Infinite Runs in Weighted Timed Automata with Energy Constraints

Patricia Bouyer[1]*, Uli Fahrenberg[2], Kim G. Larsen[2], Nicolas Markey[1]*, and Jiří Srba[2]**

[1] Lab. Spécification et Vérification, CNRS & ENS Cachan, France
{bouyer,markey}@lsv.ens-cachan.fr
[2] Dept. of Computer Science, Aalborg University, Denmark
{uli,kgl,srba}@cs.aau.dk

Abstract. We study the problems of existence and construction of infinite schedules for finite weighted automata and one-clock weighted timed automata, subject to boundary constraints on the accumulated weight. More specifically, we consider automata equipped with positive and negative weights on transitions and locations, corresponding to the production and consumption of some resource (*e.g.* energy). We ask the question whether there exists an infinite path for which the accumulated weight for any finite prefix satisfies certain constraints (*e.g.* remains between 0 and some given upper-bound). We also consider a game version of the above, where certain transitions may be uncontrollable.

1 Introduction

The overall motivation of the research underlying this paper is the quest of developing weighted (or priced) timed automata and games [3, 2] into a universal formalism useful for formulating and solving a broad range of resource scheduling problems of importance in application areas such as, *e.g.*, embedded systems. In this paper we introduce and study a new resource scheduling problem, namely that of constructing infinite schedules or strategies subject to boundary constraints on the accumulation of resources.

More specifically, we propose finite and timed automata and games equipped with positive as well as negative weights, respectively weight-rates. With this extension, we may model systems where resources are not only consumed but also occasionally produced or regained, *e.g.* useful in modelling autonomous robots equipped with solar-cells for energy-harvesting or with the ability to search for docking-stations when energy-level gets critically low. Main challenges are now to synthesize schedules or strategies that will ensure indefinite safe operation with the additional guarantee that energy will always be available, yet never exceeds a possible maximum storage capacity.

* This author is partially supported by project DOTS (ANR-06-SETI-003).
** This author is partially supported by grant no. MSM-0021622419.

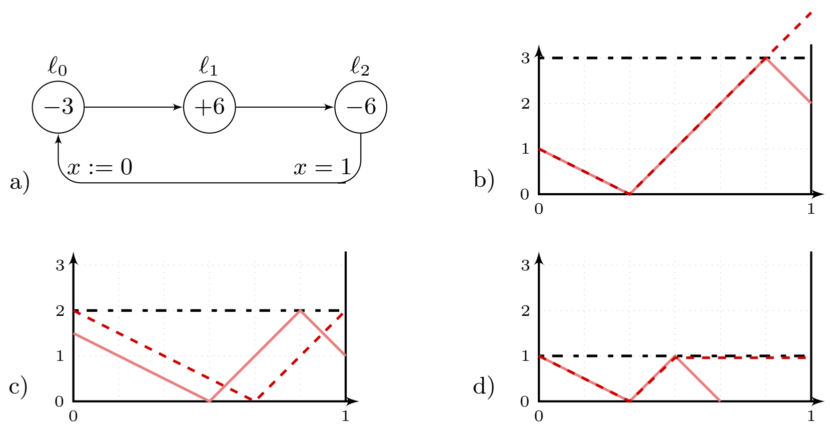


Fig. 1. a) Weighted Timed Automaton (global invariant $x \leq 1$).
b) $U = +\infty$ and $U = 3$. c) $U = 2$. d) $U = 1$ and $W = 1$.

As a basic example, consider the weighted timed automaton in Fig. 1a) with infinite behaviours repeatedly delaying in ℓ_0, ℓ_1 and ℓ_2 for a total of precisely one time-unit. The negative weights (-3 and -6) in ℓ_0 and ℓ_2 indicate the rates by which energy will be consumed, and the positive rate ($+6$) in ℓ_1 indicates the rate by which energy will be gained. Thus, for a given iteration the effect on the amount of energy remaining will depend highly on the distribution of the one time-unit over the three locations. Let us observe the effect of lower and upper constraints on the energy-level on so-called bang-bang strategies, where the behaviour remains in a given location as long as permitted by the given bounds. Fig. 1b) shows the bang-bang strategy given an initial energy-level of 1 with no upper bound (dashed line) or 3 as upper bound (solid line). In both cases, it may be seen that the bang-bang strategy yields an infinite behaviour.

In Fig. 1c) and d), we consider the upper bounds 2 and 1, respectively. For an upper bound of 2, we see that the bang-bang strategy reduces an initial energy-level of $1\frac{1}{2}$ to 1 (solid line), and yet another iteration will reduce the remaining energy-level to 0. In fact, the bang-bang strategy—and it may be argued, any other strategy—fails to maintain an infinite behaviour for any initial energy-level *except for* 2 (dashed line). With upper-bound 1, the bang-bang strategy—and any other strategy—fails to complete even one iteration (solid line).

We also propose an alternative *weak* notion of upper bounds, which does not prevent energy-increasing behaviour from proceeding once the upper bound is reached but merely maintains the energy-level at the upper bound. In this case, as also illustrated in Fig. 1d) (dashed line), the bang-bang strategy is quite adequate for yielding an infinite behaviour.

In this paper, we ask the question whether some, or all, of the infinite paths obey the property that the weight accumulated in any finite prefix satisfies certain constraints. The *lower-bound problem* requires the accumulated weight never drop below zero, the *interval-bound problem* requires the accumulated weight to stay within a given interval, and in the *lower-weak-upper-bound problem* the accumulated weights may not drop below zero, with the additional feature that the weights never accumulate to more than a given upper bound; any increases above this bound are simply truncated. We also consider a game version of the above setting, where certain transitions may be uncontrollable.

For finite weighted automata, we show that the lower- and lower-weak-upper-bound problems are decidable in polynomial time. In the game setting we prove these problems to be P-hard but decidable in NP ∩ coNP. The interval-bound problem is on the other hand NP-hard but decidable in PSPACE and in the game setting it is EXPTIME-complete. For one-clock weighted timed automata, the lower- and lower-weak-upper-bound problems remain in P, but the interval-bound problem in the game setting is already undecidable.

Related Work. Recently, extensions of timed automata with costs (or prices) to measure quantities like energy, bandwidth, etc. have been introduced and subject to significant research. In these models—so-called *priced* or *weighted timed automata*—the cost can be used to measure the performance of a system allowing various optimization problems to be considered. Examples include cost-optimal reachability problem in a single-cost setting [2, 3], as well as in a multi-cost setting [19], or the computation of (mean-cost or discounted) cost-optimal infinite schedules [6, 15]. Also games where players try to optimize the cost have been considered [1, 7, 12, 5, 9], and several model-checking problems have been studied [11, 5, 8, 10]. It is worth noticing that in these last frameworks (model-checking and games), most of the problems are undecidable, and they can be solved algorithmically only for the (restricted) class of one-clock automata.

However, in the priced extensions considered so far, only non-negative costs are allowed restricting the types of quantities one can measure. The only exception concerns the optimal-cost reachability problem, which has been solved even when negative costs are allowed [4], but the proof is basically not much more involved than with only non-negative costs. In the current work, we pursue this line of research, and focus on non-trivial problems that can be posed on timed automata with arbitrary costs, which allow to model interesting problems on timed systems with (bounded) resources.

2 Models and Problems

2.1 Weighted Automata and Games

A *weighted automaton* is a tuple $A = (S, s_0, T)$ consisting of a set of locations (or states) S, an initial location (state) $s_0 \in S$, and a set of transitions $T \subseteq S \times \mathbb{R} \times S$. A transition (s, p, s') is customarily denoted $s \xrightarrow{p} s'$, where $p \in \mathbb{R}$ is the *weight* of that transition. We implicitly consider only *non-blocking* automata where every location has at least one outgoing transition. If S and T are finite sets and $T \subseteq S \times \mathbb{Z} \times S$ then A is called a *finite* weighted automaton.

A run in A starting from $s \in S$ is a finite or infinite sequence $s = s_1 \xrightarrow{p_1} s_2 \xrightarrow{p_2} s_3 \xrightarrow{p_3} \cdots$. We write $\mathsf{Runs}(A, s)$ for the set of all runs in A starting from s. Given a finite run $\gamma = s_1 \xrightarrow{p_1} \cdots \xrightarrow{p_{n-1}} s_n$, we write $\mathsf{last}(\gamma)$ to denote its final location s_n.

Definition 1. *Let $A = (S, s_0, T)$ be a weighted automaton, $c \in \mathbb{R}$, $b \in \mathbb{R} \cup \{\infty\}$, and let $\gamma = s_1 \xrightarrow{p_1} \cdots \xrightarrow{p_{n-1}} s_n \in \mathsf{Runs}(A, s_1)$. The* accumulated weight with initial credit c under weak upper bound b *is $p_{c\downarrow b}(\gamma) = r_n$, where $r_1, \ldots, r_n \in \mathbb{R}$ are defined inductively by $r_1 = \min(c, b)$, $r_{i+1} = \min(r_i + p_i, b)$.*

So for computing $p_{c\downarrow b}(\gamma)$, costs are accumulated along the transitions of γ, but only up to a maximum accumulated cost b; possible increases above b are simply discarded. The case $b = \infty$ is special; we will denote $p_c(\gamma) = p_{c\downarrow\infty}(\gamma) = c + \sum_{i=1}^{n-1} p_i$. Also, we will write $p(\gamma) = p_0(\gamma)$.

A *weighted game* is a tuple $G = (S_1, S_2, s_0, T)$ where S_1 and S_2 are two disjoint sets of locations, and $A_G = (S_1 \cup S_2, s_0, T)$ is a weighted automaton. Note that we only introduce *turn-based* weighted games here.

A run of G is a run of A_G, and we write $\mathsf{Runs}(G)$ for $\mathsf{Runs}(A_G)$. A strategy σ for Player i, where $i = 1$ or $i = 2$, maps each finite run γ ending in S_i to a transition departing from $\mathsf{last}(\gamma)$. Given a location s in G and a strategy σ for Player i, an outcome of σ from s is any run $s_1 \xrightarrow{p_1} s_2 \xrightarrow{p_2} \cdots \xrightarrow{p_{n-1}} s_n$ starting in s such that for any k, if $s_k \in S_i$ then $s_k \xrightarrow{p_k} s_{k+1} = \sigma(s_1 \xrightarrow{p_1} s_2 \xrightarrow{p_2} \cdots \xrightarrow{p_{k-1}} s_k)$.

We are now able to formulate the problems which we shall be concerned with.

