocks compare to that of universal clock u. A valuation v is said Y*-small* (for some $Y \subseteq X$) whenever for every $y \in Y$, it holds $0 \le \mathsf{frac}(v(y)) \le \mathsf{frac}(v(u))$. Notice that for any $t < 1 - \mathsf{frac}(v(u))$, any valuation reached from v within t time units is Y-small. The following lemma characterizes equivalent constraints:

Lemma 15. *Assume that we are given a constraint $\gamma = ((\ell, v, c), \alpha, e, B, R, c')$. Fix $Y \subseteq X$ such that v is Y-small, and fix another Y-small valuation $\widehat{v}$ that agrees with v on $X \setminus Y$, and s.t. for every $x \in Y$, $v(x) = 0$ implies $\widehat{v}(x) = 0$. We define the constraint $\widehat{\gamma}$ as $((\ell, \widehat{v}, c), \alpha, e, B, R, c')$. Then:*

1) *If $R_\gamma(\mathcal{A}) \neq \emptyset$, then $R_{\widehat{\gamma}}(\mathcal{A}) \neq \emptyset$. Also, if there is a run in $R_\gamma(\mathcal{A})$ with final accumulated weight e, then there is one in $R_{\widehat{\gamma}}(\mathcal{A})$ with the same final accumulated weight.*
2) *If $\varrho \in R_\gamma(\mathcal{A})$ and $\widehat{\varrho} \in R_{\widehat{\gamma}}(\mathcal{A})$, writing v' (resp. $\widehat{v}'$) for the valuation at the end of ϱ (resp. $\widehat{\varrho}$), we have that:*

$$\begin{cases} \forall x \in (X \setminus Y) \cap B,\ v'(x) = \widehat{v}'(x) \\ \forall x \in R,\ v'(x) = \widehat{v}'(x) = 0 \\ v' \text{ and } \widehat{v}' \text{ are } ((X \setminus B) \cup Y)\text{-small} \end{cases}$$

Proof: Pick a flexible run ϱ' in $R_\gamma(\mathcal{A})$. We explain why this run can be mimicked (with the same delay- and action transitions) from $(\ell, \widehat{v}, c)$, yielding a flexible run in $R_{\widehat{\gamma}}(\mathcal{A})$. Write $\widehat{\varrho'}$ for the run (we prove below that all guards are fulfilled, so that this is really a run in $\mathcal{A}$) obtained by mimicking ϱ' from $(\ell, \widehat{v}, c)$. First, all valuations along ϱ' are Y-small, since $v(u) = \widehat{v}(u)$ and both ϱ' and $\widehat{\varrho'}$ have duration α. Furthermore, for $y \in Y$, if $v(y) = 0$ then $\widehat{v}(y) = 0$. Therefore for clocks in Y, all constraints that are satisfied along ϱ' are satisfied along $\widehat{\varrho'}$. For clocks not in Y, v and $\widehat{v}$ agree, so that all conditions are fulfilled for being a flexible run in $R_{\widehat{\gamma}}(\mathcal{A})$.

We now prove the second point. Along both ϱ and $\widehat{\varrho}$, clocks in B are not reset (except possibly for those also in R). Therefore as v and $\widehat{v}$ initially agree on $X \setminus Y$, we get the two first lines. The third point is straightforward. ∎

[5]Notice that, using similar techniques, we could easily handle different variants of our time-bounded problem, for instance if it is required that the witness run must have duration at least T: when a state (ℓ, v, c) with $\mathsf{profit}((\ell, v), c)$ is visited, the weight variable can be made arbitrarily large, and the lower-bound constraint is lifted. It just remains to decide the existence of a run along which a sufficient amount of time elapses, with no weight constraint.

4) Proof of Lemma 5. Pick a finite run ϱ in $R_\gamma(\mathcal{A})$, and modify it as follows: for every location ℓ_1 in which a positive delay is spent along ϱ, do the following:

- if $\mathsf{rate}(\ell_1) \ge 0$, then transfer all delays spent in such a location to the first occurrence of ℓ_1 along ϱ where some positive delay is already spent;
- if $\mathsf{rate}(\ell_1) < 0$, transfer all delays spent in such a location to the last occurrence of ℓ_1 where some positive delay is already spent.

We can do so since none of the guards along ϱ is violated because of this change: indeed, if $x \in B$, from the first positive delay on, x is in $(c; c+1)$ all along ϱ (as ϱ is flexible and x is not reset along ϱ). If $x \notin B$, then v is $\{x\}$-small. Assume that some transition has a guard $x = c$ (with $c > 0$) along ϱ. Then no delay is spent before this transition, or no delay is spent after. Hence there is no transfer of delay from one side of this transition to the other, so that the guard is still fulfilled in the new run. Similarly, if there is a guard $x = 0$ in the original run, then no time is elapsed since the previous reset of x (or since the beginning if x is zero at the beginning), and the guard is still fulfilled in the new run. Finally, if there is a guard $x \in [c; c+1]$, then transferring delays to the beginning of the run (before a reset of x) may increase the value of x, but as $v(x) < v(u)$ and $\mathsf{frac}(v(u)) < 1$ all along ϱ, then $x \in [c; c+1]$ is still fulfilled along the new run. Transferring delay after a reset of x would still keep $x \in [0; 1]$.