The lower-bound problem: Given a weighted game G and an initial credit c, does there exist a strategy for Player 1 such that any infinite outcome γ from the initial state of G has $p_c(\gamma') \geq 0$ for all finite prefixes γ' of γ?

The lower-weak-upper-bound problem: Given a weighted game G, a weak upper bound b and an initial credit $c \leq b$, does there exist a strategy for Player 1 such that any infinite outcome γ from the initial state of G has $p_{c\downarrow b}(\gamma') \geq 0$ for all finite prefixes γ' of γ?

The interval-bound problem: Given a weighted game G, an upper bound b, and an initial credit $c \leq b$, does there exist a strategy for Player 1 such that any infinite outcome γ from the initial state of G has $0 \leq p_c(\gamma') \leq b$ for all finite prefixes γ' of γ?

Note that by Martin's determinacy theorem [20], our games are *determined*: Player 1 has a strategy for winning one of the above games if, and only if, Player 2 does not have a strategy for making Player 1 lose.

Special variants of the above problems are obtained when one of the sets S_1 and S_2 is empty. In case $S_2 = \emptyset$, they amount to asking for the *existence* of an infinite path adhering to the given bounds; in case $S_1 = \emptyset$, one asks whether *all* infinite paths stay within the bounds. The former problems will be referred to as *existential* problems, the latter as *universal* ones.

2.2 The Timed Setting

A *weighted timed automaton* is a tuple $A = (Q, q_0, C, I, E, \mathit{rate})$, with Q a finite set of locations, $q_0 \in Q$ the initial location, C a finite set of clocks, $I \colon Q \to \Phi(C)$

location invariants, $E \subseteq Q \times \Phi(C) \times 2^C \times Q$ a finite set of transitions, and *rate*: $Q \to \mathbb{Z}$ location weight-rates. Here the set $\Phi(C)$ of clock constraints φ is defined by the grammar $\varphi ::= x \bowtie k \mid \varphi_1 \wedge \varphi_2$ with $x \in C$, $k \in \mathbb{Z}$ and $\bowtie \in \{\le, <, \ge, >, =\}$. The semantics of a weighted timed automaton A is given by a weighted automaton $[\![A]\!]$ with states (q, v), where $q \in Q$ and $v\colon C \to \mathbb{R}_{\ge 0}$, and transitions of two types:

– *Delay transitions* $(q, v) \xrightarrow{p}_t (q, v + t)$ for some $t \in \mathbb{R}_{\ge 0}$. Such a transition exists whenever $v + t' \models I(q)$ for all $t' \in [0, t]$, and its weight is $p = t \cdot \mathit{rate}(q)$.
– *Switch transitions* $(q, v) \xrightarrow{0}_e (q', v')$, where $e = (q, g, r, q')$ is a transition of A. Such a switch transition exists whenever $v \models g \wedge I(q)$, $v' = v[r \leftarrow 0]$, and $v' \models I(q')$. Switch transition have always the weight 0.

A *run* of A is a sequence $(q_1, v_1) \xrightarrow{p_1}_{t_1} (q_1, v_1') \xrightarrow{0}_{e_1} (q_2, v_2) \xrightarrow{p_2}_{t_2} (q_2, v_2') \cdots$ of alternating delay and switch transitions in $[\![A]\!]$. We write $\mathsf{Runs}(A)$ for the set of all runs of A. The accumulated weight (with initial credit c and weak upper bound b) for runs of a weighted timed automaton are given by Definition 1.

A *weighted timed game* is a tuple $G = (Q, q_0, C, I, E_1, E_2, \mathit{rate})$ such that $A_G = (Q, q_0, C, I, E_1 \cup E_2, \mathit{rate})$ is a weighted timed automaton. States and runs of a weighted timed game are those of the underlying weighted timed automaton, and we again write $\mathsf{Runs}(G)$ for $\mathsf{Runs}(A_G)$.

Given a weighted timed game G, a strategy for Player i ($i \in \{1, 2\}$) is a partial function σ mapping each finite run of $\mathsf{Runs}(A_G)$ to $E_i \cup \{\lambda\}$, where $\lambda \notin E_1 \cup E_2$ is the "wait" action. For a finite run γ ending in (q, v), it is required that if $\sigma(\gamma) = \lambda$ then $(q, v) \xrightarrow{p}_t (q, v + t)$ for some $t > 0$, and if $\sigma(\gamma) = e$ then $(q, v) \xrightarrow{0}_e (q', v')$.

Given a state $s = (q, v)$ of a weighted timed game G and a strategy σ for Player i, the set $\mathsf{Out}(\sigma, s)$ of outcomes of σ from s is the smallest subset of $\mathsf{Runs}(G, s)$ such that

– (q, v) is in $\mathsf{Out}(\sigma, s)$;
– if $\gamma \in \mathsf{Out}(\sigma, s)$ is a finite run ending in (q_n, v_n), then a run of the form $\gamma \to (q_{n+1}, v_{n+1})$ is in $\mathsf{Out}(\sigma, s)$ if either
  - $(q_n, v_n) \to_e (q_{n+1}, v_{n+1})$ with $e \in E_{3-i}$,
  - $(q_n, v_n) \to_e (q_{n+1}, v_{n+1})$ with $e = \sigma(\gamma) \in E_i$, or
  - $(q_n, v_n) \to_t (q_{n+1}, v_{n+1})$ with $t \in \mathbb{R}_{\ge 0}$, and $\sigma(\gamma \to_{t'} (q_n, v_n + t')) = \lambda$ for all $t' \in [0, t)$;
– an infinite run is in $\mathsf{Out}(\sigma, s)$ if all its finite prefixes belong to $\mathsf{Out}(\sigma, s)$.

A weighted timed game $G = (Q, q_0, C, I, E_1, E_2, r)$ is said to be *turn-based* if the set of edges leaving any $q \in Q$ is a subset of either E_1 or E_2. In that case, the semantics of G is a (turn-based, infinite) weighted game as introduced in the previous section. However, weighted timed games introduced above are more general than turn-based ones, and we shall later use the more general notion.

For weighted timed games, we will be interested in the same three problems as in the previous section; the existence of a strategy whose outcomes are infinite runs remaining within given bounds. As in the previous section, we have the special cases of *existential* and *universal* problems.

3 Fixed-Point Characterization

Let us introduce some terminology. For the lower-bound problem, we say that an infinite path γ is *c-feasible* for some initial credit $c < +\infty$, if $p_c(\gamma') \geq 0$ for all finite prefixes γ' of γ, and that γ is *feasible* if it is c-feasible for some c.

Given a weighted game (S_1, S_2, s_0, T) and $b \in \mathbb{R}_{\geq 0}$, we define three predicates $L, W_b, U_b : S \times \mathbb{R}_{\geq 0} \to \{tt, \mathit{ff}\}$ to be the respective maximal fixed points to the following equations:

$$L(s,c) = \; 0 \leq c \wedge \begin{cases} s \in S_1 \implies \exists s \xrightarrow{p} s' \in T : L(s', c+p) \\ s \in S_2 \implies \forall s \xrightarrow{p} s' \in T : L(s', c+p) \end{cases}$$

$$W_b(s,c) = \; 0 \leq c \leq b \wedge \begin{cases} s \in S_1 \implies \exists s \xrightarrow{p} s' \in T : W_b(s', \max(b, c+p)) \\ s \in S_2 \implies \forall s \xrightarrow{p} s' \in T : W_b(s', \max(b, c+p)) \end{cases}$$

$$U_b(s,c) = \; 0 \leq c \leq b \wedge \begin{cases} s \in S_1 \implies \exists s \xrightarrow{p} s' \in T : U_b(s', c+p) \\ s \in S_2 \implies \forall s \xrightarrow{p} s' \in T : U_b(s', c+p) \; . \end{cases}$$

The right-hand side of each of these equations defines a monotone functional on the power set lattice of $S \times \mathbb{R}_{\geq 0}$, hence indeed the maximal fixed points exist. Also, if the weighted game under investigation is *image finite*, i.e. if the sets $\{(p, s') \mid s \xrightarrow{p} s' \in T\}$ are finite for all states s, then it can be shown that these respective functionals are continuous, implying that the maximal fixed points can be obtained as the limits of the iterated application of the respective functionals to the maximal element $S \times \mathbb{R}_{\geq 0}$ of the power set lattice. The proof of the lemma below is immediate from the definition of the predicates.

Lemma 2. *Let (S_1, S_2, s_0, T) be a weighted game and $s \in S_1 \cup S_2$, $b, c \in \mathbb{R}_{\geq 0}$. Then $(s,c) \in L$ (or $(s,c) \in W_b$, or $(s,c) \in U_b$) if and only if there exists a strategy for Player 1 such that any infinite path γ from s consistent with the strategy is c-feasible for the lower-bound problem (or lower-weak-upper-bound problem, or interval-bound problem, respectively).*

For the lower-bound problems, the above fixed-point characterization can be stated in a different way by defining recursively the *infimum credits* sufficient for feasibility—note that such a notion does not make sense for the interval-bound problem, as here the set of sufficient credits is not necessarily upward-closed. For a given weighted game (S_1, S_2, s_0, T) and $b \in \mathbb{R}_{\geq 0}$, let $L, W_b : S \to \mathbb{R} \cup \{\infty\}$ be the functions defined as respective minimal fixed points to the following equations:

$$L(s) = \begin{cases} \min\{L(s') - p \mid s \xrightarrow{p} s'\} & \text{if } s \in S_1 \\ \max\{L(s') - p \mid s \xrightarrow{p} s'\} & \text{if } s \in S_2 \end{cases}$$

$$W_b(s) = \begin{cases} \min\big(b, \min\{W(s') - p \mid s \xrightarrow{p} s'\}\big) & \text{if } s \in S_1 \\ \min\big(b, \max\{W(s') - p \mid s \xrightarrow{p} s'\}\big) & \text{if } s \in S_2 \; . \end{cases}$$

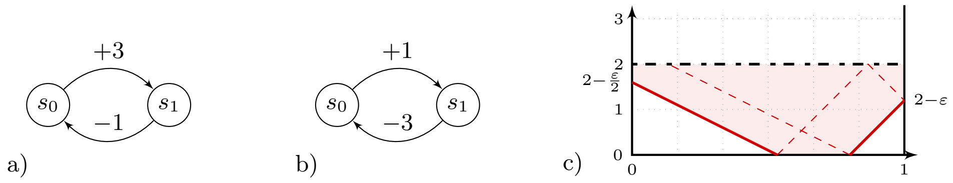


Fig. 2. a), b) Weighted automata with and without feasible paths. c) One fixed-point iteration on the weighted timed automaton of Fig. 1.