The modified run delays in at most $|L|$ locations, the lower-bound constraint is still satisfied (because the accumulated weight in any location of the new run is at least as high as in the original run).

We name ϱ' the new run, which is a new witness for (i). Consider a sub-run of ϱ' that is taken in 0-delay: if a state (ℓ_1, v_1) is visited twice along that sub-run and if the corresponding cycle has a non-positive effect, then we can remove that part of the run, the overall weight will be larger and we still have a witness for (i); if a such a cycle has a positive effect, then $\mathsf{profit}((\ell_1, v_1), c_1)$ is true (c_1 is the credit at the first visit), and the rest of the run can be dropped.

The resulting run delays in at most $|L|$ locations, and is acyclic in the 0-delay segments. In a 0-delay segment, once a clock is reset, it remains equal to 0: therefore, the length of such an acyclic 0-delay segment is at most $|L| \cdot (|X|+1)$. The length of the resulting run is thus bounded by $(|L|+1) \cdot (|L| \cdot (|X|+1))$. This run is a witness for (ii). ∎

5) Flexible segments. We come back to our flexible segments $\varrho[i..j] = (\ell_i, v_i, c_i) \xrightarrow{\delta(t_i)} \dots \xrightarrow{e_j} (\ell_j, v_j, c_j)$, and consider the checkpoints defined earlier. We write $I = \{i, j\} \cup \{m_x, n_x \mid x \in X\}$ for the indices corresponding to checkpoints, and we assume that it is ordered as $I = \{i_0 < i_2 < \cdots < i_p\}$. For each $0 \le h < p$, we consider the constraint

$$\gamma_h = ((\ell_{i_h}, v_{i_h}, c_{i_h}), \delta_{i_{h+1}} - \delta_{i_h}, e_{i_{h+1}}, B_h, R_h, c_{i_{h+1}})$$

where $B_h = \{x \in X \mid h < m_x \text{ or } h \ge n_x\}$ and $R_h = \{x \in X \mid v_{i_{h+1}}(x) = 0\}$. The set B_h is the set of clocks that should not be reset. We first argue why this is a proper constraint. Condition (b) is by assumption on clock u whereas condition (d) requires some arguments that we give now. Pick $y \notin B_h$: it means that $m_y \le h < n_y$. We have that $v_{i_h}(y) \le \delta_{i_h} - \delta_{m_y}$ since y was reset at edge m_y (and also possibly later). In the meantime $\mathsf{frac}(v_{i_h}(u)) = \mathsf{frac}(v(u)) + \delta_{i_h} \le \delta_{i_h}$. Therefore $0 \le v_{i_h}(y) \le \mathsf{frac}(v_{i_h}(u))$, which implies condition (d).

We build a run ϱ solving the constrained problem γ_0, then γ'_1, then γ'_2, etc. Constraints γ'_h will only differ from γ_h in the initial valuation (which will still satisfy the hypotheses of Lemma 15).

First note that for every h, v_{i_h} is $\big(\bigcup_{k<h}(X \setminus B_k)\big)$-small. By induction on h, we prove the following: if v'_{i_h} is related to v_{i_h} as in the hypotheses of Lemma 15 (with $Y_h = \bigcup_{k<h}(X \setminus B_k)$). We build a run in $R_{\gamma'_h}(\mathcal{A})$ (γ'_h is same as γ_h except for the initial valuation which is v'_{i_h} instead of v_{i_h}). The run $\varrho[i_h..i_{h+1}]$ is in $R_{\gamma_h}(\mathcal{A})$. Applying the first part of Lemma 15 we get that $R_{\gamma'_h}(\mathcal{A})$ is non-empty. Applying Lemma 5, we select a run $\varrho'_{i_h \to i_{h+1}}$ of length bounded by K in $R_{\gamma'_h}$ (or ending in a state carrying a profitable zero-delay cycle, which concludes the proof). Applying the second part of Lemma 15, we get that the final valuation of $\varrho'_{i_h \to i_{h+1}}$, say $v'_{i_{h+1}}$, is $((X \setminus B_h) \cup Y_h)$-small. By construction, for all clocks x such that $v_{i_{h+1}}(x) = 0$ (this is R_h), $v'_{i_{h+1}}(x) = 0$. Valuation $v'_{i_{h+1}}$ therefore satisfies the hypotheses of Lemma 15.