Proposition 3. *We have $L(s, c)$ if and only if $c \geq L(s)$, and $W_b(s, c)$ if and only if $W_b(s) \leq c \leq b$.*

The following lemma shows that for finite weighted games, there is a predescribed upper limit for the values of L and W_b. This implies that the fixed-point computations can be terminated after a finite number of iterations. For *timed* games, the second part of the next example below shows that this is *not* necessarily the case.

Lemma 4. *Let (S_1, S_2, s_0, T) be a finite weighted game and $b \in \mathbb{R}_{\geq 0}$. Let $M = \sum\{-p \mid s \xrightarrow{p} s' \in T, p < 0\}$. Then for any state s, $L(s) < \infty$ if and only if $L(s) \leq M$, and $W_b(s) < \infty$ if and only if $W_b(s) \leq M$.*

Example 5. Consider the weighted automata in Fig. 2a) and 2b). Let us attempt to compute L using iterative application of the recursive definition. For a) we find the following fixed points after two iterations: $L(s_0) = 0$ and $L(s_1) = 1$. For b) we get the following sequence of decreasing approximations: $L^{2n}(s_0) = L^{2n+1}(s_0) = 2n$ and $L^{2n}(s_1) = L^{2n-1}(s_1) = 2n+1$. Hence clearly $L(s_0) = L(s_1) = \infty$, though this fixed point is not reached within a finite number of iteration.

In Fig. 2c) we reconsider the weighted timed automaton from Fig. 1 under the interval-bound problem with upper bound 2. If we assume that after n iterations of the fixed-point computation, $U_2^n((\ell_2, 1), c) = tt$ if and only if $c \in [2-\varepsilon, 2]$ for some ε, then $U_2^{n+1}((\ell_0, 0), c) = tt$ if and only if $c \in [2 - \frac{1}{2}\varepsilon, 2]$. The largest fixed point—$U_2((\ell_2, 1), c) = tt$ if and only if $c = 2$—is only reached as the limit of this infinite sequence of approximations. □

srba: I don't quite understand what is depicted on the picture, can anybody add more explanation here?

4 Lower-Bound Problems

In this section we treat the lower-bound and lower-weak-upper-bound problems for finite automata, one-clock timed automata, and for finite games. For one-clock timed games, these problems are open.

4.1 Finite Weighted Automata

For a given finite weighted automaton $A = (S, s_0, T)$, we denote by $\mathsf{MinCr}(\gamma)$, for a path γ in A, the minimum $c \geq 0$ for which γ is c-feasible, and by $\mathsf{MinCr}(s)$, for a state $s \in S$, the minimum of $\mathsf{MinCr}(\gamma)$ over all feasible paths γ emerging from s.

A *cycle* $\gamma = s_0 \to s_1 \to \cdots \to s_0$ in A is *non-losing* if $p(\gamma) \geq 0$ (or equivalently, $p_c(\gamma) \geq c$ for any initial credit c). A *lasso* λ from a given state s_0 is an infinite path of the form $\gamma_1(\gamma_2)^\omega$, where $\gamma_1 = s_0 \to s_1 \to \cdots \to s_{i-1}$, $\gamma_2 = s_i \to \cdots \to s_k \to s_i$, and $s_{i-1} \to s_i$ are paths in A, and with $s_i \neq s_j$ whenever $i \neq j \leq k$. It is clear that if a lasso $\gamma_1(\gamma_2)^\omega$ constitutes a feasible path then the cycle γ_2 must be non-losing.

Lemma 6. *Let $A = (S, s_0, T)$ be a finite weighted automaton and $s \in S$. For any feasible path γ from s, there exists a feasible lasso λ also from s such that $\mathsf{MinCr}(\lambda) \leq \mathsf{MinCr}(\gamma)$.*

Proof. Let $c = \mathsf{MinCr}(\gamma)$, and assume first that there is a cycle π in γ such that $p(\pi) \geq 0$. Write $\gamma = \gamma_1\pi\gamma_2$, then $\lambda = \gamma_1\pi^\omega$ is a feasible lasso from s, and $\mathsf{MinCr}(\lambda) \leq \mathsf{MinCr}(\gamma)$.

Assume now that there is no cycle π in γ with $p(\pi) \geq 0$. We can write $\gamma = \gamma_1\gamma_2$, where γ_2 only takes transitions that appear infinitely often along γ. As there is no cycle with non-negative weight in γ, the weights of prefixes of γ_2 decrease to $-\infty$. Hence there is a prefix γ_2' of γ_2 such that $p(\gamma_2') < -p(\gamma_1)$. But then the accumulated weight of $\gamma_1\gamma_2'$, which is a prefix of γ, is negative, which contradicts the feasibility of γ. □

Thus, determining $L(s, c)$ corresponds to determining whether there is a feasible lasso λ out of s with $\mathsf{MinCr}(\lambda) \leq c$. This may be done in polynomial time in the size of the weighted automaton using a slightly modified Bellman-Ford algorithm. Hence, this allows us to solve the existential lower-bound and lower-weak-upper-bound problems in polynomial time. The corresponding universal problems can be solved using very similar techniques that we do not detail here (see the full version). Hence we get the following result.

Theorem 7. *For finite weighted automata, the existential and universal lower-bound and lower-weak-upper-bound problems are decidable in* P. *Also,* $\mathsf{MinCr}(s_0)$ *is computable in polynomial time.*

4.2 One-Clock Weighted Timed Automata

Let A be a one-clock weighted timed automaton. Without loss of generality we shall assume that for any location, the value of the (single) clock x is bounded by some constant M (see [3]). Let $0 \leq a_1 < a_2 < \ldots < a_n < a_{n+1} = M$ where $\{a_1, \ldots, a_n\}$ are the constants occurring in A. Then the one-dimensional *regions* of A are all elementary open intervals (a_i, a_{i+1}) and singletons $\{a_i\}$ (see [18]). In particular, two states (q, v) and (q, v') of A are time-abstract bisimilar whenever v and v' belong to the same region.

A corner-point region is a pair $\langle \rho, e \rangle$, where ρ is a region and e an end-point of ρ. We say that $\langle \rho', e' \rangle$ is the *successor* of $\langle \rho, e \rangle$ if either ρ' is the successor region of ρ and $e = e'$ or $\rho = \rho' = (a, b)$, $e = a$, and $e' = b$. Now the *corner-point* *abstraction* [3,6], $\mathsf{cpa}(A)$, of A is the finite weighted automaton with states $(q, \langle \rho, e\rangle)$ and with transitions $(q, \langle \rho, e\rangle) \to (q', \langle \rho', e'\rangle)$ if $\rho \models I(q)$, $\rho' \models I(q')$ and one of the following applies:

- $q = q'$ and $\langle \rho', e'\rangle$ is the successor of $\langle \rho, e\rangle$, or
- $(q, \varphi, \emptyset, q') \in E$ with $\rho \models \varphi$ and $\langle \rho', e'\rangle = \langle \rho, e\rangle$, or
- $(q, \varphi, \{x\}, q') \in E$ with $\rho \models \varphi$, $\rho' = \{0\}$ and $e' = 0$.

The weight of the first (delay) transition is $\mathit{rate}(q) \cdot (e' - e)$, the weights of the two last (discrete) transitions are 0. The above corner-point abstraction is sound and complete with respect to the lower-bound problem in the following sense (proof in the full version):

Proposition 8. *Let A be a one-clock weighted timed automaton.*

Completeness: *Let γ be an infinite run in A from $(q_0, 0)$ which is c-feasible for some $c < +\infty$. Then there exists a c-feasible infinite run γ' from $(q_0, \langle\{0\}, 0\rangle)$ in $\mathsf{cpa}(A)$.*

Soundness: *Let γ' be an infinite run in $\mathsf{cpa}(A)$ from $(q_0, \langle\{0\}, 0\rangle)$ which is c-feasible for some $c < +\infty$. Then, for any $\varepsilon > 0$, there exists a $(c+\varepsilon)$-feasible infinite run γ from $(q_0, 0)$ in A.*

Let us introduce the *lower-bound* (and lower-weak-upper-bound) *infimum problem* to be the problem as to whether for a given initial credit c, $L(s_0, c+\varepsilon)$ $(W_b(s_0, c+\varepsilon))$ holds for any $\varepsilon > 0$. Based on the above propositions we have:

Theorem 9. *For one-clock weighted timed automata, the existential and universal lower-bound and lower-weak-upper-bound infimum problems are decidable in P. Also, $\mathsf{MinCr}(s_0)$ may be computed in polynomial time.*

Proof (sketch). For the existential problem, note that for a given one-clock weighted timed automaton A, the size of $\mathsf{cpa}(A)$ is polynomial in the size of A, as the regions are constructed from the constants appearing in A. Then apply Theorem 7. For the universal problem, note that Proposition 8 can be modified to show that for any run in A there exist a run in $\mathsf{cpa}(A)$ which has always less remaining credit (within any region we simply choose to do the delay at the location with the smallest rate rather than the largest one). $\square$

4.3 Finite Weighted Games

Recall that a strategy σ is said to be *memoryless* if $\sigma(\gamma)$ only depends on $\mathsf{last}(\gamma)$ for any finite path γ. The proof of the following lemma can be found in the full version.

Lemma 10. *Let (S_1, S_2, s_0, T) be a finite weighted game, $c \in \mathbb{N}$ and $b \in \mathbb{N} \cup \{\infty\}$. If there exists a strategy σ for Player 1 which ensures that $p_{c\downarrow b}(\gamma') \geq 0$ for any finite prefix γ' of any infinite outcome γ of σ, then there is also a memoryless strategy with the same property. Symmetrically, if Player 2 has a strategy to ensure that for any outcome γ of σ, there is a finite prefix γ' of γ such that $p_{c\downarrow b}(\gamma') < 0$, then she has a memoryless strategy with the same property.*

Proposition 11. *For finite weighted games, the lower-bound and lower-weak-upper-bound problems are decidable in* $\mathsf{NP} \cap \mathsf{coNP}$.

Proof. The NP algorithm consists in nondeterministically guessing a memoryless strategy, pruning the transitions that are not selected by that strategy in G, and checking whether for any finite prefix γ of any infinite execution, we have $p_{c\downarrow b}(\gamma) \geq 0$, which is polynomial (Theorem 7). If Player 1 has a winning strategy, then this algorithm will answer positively.

The coNP algorithm follows from the determinacy of the game: if Player 1 has no winning strategy, then Player 2 has one, which can be chosen memoryless. It can then be guessed and checked in nondeterministic polynomial time. □

For the lower-bound problem, we can do better, and prove its equivalence with the mean-payoff problem.

Proposition 12. *The mean-payoff game problem is log-space equivalent to the lower-bound problem, and is log-space reducible to the lower-weak-upper-bound problem.*

Proof (sketch). Mean-payoff games are defined as follows [16]: given a weighted game $G = (S_1, S_2, s_0, T)$ and an integer m, is there a strategy for Player 1 s.t. for any infinite outcome $s_0 \xrightarrow{p_0} s_1 \xrightarrow{p_1} \cdots$, we have $\liminf_{n\to\infty} \sum_{j\leq n} p_j/n \geq m$. By shifting all weights by $-m$, we can simplify the problem by assuming $m = 0$. An important feature of mean-payoff games is that they admit memoryless winning strategies in case they admit winning strategies [14].

Transforming a mean-payoff problem into a lower-bound (or lower-weak-upper-bound) problem is easy: the mean weight of an infinite path is negative iff its accumulated weight is $-\infty$. Hence a winning strategy for Player 1 for the mean-payoff problem is also winning in the lower-bound (or lower-weak-upper-bound) problem, provided that the initial credit is sufficient (at least the opposite of the sum of all negative weights).

Conversely, given an instance $G = (S_1, S_2, s_0, T)$ of the lower-bound problem (assuming the initial credit is zero, by possibly adding an extra initial transition setting the initial credit), we construct an equivalent instance of the mean-payoff problem as follows: each transition $t = s \xrightarrow{p} s'$ is replaced by a new uncontrollable state s_t and three transitions $s \xrightarrow{p} s_t$, $s_t \xrightarrow{0} s'$ and $s_t \xrightarrow{0} s_0$. In other words, at each transition of the original game, Player 2 is given the opportunity to go back to the initial state. If the accumulated weight goes below zero at some point, Player 2 will decide to go back to the initial state. Since strategies can be assumed to be memoryless, there will be a periodic outcome with negative mean weight. □

The exact complexity of the mean-payoff problem is not known, but it is P-hard and in $\mathsf{UP} \cap \mathsf{coUP}$ [16]. Hence we get the following theorem.

Theorem 13. *For finite weighted games, the lower-bound problem is* P*-hard and in* $\mathsf{UP} \cap \mathsf{coUP}$ *(thus in* $\mathsf{NP} \cap \mathsf{coNP}$*). The lower-weak-upper-bound problem is* P*-hard and in* $\mathsf{NP} \cap \mathsf{coNP}$.

5 Interval-Bound Problems

We shall now study the interval-bound problems for the finite and timed cases.

5.1 Finite Weighted Automata

The following theorem summarizes all our results on the various interval-bound problems for finite weighted automata and games.

Theorem 14. *Under the interval-bound constraint,*

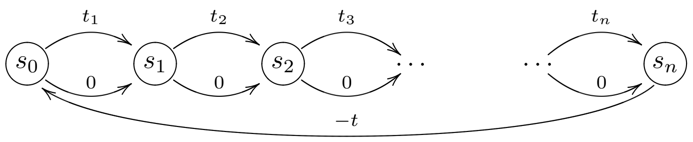
1. *the universal problem for finite weighted automata is decidable in* P*,*
2. *the existential problem for finite weighted automata is decidable in* PSPACE *and* NP*-hard, and*
3. *the problem for finite weighted games is* EXPTIME*-complete.*

The complete proofs of all the results mentioned in this theorem can be found in the full version. We only mention some ideas, and then give the complete NP-hardness proof, as we think this can be rather instructive.

1. Using Bellman-Ford-like algorithms, we detect whether there is a negative (resp. positive) cycle or a shortest (resp. longest) path whose cost becomes negative (resp. goes above the maximal bound of the interval). This can be done in polynomial time.
2. From a finite weighted automaton, we can construct a finite graph whose set of states is a pair (s, c) with s a state of the automaton, and c a cost value (in the interval $[0, b]$), and in which $(s, c) \to (s', c')$ is a transition whenever $s \xrightarrow{c'-c} s'$ is a transition of the weighted automaton. Guessing an infinite path in that graph gives a solution to the existential problem. This can be done in polynomial space (due to the encoding of b in binary). The NP-hardness proof will be detailed below.
3. Similarly as for the PSPACE upper bound, we construct the graph described above, and we add alternation: a state belonging to Player 1 (resp. 2) becomes existential (resp. universal). The interval-bound problem for finite weighted games reduces to an alternating graph accessibility problem (see [13]) which can be solved in P, hence the EXPTIME upper bound (the exponential blow-up is still due to the encoding of b in binary). EXPTIME-hardness is by reduction from the problem of countdown games which was very recently shown to be EXPTIME-complete [17].

Proof (of the NP*-hardness).* We reduce the NP-complete problem SUBSET-SUM (see *e.g.* [21]) into our existential interval-bound problem. An instance of SUBSET-SUM is a pair (A, t) where $A \subseteq \mathbb{N}$ is a finite set and $t \in \mathbb{N}$. The question is whether there is a subset of A which adds exactly to t. Assume a given instance of SUBSET-SUM (A, t) where $A = \{t_1, t_2, \ldots, t_n\}$. We construct a weighted automaton (S, s_0, T) where $S = \{s_0, s_1, \ldots, s_n\}$ and where $T = \{s_i \xrightarrow{t_{i+1}} s_{i+1} \mid$

$0 \leq i < n\} \cup \{s_i \xrightarrow{0} s_{i+1} \mid 0 \leq i < n\} \cup \{s_n \xrightarrow{-t} s_0\}$. The construction is depicted below.



Now consider the existential interval-bound problem with upper bound t, and let the initial credit be 0. It is clear that there is an infinite path with the accumulated weight staying between 0 and t if and only if the SUBSET-SUM problem has a solution. □

5.2 Timed Games

We prove in this section that the interval-bound problem for timed games is undecidable, even for games involving only one clock.

Theorem 15. *The interval-bound problem is undecidable for one-clock weighted timed games.*

Proof (sketch). This is achieved by simulating a two-counter machine. If c_1 and c_2 are the values of the counters, the accumulated weight along runs of the simulating weighted timed game will be $E(c_1, c_2) = 5 - 1/(2^{c_1}3^{c_2})$, and we will work with the upper bound of 5. We start with accumulated weight 4 (encoding that the two counters are initialized to 0).

We will describe modules that increment respectively decrement a counter, and then informally describe how we can test that the value of a counter is zero. We assume that there is a global invariant $x \leq 1$ in all the modules. Moreover, for the purpose of this reduction, a *configuration* is a triple (ℓ, x, W) where ℓ is a discrete location, x is the value of the clock, and W is the accumulated weight.

We first assume we have a module ok that is winning for Player 1 as soon as this module is entered with an accumulated weight in $[0, 5]$ (it is nothing more than a location, with weight-rate 0, and a controllable self-loop that checks whether $x = 1$ and resets clock x).

We now consider the module Mod_n that is described in Fig. 3 (n is an integer that will be fixed later in the set $\{2, 3, 12, 18\}$). A strategy in Mod_n for Player 1 which has the property that any outcome either reaches a module ok or exits the module while always satisfying the lower- and upper-bound conditions will be said to be *locally safe* in that module.

srba: I cannot see why ≤ 30 ? Shall not we just say that e is in the interval 0 to 1? – also later on in the text.

We assume that $0 \leq ne \leq 30$, and we consider the following strategy σ_n, depicted on Fig. 4:

– from location $(m, 0, 5 - e)$, it delays during $(5 - e)/6$ time units in m, then leaves to m_1 (thus to configuration $(m_1, (5 - e)/6, 0)$);
– from m_1, it directly goes to m_2;

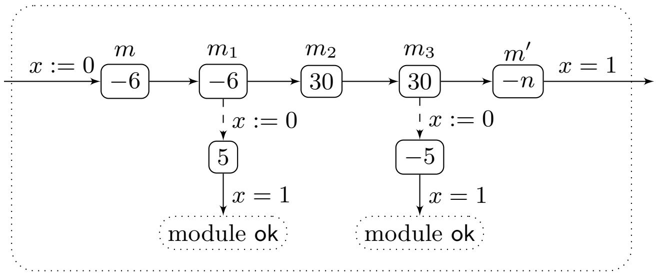


Fig. 3. Generic module Mod_n

– from $(m_2, (5-e)/6, 0)$, it waits for $1/6$ time units and then leaves to m_3 (hence to configuration $(m_3, 1-e/6, 5)$);
– from $(m_3, 1-e/6, 5)$, it directly goes to m';
– in m', it delays until $x = 1$, and fires the last transition to the next module.

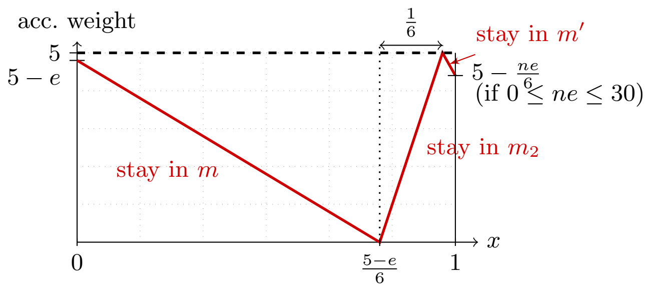


Fig. 4. The effect of (strategy σ_n in) module Mod_n

In Mod_n, from locations m_1 (resp. m_3), Player 2 can decide to bring the game to the first (resp. second) vertical branch, and the only way to stay safe is to arrive in m_1 (resp. m_3) with accumulated cost 0 (resp. 5) and to leave in 0-delay. This is precisely what does strategy σ_n, which is then the unique locally safe strategy in module Mod_n.

Proposition 16. *Starting from configuration $(m, 0, 5-e)$ in Mod_n, with $0 \leq ne \leq 30$, Player 1 has a unique locally safe strategy. Under that strategy, the only outcome that visits m' exits the module with accumulated weight $5 - ne/6$.*

We define the following shorthands: $\mathsf{Inc}(c_1) = \mathsf{Mod}_3$, $\mathsf{Inc}(c_2) = \mathsf{Mod}_2$, $\mathsf{Dec}(c_1) = \mathsf{Mod}_{12}$, and $\mathsf{Dec}(c_2) = \mathsf{Mod}_{18}$. When starting with accumulated weight $5-e$ in module $\mathsf{Inc}(c_1)$, the accumulated weight when leaving this module is $5 - e/2$, hence $\mathsf{Inc}(c_1)$ increments the counter c_1. Similarly, $\mathsf{Inc}(c_2)$ increments c_2, and $\mathsf{Dec}(c_1)$ and $\mathsf{Dec}(c_2)$ decrement the respective counters. In the last two modules, no check is performed whether the counter to be decremented actually has a positive value.