Initially we assume $v_0 = v'_0$. We have thus constructed by induction runs $\varrho'_{i_h \to i_{h+1}}$ that can be glued together, yielding a run $\varrho'_{i \to j}$ of duration δ. The final state of $\varrho'_{i \to j}$ is (ℓ_j, v_j, c'_j) (by construction), and $c'_j \ge c_j$. Furthermore the length of $\varrho'_{i \to j}$ is at most $(2|X|+1) \cdot K$. We have proved:

Proposition 16. *Take a flexible segment $\varrho[i..j]$ of ϱ, such that clock u has constant integral part along $\varrho[i..j]$. We can construct a run $\varrho'_{i \to j}$ such that the following holds:*

- *it starts in (ℓ_i, v_i, c_i);*
- *its duration is that of $\varrho[i..j]$;*
- *its length is at most $(2|X|+1) \cdot K$;*
- *it satisfies the lower-bound problem;*
- *if (ℓ_j, v'_j, c'_j) is its final state, then $v'_j = v_j$, and $c'_j \ge c_j$.*

6) Case where the witness ϱ is infinite. We now assume that $\varrho = (\ell_0, v_0, c_0) \to \dots (\ell_n, v_n, c_n) \dots$ is an infinite witness of total duration no more than T. We will show that there is a finite run which satisfies the lower-bound constraint $L(c_0)$ and which ends in a state from which it is possible to follow a zero-delay cycle while satisfying the lower-bound constraint and whose accumulated weight is non-negative (we can define a predicate $\mathsf{profit}^{\ge 0}$ similar to profit, but where the accumulated weight is non-negative instead of positive).

There exists n such that the tail $\varrho[n..\infty)$ of the witness is flexible (and clock u lies within one time unit). For all locations ℓ_1 with $\mathsf{rate}(\ell_1) > 0$ in which some delay is spent along $\varrho[n..\infty)$, we transfer all delays in location ℓ_1 to its first occurrence where some delay is spent. We remove all delays spent in some location ℓ_2 with $\mathsf{rate}(\ell_2) \le 0$. This yields a new witness ϱ' such that $\varrho'_{m \to +\infty}$ is zero-delay: we can extract from that tail a zero-delay cycle which satisfies the lower-bound constraint and has a non-negative accumulated weight.

B. Proof of Theorem 11

1) PSPACE ***upper bound for $\exists c.\exists_T L(c)$.***

Lemma 17. *$\mathcal{A} \models \exists c.\exists_T L(c)$ if, and only if, there is a finite initial. run ϱ s.t.*

(i) *either its duration is T;*
(ii) *or its duration is no more than T, and it ends in configuration (ℓ, v) s.t. from (ℓ, v), there is a profitable zero-delay cycle.*

Proof: Assume $\mathcal{A} \models \exists c.\exists_T L(c)$, and take a witness run ϱ. If ϱ is finite, then (i) holds. If ϱ is infinite, we apply a construction similar to the proof presented in Annex A6, and we get a finite run which satisfies the condition (ii).

Assume that either (i) or (ii) holds. If (i) holds, then we can choose a large enough initial credit to compensate any decrease in the weight along ϱ: if the length of ϱ is ℓ, then if we start with initial credit 0, the cost never goes below $-(\ell + T) \cdot R$, where R is the maximal absolute value for a rate or a weight decorating a location or an edge of $\mathcal{A}$. Therefore by setting $c_0 = (\ell + T) \cdot R$, we get that $\varrho \models L(c_0)$. If (ii) holds, then we easily get an infinite witness for the property. The initial credit will only be used to compensate any loo along the finite run. Therefore $\mathcal{A} \models \exists c.\exists_T L(c)$. ∎

Note that in the proof above, by applying a construction to that of Lemma 3, we can bound the length of the finite paths by N. Therefore, if there is some initial credit, then there is one which is bounded by $(N + T) \cdot R$.

The two conditions of the lemma can be checked in polynomial space.

2) PSPACE ***upper bound for $\exists c.\forall_\infty L(c)$.*** Write $\mathcal{R}_{cp}(\mathcal{A})$ for the corner-point abstraction of $\mathcal{A}$ (see [7]). Then we show:

Lemma 18. *$\mathcal{A} \models \exists c.\forall_\infty L(c)$ if, and only if, $\mathcal{R}_{cp}(\mathcal{A}) \models \exists c.\forall_\infty L(c)$.*

This is an obvious consequence of Lemma 9. This condition can be easily checked in polynomial space by detecting cycles with negative cost in $\mathcal{R}_{cp}(\mathcal{A})$.

3) PSPACE ***lower bound for $\exists c.\forall_\alpha L(c)$.*** We fix a timed automaton $\mathcal{A}$ and we build the weighted timed automaton $\mathcal{B}$ by assigning weight and rate zero everywhere in $\mathcal{A}$, and from the final location of $\mathcal{A}$ we go to a sink location, with a self-loop labelled with weight -1. We define $T = |\mathcal{R}(\mathcal{A})|$. The following four properties are then equivalent:

(i) the final location of $\mathcal{A}$ is reachable
(ii) the final location of $\mathcal{A}$ is reachable in no more than T time units
(iii) $\mathcal{B} \not\models \exists c.\forall_\infty L(c)$
(iv) $\mathcal{B} \not\models \exists c.\forall_T L(c)$

4) PSPACE ***lower bound for $\exists c.\exists_\alpha L(c)$.*** the argument is similar to the previous one: automaton $\mathcal{B}$ is now obtained from $\mathcal{A}$ by assigning weight and rate -1 everywhere: for any initial credit the weight of any run will decrease to infinity unless we allow to escape to a rate-zero location. From the final location of $\mathcal{A}$ we go to a sink location, with a self-loop of weight zero. The very same equivalent properties can be stated for this case.