We briefly describe how such a test can be done. First we construct a module that has two output branches, one where the two counters are equal to 0, and one where one of the counters is positive. This is to Player 1 to decide in which branch we go, but Player 2 can easily check that this is correct (checking that the two counters are equal to zero reduces to checking that the accumulated weight is 4, whereas checking that one of the counters is positive reduces to checking that the accumulated cost be in the open interval $(4, 5)$). Then, using that module, we can for instance implement a test that counter c_2 is equal to zero as follows: either both counters are equal to zero, or under the assumption that one of the counters is positive, and by decrementing the first counter, we will eventually reach a point where both counters are equal to zero. For lack of space, we do not detail the construction of those modules, which use ideas quite similar to Mod_n.

Hence, fixing $\mathcal{M}$ to be a two-counter machine, we can construct a one-clock weighted timed game $G_{\mathcal{M}}$ which is a positive instance of the interval-bound problem if and only if $\mathcal{M}$ has an infinite computation. □

6 Conclusion

A summary of the results proved in this paper is provided in the following table. The fields in gray remain open. Matching the complexity lower and upper bounds for some of the problems is left open: the lower-bound problems for finite games are strongly related to the well known open problem of complexity of mean-payoff games; closing the gap between NP-hardness and containment in PSPACE for the existential interval-bound problem seems intricate and it is a part of our future work.

	games		**existential problem**		**universal problem**	
	finite	**1-clock**	**finite**	**1-clock**	**finite**	**1-clock**
L	$\in$ UP $\cap$ coUP P-h (Th. 13)	▨	$\in$ P (Th. 7)	$\in$ P (Th. 9)	$\in$ P (Th. 7)	$\in$ P (Th. 9)
L+W	$\in$ NP $\cap$ coNP P-h (Th. 13)	▨	$\in$ P (Th. 7)	$\in$ P (Th. 9)	$\in$ P (Th. 7)	$\in$ P (Th. 9)
L+U	EXPTIME-c (Th. 14)	Undec. (Th. 15)	$\in$ PSPACE NP-h (Th. 14)	▨	$\in$ P (Th. 14)	▨

Note that the results related to Theorem 9 hold for the initial credits arbitrarily close (for any given $\varepsilon > 0$) to the given ones.

References

1. R. Alur, M. Bernadsky, and P. Madhusudan. Optimal reachability in weighted timed games. In *Proc. of ICALP'04*, LNCS 3142, p. 122–133. Springer, 2004.
2. R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. In *Proc. 4th Int. Workshop Hybrid Systems: Computation and Control (HSCC'01)*, LNCS 2034, p. 49–62. Springer, 2001.
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 *Proc. 4th Int. Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, LNCS 2034, p. 147–161. Springer, 2001.

4. P. Bouyer, Th. Brihaye, V. Bruyère, and J.-F. Raskin. On the optimal reachability problem. *Formal Methods in System Design*, 31(2):135–175, 2007.
5. P. Bouyer, Th. Brihaye, and N. Markey. Improved undecidability results on weighted timed automata. *Inf. Proc. Letters*, 98(5):188–194, 2006.
6. P. Bouyer, E. Brinksma, and K. G. Larsen. Staying alive as cheaply as possible. In *Proc. 7th Int. Workshop Hybrid Systems: Computation and Control (HSCC'04)*, LNCS 2993, p. 203–218. Springer, 2004.
7. P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *Proc. of FSTTCS'04*, LNCS 3328, p. 148–160. Springer, 2004.
8. P. Bouyer, K. G. Larsen, and N. Markey. Model-checking one-clock priced timed automata. In *Proc. 10th Int. Conf. Foundations of Software Science and Computation Structures (FoSSaCS'07)*, LNCS 4423, p. 108–122. Springer, 2007.
9. P. Bouyer, K. G. Larsen, N. Markey, and J. I. Rasmussen. Almost optimal strategies in one-clock priced timed automata. In *Proc. of (FSTTCS'06)*, LNCS 4337, p. 345–356. Springer, 2006.
10. P. Bouyer and N. Markey. Costs are expensive! In *Proc. of FORMATS'07*, LNCS 4763, p. 53–68. Springer, 2007.
11. Th. Brihaye, V. Bruyère, and J.-F. Raskin. Model-checking for weighted timed automata. In *Proc. Joint Conf. Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, LNCS 3253, p. 277–292. Springer, 2004.
12. Th. Brihaye, V. Bruyère, and J.-F. Raskin. On optimal timed strategies. In *Proc. of (FORMATS'05)*, LNCS 3821, p. 49–64. Springer, 2005.
13. A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.
14. A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *Int. J. Game Theory*, 8(2):109–113, 1979.
15. U. Fahrenberg and K. G. Larsen. Discount-optimal infinite runs in priced timed automata. Submitted, 2008.
16. M. Jurdziński. Deciding the winner in parity games is in UP$\cap$co-UP. *Inf. Proc. Letters*, 68(3):119–124, 1998.
17. M. Jurdziński, F. Laroussinie, and J. Sproston. Model checking probabilistic timed automata with one or two clocks. In *Proc. of TACAS'07)*, LNCS 4424, p. 170–184. Springer, 2007.
18. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model checking timed automata with one or two clocks. In *Proc. 15th Int. Conf. Concurrency Theory (CONCUR'04)*, LNCS 3170, p. 387–401. Springer, 2004.
19. K. G. Larsen and J. I. Rasmussen. Optimal conditional scheduling for multi-priced timed automata. In *Proc. of FOSSACS'05*, LNCS 3441, p. 234–249. Springer, 2005.
20. D. Martin. Borel determinacy. *Annals Math.*, 102(2):363–371, 1975.
21. Ch. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

7 Appendix

7.1 Proofs of Section 3

Lemma 4. *Let (S_1, S_2, s_0, T) be a finite weighted game and $b \in \mathbb{R}_{\geq 0}$. Let $M = \sum\{-p \mid s \xrightarrow{p} s' \in T, p < 0\}$. Then for any state s, $L(s) < \infty$ if and only if $L(s) \leq M$, and $W_b(s) < \infty$ if and only if $W_b(s) \leq M$.*

Proof. We prove the lemma for the lower-bound case; the proof for the case with weak upper bound is similar. Suppose that $L(s) < \infty$, but $L(s) > M$. Let σ be a strategy for Player 1 which realizes $L(s)$ from s, and let γ be an outcome of σ under an optimal strategy for Player 2.

There must be a state for which the remaining credit is 0 in γ, otherwise the strategy of Player 2 would not be optimal. Let s_2 be the first such state, and let $s_1 \xrightarrow{p_1} s_2$ be the transition in γ leading to that occurrence of s_2. Then $p_1 < 0$.

On the finite prefix of γ leading from s with initial credit $L(s)$ to s_2 with remaining credit 0, weight has dropped by more than M, hence at least one transition with negative weight has been taken twice. Let $s_3 \xrightarrow{p_3} s_4$ be one such transition, and let c_3 be the remaining credit at the first occurrence of s_3 as part of this transition, c_3' the remaining credit at the second occurrence.

If $c_3' \leq c_3$, we can discard the cycle we just found: In case $s_4 \in S_1$, we can delete the cycle beginning and ending at $s_3 \xrightarrow{p} s_4$ from the strategy σ and arrive at a strategy which succeeds from s with initial credit not greater than $L(s)$. If $s_4 \in S_2$ instead, Player 2 can add another occurrence of the cycle to her strategy without losing optimality. Hence for $c_3' = c_3$, we arrive at a setting which is equally good for both players, and for $c_3' < c_3$ we are in contradiction to $L(s)$ being the minimum credit necessary from s.

The above argument specialises to the transition $s_1 \xrightarrow{p_1} s_2$ leading to a remaining credit of 0: if this edge also occurs in γ before, we have a cycle like the one above. Hence we can conclude that there must be a cycle in the finite prefix of γ leading to s_1 with remaining credit $-p_1$, *i.e. before* the remaining credit drops to 0.

We are left with considering the case $c_3' > c_3$. However as the remaining credit does not drop to 0 during the cycle, we can use an argument as above and either repeat the cycle (in case $s_4 \in S_1$) or delete it (in case $s_4 \in S_2$), arriving at a contradiction in both cases. $\square$

7.2 Proofs of Section 4.1

Theorem 7. *For finite weighted automata, the existential and universal lower-bound and lower-weak-upper-bound problems are decidable in* P. *Also,* $\mathsf{MinCr}(s_0)$ *is computable in polynomial time.*

Algorithm 1 Modified Bellman-Ford Algorithm for Detecting Feasible Lassoes

Input: A finite weighted automaton (S, s_0, T) and initial credit c_0.

Output: TRUE if a feasible lasso from s_0 with the credit c_0 exists, FALSE otherwise.

```
1: procedure INIT($s_0, c_0$);
2: for all $s \in S$ do
3:     $C(s) := -\infty$
4: end for
5: $C(s_0) := c_0$; $b$ :=FALSE;
6:
7: procedure ROUND($s_0$);
8: for $i := 1$ to $|S| - 1$ do
9:     for each edge $s \xrightarrow{p} s'$ in $T$ do
10:        if $C(s') \leq C(s) + p$ and $C(s) + p \geq 0$ then
11:            $C(s') := C(s) + p$
12:            if $s' = s_0$ then
13:                $b$ :=TRUE
14:            end if
15:        end if
16:    end for
17: end for
18:
19: function MAXRMCR($s_0, c_0$)
20: INIT($s_0, c_0$); ROUND($s_0$); $C' := C$; ROUND($s_0$);
21: for all $s \in S$ do
22:     if $C(s) > C'(s)$ then
23:         $C(s) := +\infty$
24:     end if
25: end for
26: return $(C, b)$
27:
28: $(C_0, b_0)$ :=MAXRMCR($s_0, c_0$)
29: if $C_0(s) = \infty$ for some $s \in S$ then
30:     return TRUE
31: else
32:     for all $s \in S$ such that $C_0(s) \neq -\infty$ do
33:         $(C_s, b_s)$ :=MAXRMCR($s, C_0(s)$)
34:         if $C_s(s) \geq C_0(s)$ and $b_s$ =TRUE then
35:             return TRUE
36:         end if
37:     end for
38:     return FALSE
39: end if
```

Proof. We first focus on the existential problems.

We shall use Algorithm 1 where C and C' are global hashes that for every location store an integer value (possibly also $-\infty$ or ∞) and b is a global boolean variable.

The procedure INIT initializes the current weights in the hash C to $-\infty$ for all locations except for s_0 which is initialized to the initial credit c_0. The bit b, specifying that the initial location got updated during the relaxation of edges, is set to FALSE.

Procedure ROUND performs $(|S|-1)$ times the inner loop where every edge is relaxed. Possible improvement, which however does not go below zero (the condition on line 10), is performed, and if the initial node s_0 was improved, the bit b is set to TRUE. This means that the procedure ROUND computes for each location s the maximal remaining credit $C(s)$ with which this location can be reached without the accumulated cost dropping below zero, provided that the location s is not on any positive loop.

Detecting all nodes that are on a positive loop is performed in the function MAXRMCR, which calls the procedure ROUND twice and all locations that were improved by the second call get the value $+\infty$.

The main code first runs MAXRMCR for the initial location s_0 and the initial credit c_0 and remembers the computed hash C in C_0. Should any node be marked by $+\infty$ then we terminate because there is surely a feasible lasso in the automaton. Otherwise we consider every node s with the computed maximal achievable credit $C_0(s) \geq 0$ as a candidate for the first node forming the infinite loop in the lasso. This is verified on line 33 by calling MAXRMCR with the argument s and the initial credit $C_0(s)$ and checking whether by doing at least one step (the bit b_s being TRUE) the location s can reach itself without losing the credit it has. If this is the case for some s then a feasible lasso is found and we return TRUE, otherwise we return FALSE.

So for given a s_0 and an initial credit c_0 we can in polynomial time answer the existence of a feasible lasso and this proves the first part of the theorem.

To find the smallest initial credit c_0 for which there is a feasible lasso, we use Lemma 4 which gives us the upper bound M on c_0. Now by using binary search, we have to make at most $O(\log M)$ queries, *i.e.*, polynomially many times run the algorithm described above.

For the lower-weak-upper-bound problem, we have to make sure that the weights in the hash C never get above a given weak upper bound. This means a simple modification on line 11 of the algorithm where in the assignment we assign the minimum of $C(s)+p$ and the given weak upper bound.

We now turn to the universal problems. The dual problem to the lower-bound problem (*i.e.*, exhibiting a finite path with negative accumulated cost) can be solved in deterministic polynomial time by applying the Bellman-Ford shortest path algorithm. Hence the polynomial time algorithm for the lower-bound problem.

Similarly, the (dual problem to the) lower-weak-upper-bound problem is solved in deterministic polynomial time by adapting the Bellman-Ford algorithm as follows: whenever the weight of some node should be updated to a larger value than is the given weak upper bound b, we update it only to b. $\square$

7.3 Proofs of Section 4.2

Proposition 8. *Let A be a one-clock weighted timed automaton.*

Completeness: *Let γ be an infinite run in A from $(q_0, 0)$ which is c-feasible for some $c < +\infty$. Then there exists a c-feasible infinite run γ' from $(q_0, \langle\{0\}, 0\rangle)$ in $\mathsf{cpa}(A)$.*

Soundness: *Let γ' be an infinite run in $\mathsf{cpa}(A)$ from $(q_0, \langle\{0\}, 0\rangle)$ which is c-feasible for some $c < +\infty$. Then, for any $\varepsilon > 0$, there exists a $(c + \varepsilon)$-feasible infinite run γ from $(q_0, 0)$ in A.*

Proof. Completeness: γ' is obtained from γ by essentially pushing delay transitions within the same region $\rho = (a, b)$ towards the most profitable end-point. Consider some maximal sub-sequence of γ of alternating delay and switch transitions within ρ:

$$\gamma : \cdots \to (q_1, v_1) \to_{d_1} (q_1, v_2) \to_{e_1} (q_2, v_2) \to_{d_2} (q_2, v_3) \to_{e_2} \cdots \\ \to_{d_n} (q_n, v_{n+1}) \to \cdots$$

where e_i are non-resetting switch transitions, and $a \leq v_1 \leq v_2 \leq \cdots \leq v_{n+1} \leq b$. Now with $j \in \{1, \ldots, n\}$ maximizing the rate $\mathit{rate}(q_j)$, we construct the following sub-sequence of γ':

$$\gamma' : \cdots \to (q_1, \langle\rho, a\rangle) \to (q_2, \langle\rho, a\rangle) \to \cdots \to \\ (q_j, \langle\rho, a\rangle) \to (q_j, \langle\rho, b\rangle) \to (q_{j+1}, \langle\rho, b\rangle) \to \cdots \to (q_n, \langle\rho, b\rangle) \to \cdots$$

Given an initial credit, it is easy to see that the remaining credit at a state in γ' is always greater than or equal to the remaining credits at the corresponding states in γ.

Soundness: Let K be the maximum of the absolute values of weight-rates of A. If $(q, \langle\{a\}, a\rangle) \to (q, \langle(a, b), a\rangle)$ is the n-th transition in γ', the corresponding n-th transition of γ is taken to be $(q, a) \to (q, a + \varepsilon(2^{n+1}K)^{-1})$. Similarly, if $(q, \langle(a, b), a\rangle) \to (q, \langle(a, b), b\rangle)$ is the n-th transition of γ', the corresponding n-th transition of γ' is taken to be $(q, a + \varepsilon(2^{n+1}K)^{-1}) \to (q, b - \varepsilon(2^{n+1}K)^{-1})$. □

7.4 Proofs of Section 4.3

Lemma 10. *Let (S_1, S_2, s_0, T) be a finite weighted game, $c \in \mathbb{N}$ and $b \in \mathbb{N} \cup \{\infty\}$. If there exists a strategy σ for Player 1 which ensures that $p_{c\downarrow b}(\gamma') \geq 0$ for any finite prefix γ' of any infinite outcome γ of σ, then there is also a memoryless strategy with the same property. Symmetrically, if Player 2 has a strategy to ensure that for any outcome γ of σ, there is a finite prefix γ' of γ such that $p_{c\downarrow b}(\gamma') < 0$, then she has a memoryless strategy with the same property.*

Proof. Let $s \in S_1$. Assuming s can be reached by applying σ, we let C_s be the set of integers c' s.t. there is an outcome of σ starting in s_0 with credit c and reaching s with credit c'. Since σ is winning, $c_s = \min C_s$ is non-negative. When s is reachable under σ, we let γ_s be an outcome of σ that reaches s with credit c_s, and we define the memoryless strategy $\widetilde{\sigma}(s) = \sigma(\gamma_s)$. When s is not reachable under σ, the value of $\widetilde{\sigma}(s)$ is irrelevant. It can be easily proved that along every outcome γ' of $\widetilde{\sigma}$, the accumulated cost in a state s is larger than or equal to c_s. The proof for the second claim is similar. □

Proposition 12. *The mean-payoff game problem is log-space equivalent to the lower-bound problem, and is log-space reducible to the lower-weak-upper-bound problem.*

Proof. Mean-payoff games are defined as follows [16]: given a weighted game $G = (S_1, S_2, s_0, T)$ and an integer threshold m, is there a strategy for Player 1 such that for any infinite outcome $s_0 \xrightarrow{p_0} s_1 \xrightarrow{p_1} \cdots$, we have $\liminf_{n\to\infty} \sum_{j\leq n} p_j/n \geq m$. Shifting all weights by $-m$, we can simplify the problem by assuming $m = 0$.

Let (G, m) be an instance of the mean-payoff problem. Consider the lower-bound problem played on the same weighted game G, with initial credit $M = \sum\{-p \mid s \xrightarrow{p} s' \in T, p < 0\}$.

Assume Player 1 has a winning strategy σ for the mean-payoff problem (G, m). Following [14], we may assume that σ is memoryless, and that the weight of any cycle in any outcome of σ is nonnegative. Assume that the same strategy σ is not winning for the lower-bound (resp. lower-weak-upper bound) problem: there is an outcome that eventually reaches a point where the accumulated weight is negative. Consider one of the outcomes, γ, that reaches this negative accumulated weight with the least number of transitions. Since the initial credit is M, it must be the case that a state is visited twice before reaching the negative weight. Along that cycle, the accumulated weight must be positive. Since σ is memoryless, removing this cycle yields another outcome of σ, which reaches a negative accumulated weight with fewer transitions than γ, which is a contradiction.

Conversely, given a winning strategy for Player 1 for the lower-bound problem, the accumulated weight along any outcome remains nonnegative. The sum of the weights of the transitions along any prefix is then larger than or equal to $-M$, hence the mean payoff for a prefix of length n is larger than or equal to $-M/n$, which converges to 0 when n goes to infinity.

We now present the converse reduction, starting from a weighted game $G = (S_1, S_2, s_0, T)$ and an initial credit c_0 for the lower-bound problem. Even if it means adding an extra initial transition of weight c_0, we may assume that the initial credit is zero. We construct a weighted game G' from G by giving a way to Player 2 to "reset" the game at any transition. Formally, $G' = (S_1, S_2 \cup S_T, s_0, T')$ with $S_T = \{s_t \mid t \in T\}$ and $T' = T \cup \{(s, p, s_t), (s_t, 0, s'), (s_t, 0, s_0) \mid t = (s, p, s') \in T\}$. We claim that Player 1 has a winning strategy in G for the lower bound problem iff she has a winning strategy in G' for the mean-payoff problem (with threshold 0).

Assume first that there is a winning strategy σ in G for the lower bound problem, which we may require to be memoryless. From σ, we define a strategy σ' in G' in the obvious way: if $\sigma(s) = t$ for $s \in S_1$, then $\sigma'(s) = (s, p, s_t)$. Consider an outcome γ' of σ'. If it contains infinitely many "reset" transitions from Player 2, then since σ is winning, the accumulated weight from s_0 to the next "reset" transition is nonnegative, and so is the mean payoff along γ'. If Player 2 plays only finitely many "reset" transitions, then there is a suffix π' of γ' that corresponds to an outcome of σ in G. Hence the accumulated weight is nonnegative along π', and so is the mean payoff along γ'.

Now, if there is a winning strategy σ' in G' for the mean payoff problem, which we require to be memoryless, then this strategy gives rise to a strategy σ in G in the obvious way. If this strategy were not winning in G, then some finite prefix of an outcome would have negative accumulated weight. After the corresponding prefix played in G', Player 2 could play its "reset" transitions back to s_0. This yields a negative cycle, entailing that σ' is not winning in G'. □

7.5 Proofs of Section 5.1

Theorem 14. *Under the interval-bound constraint,*

1. *the universal problem for finite weighted automata is decidable in* P*,*
2. *the existential problem for finite weighted automata is decidable in* PSPACE *and* NP*-hard, and*
3. *the problem for finite weighted games is* EXPTIME*-complete.*

We decompose the proof of this theorem into several lemmas.

Lemma 17. *The interval-bound problem for finite weighted games is* EXPTIME*-hard.*

Proof. We reduce the EXPTIME-complete problem of Countdown Games [17] into our interval-bound problem. An instance of a countdown game is a pair (S, E) where S is a finite set of states and $E \subseteq S \times (\mathbb{N} \setminus \{0\}) \times S$ is a finite transition relation. A configuration of the game is a pair $(s, c) \in S \times \mathbb{N}$. The game is between two players that change the current configuration (s, c) into (s', c') according to the following rule: first Player 1 chooses a number $0 < d \leq c$ for which there is at least one transition $(s, d, s'') \in E$, then Player 2 chooses one of the states s' for which $(s, d, s') \in E$; now the game continues from a new current configuration $(s', c - d)$. If the game reaches a configuration $(s, 0)$, Player 1 is the winner; if the game reaches a configuration (s, c) with $c > 0$ and no possible moves, Player 2 is the winner. The question is to decide whether Player 1 has a winning strategy in the countdown game starting from a given configuration (s_0, c_0).

Given a countdown game (S, E) with initial configuration (s_0, c_0), we construct a weighted game as follows: let $S_1 = S$, $S_2 = \{(s, d) \mid (s, d, s') \in E\}$, and

$$T = \big\{s \xrightarrow{d} (s, d) \;\big|\; (s, d) \in S_2\big\} \cup \big\{(s, d) \xrightarrow{0} s' \;\big|\; (s, d, s') \in E\big\} \cup \big\{s \xrightarrow{-c_0} s_0 \;\big|\; s \in S\big\}$$

The upper bound is set to c_0. Player 1 can now from a state $s \in S$ choose a particular number d and Player 2 from the temporary state (s, d) choose a transition to a state $s' \in S$. The number d is added to the accumulated weight and the same repeats. As the accumulated weight is bounded by c_0, Player 1 has to eventually take some transition labeled $-c_0$ and return to the initial state s_0. In order not to drop below zero, this is only possible if the accumulated weight is exactly c_0, hence the first player in the countdown game has a winning strategy if and only if Player 1 has a winning strategy in the interval-bound game. □

For the rest of this subsection we will need the definition of a P-complete Alternating Graph Accessibility Problem (AGAP) [13]. An instance of AGAP is a directed graph $G = (V, E)$ together with $s, t \in V$ such that $E \subseteq V \times V$ and V is partitioned into two disjoint sets V_1 and V_2. The question is whether the predicate $\mathsf{apath}(s, t)$ is true, where $\mathsf{apath}(x, y)$ holds if and only if

- $x = y$, or
- $x \in V_1$ and there is $z \in V$ such that $(x, z) \in E$ and $\mathsf{apath}(z, y)$ holds, or
- $x \in V_2$ and for all $z \in V$ such that $(x, z) \in E$ the predicate $\mathsf{apath}(z, y)$ holds.

We shall provide a translation of the interval-bound problem for finite weighted games into several AGAP questions. Let (S_1, S_2, s_0, T) be a finite weighted game and let b be a given upper bound. We construct a directed graph $G = (V = V_1 \cup V_2, E)$ such that $V_1 = \{(s, c) \mid s \in S_1, c \in \{0, 1, \dots, b\}\}$, $V_2 = \{(s, c) \mid s \in S_2, c \in \{0, 1, \dots, b\}\} \cup \{\mathit{sink}\}$ and

$$(s, c) \to (s', c') \in E \quad \text{whenever} \quad s \xrightarrow{c'-c} s' \in T$$

and

$$(s, c) \to \mathit{sink} \in E \quad \text{whenever} \quad s \xrightarrow{c'} s' \in T \text{ s.t. } (c + c') \notin [0, b]$$

and

$$\mathit{sink} \to \mathit{sink} \in E\ .$$

This means that as long as the accumulated weight stays within the given interval $[0, b]$, it is simply recorded in the second component of the state. Should the accumulated weight reach beyond of the given interval, we enter a special state *sink* with a self-loop.

Notice also that the reduction produces a graph of size exponential in the size of the given weighted game, and that the blow-up depends only on the bound b. Nevertheless, the graph G can help us to introduce the following complexity upper-bounds.

Lemma 18. *The interval-bound problem for finite weighted games is decidable in* EXPTIME. *The existential interval-bound problem for finite automata is decidable in* PSPACE.

Proof. Notice that the graph G contains exactly those paths that are between the lower bound 0 and the upper bound b (with the only exception when the state *sink* is entered). Assume a given initial location s_0 with initial credit c_0. We can now for all possible pairs (s, c), where $s \in S_1 \cup S_2$ and $c \in \{0, 1, \ldots, b\}$, ask two AGAP questions: $\mathsf{apath}((s_0, c_0), (s, c))$ and $\mathsf{apath}((s, c), (s, c))$ where in the second question we require that at least one step was taken before the loop closes. If the answer to both questions is positive then we know that Player 1 has a strategy in the weighted game to stay within the given interval. If for none of the pairs (s, c) this happens then we know that the first player does not have such strategy. As the AGAP questions can be answered in time polynomial in the size of G, and because these questions are asked only exponentially many times, we conclude that the interval-bound problem for finite weighted games is in EXPTIME.

Now if we are in the existential automata setting, we can guess the pair (s, c) and then nondeterministically verify whether there is a path between $(s_0, 0)$ and (s, c) and a nontrivial path between (s, c) and (s, c). This gives a PSPACE algorithm. □

Lemma 19. *The universal interval-bound problem for finite weighted automata is decidable in* P.

Proof. From a given initial location s_0, we can run *e.g.* the Bellman-Ford shortest path algorithm. Either a negative cycle or a location with negative shortest path is discovered, in which case there is a path which does not adhere to the lower bound (additionally it might violate the upper bound). If none of these are discovered, we run the Bellman-Ford algorithm another time, this time with max instead of min, to find the longest path to every other location. Again, if there is a positive cycle or a location with a longest path larger then the upper bound b, we have a path violating the upper-bound. Otherwise we know that all paths are within the required bounds. □

7.6 Proofs of Section 5.2

Theorem 15. *The interval-bound problem is undecidable for one-clock weighted timed games.*

This is achieved by simulating a two-counter machine. If c_1 and c_2 are the values of the counters, the accumulated weight along runs of the simulating weighted timed game will be

$$E(c_1, c_2) = 5 - \frac{1}{2^{c_1} 3^{c_2}} ,$$

and we will work with an upper bound of 5. We start with accumulated weight 4 (encoding that the two counters are initialized to 0).

The instructions of the two-counter machine will be encoded as modules which we describe below. All our modules have invariant "$x \leq 1$". For the purpose of this section, a *configuration* is a triple (ℓ, x, W) where ℓ is a discrete location, x is the value of the clock, and W is the accumulated weight.

***Module* ok.** Our first module is the "accepting" (ok) module, which is depicted on Fig. 5. Clearly enough, entering module ok with accumulated weight in $[0,5]$ is winning.

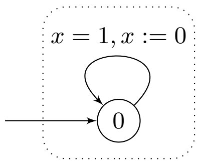


Fig. 5. Module ok

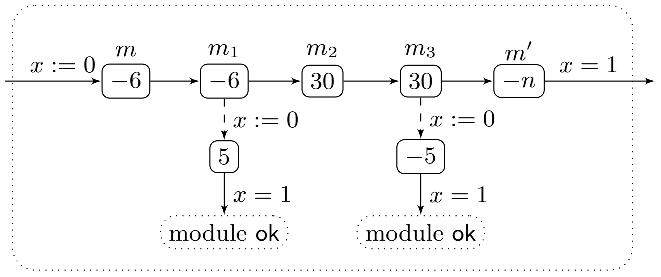


Fig. 6. Generic module Mod_n

A strategy for Player 1 which is defined on a module and has the property that any outcome either reaches a module ok or exits the module while always satisfying the lower- and upper-bound conditions will be said to be *locally safe* in that module.

***Generic module* Mod_n** Consider the generic module Mod_n, as depicted on Figure 6, for some given value of n (which will be fixed later among $\{2, 3, 12, 18\}$), and assume that $0 \leq ne \leq 30$. Consider the following strategy σ_n, depicted on Fig. 7:

– from location $(m, 0, 5 - e)$, it delays during $(5 - e)/6$ time units in m, then leaves to m_1 (thus to configuration $(m_1, (5 - e)/6, 0)$);
– from m_1, it directly goes to m_2;
– from $(m_2, (5 - e)/6, 0)$, it waits for $1/6$ time units and then leaves to m_3 (hence to configuration $(m_3, 1 - e/6, 5)$);
– from $(m_3, 1 - e/6, 5)$, it directly goes to m';
– in m', it delays until $x = 1$, and fires the last transition to the next module.

The only memory that is needed to define strategy σ_n is the accumulated weight so far.

Lemma 20. *For $0 \leq ne \leq 30$, the strategy σ_n is locally safe in Mod_n and has the following three outcomes, all of which adhere to the lower and upper bound conditions:*

– *one where Player 2 takes her move down from location m_1;*
– *one where Player 2 takes her move down from location m_3;*
– *one that leaves the module to the right.*

Moreover, any other strategy for Player 1 is not locally safe.

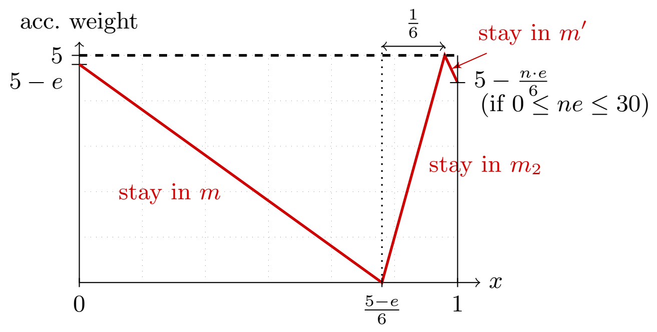


Fig. 7. The effect of module Mod_n

Proof. It is clear that σ_n satisfies the properties above. To prove the last claim, we exhaustively list the possible other strategies:

- The maximal delay that can be waited in m before violating the lower-bound constraint is $(5-e)/6$ time units. Assume that we wait strictly less than $(5-e)/6$ time units in m before leaving to m_1; then the accumulated weight when reaching m_1 is positive, and the second player can immediately go down and increase it above 5.
- If the strategy waits for $(5-e)/6$ time units in m before reaching m_1, then waiting any positive delay in m_1 will also take the accumulated weight above 5.
- The cases of m_2 and m_3 are similar to the cases of m and m_1, respectively (but instead, the lower bound will be violated).
- Last, waiting until $x = 1$ is the only possible move in m'. □

We have shown the following:

Proposition 21. *Starting from configuration* $(m, 0, 5-e)$ *in* Mod_n*, with* $0 \leq ne \leq 30$*, Player* 1 *has a unique locally safe strategy. Under that strategy, the only outcome that visits* m' *exits the module with accumulated weight* $5 - ne/6$.

In the sequel, we consider the modules $\mathsf{Inc}(c_1) = \mathsf{Mod}_3$, $\mathsf{Inc}(c_2) = \mathsf{Mod}_2$, $\mathsf{Dec}(c_1) = \mathsf{Mod}_{12}$, and $\mathsf{Dec}(c_2) = \mathsf{Mod}_{18}$. Note that when starting with accumulated weight $5-e$ in module $\mathsf{Inc}(c_1)$, the accumulated weight when leaving this module is $5-e/2$, hence $\mathsf{Inc}(c_1)$ increments the counter c_1. Similarly, $\mathsf{Inc}(c_2)$ increments c_2, and $\mathsf{Dec}(c_1)$ and $\mathsf{Dec}(c_2)$ decrement the respective counters. In the last two modules, no check is performed whether the counter to be decremented actually has a positive value. We now describe modules that will test the values of the counters (to implement an instruction of the form "`if` $c_1 = 0$ `then goto` p_1 `else goto` p_2"), and for the global reduction we will assume that a decrementing instruction is always preceded by such an instruction.

***Module for checking* $\mathbf{c_1 = c_2 = 0}$.** As a first step, we construct a module implementing the instruction "`if` $c_1 = c_2 = 0$ `then goto` p_1 `else goto` p_2" where p_1 and p_2 are instructions of the two-counter machine. This module is depicted on Fig. 8.

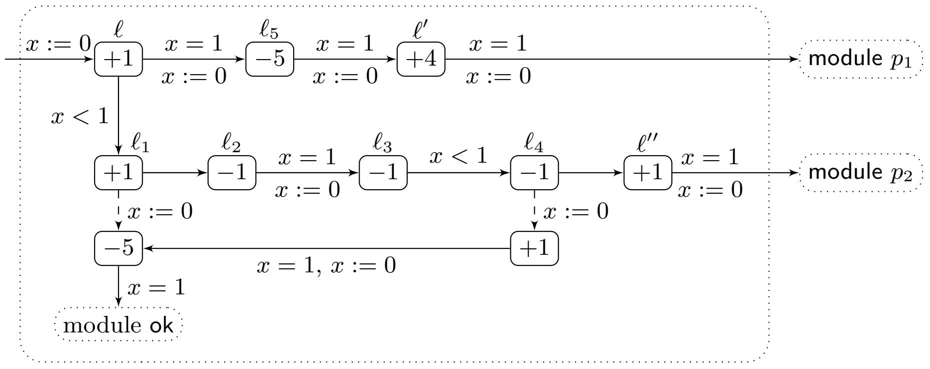


Fig. 8. Module $\mathsf{Test}(c_1 = c_2 = 0)$

The following result is easily proved:

Proposition 22. *Starting from configuration $(\ell, 0, 5 - e)$ in module* $\mathsf{Test}(c_1 = c_2 = 0)$*, Player 1 has a locally safe strategy if and only if $e \in [0, 1]$. In that case, the strategy is unique and has three outcomes, two of which end in module* ok*, and*

- *if $e = 1$, the third outcome exits the module to module p_1 via configuration $(\ell', 1, 4)$;*
- *otherwise, the third outcome exits to p_2 via $(\ell'', 1, 5 - e)$.*

***Module for testing* $\mathbf{c_1 = 0}$.** In this module, Player 1 will have a winning strategy if, and only if, the value of c_1 is zero when entering the module (*i.e.*, the accumulated weight is $5 - 1/3^k$ for some $k \in \mathbb{N}$). This is achieved by repeatedly decrementing c_2 until $c_1 = c_2 = 0$, as implemented in the module depicted on Fig. 9. A similar module can be constructed for counter c_2. Note that in this module, and also in module $\mathsf{Test}(c_1 > 0)$, we might decrement a counter which already has value 0. We show in the proof of the next proposition that this is not a problem.

Proposition 23. *Starting from configuration $(\ell, 0, 5 - e)$ in module* $\mathsf{Test}(c_1 = 0)$*, Player 1 has a locally safe strategy in this module if and only if $e = 1/3^k$ for some $k \in \mathbb{N}$. In that case, the strategy is unique.*

Proof. The reverse direction of the equivalence is immediate, applying the strategies from Prop. 21 and 22.

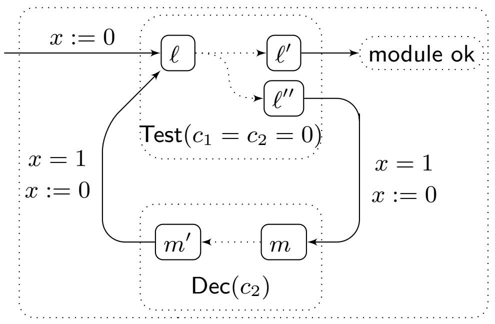


Fig. 9. Module Test($c_1 = 0$)

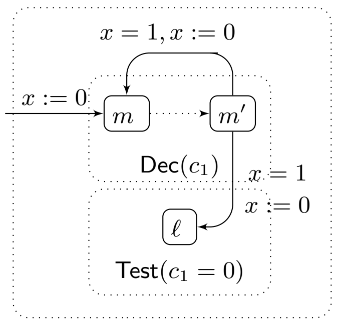


Fig. 10. Module Test($c_1 > 0$)

Conversely, from Prop. 22, if $e > 1$ then Player 1 has no winning strategy. We now prove by induction that if $3^{-k} < e < 3^{1-k}$ for some $k \in \mathbb{N}$, then there is no safe strategy. The base case ($k = 0$) has been handled above. Now, if $k > 0$, then from Prop. 22 there is only one safe strategy to exit submodule Test($c_1 = c_2 = 0$), yielding an outcome entering module Dec(c_2) in configuration $(m, 0, 5-e)$. Prop. 21 applies, and Player 1 has a unique strategy to safely exit that module. One of the outcomes then re-enters module Test($c_1 = c_2 = 0$) in configuration $(\ell, 0, 5 - 3e)$. From the induction hypothesis, there is no safe strategy from that configuration, which concludes the proof. □

Module for checking $\mathbf{c_1 > 0}$***.*** The module Test($c_1 > 0$), depicted on Fig. 10, uses the previous one: it first lets Player 1 decrement counter c_1 a positive number of times, and then checks whether $c_1 = 0$ using the previous module. The following proposition, whose proof is similar to the previous one, expresses the correctness of this module:

Proposition 24. *Starting from configuration $(\ell, 0, 5-e)$ in module* Test($c_1 > 0$)*, Player* 1 *has a locally safe strategy in this module if and only if $e = 1/(2^k \cdot 3^{k'})$ for some $k, k' \in \mathbb{N}$ with $k > 0$. In that case, the strategy is unique.*

Modules for conditional instructions. We now implement conditional instructions of the form "`if` $c_1 = 0$ `then goto` p_1 `else goto` p_2" of the two-counter machine. The corresponding module gives the choice to Player 1 to go either to p_1 or to p_2, and lets Player 2 check that the value of c_1 meets the requirement before proceeding to module p_1 or p_2. It is depicted on Fig. 11.

Proposition 25. *Starting from configuration $(t, 0, 5-e)$ in module* Cond($c_1 = 0$)*, Player* 1 *has a locally safe strategy in this module if and only if $e = 1/(2^{k_1} \cdot 3^{k_2})$ for some $k_1, k_2 \in \mathbb{N}$. In that case, the strategy is unique, and*

– *if $k_1 = 0$, one of the outcomes of the strategy visits module p_1, and all the other ones eventually reach module* ok*;*
– *if $k_1 > 0$, one of the outcomes of the strategy visits module p_2, and all the other ones eventually reach module* ok*.*

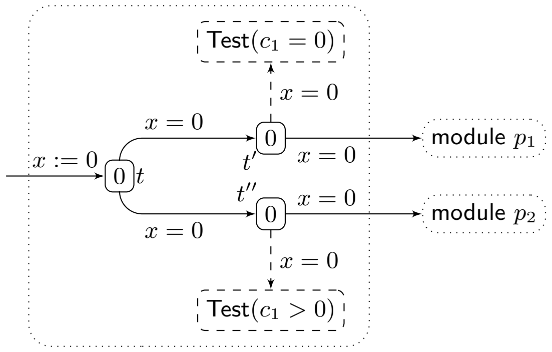


Fig. 11. Module Cond($c_1 = 0$)

srba: why it cannot elapse? Perhaps invariant $x \leq 0$ should be added?

Proof. Letting time elapse in t would block the game in that location.$\surd$ Thus, from t, Player 1 has only two possible strategies, visiting t' or t''. If e is not of the form $1/(2^{k_1}3^{k_2})$ with $k_1, k_2 \in \mathbb{N}$, then both strategies are losing since Player 2 will be able to enter one of the Test modules with that incorrect value.

If $e = 1/(2^{k_1}3^{k_2})$ with $k_1, k_2 \in \mathbb{N}$, then if $k_1 = 0$, the strategy visiting t' is locally safe, while if $k_1 > 0$, the other strategy is locally safe. $\square$

Simulation of the two-counter machine. Let $\mathcal{M}$ be a two-counter machine. We simulate an increment operation of the first (resp. second) counter by Inc(c_1) (resp. Inc(c_2)). We simulate a decrement operation of the form "`if` $c_1 = 0$ `then goto` p_i `else decrement(`c_1`); goto` p_j" by plugging the corresponding modules on module Cond($c_1 = 0$), and similarly for counter c_2. From the results above, we have the following proposition, entailing undecidability of the 1-clock interval-bound game problem.

Proposition 26. *Let $\mathcal{M}$ be a two-counter machine. We can construct a one-clock weighted timed game $G_{\mathcal{M}}$ which is a positive instance of the interval-bound problem if and only if $\mathcal{M}$ has an infinite computation.*